EDITORA UFRJ

Tópicos em otimização inteira

Ana Flavia Uzeda Macambira
Luidi Simonetti
Rosiane de Freitas Rodrigues
Nelson Maculan

Tópicos em Otimização Inteira

UFRJ Universidade Federal do Rio de Janeiro

Reitora Denise Pires de Carvalho

Vice-reitor Carlos Frederico Leão Rocha

Coordenadora do Fórum de Ciência e Cultura Tatiana Roque

Editora Universidade Federal do Rio de Janeiro

Diretor Marcelo Jacques de Moraes

Diretora adjunta Fernanda Ribeiro

Conselho editorial Marcelo Jacques de Moraes (presidente)
Cristiane Henriques Costa
David Man Wai Zee
Flávio dos Santos Gomes
João Camillo Barros de Oliveira Penna
Tania Cristina Rivera

Tópicos em Otimização Inteira

organização
Ana Flavia Uzeda Macambira
Luidi Simonetti
Rosiane de Freitas Rodrigues
Nelson Maculan

EDITORA UFRJ

©2022 Ana Flávia Uzeda Macambira, Luidi Simonetti, Rosiane de Freitas Rodrigues e Nelson Maculan

Ficha catalográfica elaborada por Maria Luiza Cavalcanti Jardim (CRB7-1878)

T674 Tópicos em otimização inteira [recurso eletrônico] / organizadores, Ana Flávia Uzeda Macambira, Luidi Simonetti, Rosiane de Freitas Rodrigues, Nelson Maculan. – Rio de Janeiro : Ed. UFRJ, 2022.
1 recurso eletrônico (258 p.) ; digital.

Bibliografia: p. 237-258.
ISBN 978-65-88388-10-5

1. Otimização matemática. I. Macambira, Ana Flávia Uzeda.

CDD: 519.6

Coordenação editorial
Maíra Alves
Thiago de Morais Lins

Revisão
Paula Halfeld

Capa
Louise Xavier Dantas

Fotografia da capa
Anne-Marie Maculan

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
FÓRUM DE CIÊNCIA E CULTURA

EDITORA UFRJ
Rua Lauro Müller, 5
Rio de Janeiro, RJ – CEP 22290-902

www.editora.ufrj.br

LIVRARIA EDITORA UFRJ
Av. Pasteur, 250 – Palácio Universitário
Rio de Janeiro, RJ – CEP 22290-160

vendas@editora.ufrj.br

Dedicatória

Dedicamos este livro à memória da nossa querida colega e coautora Laura Patuzzi.

Agradecimento

Gostaríamos de agradecer a Luiz Mariano Carvalho (Professor Titular da Universidade Estadual do Rio de Janeiro) pelo seu esforço e entusiasmo na publicação deste livro.

Lista de Figuras

2.1 Exemplo de um grafo planar 12
2.2 Exemplos de grafos rotulados: (a) Grafo G_1 com rótulos de identificação nos vértices; (b) Grafo G_2 com rótulos indicando peso nos vértices; (c) Grafo G_3 com rótulos indicando peso nas arestas. 13
2.3 Exemplo de grafos (a) G_1 e (b) G_2, que são isomorfos; e (c) G_3, que é não isomorfo. 14
2.4 Exemplo de um multigrafo G_1 e o grafo G_2 com laço. . . . 15
2.5 Exemplos de grafos que são (a) caminho e (b) ciclo. . . . 15
2.6 Grafo desconexo com cinco componentes conexas. 16
2.7 Operações de exclusão e inclusão de arestas e vértices em um grafo G. 17
2.8 Exemplo de um grafo e seu complemento. 19
2.9 Grafos completos. 19
2.10 Exemplos de grafos bipartidos em (a) e (b), que neste caso é também completo ($K_{3,3}$). 20
2.11 Exemplo de (a) um grafo G, (b) um subgrafo de G em (a) e (c) um subgrafo induzido de G. 21
2.12 Exemplos de árvores, que são grafos conexos e acíclicos. . 22
2.13 Todas as árvores com $n = 6$ vértices. 22
2.14 Os grafos (b) e (c) são subgrafos geradores de (a), e o grafo em (c) é uma árvore geradora dos outros dois grafos. 24
2.15 Exemplos de (a) árvore livre e (b) árvore enraizada. . . . 26
2.16 Exemplos de (a) subárvore e (b) subárvore parcial (relativas à árvore enraizada da figura 2.15(b)). 26
2.17 Árvores enraizadas isomorfas. 27
2.18 Exemplo de (a) um grafo conexo, em que (b) e (c) são exemplos de seus cortes. 28
2.19 Exemplo de um grafo planar 31

2.20 Exemplo de fluxo em redes. 39
2.21 Fluxos (a) maximal e (b) máximo. 40
2.22 Prova do lema 2.9.1. 41
2.23 Redes residuais. 42
2.24 Um exemplo de pior caso para o algoritmo 2.1. 45
2.25 Entrada para o exemplo da figura 2.26. 50
2.26 1ª iteração. Fluxo maximal 3. 51
2.27 2ª iteração. Fluxo maximal 2. 52
2.28 3ª iteração. Fluxo maximal 2. 53
2.29 4ª iteração. Fluxo maximal: $3+2+2=7$. 53

3.1 Grafo de precedências. 71
3.2 Ilustração da solução obtida. 79

4.1 Exemplo de um grafo não orientado $G=(V,E)$ com onze vértices. 85
4.2 Adicionando um vértice 0 ao grafo G_d e o conjunto de arcos $\{(0,j) \mid j \in V\}$. 86
4.3 Grafo conexo G_{SC}, subgrafo de G, obtido a partir de P_1. . 88
4.4 Soluções viáveis para o problema P_2, obtidas a partir do grafo G, considerando $v_0=2$ e $v_p=7$. 89
4.5 Grafo $\bar{G}=(V,E)$ obtido a partir de G. Neste grafo, consideramos $c_i=0$, para $i \in \{1,2,\ldots,11\}$, $w_{78}=w_{87}=-5$, $w_{79}=w_{97}=-5$ e $w_{89}=w_{98}=-2$. 90
4.6 Solução obtida pela resolução de P_2 a partir do grafo $\bar{G}$ (figura 4.5). 91
4.7 Ciclo elementar obtido a partir da solução de P_3, utilizando-se o grafo G, para $p=9$. 92
4.8 Solução viável obtida a partir do grafo G da figura 4.1 para o PCV_2, excluindo a restrição (4.31). Observamos aqui a ocorrência de subciclos. 94
4.9 Conjunto de caminhos de 1 a k, de onde se obtém um ciclo hamiltoniano de menor custo. A figura 4.9(k) representa a mesclagem de todos os caminhos obtidos. Note que a presença da aresta $\{9,10\}$, na figura 4.9(l), se deve à restrição (4.21). A figura 4.9(l) representa a solução ótima do PCV_5 para o grafo da figura 4.1, considerando-se todas as componentes c_i do vetor c iguais a zero. 100
4.10 Nas figuras 4.10(a) e 4.10(b), vemos duas árvores mínimas de Steiner obtidas como solução da formulação (4.77)–(4.83), utilizando-se como entrada o grafo G da figura 4.1. Os vértices em vermelho representam vértices de Steiner. . 104

5.1 Raios extremos. 108
5.2 Região viável limitada. 108
5.3 Divisão do conjunto de restrições. 109
5.4 Melhora da relaxação linear da geração de colunas. 118

7.1 Fechos de dois pontos. 134
7.2 Exemplos de conjuntos afins, cones e conjuntos convexos. 136
7.3 Multiplicação por escalar e soma de Minkowski. 137
7.4 Exemplo de conjunto convexo e seu cone de recessão. . . . 138
7.5 Dependência e independência. 140
7.6 Posto $\times$ posto-afim. 141
7.7 Hiperplano, semiespaço e separação. 142
7.8 Exemplos de poliedros em $\mathbb{R}^2$. Em particular, P_1, P_2 e P_3 são politopos. 143
7.9 Diferentes representações de um poliedro. 144
7.10 Pontos internos: x^1, y^2, z^3. Ponto interior: z^3. Não são internos: y^1, z^1 e z^2. P_1 e P_2 não têm ponto interior. 145
7.11 Ilustração do lema 7.3.1. 172
7.12 Desigualdade válida. 173
7.13 Faces de um poliedro. 173
7.14 Vértices e direções extremas. 173
7.15 Ilustração do Teorema de Minkowski. 174
7.16 Lifting. 174
7.17 Roda ímpar com sete vértices periféricos e um central. . . 175
7.18 Separação e plano de corte. 175
7.19 Resolução por planos de corte. 176

Lista de Tabelas

4.1 Resultados computacionais das formulações PAS_3 e PAS_3 Relaxado para instâncias STP OR-Library *b*. 105
4.2 Resultados computacionais das formulações PAS_3 e PAS_3 Relaxado para instâncias STP OR-Library *c*. 105
4.3 Resultados computacionais das formulações PAS_3 e $RPAS_3$ para instâncias STP OR-Library *d*. 106
4.4 Resultados computacionais das formulações PAS_3 e PAS_3 Relaxado para instâncias STP OR-Library *e*. 106

Sumário

Prefácio
Carlile Lavor
Presidente da SBMAC (biênio 2018-2019)

1 Tópicos em Otimização Linear 1
Ana Flávia Uzeda Macambira
Nelson Maculan

1.1 Programação Linear 1
1.1.1 Método primal do Simplex 2
1.1.2 Método dual do Simplex 4

2 Uma introdução à Teoria dos Grafos 11
Jayme Szwarcfiter
Rosiane de Freitas Rodrigues

2.1 Introdução 11
2.2 Definições básicas 12
2.3 Árvores 21
2.4 Conectividade 27
2.5 Planaridade 30
2.6 Coloração 33
2.7 Grafos direcionados 34
2.8 Fluxo máximo em redes 36
2.9 O Problema do Fluxo Máximo 37
2.9.1 O Teorema do Fluxo Máximo – Corte Mínimo . . 41
2.9.2 Um primeiro algoritmo 44
2.9.3 Um algoritmo $O(nm^2)$ 46
2.9.4 Um algoritmo $O(n^2m)$ 47
2.9.5 Um algoritmo $O(n^3)$ 51
2.10 Notas sobre este capítulo 54

3 Modelagem com variáveis 0-1 57

Ana Flávia Uzeda Macambira
Virginia Costa
Nelson Maculan
Gérard Plateau
Viviane Cátia Köhler

3.1 Introdução 57
3.2 Linearização 57
3.3 Variáveis 0-1 em equações diofantinas 63
3.4 Transformação em programação 0-1 66
3.5 Exemplo de aplicação 69
3.6 Linearização com o uso frações 0-1 80
3.7 Conclusões 83

4 Problemas NP-difíceis em Otimização em grafos 84
Virginia Costa
Ana Flávia Uzeda Macambira
Laura Bahiense
Laura Patuzzi
Nelson Maculan

4.1 Subgrafo conexo de G 85
4.2 Caminhos elementares 87
4.3 Ciclos elementares 90
4.3.1 Ciclos Hamiltonianos 91
4.4 Problema do Caixeiro Viajante 92
4.5 Árvores 99
4.5.1 Árvores geradoras 101
4.5.2 Problema da Árvore de Steiner em grafos 101

5 Geração de Colunas em Otimização Inteira 107
Luidi Simonetti
Nelson Maculan
Ana Flávia Uzeda Macambira
Pedro Henrique González

5.1 Geração de Colunas em Programação Linear 107
5.1.1 Decomposição Dantzig-Wolfe 116
5.2 Geração de Colunas em Programação Inteira 117
5.2.1 Geração de colunas do tipo Dantzig-Wolfe 117
5.2.2 Geração de colunas do tipo Gilmore-Gomory . . . 119
5.2.3 Ramificação e cortes 119
5.2.4 Estratégias de aceleração 120

6 Dualidade em Otimização Linear Inteira 122

Nelson Maculan
Ana Flávia Uzeda Macambira
Luidi Simonetti
Abilio Lucena

6.1 Introdução 122
6.2 Relaxação lagrangiana 123
6.3 Resolução do problema dual 124
6.3.1 Técnica de geração de colunas 124
6.3.2 Método utilizando subgradientes 126
6.4 Relaxação lagrangiana e enumeração implícita 127
6.4.1 Métodos de branch-and-bound 128
6.4.2 Um exemplo ilustrando a resolução do dual lagrangiano 129

7 Combinatória Poliédrica 132
Manoel Campêlo

7.1 Introdução 132
7.2 Conceitos e propriedades preliminares 133
7.2.1 Conjuntos afins, cônicos e convexos 133
7.2.2 Dimensão e posto 139
7.2.3 Hiperplanos e semiespaços 141
7.3 Poliedros 142
7.3.1 Introdução 142
7.3.2 Dimensão 144
7.3.3 Desigualdades válidas e faces 147
7.3.4 Facetas 149
7.3.5 Descrição de poliedros via desigualdades 150
7.3.6 Vértices e direções extremas 151
7.3.7 Descrição de poliedros por vértices e direções extremas 153
7.4 Fecho inteiro 155
7.4.1 Caracterização 155
7.4.2 Determinação da dimensão 157
7.4.3 Obtenção de facetas 159
7.4.4 Lifting 160
7.5 Poliedros inteiros 162
7.5.1 Caracterização 162
7.5.2 Sistemas TDI 163
7.5.3 Total unimodularidade 164
7.5.4 Reconhecimento de poliedros inteiros 167
7.6 Método de Planos de Corte 169

8 Otimização Não Linear Inteira Mista 177
Wendel Melo
Marcia Fampa
Fernanda Maria Pereira Raupp

8.1 Introdução 177
8.2 Branch-and-bound baseado em PNL 178
8.3 Algoritmos de aproximação linear 181
8.3.1 O problema mestre 182
8.3.2 Algoritmo Plano de Corte Estendido 183
8.3.3 Algoritmo Aproximação Externa 183
8.3.4 Algoritmo Decomposição de Benders Generalizada 186
8.3.5 Algoritmo branch-and-bound baseado em PL/PNL 187
8.3.6 Algoritmo Hiperplano de Suporte Estendido 189
8.4 Algoritmos de subproblemas contínuos 191
8.4.1 Algoritmo de Minimização de Gap de Integralidade 1 (para problemas binários) 191
8.5 Algoritmos híbridos 193
8.5.1 Algoritmo branch-and-bound Híbrido Bonmin . . . 193
8.5.2 Branch-and-bound Híbrido com Aproximação Externa 196
8.6 Considerações finais 198
8.6.1 Heurísticas 198
8.6.2 Pacotes computacionais 198

9 Algoritmos de Aproximação 199
Flavio Miyazawa
Lehilton Pedrosa

9.1 Introdução 199
9.2 Definições e algoritmos básicos 200
9.2.1 Um primeiro algoritmo de aproximação 200
9.2.2 Um segundo algoritmo de aproximação 202
9.2.3 Problemas em grafos e em espaços métricos 204
9.2.4 Um esquema de aproximação 206
9.3 Método do arredondamento 209
9.3.1 Limitantes por programação linear 209
9.3.2 Um algoritmo de arredondamento determinístico . 210
9.3.3 Arredondamento probabilístico 212
9.3.4 Explorando o politopo do programa linear 215
9.4 Método primal-dual 218
9.4.1 Um algoritmo de crescimento dual 218
9.4.2 Algoritmos primais-duais com duas fases 220

9.4.3 Programas lineares de tamanho exponencial 224
9.4.4 Método primal-dual e relaxação lagrangiana 229
9.5 Comentários finais 235

Prefácio

Carlile Lavor
Presidente da SBMAC (biênio 2018-2019)

Em nosso mundo digital e cada vez mais interligado, dois conceitos matemáticos se destacam por trás desse novo estilo de vida: algoritmos e grafos.

A todo instante, problemas em diversos setores da economia são resolvidos, maximizando ou minimizando algum critério. Tais problemas são representados matematicamente por meio de modelos de Otimização, que necessitam de algoritmos e grafos para a sua resolução computacional.

Contando com autores de várias gerações, de dezenas de universidades brasileiras e centros de pesquisa, incluindo pesquisadores da América Latina e França, este livro reúne uma combinação ímpar envolvendo três conceitos que fazem parte da fundamentação tecnológica do século XXI: algoritmos, grafos e otimização.

Para enfrentar os desafios de uma sociedade que se automatiza de forma acelerada, vai ganhar destaque o profissional com uma cultura diversificada, capaz de transitar entre as várias áreas do conhecimento.

Este livro apresenta uma diversidade de tópicos em Otimização, área estratégica e capaz de conectar vários ramos do conhecimento humano. Além de resultados clássicos e atuais, a obra enfatiza a Otimização Inteira e suas conexões com a Teoria dos Grafos, a Complexidade de Algoritmos, a Combinatória Poliédrica e os Algoritmos Aproximativos.

Não posso deixar de destacar o espírito jovem dos organizadores e, de forma especial, a grande capacidade de agregar talentos, cada vez mais importante em nossos dias, do professor Nelson Maculan.

Autores(as) que colaboraram com este livro

Abílio Lucena (Universidade Federal do Rio de Janeiro)
Ana Flávia Uzeda Macambira (Universidade Federal da Paraíba)
Fernanda Maria Pereira Raupp (Laboratório Nacional de Computação Científica)
Flávio Keidi Miyazawa (Universidade Estadual de Campinas)
Gérard Plateau (Université Paris XIII)
Jayme Luiz Szwarcfiter (Universidade Federal do Rio de Janeiro)
Laura Bahiense (Universidade Federal do Rio de Janeiro)
Laura Patuzzi (Universidade Federal do Rio de Janeiro)
Lehilton Lelis Chaves Pedrosa (Universidade Estadual de Campinas)
Luidi Simonetti (Universidade Federal do Rio de Janeiro)
Manoel Campêlo (Universidade Federal do Ceará)
Marcia Helena Costa Fampa (Universidade Federal do Rio de Janeiro)
Nelson Maculan (Universidade Federal do Rio de Janeiro)
Pedro Henrique González Silva (Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - CEFET-RJ)
Rosiane de Freitas Rodrigues (Universidade Federal do Amazonas)
Virgínia Costa (Universidade Federal do Rio de Janeiro)
Viviane Cátia Köhler (Universidade Federal de Santa Maria)
Wendel Melo (Universidade Federal de Uberlândia)

1

Tópicos em Otimização Linear

Ana Flávia Uzeda Macambira
Nelson Maculan

Para a resolução de problemas de programação linear, os métodos mais utilizados são o primal e o dual do Simplex. Ver [21, 23, 50, 57, 13, 154, 178, 195, 35, 100, 152, 14].

1.1 Programação Linear

Um problema de programação linear (PL) é definido da seguinte maneira:

$$(PL): \text{Minimizar} \quad z = c^\top x, \tag{1.1}$$

$$\text{sujeito a:} \quad Ax = b, \tag{1.2}$$

$$x \geq 0, \tag{1.3}$$

em que são conhecidos $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$, $b \in \mathbb{R}^m$, e $x \in \mathbb{R}^n$ é o vetor das variáveis $x_1, x_2, ..., x_n$.
Sejam $A = (a_1, a_2, ..., a_n)$, $a_j \in \mathbb{R}^m$, $a_j = (a_{1j}, a_{2j}, ..., a_{mj})^\top$, $c = (c_1, c_2, ..., c_n)^\top$, $b = (b_1, b_2, ..., b_m)^\top$ e $x = (x_1, x_2, ..., x_n)^\top$. O sistema das restrições de igualdades em (6.2) pode também conter desigualdades, isto é, $\sum_{j=1}^{n} a_{ij}x_j \leq b$ que podem ser substituídas por $\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i$ e $x_{n+i} \geq 0$, em que x_{n+i} será uma variável de folga. A equação $z = c^\top x$, $z \in \mathbb{R}$, em (6.1), é

a função-objetivo que desejamos minimizar. As restrições (6.2) e (6.3) formam o conjunto de soluções viáveis do problema.

1.1.1 Método primal do Simplex

Supondo que $\rho(A) = m$, em que $\rho(A)$ representa o posto da matriz A, então haverá uma matriz $B \in \mathbb{R}^{m\times m}$, formada por m colunas de A, tal que exista B^{-1}. Particionaremos A e c :

$A = (B\ N)$ e $c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}$, $N \in \mathbb{R}^{m\times(n-m)}$, $c_B \in \mathbb{R}^m$ e $c_N \in \mathbb{R}^{m\times(n-m)}$.

$x_B = (x_{B(1)},\ x_{B(2)}, ...,\ x_{B(m)})^\top \in \mathbb{R}^m$

$y_j = B^{-1}a_j$, $y_j = (y_{1j},\ y_{2j}, ...,\ y_{mj})^\top$, $j = 1, 2, ...n$;

(PL) pode ser escrito da seguinte maneira:

$$(PL): \text{Minimizar} \quad z = c_B^\top x_B + c_N^\top x_N, \tag{1.4}$$
$$\text{sujeito a:} \quad Bx_B + Nx_N = b, \tag{1.5}$$
$$x_B \geq 0 \text{ e } x_N \geq 0. \tag{1.6}$$

Como existe B^{-1}, podemos reescrever (6.5):

$x_B = B^{-1}b - B^{-1}Nx_N$ ou ainda

$$(PL): \text{Minimizar} \quad z = c_B^\top B^{-1}b + (c_N^\top - c_B^\top B^{-1}N)x_N, \tag{1.7}$$
$$\text{sujeito a:} \quad x_B = B^{-1}b - B^{-1}Nx_N, \tag{1.8}$$
$$x_B \geq 0 \text{ e } x_N \geq 0. \tag{1.9}$$

Em (6.18), (6.19) e (6.20), fazendo $x_N = 0$, obtemos uma solução

$$\bar{x} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$$

denominada solução básica de (6.2) ou (6.5) ou ainda (6.19). Quando $B^{-1}b \geq 0$ dizemos que $\bar{x}$ é uma solução básica de (6.2) e viável do (PL).

Sejam $u^\top = c_B^\top B^{-1}$, onde $u \in \mathbb{R}^m$, e I_N o conjunto de índices das colunas da matriz N:

$$(c_N^\top - c_B^\top B^{-1}N)x_N = \sum_{j\in I_N}(c_j - ua_j)x_j = \sum_{j\in I_N}(c_j - z_j)x_j,$$

onde $z_j = u^\top a_j, \quad j = 1, 2, ..., n-m.$

Sejam $\bar{x}_B = B^{-1}b = (\bar{x}_{B(1)},\ \bar{x}_{B(2)}, ...,\ \bar{x}_{B(m)})^\top$ e $I_B = \{B(1),\ B(2), ...,\ B(m)\}$ o conjunto de índices das colunas da matriz B, podemos ainda escrever (PL) da seguinte forma:

$$(PL): \text{Minimizar} \quad z = u^\top b + \sum_{j \in I_N} (c_j - z_j)x_j, \tag{1.10}$$

$$\text{sujeito a:} \quad x_{B(i)} = \bar{x}_{B(i)} - \sum_{j \in I_N} y_{ij}x_j, \quad i = 1, 2, ..., m, \tag{1.11}$$

$$x_{B(i)} \geq 0, \quad i = 1, 2, ..., m, \quad x_j \geq 0,\ j \in I_N. \tag{1.12}$$

Supondo que uma solução básica (primal viável) $\bar{x} = \begin{pmatrix} \bar{x}_B \\ 0 \end{pmatrix} \geq 0$ e que $(c_j - z_j) \geq 0,\ j \in I_N$, então $\bar{x}$ é uma solução ótima de (PL), isto é, val(PL)=$u^\top b = c_B^\top B^{-1} b = c_B^\top \bar{x}_B$, em que val$(\cdot)$ representa o valor ótimo da função-objetivo do problema $(\cdot)$.

Demonstração. Como $z = u^\top b + \sum_{j \in I_N} (c_j - z_j)x_j$ e $(c_j - z_j) \geq 0$, temos que qualquer $z \geq u^\top b = c_B^\top \bar{x}_B$ significará $\bar{x}$ ótimo. □

Teorema 1.1.1. Se houver solução ótima finita, então uma será básica.

Demonstração. Suponha que exista $\hat{x}$ solução viável do (PL), então temos $\hat{z} = c^\top \hat{x}$ e podemos, por operações polinomiais, transformar $\hat{x}$ numa solução básica $\bar{x}$ de (PL) tal que $c^\top \bar{x} \leq c^\top \hat{x}$, ver [154]. □

Caso haja em (6.21) um $(z_k - c_k) < 0$ para $k \in I_N$, a solução $\bar{x} = \begin{pmatrix} \bar{x}_B \\ 0 \end{pmatrix}$ pode não ser mais ótima.

$$z = u^\top b + (c_k - z_k)x_k + \sum_{j \in I_N \setminus \{k\}} (c_j - z_j)x_j$$

Se $x_k > 0$ e $x_j = 0,\ j \in I_N \setminus \{k\}$, o valor de z decrescerá. Podemos considerar

$x_{B(i)} = \bar{x}_{B(i)} - y_{ik}x_k \geq 0, \quad i = 1, 2, ..., m$

e podem ocorrer duas situações:

- $y_{ik} \leq 0 \rightarrow x_{B(i)} = \bar{x}_{B(i)} - (y_{ik}x_k) = \bar{x}_{B(i)} + \alpha,\ \alpha \geq 0;$
- $y_{ik} > 0 \rightarrow y_{ik}x_k \leq \bar{x}_{B(i)} \rightarrow x_k \leq \frac{\bar{x}_{B(i)}}{y_{ik}}.$

Sejam $L_< = \{i \mid y_{ik} \leq 0\}$ e $L_> = \{i \mid y_{ik} > 0\}$. Se $L_> = \varnothing$, o ótimo de (PL) é ilimitado, pois $x_k \to \infty$, fazendo com que $z \to -\infty$. Caso contrário, tomamos $\alpha_k = \frac{\bar{x}_{B(s)}}{y_{sk}} = \min\limits_{i \in L_>} \left\{ \frac{\bar{x}_{B(i)}}{y_{ik}} \right\}$. Faremos $x_k = \alpha_k \to \bar{x}_{B(s)} = 0$, o que significa remover da base $B = (a_{B(1)}\ a_{B(2)}\ \ldots\ a_{B(s-1)}\ a_{B(s)}\ a_{B(s+1)}\ \ldots\ a_{B(m)})$ a coluna $a_{B(s)}$ e colocar no seu lugar a coluna a_k, ficando a base $B = (a_{B(1)}\ a_{B(2)}\ \ldots\ a_{B(s-1)}\ a_k\ a_{B(s+1)}\ \ldots\ a_{B(m)})$. Para a nova matriz B, temos que calcular a sua inversa B^{-1} e atualizar os conjuntos I_B e I_N. Colocaremos o (PL) (6.21)-(6.23) sob uma outra forma. Como o número de soluções básicas é finito, e se não houver ciclagem, o método encontrará uma solução ótima.

1.1.2 Método dual do Simplex

A cada problema primal de programação linear está associado a um outro problema, denominado problema dual. Esse par de problemas está intimamente interligado. Seguem alguns resultados.

Problemas duais

Consideremos o par de problemas primal (PL) e dual (DL) abaixo:

$$\begin{aligned} (PL): \text{Minimizar} \quad & z = c^\top x, \\ \text{sujeito a:} \quad & Ax = b, \\ & x \geq 0. \end{aligned}$$

e

$$\begin{aligned} (DL): \text{Maximizar} \quad & w = b^\top v, \\ \text{sujeito a:} \quad & A^\top v \leq c, \\ & v \in \mathbb{R}^m. \end{aligned}$$

Observação: O dual do problema dual é o problema primal.

Tomemos uma submatriz $B \in \mathbb{R}^{m \times m}$ de A, tal que exista B^{-1}. Consideremos $u^\top = c_B B^{-1}$ tal que $A^\top u \leq c$, ou seja, $u^\top A \leq c^\top \to u a_j \leq c_j,\ j = 1, 2, \ldots, n$. Então

$c_j - u a_j \geq 0$, ou seja, $c_j - z_j \geq 0$.

Dizemos que B é uma base dual viável de (PL) quando u tal que $u^\top = c_B B^{-1}$ for uma solução viável de (DL).

Uma base B primal e dual viável é ótima.

Teorema 1.1.2 (Teorema Fundamental da Dualidade). Com relação aos problemas de programação linear dual e primal, exatamente uma das seguintes cláusulas é verdadeira:

- Ambos possuem soluções ótimas x^* e w^*, com $cx^* = w^*b$.
- Um dos problemas tem valor de função-objetivo ilimitado. Nesse caso, o outro problema é inviável.
- Ambos os problemas são inviáveis.

Teorema 1.1.3 (Teorema das Folgas Complementares). Dados os problemas primal e dual abaixo:

$$\begin{array}{c} (P): \text{ Minimizar } z = cx, \\ \text{sujeito a: } Ax \geq b, \\ x \geq 0 \end{array} \qquad \begin{array}{c} (D): \text{ Maximizar } d = wb, \\ \text{sujeito a: } wA \leq c, \\ w \geq 0 \end{array}$$

e sejam x^* e w^* soluções viáveis para (P) e (D) respectivamente. Essas soluções serão ótimas para (P) e (D) se, e somente se,

(i) $Ax^* \geq b,\ x^* \geq 0$;

(ii) $w^*A \leq c,\ w^* \geq 0$;

(iii)

$$w^*(Ax^* - b) = 0 \tag{1.13}$$
$$(w^*A - c)x^* = 0 \tag{1.14}$$

Supondo que a matriz $A \in \mathbb{R}^{m\times m}$, podemos escrever:
$w^*(Ax^* - b) = \sum_{i=1}^{m} w_i^*[(\sum_{j=1}^{n} a_{ij}x_j^*) - b_i] = 0.$
Como todos os m termos
$w_i^*[(\sum_{j=1}^{n} a_{ij}x_j^*) - b_i]$ desta soma são não negativos, isso implica que
$w_i^*[(\sum_{j=1}^{n} a_{ij}x_j^*) - b_i] = 0,\ i = 1,2,...,m.$
Assim, temos: $(w^*A - c)x^* = \sum_{j=1}^{n}[(\sum_{i=1}^{m} w_i^*a_{ij}) - c_j]x_j^* = 0.$
Como todos os n termos desta soma são não positivos, temos:
$[(\sum_{i=1}^{m} w_i^*a_{ij}) - c_j]x_j^* = 0,\ j = 1,2,...,n.$
Este item é denominado Teorema das Folgas Complementares.

A condição (i) exige que a solução x^* seja viável para o problema primal.
A condição (ii) exige que a solução w^* seja viável para o problema dual.
A condição (iii) exige que um dos termos seja nulo, ou seja:
em relação à equação 1.13, se $Ax^* - b \neq 0 \rightarrow w^* = 0$ e, da mesma forma, se $w^* \neq 0 \rightarrow Ax^* - b = 0$.
Em relação à equação 1.14, se $w^*A - c \neq 0 \rightarrow x^* = 0$ e, da mesma forma, se $x^* \neq 0 \rightarrow w^*A - c = 0$.

Uma observação associada ao método Simplex
Suponha o par de problemas abaixo:

$$(P): \begin{array}{c} \text{Minimizar } z = cx, \\ \text{sujeito a: } Ax = b, \\ x \geq 0. \end{array} \qquad (D): \begin{array}{c} \text{Maximizar } D = wb, \\ \text{sujeito a: } wA \leq c, \\ w \text{ livre.} \end{array}$$

Tomemos qualquer solução básica $\bar{x}$ de $Ax = b$, satisfazendo ou não $\bar{x} \geq 0$, que será da forma $x = (B^{-1}b \mid 0)$, $0 \in \mathbb{R}^{n-m}$. Nas condições do teorema 1.1.3, item (iii), acima, só nos interessa a equação 1.14, pois a equação 1.13 $(A\bar{x} = b)$ será neste caso sempre verificada. Veremos o que se passa com a condição 1.14 quando consideramos um $\bar{w} = c_B B^{-1}$ e $z_j = \bar{w}a_j$, $j = 1, 2, ..., n$; assim podemos escrever: $\sum_{j=1}^{n}(z_j - c_j)\bar{x}_j = 0$, pois quando a variável $\bar{x}_j$ for básica, $z_j - c_j = 0$ e, quando não básica, $\bar{x}_j = 0$, logo $(\bar{w}A - c)\bar{x} = 0$. Observamos que $\bar{x}$ e $\bar{w}$ satisfazem 1.14. Isso é exatamente o que se passa nos métodos primal e dual do Simplex, e o Teorema das Folgas Complementares sempre é respeitado.
Exemplo (Teorema das Folgas Complementares):
Resolvendo o primal através do dual. Considere os seguintes problemas primal e dual respectivamente:

$$\begin{aligned} \text{Minimizar } 2x_1 + 3x_2 + 5x_3 + 2x_4 + 3x_5, \\ \text{sujeito a: } x_1 + x_2 + 2x_3 + x_4 + 3x_5 \geq 4, \\ 2x_1 - 2x_2 + 3x_3 + x_4 + x_5 \geq 3, \\ x_1, x_2 \geq 0. \end{aligned}$$

$$\begin{aligned} \text{Maximizar } 4w_1 + 3w_2, \\ \text{sujeito a: } w_1 + 2w_2 \leq 2, \\ w_1 - 2w_2 \leq 3, \\ 2w_1 + 3w_2 \leq 5, \\ w_1 + w_2 \leq 2, \\ 3w_1 + w_2 \leq 3, \\ w_1, w_2 \geq 0. \end{aligned}$$

A solução ótima do problema dual é: $w_1^* = 4/5, w_2^* = 3/5$. Com essas soluções dadas, vamos verificar quais restrições têm folga. A primeira restrição do problema dual é:
$w_1 + 2w_2 \leq 2$.
Substituindo os valores da solução ótima obtemos:
$\frac{4}{5} + 2\frac{3}{5} = \frac{10}{5} = 2$, portanto, a primeira restrição não tem folga, pois o lado esquerdo fica igual a 2, resultando numa igualdade, não tendo, desta

forma, como afirmarmos nada sobre o valor da variável x_1^*.
A segunda restrição do dual é: $w_1 - 2w_2 \leq 3$.
Da mesma forma, substituindo os valores da solução ótima, obtemos:
$\frac{4}{5} - 2\frac{3}{5} = -\frac{2}{5}$.
Então, na segunda restrição, temos folga, já que, no lado esquerdo, temos o valor $2/5$ e, no lado direito, temos o valor 3.
Segundo o Teorema das Folgas Complementares, $(w^*A - c)x^* = 0$.
Na segunda restrição já vimos que $w^*A - c \neq 0$, portanto, a variável do problema primal relativa a essa restrição dual, a saber, x_2^*, tem que ser igual a zero. Logo, já temos que $x_2^* = 0$.
Analisando a terceira restrição do problema dual, temos:
$2w_1 + 3w_2 \leq 5$,
e substituindo os valores ótimos do dual temos que o lado esquerdo da restrição vale
$2\frac{4}{5} + 3\frac{3}{5} = \frac{8}{5} + \frac{9}{5} = \frac{17}{5}$.

Então, na terceira restrição, temos folga também, já que, no lado esquerdo, temos o valor de $17/5$ e, no lado direito, temos o valor 5. De forma análoga ao que foi feito para a segunda restrição, temos que $x_3^* = 0$.
Para a quarta restrição dual $w_1 + w_2 \leq 2$, temos:
$\frac{4}{5} + \frac{3}{5} = \frac{7}{5}$, e como essa restrição também tem folga, $x_4^* = 0$.
Para a quinta e última restrição dual, $3w_1 + w_2 \leq 3$
$3\frac{4}{5} + \frac{3}{5} = \frac{15}{5} = 3$, e como essa restrição não tem folga, não temos como dizer nada sobre o valor da variável x_5^*.
Agora, vamos analisar a outra equação do Teorema das Folgas Complementares, ou seja, a equação $w^*(Ax^* - b) = 0$.
Como a variável dual $w_1^* \neq 0$, temos que a primeira restrição do problema primal não pode ter folga, ou seja, $Ax^* = b$, então temos que
$x_1^* + x_2^* + 2x_3^* + x_4^* + 3x_5^* = 4$.
Da mesma forma, como $w_2^* \neq 0$, temos que a segunda restrição do problema primal também não pode ter folga, então,
$2x_1^* - 2x_2^* + 3x_3^* + x_4^* + x_5^*\partial = 3$.
Como já temos que $x_2^* = x_3^* = x_4^* = 0$, temos duas equações e duas variáveis, ou seja, um sistema possível determinado

$$\begin{aligned} x_1^* + 3x_5^* &= 4 \\ 2x_1^* + x_5^* &= 3 \end{aligned}$$

que resulta em $x_1^* = 1$ e $x_5^* = 1$.
Portanto, através da solução ótima do problema dual e usando o Teorema das Folgas Complementares descobrimos a solução ótima para o problema primal.

O problema descrito por (6.21), (6.22) e (6.23) também poderá ser resolvido por um outro método: o dual do Simplex.

$$
\begin{aligned}
(PL): \text{Minimizar} \quad & z = u^\top b + \sum_{j \in I_N} (c_j - z_j) x_j, \\
\text{sujeito a:} \quad & x_{B(i)} = \bar{x}_{B(i)} - \sum_{j \in I_N} y_{ij} x_j, \ i \in I_B, \\
& x_{B(i)} \geq 0, \quad i = 1, 2, ..., m, \quad x_j \geq 0, \ j \in I_N,
\end{aligned}
$$

em que supomos que $c_j - z_j \geq 0, \quad j \in I_N$ (é fácil verificar que $c_j - z_j = 0, \ j \in I_B$).

Se $\bar{x}_{B(i)} \geq 0, \quad i = 1, 2, ..., m$ a solução $\bar{x} = \begin{pmatrix} \bar{x}_B \\ 0 \end{pmatrix} \in \mathbb{R}^n$ é primal e dual viável, então é ótima.

Vamos supor que exista p tal que $\bar{x}_{B(p)} < 0$, logo $\bar{x}_{B(p)}$ não é primal viável.

$x_{B(p)} = \bar{x}_{B(p)} - \sum_{j \in I_N} y_{pj} x_j$

se $y_{pj} \geq 0$ para todo $j \in I_N$ o problema (PL) é vazio e o (DL) é ilimitado.

Seja $L_< = \{j \mid y_{pj} < 0\}$. Consideremos

$$
\left| \frac{c_k - z_k}{y_{pk}} \right| = \min_{j \in L_<} \left\{ \left| \frac{c_j - z_j}{y_{pj}} \right| \right\} \tag{1.15}
$$

A coluna a_k substituirá a coluna $a_{B(p)}$ na nova matriz B. O processo continua até que $\bar{x}_B \geq 0$.

A equação (6.24) garantirá que os novos $(c_j - z_j) \geq 0, \quad j \in I_N$.

Consideremos (PL):

$$
\begin{aligned}
(PL): \text{Minimizar} \quad & z = c^\top x, && (1.16) \\
\text{sujeito a:} \quad & Ax = b, && (1.17) \\
& x \geq 0, && (1.18)
\end{aligned}
$$

e B ótima para (PL).

Suponhamos que quiséssemos introduzir no (PL) já resolvido acima (6.25)-(1.18) uma restrição do tipo

(i) $d^\top x \leq \alpha$, onde $d \in \mathbb{R}^m$ e $\alpha \in \mathbb{R}$,

dando origem a $(\overline{\text{PL}})$.

$$
\begin{aligned}
(\overline{\mathrm{PL}}): \text{Minimizar} \quad & z = c^\top x, \\
\text{sujeito a:} \quad & Ax = b, \\
& d^\top x + x_{n+1} = \alpha, \\
& x_j \geq 0, \quad j = 1,2,...,n+1,
\end{aligned}
$$

$$
\begin{array}{c} Ax = b \\ d^\top x + x_{n+1} = \alpha \\ x_j \geq 0,\ j = 1,2,...,n+1 \end{array}
\quad \left| \begin{pmatrix} A & 0 \\ d^\top & 1 \end{pmatrix} \begin{pmatrix} x \\ x_{n+1} \end{pmatrix} = \begin{pmatrix} b \\ \alpha \end{pmatrix},\ x_j \geq 0\ j = 1,...,n,n+1 \right.
$$

e $c_{n+1} = 0$. Consideramos $\bar{B} = \begin{pmatrix} B & 0 \\ d_B^\top & 1 \end{pmatrix}$, onde $d = \begin{pmatrix} d_B \\ d_N \end{pmatrix}$.

Caso exista B^{-1}, então existirá também $\bar{B}^{-1}$. Por quê? ?

Logo, $\bar{B}^{-1} = \begin{pmatrix} B^{-1} & 0 \\ -d_B^\top B^{-1} & 1 \end{pmatrix}$.

Fazemos $\bar{u} = \bar{c}_B^\top (\bar{B}^{-1}) = \begin{pmatrix} c_B^\top & 0 \end{pmatrix} \begin{pmatrix} B^{-1} & 0 \\ -d_B^\top B^{-1} & 1 \end{pmatrix} = \begin{pmatrix} c_B^\top B^{-1} & 0 \end{pmatrix}$.

$\bar{z}_j = \bar{u}^\top \bar{a}_j = u^\top a_j = z_j$, logo, $(\bar{c}_j - \bar{z}_j) \geq 0,\ j = 1,2,...,n,n+1$ e $c_{n+1} - z_{n+1} = 0$, pois x_{n+1} está na base. Logo, a base $\bar{B}$ é dual viável e o procedimento de otimização via o método dual do Simplex pode continuar. Se $\bar{B}$ for também primal viável será uma base ótima.

Suponhamos agora que quiséssemos introduzir no (PL) já resolvido (6.25)-(1.18) uma restrição do tipo

(ii) $d^\top x = \alpha$, onde $d \in \mathbb{R}^m$ e $\alpha \in \mathbb{R}$, isto é,

$d^\top x \leq \alpha$ e $d^\top x \geq \alpha \leftrightarrow (-d^\top x \leq -\alpha)$, o que dá origem a:

$$
\left\{ \begin{array}{c} d^\top x + x_{n+1} \qquad = \alpha \\ -d^\top x + \qquad x_{n+2} = -\alpha \\ x_{n+1} \geq 0,\ x_{n+2} \geq 0 \end{array} \right.
$$

E o novo (PL) é:

$$
\begin{aligned}
(\overline{\overline{\mathrm{PL}}}): \text{Minimizar} \quad & z = c^\top x + c_{n+1} x_{n+1} + c_{n+2} x_{n+2} \\
\text{sujeito a:} \quad & Ax = b, \\
& d^\top x + x_{n+1} = \alpha, \\
& -d^\top x + x_{n+2} = -\alpha, \\
& x_j \geq 0, \quad j = 1,2,...,n+2,
\end{aligned}
$$

onde $c_{n+1} = c_{n+2} = 0$. A base inicial

$$\bar{\bar{B}} = \begin{pmatrix} B & 0 & 0 \\ d^\top & 1 & 0 \\ -d^\top & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} B^{-1} & 0 & 0 \\ -d^\top B^{-1} & 1 & 0 \\ d^\top B^{-1} & 0 & 1 \end{pmatrix}$$

$$\bar{\bar{B}}^{-1} \begin{pmatrix} x_B \\ x_{n+1} \\ x_{n+2} \end{pmatrix} = \begin{pmatrix} b \\ \alpha \\ -\alpha \end{pmatrix} \Rightarrow \begin{pmatrix} x_B \\ x_{n+1} \\ x_{n+2} \end{pmatrix} = \begin{pmatrix} B^{-1}b \\ -d^\top B^{-1}b + \alpha \\ d^\top B^{-1}b + \alpha \end{pmatrix} = \bar{\bar{x}}_B.$$

$\bar{\bar{u}}^\top = (c_B^\top B^{-1}, 0, 0)$, que é uma solução viável para o dual de $(\overline{\overline{\mathrm{PL}}})$, portanto, caso essa solução seja primal viável também, ela é otima. Caso contrário, prosseguimos com o algoritmo até que a solução ótima seja encontrada, caso exista.

2

Uma introdução à Teoria dos Grafos

Jayme Szwarcfiter
Rosiane de Freitas Rodrigues

2.1 Introdução

Neste capítulo são apresentados alguns conceitos básicos de Teoria dos Grafos, de modo a cobrir o necessário à compreensão dos problemas e algoritmos em grafos abordados nos capítulos seguintes. A terminologia e notação utilizadas posteriormente nos problemas são também aqui introduzidas.

Alguns tópicos de fundamental importância para a modelagem e resolução de diversos problemas de diferentes áreas recebem um maior destaque. Por exemplo, algumas classes especiais de grafos, como árvores, digrafos (grafos direcionados) e grafos planares. Do mesmo modo, problemas clássicos em grafos, como coloração de vértices e fluxo em redes, são também abordados. No caso de fluxo em redes, diversos problemas de Otimização Combinatória podem ser resolvidos mediante sua transformação em um caso de fluxo máximo em redes. O Teorema do Fluxo Máximo de Corte Mínimo, fundamental na teoria, bem como alguns algoritmos de resolução de tal problema, são discutidos ao final.

Este capítulo está baseado, fortemente, no livro [201].

2.2 Definições básicas

Um grafo $G = (V, E)$ é um conjunto finito não vazio V e um conjunto E de pares não ordenados de elementos distintos de V. G é dito trivial quando $|V| = 1$. Quando necessário, utiliza-se o termo grafo não direcionado para designar um grafo simples, como o descrito. Os elementos de V são os vértices e os de E são as arestas de G, respectivamente. Cada aresta $e \in E$ será denotada pelo par de vértices $e = (v, w)$ que a forma. Nesse caso, os vértices v, w são os extremos (ou extremidades) da aresta e, denominados adjacentes. A aresta e é dita incidente a ambos, v e w. Duas arestas que possuem um extremo comum são chamadas de adjacentes. Utilizaremos a notação $n = |V|$ e $m = |E|$.

Podemos visualizar graficamente um grafo através de sua representação geométrica, na qual seus vértices correspondem a pontos distintos do plano em posições arbitrárias, enquanto que a cada aresta (v, w) é associada uma linha arbitrária unindo os pontos correspondentes a v, w. Para maior facilidade de exposição, no decorrer do texto será utilizado o termo grafo, significando também a sua representação geométrica. E, seja G um grafo e R uma representação geométrica de G em um plano, a representação R é chamada plana quando não houver cruzamento de linhas em R, a não ser nos extremos (vértices), naturalmente. Um grafo planar é aquele que admite alguma representação plana.

Um grafo conexo $G(V, E)$ é aquele em que existe caminho entre cada par de vértices de G, caso contrário G é desconexo. Um caminho em um grafo é uma sequência finita ou infinita de vértices conectados por uma sequência de arestas. Observe que a representação geométrica de um grafo desconexo é necessariamente descontígua. Em especial, o grafo G será totalmente desconexo quando não possuir arestas.

Na figura 2.1 é apresentado um grafo simples, não direcionado, conexo e que também é planar.

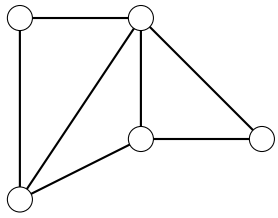
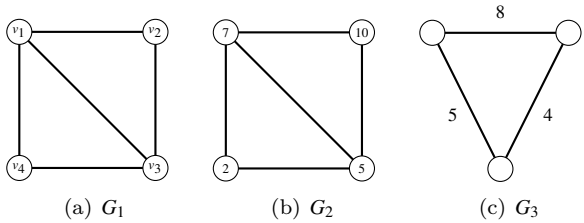


Figura 2.1: Representação gráfica de um grafo simples, não direcionado e conexo, com $n = 5$ vértices e $m = 7$ arestas. Tal grafo também é planar.

Um grafo será denominado rotulado em vértices (ou arestas) quando a cada vértice (ou aresta) estiver associado um conjunto, ou função, denominado rótulo. Um grafo ponderado é um tipo especial de grafo rotulado, no qual os rótulos são números, ou seja, é um grafo no qual cada aresta (vértice) recebe um valor numérico, geralmente um inteiro positivo, denominado peso da aresta (ou vértice). Por exemplo, os grafos da figura 2.2 são todos rotulados, sendo que o grafo 2.2(a) possui rótulos de identificação nos vértices, o grafo 2.2(b) possui rótulos numéricos nos vértices (geralmente usados para identificá-los unicamente, mas que podem também ser pesos associados) e, por fim, o grafo 2.2(c) possui pesos nas arestas.



(a) G_1 (b) G_2 (c) G_3

Figura 2.2: Exemplos de grafos rotulados: (a) Grafo G_1 com rótulos de identificação nos vértices; (b) Grafo G_2 com rótulos indicando peso nos vértices; (c) Grafo G_3 com rótulos indicando peso nas arestas.

A partir dessas definições iniciais, podemos formular a seguinte questão: dadas duas representações geométricas, elas correspondem a um mesmo grafo ou não? Em outras palavras, é possível fazer coincidir, respectivamente, os pontos de duas representações geométricas, de modo a preservar adjacência (ou seja, de modo a fazer também coincidir as arestas)? Formalmente, o problema pode ser enunciado da seguinte maneira: dados dois grafos $G_1(V_1,E_1)$, $G_2(V_2,E_2)$, com $|V_1| = |V_2| = n$, existe uma função unívoca $f : V_1 \to V_2$, tal que $(v,w) \in E1$ se, e somente se, $(f(v), f(w)) \in E_2$, para todo $v,w \in V1$. Em caso positivo, G_1 e G_2 são ditos isomorfos entre si. Por exemplo, as representações geométricas dos grafos da figura 2.3(a) G_1 e figura 2.3(b) G_2 podem se tornar coincidentes, mediante a aplicação da função f indicada na figura. Logo, 2.3(a) G_1 e 2.3(b) G_2 são isomorfos entre si. Contudo, não existe função f que faça coincidir as representações 2.3(a) G_1 e 2.3(c) G_3. Logo, 2.3(c) G_3 é não isomorfo a ambos G_1 e G_2.

Esse problema, denominado isomorfismo de grafos, pode ser facil-

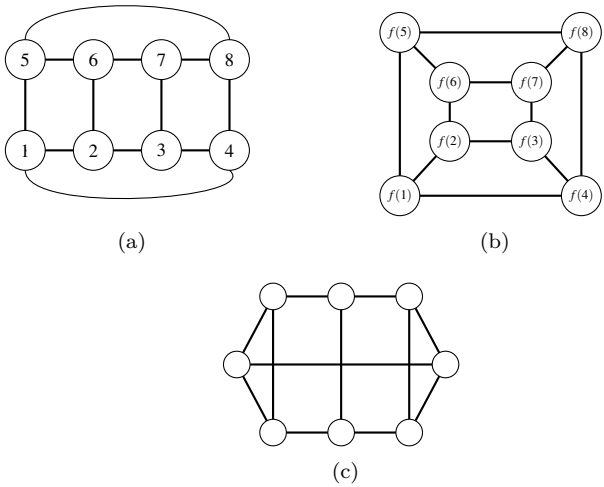
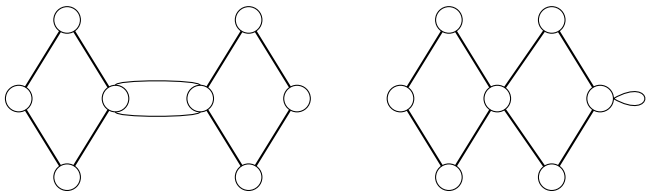


Figura 2.3: Exemplo de grafos (a) G_1 e (b) G_2, que são isomorfos; e (c) G_3, que é não isomorfo.

mente resolvido por força bruta, ou seja, por um algoritmo que enumere todas as possibilidades explicitamente, examinando-se cada uma das $n!$ permutações de V_1 (ou seja, cada função f possível). Esse algoritmo necessita de pelo menos $n!=O(n^n)$ passos, no pior caso. É desconhecido se existe ou não algum algoritmo eficiente para o problema geral de isomorfismo de grafos.

Algumas vezes é útil relaxar a definição de grafos, de modo a permitir uma aresta do tipo $e = (v,v)$, isto é, formada por um par de vértices idênticos. Uma aresta dessa natureza é chamada laço (figura 2.4(b)). Uma outra extensão possível consiste em substituir, na definição de grafo, o conjunto de arestas E por um multiconjunto. O efeito dessa alteração é, naturalmente, permitir a existência de mais de uma aresta entre o mesmo par de vértices, as quais são então denominadas arestas paralelas. A estrutura (V,E) assim definida é um multigrafo, tal como apresentado na figura 2.4.

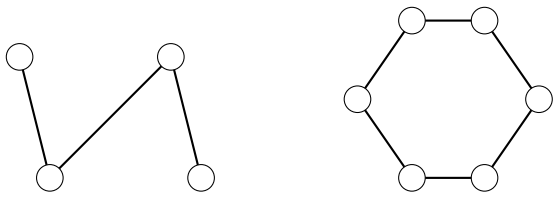
Em um grafo $G(V,E)$, define-se grau de um vértice $v \in V$, denotado por grau(v), como o número de vértices adjacentes a v. Um grafo é regular de grau r quando todos os seus vértices possuírem o mesmo grau r. Por exemplo, cada grafo da figura 2.3 é regular de grau 3. Observe que cada vértice v é incidente a grau(v) arestas e cada aresta sempre é incidente a dois vértices, logo, $\sum_{v \in V} \text{grau}(v) = 2|E|$. Um vértice que possui



(a) G_1 (b) G_2

Figura 2.4: Exemplo de um multigrafo G_1 e o grafo G_2 com laço.

grau zero é chamado isolado. Agora, mais formalmente, uma sequência de vértices $v1,\ldots,vk$ tal que $(v_j,v_{j+1}) \in E$, $1 \leq j < |k-1|$ é denominado caminho de $v_1,\ldots,v_k$. Diz-se então que v_1 alcança ou atinge v_k. Um caminho de k vértices é formado por $k-1$ arestas $(v_1,v_2),(v_2,v_3),\ldots,(v_{k-1},v_k)$. O valor $k-1$ é o comprimento do caminho. Se todos os vértices do caminho $v1,\ldots,vk$ forem distintos, a sequência recebe o nome de caminho simples ou elementar. Se as arestas forem distintas, a sequência denomina-se trajeto. Por exemplo, no grafo da figura 2.3(a), a sequência $1,2,3,7$ é um caminho simples, enquanto $2,3,7,6,2,1,5$ é um trajeto. Um ciclo é um caminho $v_1,\ldots,v_k,v_{k+1}$, sendo $v_1 = v_{k+1}$ e $k \geq 3$. Se o caminho $v_1,\ldots,v_k$ for simples, o ciclo $v_1,\ldots,v_k,v_{k+1}$ também é denominado simples ou elementar. Um grafo que não possui ciclos simples é acíclico. Um triângulo é um ciclo de comprimento 3. Dois ciclos são considerados idênticos se um deles puder ser obtido do outro, através de uma rotação de seus vértices. A figura 2.5 apresenta exemplos de grafos que são caminho (2.5.a) e ciclo (2.5.b).



(a) (b)

Figura 2.5: Exemplos de grafos que são (a) caminho e (b) ciclo.

Um caminho que contenha cada vértice do grafo exatamente uma vez é chamado hamiltoniano. Por outro lado, algum caminho ou ciclo que contenha cada aresta do grafo, também exatamente uma vez cada,

é denominado euleriano. Um ciclo $v_1, \ldots, v_k, v_{k+1}$ é hamiltoniano quando o caminho $v_1, \ldots, v_k$ o for. Se G é um grafo que possui ciclo hamiltoniano ou euleriano, então G é denominado hamiltoniano ou euleriano, respectivamente. Por exemplo, no grafo da figura 2.3(a), os ciclos 2, 3, 7, 6, 2 e 7, 6, 2, 3, 7 são idênticos, e o caminho 1, 5, 6, 2, 3, 7, 8, 4 é hamiltoniano. Para maior simplicidade de apresentação, quando não houver ambiguidade os termos "caminho"e "ciclo"serão utilizados com o significado de "caminho simples"e "ciclo simples", respectivamente.

Denomina-se distância $d(v,w)$ entre dois vértices v,w de um grafo o comprimento do menor caminho entre v e w. Assim, a distância entre os vértices 1 e 8, no grafo da figura 2.3(a), é igual a 2, isto é, $d(1,8) = 2$.

Um grafo $G(V,E)$ é denominado conexo quando existe caminho entre cada par de vértices de G, caso contrário G é desconexo. Observe que a representação geométrica de um grafo desconexo é necessariamente descontígua. Em especial, o grafo G será totalmente desconexo quando não possuir arestas. Seja S um conjunto e $S' \subseteq S$. Diz-se que S' é maximal em relação a uma certa propriedade P quando S' satisfaz a propriedade P e não existe subconjunto $S'' \to S'$ que também satisfaça P. Observe que a definição acima não implica necessariamente que S' seja o maior subconjunto de S satisfazendo P, ela significa apenas que S' não está propriamente contido em nenhum subconjunto de S que satisfaça P. De maneira análoga, define-se, também, conjunto minimal em relação a uma certa propriedade. A noção de conjunto maximal (ou minimal) é frequentemente encontrada em combinatória e pode ser aplicada, por exemplo, na seguinte definição: denominam-se componentes conexas de um grafo G os subgrafos maximais de G que sejam conexos. Observe que a propriedade P, nesse caso, é equivalente a "ser conexo". As componentes conexas de um grafo G são os subgrafos de G correspondentes às porções contíguas de sua representação geométrica. Os grafos da figura 2.5 são ambos conexos, ao contrário do grafo da figura 2.6, que possui cinco componentes conexas, sendo dois delas com um único vértice (isolados).

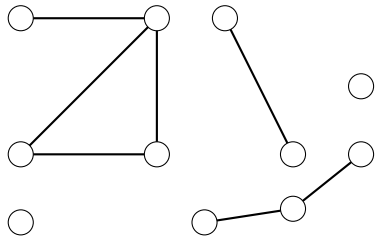
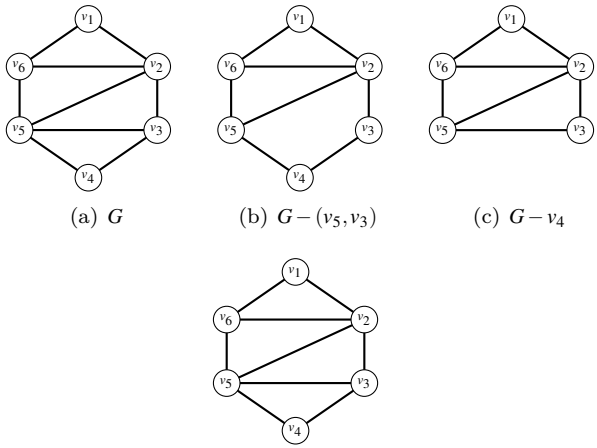


Figura 2.6: Grafo desconexo com cinco componentes conexas.

Seja $G(V,E)$ um grafo, $e \in E$ uma aresta. Denota-se por $G-e$ o grafo obtido de G pela exclusão da aresta e. Se v,w é um par de vértices não adjacentes em G, a notação $G+(v,w)$ representa o grafo obtido adicionando-se a G a aresta (v,w). Analogamente, seja $v \in V$ um vértice de G. O grafo $G-v$ denota aquele obtido de G pela remoção do vértice v. Observe que excluir um vértice implica remover de G o vértice em questão e as arestas nele incidentes. Da mesma forma, $G+w$ representa o grafo obtido adicionando-se a G o vértice w. Observa-se ainda que essas operações de inclusão e exclusão de vértices e arestas podem ser generalizadas. De um modo geral, se G é um grafo e S um conjunto de arestas ou vértices, $G-S$ e $G+S$ denotam, respectivamente, o grafo obtido de G pela exclusão e inclusão de S, tal como apresentado na figura 2.7.

Dado um grafo G, pode-se indagar quais são as condições para que G possua um ciclo euleriano. A resposta a essa questão culminou no primeiro resultado teórico de Teoria dos Grafos, cujo problema é conhecido por "Problema das 7 Pontes (de Königsberg)". Seu teorema é reproduzido a seguir.



(a) G (b) $G-(v_5,v_3)$ (c) $G-v_4$

(d) $G+(v_1,v_4)$

Figura 2.7: Operações de exclusão e inclusão de arestas e vértices em um grafo G.

Teorema 2.2.1. Um grafo G conexo possui ciclo euleriano se, e somente se, todo vértice de G possuir grau par.

Prova. Seja C um ciclo euleriano de G. Cada ocorrência de um dado vértice v em C contribui com duas unidades para computar do grau de v. Como cada aresta de G aparece exatamente uma vez em C, conclui-se que v possui grau par. Para a prova de suficiência, particionam-se inicialmente as arestas de G em um conjunto C de ciclos simples, do seguinte modo: como cada vértice de G possui grau 2, G contém necessariamente algum ciclo simples C_1. Se C_1 contém todas as arestas de G, o particionamento C está terminado. Caso contrário, removem-se de G as arestas do ciclo C_1 e os vértices isolados, porventura formados após essa operação. No novo grafo assim obtido, cada vértice possui ainda grau par. Determina-se então um novo ciclo simples, e assim por diante. Ao final, as arestas de G encontram-se particionadas em ciclos simples. Para determinar o ciclo euleriano, considera-se um ciclo, por exemplo C_1 do particionamento C. Se não há mais ciclos de C a serem considerados, a prova está concluída. Caso contrário, como G é conexo, existe um ciclo $C_2 \in P$ tal que C_1 e $C2$ possuam um vértice comum v . Então o ciclo formado pela união das arestas de C_1 e C_2 contém cada uma dessas arestas exatamente uma vez. Repete-se o processo, considerando um novo ciclo $C_3 \in C$, ainda não considerado, e assim por diante. □

Observe que a prova de suficiência do teorema 2.2.1 é construtiva. De fato, ela corresponde a um algoritmo de complexidade $O(n+m)$ para encontrar (se existir) um ciclo euleriano num grafo com n vértices e m arestas. Em contraste, a determinação de um ciclo hamiltoniano apresenta dificuldade muito maior. Na realidade, ainda hoje é desconhecido se existe ou não algum algoritmo eficiente para resolver esse problema, que é da classe de complexidade computacional NP-completo e é base para o problema clássico de Otimização Combinatória, Problema do Caixeiro Viajante, a ser abordado em capítulos posteriores.

Denomina-se complemento de um grafo $G(V,E)$ o grafo G que possui o mesmo conjunto de vértices de G e tal que, para todo par de vértices distintos $v,w \in V$, tem-se que (v,w) é aresta de G se, e somente se, não o for de G (ver figura 2.8).

Um grafo é completo quando existe uma aresta entre cada par de seus vértices. Utiliza-se a notação K_n para designar um grafo completo com n vértices (figura 2.9). O grafo K_n possui, pois, o número máximo possível de arestas para um dado n, ou seja, $\binom{n}{2}$ arestas. Um grafo $G(V,E)$ é bipartido quando o seu conjunto de vértices V puder ser particionado em dois subconjuntos V_1,V_2, tais que toda aresta de G una um vértice de V_1 a outro de V_2. É útil denotar-se o grafo bipartido G por $(V_1 \cup V_2, E)$. Um grafo bipartido completo possui uma aresta para cada par de vértices v_1, v_2, sendo $v_1 \in V_1$ e $v_2 \in V_2$. Sendo $n_1 = |V_1|$ e $n_2 = |V_2|$, um

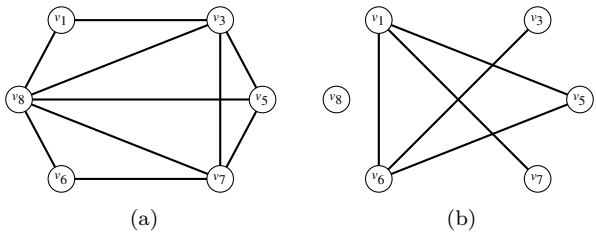


Figura 2.8: Exemplo de um grafo e seu complemento.

grafo bipartido completo é denotado por K_{n_1,n_2} e obviamente possui n_1n_2 arestas. A figura 2.10 apresenta dois grafos bipartidos, sendo que em 2.10(b) tem-se o grafo bipartido completo $K_{3,3}$.

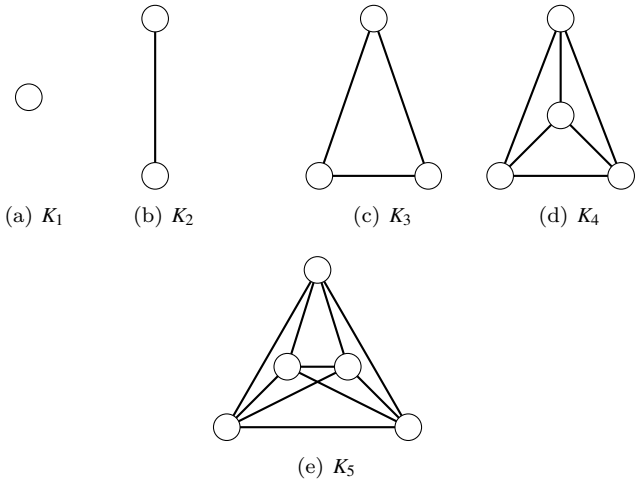
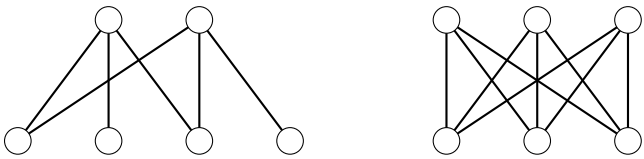


Figura 2.9: Grafos completos.

Teorema 2.2.2. Um grafo $G(V,E)$ é bipartido se, e somente se, todo ciclo de G possuir comprimento par.

Prova. Seja $v_1,\ldots,v_k,v_1$ um ciclo de comprimento k do grafo bipartido G e seja $v \in V_1$. Logo, $v_2 \in V_2$, $v_3 \in V_1$, $v_4 \in V_2$, e assim por diante. Como $(v_k,v_1) \in E$ implica $v_k \in V_2$. Portanto k é par, o que mostra a necessidade. A prova da suficiência consiste em exibir os subconjuntos V_1,V_2 que biparticionam o conjunto de vértices do grafo G, no qual todos os



(a) (b)

Figura 2.10: Exemplos de grafos bipartidos em (a) e (b), que neste caso é também completo ($K_{3,3}$).

ciclos possuem comprimento par. Selecione arbitrariamente um vértice $v_1 \in V$. Defina V_1 como contendo v_1 e todos os vértices que se encontram à distância par de v_1. Defina $V_2 = V - V_1$. Suponha que exista uma aresta (a,b) entre dois vértices $a,b \in V_1$. Então os caminhos mais curtos entre v_1 e a e entre v_1 e b unidos com a aresta (a,b) formam um ciclo de comprimento ímpar, uma contradição. As demais possibilidades são tratadas de forma análoga. □

Um subgrafo $G_2(V_2,E_2)$ de um grafo $G_1(V_1,E_1)$ é um grafo tal que $V_2 \subseteq V_1$ e $E_2 \subseteq E_1$. Se, além disso, G_2 possuir toda aresta (v,w) de G_1 tal que ambos v e w estejam em V_2, então G_2 é o subgrafo induzido pelo subconjunto de vértices V_2. Diz-se então que V_2 induz G_2. Ou seja, o subgrafo induzido G_2 de G_1 satisfaz: para $v,w \in V2$, se $(v,w) \in E_1$, então $(v,w) \in E_2$. Como exemplo, o grafo da figura 2.11(a) tem como subgrafos os grafos 2.11(b) e 2.11(c), mas somente o último é um subgrafo induzido. Similarmente, para um dado $E_2 \subseteq E_1$, se $V_2 \subseteq V_1$ for exatamente o subconjunto de vértices de V_1, incidentes a alguma aresta de E_2, então $G_2[V_2,E_2]$ é o subgrafo induzido pelo subconjunto de arestas $E_2 \subseteq E_1$.

Denomina-se clique de um grafo G um subgrafo de G que seja completo. E denomina-se conjunto independente de vértice um subgrafo induzido de G, que seja totalmente desconexo. De outro modo, em uma clique existe uma aresta entre cada par de vértices distintos e em um conjunto independente de vértices não há aresta entre qualquer par de vértices. O tamanho de uma clique ou de um conjunto independente é igual à cardinalidade de seu conjunto de vértices.

Dado um grafo G, o problema de encontrar em G uma clique ou conjunto independente de vértices com um dado tamanho k pode ser facilmente resolvido por um processo de força bruta, examinando o subgrafo induzido de cada um dos $\binom{n}{k}$ subconjuntos de V com k vértices. Esse método corresponde a um algoritmo de complexidade $O(n^k)$, e também é desconhecido se existe algum processo eficiente para resolver esse

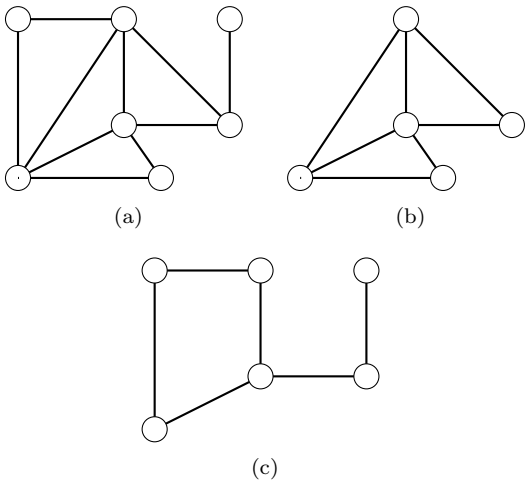


Figura 2.11: Exemplo de (a) um grafo G, (b) um subgrafo de G em (a) e (c) um subgrafo induzido de G.

problema. Ambos são da classe de problemas NP-completos.

2.3 Árvores

Um grafo que não possui ciclos é acíclico. Denomina-se árvore um grafo $T(V,E)$ que seja acíclico e conexo. Se um vértice v da árvore T possuir grau ≤ 1, então v é uma folha. Caso contrário, se grau$(v) > 1$, então v é um vértice interior (ou interno) da árvore. Um conjunto de árvores é denominado floresta. A figura 2.12 ilustra uma floresta com dois componentes conexas que são árvores. Os vértices v e w assinalados na árvore mais à esquerda (2.12(a)) são respectivamente uma folha e um vértice interior.

Observe que toda árvore T com n vértices possui exatamente $n-1$ arestas. O seguinte argumento indutivo justifica a afirmativa: se T possui 1 vértice, então obviamente seu número de arestas é zero. Suponha que T contenha $n-1$ vértices e $n-2$ arestas, $n > 1$. Considere a adição de uma nova aresta $e = (v,w)$. Como T é conexo, pelo menos um dentre v,w pertence a T. Mas como T é acíclico, pelo menos um deles não lhe pertence (caso contrário, a inclusão de uma nova aresta entre dois vértices de T produziria um ciclo). Logo, exatamente um dentre v,w pertence a T, o que significa que a árvore passa a possuir n vértices e

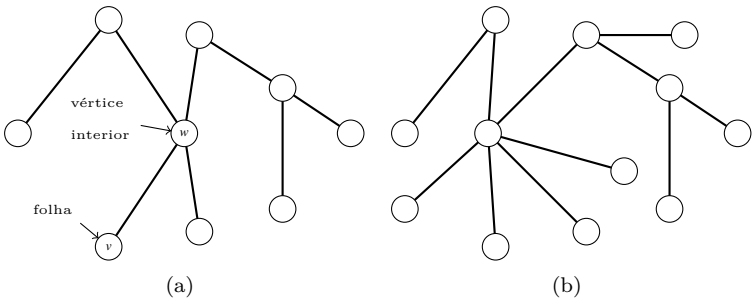


Figura 2.12: Exemplos de árvores, que são grafos conexos e acíclicos.

$n-1$ arestas. Através de um raciocínio análogo pode-se mostrar que o número de folhas de T varia entre um mínimo de 2 e máximo de $n-1$, para $n > 2$.

A figura 2.13 ilustra todas as possíveis árvores (não isomorfas entre si) com seis vértices. Observe que toda árvore T com n vértices possui exatamente $n-1$ arestas.

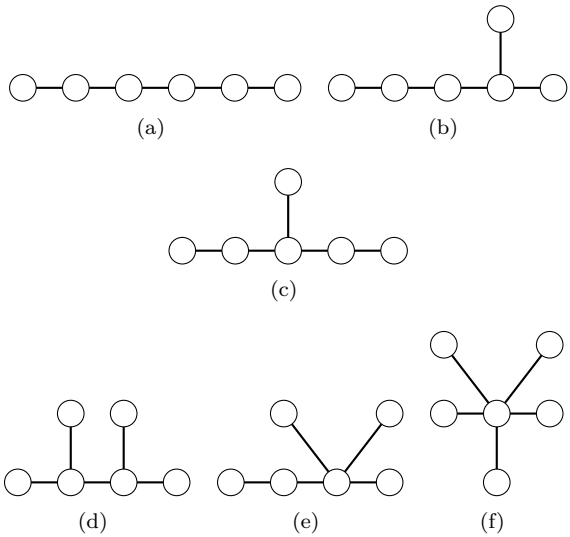


Figura 2.13: Todas as árvores com $n = 6$ vértices.

Árvores constituem uma classe extremamente importante de grafos,

principalmente devido a sua aplicação nas mais diversas áreas. O teorema 2.3.1 é uma caracterização para as árvores.

Teorema 2.3.1. Um grafo G é uma árvore se, e somente se, existir um único caminho entre cada par de vértices de G.

Prova. Seja G uma árvore. Então G é conexo e portanto existe pelo menos um caminho entre cada par v, w de vértices de G. Suponha que existam dois caminhos distintos vP_1w e vP_2w, então v e w. Então vP_1wP_2v P2v é um ciclo, o que contradiz G ser acíclico. Isso prova que, reciprocamente, se existe exatamente um caminho entre cada par de vértices de G, então o grafo é obviamente conexo e, além disso, não pode conter ciclos. Logo, G é uma árvore. □

Além do teorema acima, existem diversas outras possíveis caracterizações para as árvores, como a apresentada no teorema 2.3.2.

Teorema 2.3.2. Seja $G(V,E)$ um grafo. As possíveis afirmativas são equivalentes.

1. G é uma árvore.
2. G é conexo e $|E|$ é mínimo.
3. G é conexo e $|E| = |V| - 1$.
4. G é acíclico e $|E| = |V| - 1$.
5. G é acíclico e, para todo $v, w \in V$, a adição da aresta (v, w) produz um grafo contendo exatamente um ciclo.

A prova do teorema acima não apresenta dificuldade e será omitida.

Seja $G(V,E)$ um grafo. Denomina-se excentricidade de um vértice $v \in V$ o valor da distância máxima entre v e w, para todo $w \in V$. O centro de G é o subconjunto dos vértices de excentricidade mínima. O centro de um grafo pode possuir no mínimo um e no máximo n vértices. O centro de uma árvore possui não mais do que dois vértices. O lema 2.3.1 é utilizado na justificativa desse fato.

Lema 2.3.1. Seja T uma árvore com pelo menos três vértices. Seja T' a árvore obtida de T pela exclusão de todas as suas folhas. Então T e T' possuem o mesmo centro.

Prova. Observe inicialmente que, se um vértice f de T é uma folha, então f não pertence ao centro de T'. Isso porque o vértice g adjacente a f possui necessariamente como excentricidade uma unidade menor do que

a de f. Seja agora um vértice interior v de T. O vértice w, cuja distância a v é máxima, é necessariamente uma folha. Logo, a exclusão de todas as folhas de T faz decrescer de uma unidade a excentricidade de cada um de seus vértices interiores. Isso completa a prova. □

Tem-se então, o teorema 2.3.3 a seguir.

Teorema 2.3.3. O centro de uma árvore T possui um ou dois vértices.

Prova. Se T possui até dois vértices, o teorema é trivial. Caso contrário, aplicar repetidamente o lema 2.3.1. □

Denomina-se subgrafo gerador (ou subgrafo de espalhamento) de um grafo $G_1(V_1,E_1)$ um subgrafo $G_2(V_2,E_2)$ de G_1 tal que $V1 = V2$. Quando o subgrafo gerador é uma árvore, recebe o nome de árvore geradora (ou árvore de espalhamento). O grafo da figura 2.14 possui como exemplos de seus subgrafos geradores os grafos 2.14(b) e 2.14(c), sendo este último uma árvore geradora dos outros dois.

Todo grafo conexo G possui árvore geradora. O seguinte processo construtivo de uma árvore geradora T de G justifica essa afirmativa: considere uma aresta e de G. Remover e de G se $G-e$ for conexo. Quando todas as arestas que permaneceram já tiverem sido consideradas, o grafo resultante é uma árvore geradora de G . Observe que, se G não for conexo, pode-se aplicar essa mesma ideia a cada componente conexo de G, obtendo-se então uma floresta geradora de G, ou seja, uma árvore geradora para cada componente conexo do grafo.

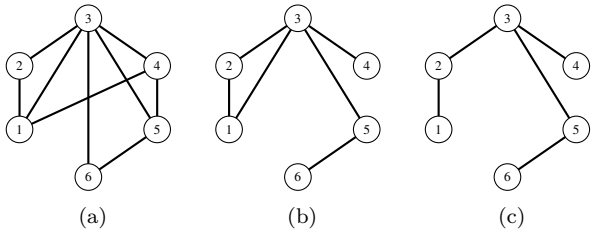


Figura 2.14: Os grafos (b) e (c) são subgrafos geradores de (a), e o grafo em (c) é uma árvore geradora dos outros dois grafos.

Seja $G(V,E)$ um grafo conexo e $T(V,E)$ uma árvore geradora de G. Uma aresta $e \in E - E_T$ é denominada elo de G em relação a T. Sendo $|V| = n$, $|E| = m$, tem-se: $|E_T| = n-1$ e, portanto, o número total de elos distintos de G é igual a $m-n+1$. O valor $m-n+1$ é denominado posto do grafo G . Observe que a adição do elo e à árvore T produz

sempre um único ciclo simples, isto é, o grafo $G+e$ possui exatamente um ciclo simples, o qual contém a aresta e. Esse ciclo é denominado ciclo fundamental de G, em relação a T. Chama-se conjunto fundamental de ciclos o conjunto de todos os ciclos fundamentais de G, em relação à árvore T. Assim sendo, um conjunto fundamental de ciclos é formado por $m-n+1$ ciclos simples, correspondentes respectivamente aos $m-n+1$ elos de G em relação a T. Cada ciclo do conjunto fundamental possui exatamente um elo. Para um certo grafo G um conjunto fundamental de ciclos depende da árvore geradora considerada. Contudo, a cardinalidade desse conjunto é invariante e igual ao posto do grafo. Por exemplo, o grafo da figura 2.14(a) possui posto igual a 4. Seus elos em relação à árvore geradora da figura 2.14(c) são respectivamente as arestas $(1,3)$, (1.4), $(3,6)$ e $(4,5)$. O conjunto fundamental de ciclos do grafo considerado em relação a essa árvore geradora é constituído pelos quatro ciclos $\{1,2,3,1;1,2,3,4,1;3,5,6,3;3,4,5,3\}$.

Uma árvore $T(V,E)$ é denominada enraizada quando algum vértice $v \in V$ é escolhido como especial. Esse vértice é então chamado de raiz da árvore. Uma árvore não enraizada é também denominada árvore livre. Por exemplo, a árvore livre da figura 2.15(a) torna-se a árvore enraizada da árvore 2.15(a)(b), quando o vértice c é escolhido como raiz. Sejam v,w dois vértices de uma árvore enraizada T de raiz r. Suponha que v pertença ao caminho de r a w em T. Então v é ancestral de w, sendo w descendente de v. Se ainda $v \neq w$, v é ancestral próprio de w, e este é descendente próprio de v. Além disso, se (v,w) é aresta de T, então v é pai de w, sendo w filho de v. Dois vértices que possuem o mesmo pai são irmãos. A raiz de uma árvore naturalmente não possui pai, enquanto todo vértice $v \neq r$ possui um único. Uma folha é um vértice que não possui filhos. Denomina-se nível de um vértice v, denotado por nível(v), o número de vértices do caminho da raiz r a v. Então nível$(r) = 1$ e, se dois vértices v,w são irmãos, nível$(v) =$ nível(w). A altura da árvore T é igual ao valor máximo de nível(v), para todo vértice v de T. Por exemplo, o conjunto dos ancestrais do vértice j da árvore enraizada da figura 2.15(b) é $\{j,e,c\}$. O conjunto dos ancestrais próprios de j é $\{c,e\}$. O dos descendentes de j é $\{j,i,k\}$, e $\{i,k\}$ é o conjunto dos descendentes próprios desse vértice. O pai de j é o vértice e, enquanto seus filhos são i,k. O nível de j é 2 e a altura da árvore é 3.

Seja $T(V,E)$ uma árvore enraizada e $v \in V$. Uma subárvore T_v de T é a árvore enraizada cuja raiz é v, definida pelo subgrafo induzido em T pelos descendentes de v. Isto é, u é pai de w em T_v se, e somente se, o for em T, para todo u,w descendentes de v em T. Seja S um subconjunto de vértices de T_v tal que $T_v - S$ seja conexo e $v \notin S$. O grafo $T_v - S$ é

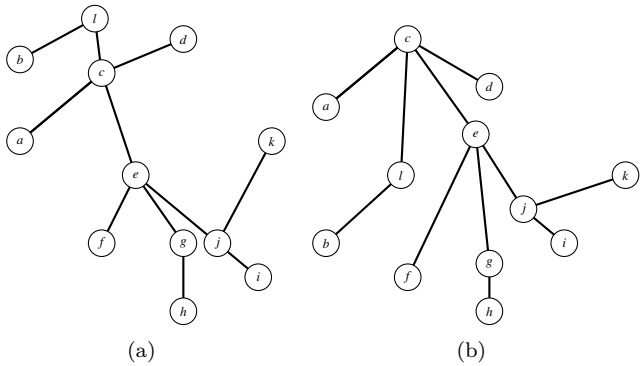


Figura 2.15: Exemplos de (a) árvore livre e (b) árvore enraizada.

então denominado subárvore parcial de raiz v. Obviamente a subárvore de raiz v é única para cada $v \in V$, enquanto a subárvore parcial não o é. A árvore da figura 2.16(a) é a subárvore de raiz e da árvore enraizada da figura 2.15(b), enquanto a da figura 2.16(b) é uma subárvore parcial da raiz e daquela mesma árvore.

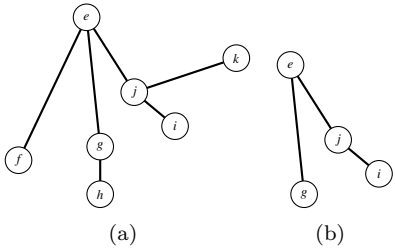


Figura 2.16: Exemplos de (a) subárvore e (b) subárvore parcial (relativas à árvore enraizada da figura 2.15(b)).

Observe que na definição da árvore enraizada é irrelevante a ordem em que os filhos de cada vértice v são considerados. Caso essa ordenação seja relevante, a árvore é denominada enraizada ordenada. Assim sendo, em uma árvore enraizada ordenada, para cada vértice v que possui filhos, pode-se identificar o primeiro filho de v (o mais à esquerda), o segundo (segundo mais à esquerda) e assim por diante. Obviamente, se duas árvores enraizadas são isomorfas, não necessariamente elas o serão como árvores enraizadas ordenadas. Para tanto, o isomorfismo deve preservar também a ordenação. Observe, por outro lado, que árvores isomorfas li-

vres também não serão necessariamente isomorfas se consideradas como enraizadas. Há diversas aplicações em árvores que requerem a mencionada ordenação. As árvores das figuras 2.17(a) e (b) são isomorfas entre si, se consideradas como enraizadas. Mas não o são como enraizadas ordenadas.

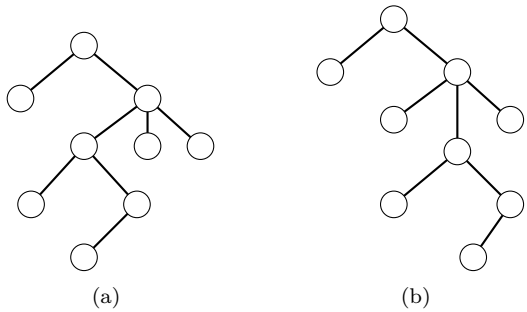


Figura 2.17: Árvores enraizadas isomorfas.

Uma árvore estritamente m-ária T é uma árvore enraizada ordenada em que cada vértice não folha possui exatamente m filhos, $m \geq 1$. Quando $m = 2$, a árvore é estritamente binária. A cada filho w de todo vértice não folha v de T atribua agora um rótulo inteiro positivo $r(w)$, $1 \leq r(w) \leq m$, igual à posição que w ocupa na ordenação dos filhos de v. Uma subárvore parcial de uma árvore estritamente m-ária que possui essa rotulação é chamada árvore m-ária. Ou seja, numa árvore m-ária T dois vértices irmãos w_j, w_{j+1} consecutivos na ordenação dos filhos de um vértice v podem ter rótulos não consecutivos (não necessariamente $r(w_{j+1}) - r(w_j) = 1$). Pode-se então considerar que uma árvore m-ária é obtida a partir de uma estritamente m-ária pela remoção de $r(w_{j+1}) - r(wj) - 1$ subárvores entre cada par de irmãos w_j, w_{j+1}, além de $r(w_1) - 1$ subárvores antes do filho w_1 de v e $r(w_f) - 1$ após o último filho w_f de v.

2.4 Conectividade

Seja $G(V,E)$ um grafo conexo. Um corte de vértices de G é um subconjunto minimal de vértices $V' \subseteq V$ cuja remoção de G o desconecta ou o transforma no grafo trivial. Assim sendo, o grafo $G - V'$ é desconexo ou trivial e, para todo subconjunto próprio $V'' \subset V', G - V''$ é conexo e não trivial. Analogamente, um corte de arestas de G é um subconjunto minimal de arestas $E \subseteq E$, cuja remoção de G o desconecta. Isto é, $G - E'$ é

desconexo e, para todo subconjunto próprio $E'' \subset E', G - V''$ é conexo. Se G é um grafo completo K_n, $n > 1$, então não existe subconjunto próprio de vértices $V \subset V$ cuja remoção desconecte G , mas, removendo-se $n-1$ vértices, o resultado é o grafo trivial. Denomina-se conectividade de vértices c_V de G a cardinalidade do menor corte de vértices de G. Analogamente, a conectividade de arestas c_E de G é igual à cardinalidade do menor corte de arestas de G. Ou seja, c_V é igual ao menor número de vértices cuja remoção desconecta G ou o transforma no grafo trivial. Se G é um grafo desconexo, então $c_V = c_E = 0$. Considere agora o grafo da figura 2.18(a). O subconjunto 3,4 é um corte de vértices, pois sua remoção desconecta o grafo nas componentes conexas ilustradas na figura 2.18(b). Nesse mesmo grafo, o subconjunto de arestas $\{(1,3),(2,3),(4,5)\}$ é um corte de arestas, porque, removendo-o de G, produz-se o grafo desconexo da figura 2.18(c). Observe, também, que, removendo de G o subconjunto de vértices 3,4,7, desconecta-se G. Contudo, $\{3,4,7\}$ não é um corte de vértices, pois contém propriamente o corte $\{3,4\}$. Por outro lado, os cortes de vértices e arestas, de cardinalidade mínima de G, são respectivamente os subconjuntos $\{5\}$ e $\{(3,5),(4,5)\}$. Portanto, as conectividades de vértices e arestas desse grafo são respectivamente iguais a 1 e 2.

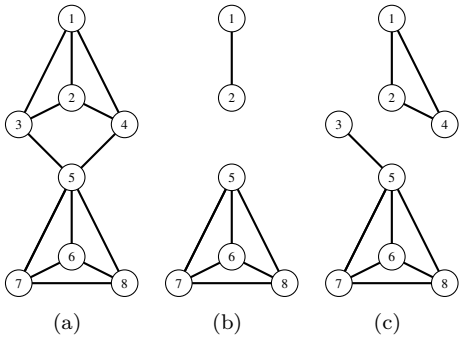


Figura 2.18: Exemplo de (a) um grafo conexo, em que (b) e (c) são exemplos de seus cortes.

Sendo k um inteiro positivo, diz-se que um grafo G é k-conexo em vértices quando a sua conectividade de vértices é $\geq k$. Analogamente, G é k-conexo em arestas quando sua conectividade de arestas é $\geq k$. Em outras palavras, se G é k-conexo em vértices (arestas), então não existe corte de vértices (arestas) de tamanho $< k$. O grafo da figura 2.18(a), por exemplo, é 1-conexo em vértices e 2-conexo em arestas.

Sejam $G(V,E)$ um grafo e $E \subseteq E$ um corte de arestas de G. Então é

sempre possível encontrar um corte de vértices V' de tamanho $|V'| \leq |E'|$. Basta escolher para V' precisamente o subconjunto de vértices v tais que toda aresta $e \in E'$ é incidente a algum $v \in V'$. Consequentemente, para todo grafo vale $c_V \leq c_E$. Seja w o vértice de grau mínimo no grafo G, suposto não completo. Então é possível desconectar G, removendo-se do grafo as $|\text{grau}(w)|$ arestas adjacentes a w. Então, $\text{grau}(w) \geq c_E \geq c_V$. Observe que, se G for o grafo completo K_n, então $\text{grau}(w) = c_E = c_V = n-1$.

O estudo de conectividade constitui um tópico básico em grafos. Os grafos 1-conexos foram denominados conexos na seção 2.2. Grafos 2-conexos (também denominados biconexos) apresentam propriedades interessantes. Seja $G(V,E)$ um grafo. Um vértice v é denominado articulação quando sua remoção de G o desconecta. Ou seja, $G-v$ é desconexo. Uma aresta $e \in E$ é chamada ponte quando sua remoção de G o desconecta. Nesse caso, $G-e$ é desconexo. Assim sendo, um grafo é biconexo em vértices (arestas) se, e somente se, não possuir articulações (pontes). Denominam-se componentes biconexos do grafo G os subgrafos maximais de G que sejam biconexos em vértices, ou isomorfos a K_2. Cada componente biconexo é também chamado *bloco* do grafo. Assim, se G é biconexo em vértices, então G possui um único bloco, que coincide com o próprio G.

Lema 2.4.1. Seja G um grafo. Então:

1. Cada aresta de G pertence a exatamente um bloco do grafo.

2. Um vértice v de G é articulação se, e somente se, v pertencer a mais de um bloco do grafo.

A prova é simples e será omitida. As articulações e pontes de um grafo podem ser caracterizadas pelo lema 2.4.2 a seguir.

Lema 2.4.2. Seja $G(V,E)$ um grafo conexo, $|V| > 2$. Então:

i Um vértice $v \in V$ é articulação de G se, e somente se, existirem vértices $w,u \neq v$ tais que v está contido em todo caminho entre w e u em G.

ii Uma aresta $(p,q) \in E$ é ponte se, e somente se, p,q for o único caminho simples entre p e q em G .

Prova. i Se v é articulação de G, então $G-v$ é desconexo. Sejam w,u dois vértices localizados em componentes conexas distintas de $G-v$. Então todo caminho entre w e u contém v.

ii $(p,q) \in E$ é uma ponte, então o grafo $G-(p,q)$ é desconexo, localizando-se p e q em componentes conexas distintas de $G-(p,q)$. Logo, (p,q) é o único caminho simples entre p e q em G.

□

O lema 2.4.3 a seguir caracteriza um grafo biconexo e pode ser generalizado como proposto no teorema 2.4.1.

Lema 2.4.3. Um grafo $G(V,E)$,$|V| > 2$, é biconexo se, e somente se, cada par de vértices de G está contido em algum ciclo.

Prova. Segue a parte 1 do lema 2.4.2. □

Teorema 2.4.1. Seja G um grafo k-conexo. Então existe algum ciclo de G passando por cada subconjunto de k vértices.

Observe que a recíproca do teorema 2.4.1 anterior não é válida para $k > 2$. O grafo definido por exatamente um ciclo de comprimento $|V|$ explica a razão. Deste modo, o teorema 2.4.2 a seguir caracteriza grafos k-conexos.

Teorema 2.4.2. Um grafo $G(V,E)$ é k-conexo, se, e somente se, existissem k caminhos disjuntos (exceto nos extremos) entre cada par de vértices de G.

As provas dos teoremas 2.4.1 e 2.4.2 são mais elaboradas e não serão aqui apresentadas.

2.5 Planaridade

Nesta seção, retorna-se à noção de representação geométrica dos grafos. Seja G um grafo e R uma representação geométrica de G em um plano. A representação R é chamada plana quando não houver cruzamento de linhas (arestas) em R, a não ser nos vértices, naturalmente. Um grafo é dito planar quando admitir alguma representação plana. Esse conceito pode ser estendido para outras superfícies. Assim, de um modo geral, diz-se que o grafo G é imersível em uma superfície S se existir uma representação geométrica R de G, desenhada sobre S, tal que duas linhas de R não se cruzem, a não ser nos vértices.

Seja G um grafo planar e R uma representação plana de G em um plano P. Então, as linhas de R dividem P em regiões, as quais são denominadas faces de R. Observe que existe exatamente uma região não limitada. Esta é chamada face externa. A representação da figura 2.19, por exemplo, possui cinco faces. Segue-se que duas representações planas

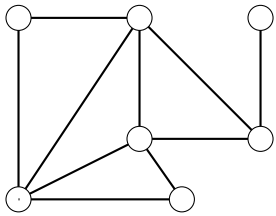


Figura 2.19: Exemplo de um grafo planar com sete vértices e dez arestas.

de um grafo possuem sempre o mesmo número de faces. Logo, esse número constitui também um invariante de um grafo, sendo denotado por f. Os números de vértices, arestas e faces de um grafo planar não são independentes. O teorema 2.5.1 a seguir os correlaciona.

Teorema 2.5.1. Seja G um grafo planar. Então $n+f=m+2$.

Uma conveniente representação plana de G corresponde a um poliedro com n vértices, m arestas e f faces. A expressão acima é conhecida como fórmula de Euler para poliedros e pode ser demonstrada por indução. Observe que quanto maior for o número de arestas de um grafo G em relação ao seu número de vértices, mais difícil se torna a obtenção de uma representação geométrica plana para G. De fato, há um limite máximo para o número de arestas de um grafo planar, dado pelo lema 2.5.1 a seguir.

Lema 2.5.1. Seja G um grafo planar. Então $m \leq 3n-6$.

Prova. Cada face é delimitada por um mínimo de três arestas e cada aresta pertence a exatamente duas faces. Logo, $2m \geq 3f$. Substituindo na fórmula de Euler, tem-se $f=m-n+2$, onde $(2/3)m \geq m-n+2$. Assim, $m \leq 3n-6$. □

Uma outra questão relacionada é: dado um grafo G planar, é admitida uma representação plana em que todas as linhas sejam retas? Todo grafo planar admite uma representação plana em que todas as linhas sejam retas. A prova disso é elaborada e não será apresentada neste texto.

Conforme foi mencionado, toda representação plana de um grafo G contém uma face externa, não limitada. Para evitar a distinção entre faces limitadas e não limitadas, pode-se considerar a representação geométrica de G em uma esfera, ao invés de em um plano. Para tal, seja uma esfera S apoiada em um plano P, sendo b o ponto de contato entre essas superfícies. Seja a o ponto de S diametralmente oposto a b.

Denote por R uma representação plana do grafo G no plano P. Então, cada ponto v de R em P possui um correspondente v' na esfera S, sendo v' definido pela interseção de S com a reta av. É imediato verificar que essa construção define uma representação geométrica de G em S, sem cruzamento de linhas (a não ser nos vértices). E reciprocamente, ou seja, um grafo G é planar se, e somente se, G for imersível em uma esfera. As representações geométricas na esfera apresentam a vantagem de não possuírem regiões (faces) ilimitadas.

Uma questão básica é naturalmente caracterizar os grafos planares. Inicialmente tem-se:

Lema 2.5.2. Os grafos K_5 e $K_{3,3}$ são não planares.

Prova. Para K_5 tem-se $n = 5$ e $m = 10$. Assim, $m > 3n - 6$. Portanto, K_5 não satisfaz à condição necessária do lema 2.5.1, logo não pode ser planar. Para $K_{3,3}$ tem-se $n = 6$ e $m = 9$. Assim, suponha que $K_{3,3}$ seja planar. Então, deve satisfazer à Fórmula de Euler, ou seja, seu número de faces é $f = m - n + 2 = 5$. Em qualquer representação plana de $K_{3,3}$, cada face deve ter pelo menos quatro arestas (pois o menor ciclo em $K_{3,3}$ tem comprimento 4). Além disso, cada aresta pertence a exatamente duas faces. Logo, $2m \geq 4f$, o que contradiz $m = 9$, $f = 5$. □

Observe que K_5 e $K_{3,3}$ são os grafos não planares com menor número de vértices e arestas, respectivamente. Denomina-se subdivisão de uma aresta (v, w) de um grafo G uma operação que transforma (v, w) em um caminho $v, z_1, ..., z_k, w$, sendo $k \geq O$, em que os z_i são vértices de grau 2 adicionados a G. Diz-se que um grafo G_2 é uma subdivisão de um grafo G_1 quando G_2 puder ser obtido de G_1 através de uma sequência de subdivisões de arestas de G_1.

O teorema 2.5.2 seguinte é o fundamental em planaridade, resolvendo o problema de caracterização dos grafos planares.

Teorema 2.5.2. Um grafo é planar se, e somente se, não contém como subgrafo uma subdivisão de K_5 ou $K_{3,3}$.

A prova da necessidade é imediata. Os grafos K_5 e $K_{3,3}$ não são planares (lema 2.5.2). Consequentemente qualquer subdivisão dos mesmos também não será. Contudo, a prova de suficiência é bastante envolvente e não será apresentada. O teorema 2.5.2 concede aos grafos K_5 e $K_{3,3}$ um papel todo especial em planaridade. A partir deste teorema 2.5.2, construíram-se os primeiros algoritmos para o reconhecimento de grafos planares. Atualmente são conhecidos algoritmos de complexidade $O(n)$ para essa finalidade.

2.6 Coloração

Seja $G(V,E)$ um grafo e $C = \{c_i\}$ um conjunto de cores. Uma coloração de G é uma atribuição de alguma cor de C para cada vértice de V, de tal modo que a dois vértices adjacentes sejam atribuídas cores diferentes. Assim sendo, uma coloração de G é uma função $f : V \to C$ tal que para cada par de vértices $v, w \in V$ tem-se $(v,w) \in E$ $f(v) \neq f(w)$. Uma k-coloração de G é uma coloração que utiliza um total de k cores. Isto é, uma k-coloração de G é uma coloração f cuja cardinalidade do conjunto imagem é igual a k. Diz-se, então, que G é k-colorível. Denomina-se número cromático $\chi(G)$ de um grafo G o menor número de cores k, para o qual existe uma k-coloração de G. Uma coloração que utiliza o número mínimo de cores é chamada mínima.

Observe que é muito fácil colorir um grafo (isto é, obter uma coloração). Basta utilizar um total de n cores, uma para cada vértice. Isso obviamente garante cores diferentes a vértices adjacentes. Contudo, o problema de encontrar um processo eficiente para encontrar uma coloração mínima é bastante difícil. A exemplo do caso da seção anterior, não é conhecida caracterização razoável para grafos k-coloríveis. Não é conhecido, pois, algoritmo eficiente para determinar o número cromático de um grafo. De fato, há fortes indícios de que não existe algoritmo dessa natureza. Contudo, o caso especial de bicoloração pode ser resolvido de forma simples.

Lema 2.6.1. Um grafo $G(V,E)$ é bicolorível se, e somente se, for bipartido.

Prova. Se G é bipartido, sejam $V_1, V_2 \subseteq V$ os subconjuntos de V que o biparticionam. Então atribua a cor c_1 aos vértices de V_1 e a cor c_2 aos de V_2. Reciprocamente, se G é bicolorível, considere uma bicoloração de G, com cores c_1 e c_2. Sejam V_1 e V_2 os subconjuntos de vértices que possuem as cores c_1 e c_2, respectivamente. Então V_1, V_2 biparticionam G. □

O lema 2.6.1 acima caracteriza grafos bicoloríveis e sugere um algoritmo eficiente para encontrar uma bicoloração, se existir. Entretanto, se o número cromático for maior do que 2, o processo se torna substancialmente mais difícil.

Os conceitos de coloração, clique e conjunto independente de vértices estão naturalmente relacionados. Por exemplo, como são necessárias p cores para colorir os p vértices de uma clique de tamanho k, conclui-se que o número cromático de um grafo G é maior ou igual do que o tamanho da maior clique de G. Considere agora uma k-coloração de $G(V,E)$. Sejam $V_1, V_2, ..., V_k$ os subconjuntos (disjuntos) de V, induzidos pela k-coloração. Então, obviamente, $\bigcup V_i = V$ e cada V_i é um conjunto

independente de vértices. Então o problema de determinar uma coloração mínima de G pode ser formulado em termos de particionar V em um número mínimo de conjuntos independentes de vértices.

Um grafo G é denominado k-crítico quando $\chi(G) = k$ e, para todo subgrafo próprio H de G, $\chi(H) < k$. Naturalmente todo grafo G admite um subgrafo $\chi(G)$-crítico. Além disso, todo grafo k-crítico é conexo. O teorema seguinte traz alguma informação sobre essa classe de grafos.

Teorema 2.6.1. Se $G(V,E)$ é k-crítico, então grau$(v) \geq k-1$, para todo $v \in V$.

Prova. Suponha que G seja k-crítico e grau$(v) \geq k-1$, para algum $v \in V$. Como G é k-crítico, $G-v$ é $(k-1)$-colorível. Seja $V_1, ..., V_{k-1}$ uma $(k-1)$-coloração de $G-v$. Isto é, V_i representa o conjunto de vértices de G com a cor i, $1 \leq i \leq k-1$. Como grau$(v) < k-1$, existe V_j tal que todo $w \in V_j$ é não adjacente a v. Então, $V_1, ..., V_j \bigcup \{v\}, ..., V_{k-1}$ é uma $(k-l)$-coloração de G, uma contradição. □

O estudo de coloração foi iniciado com o problema das quatro cores. Seja M uma região do plano. Um mapa é um particionamento de M em um número finito de sub-regiões, as quais são delimitadas por linhas. Duas sub-regiões são adjacentes quando possuírem uma linha em comum. Uma coloração de M é uma atribuição de alguma cor a cada região de M, de modo que regiões adjacentes possuam cores diferentes. Um mapa M é k-colorível quando existir uma coloração de M que utilize k cores. Observe que existe uma correspondência entre coloração de grafos G e de mapas M. De fato, defina G de modo a possuir um vértice para cada região de M, sendo um par de vértices adjacente quando as respectivas regiões o forem. Então é imediato observar que colorir o mapa M é equivalente a colorir o grafo G. Por outro lado, observe que G é necessariamente planar, pela própria definição de M. Então o problema de coloração de mapas é equivalente ao de coloração de grafos planares. O problema das quatro cores consiste, então, em provar que todo grafo planar é 4-colorível.

2.7 Grafos direcionados

Os grafos abordados até então são também denominados não direcionados (seção 2.2). Isso porque suas arestas (v,w) não são direcionadas. Assim, se (v,w) é aresta de $G(V,E)$, então tanto v é adjacente a w, como w é adjacente a v. Em contrapartida, um grafo direcionado ou digrafo $D(V,E)$ é um conjunto finito não vazio V (os vértices) e um conjunto

E (as arestas) de pares ordenados de vértices distintos. Portanto, em um digrafo cada aresta (v,w) possui uma única direção de v para w, ou seja, tem-se uma aresta direcionada ou arco. Diz-se também que (v,w) é divergente de v e convergente a w. Grande parte da nomenclatura e dos conceitos é análoga nos dois casos. Assim sendo, definem-se, por exemplo, caminho simples, trajeto, ciclo e ciclo simples de forma análoga às definições de grafos. Em todo caso, um caminho em digrafos é também chamado de cadeia, tal como um ciclo direcionado é chamado de circuito.

Ao contrário dos grafos simples, os digrafos podem possuir ciclos de comprimento 2, no caso em que ambos (v,w) e (w,v) são arestas do digrafo. Os caminhos e ciclos em um digrafo devem obedecer ao direcionamento das arestas.

Seja $D(V,E)$ um digrafo e um vértice $v \in V$. O grau de entrada de v é o número de arestas convergentes a v. O grau de saída de v é o número de arestas divergentes de v. Uma fonte é um vértice com grau de entrada nulo, enquanto um sumidouro (ou poço) é um com grau de saída nulo.

Se forem retiradas as direções das arestas de um digrafo D, obtém-se um multigrafo não direcionado, chamado grafo subjacente a D.

Um digrafo D(V,E) é fortemente conexo quando para todo par de vértices $v,w \in V$ existir um caminho em D de v para w e também de w para v. Se ao menos um desses caminhos existir para todo $v,w \in V$, então D é unilateralmente conexo. D é chamado fracamente conexo ou desconexo, conforme seu grafo subjacente seja conexo ou desconexo, respectivamente. Observe que se um digrafo é fortemente conexo, então, também é unilateralmente e fracamente conexo. Se existir algum caminho em D de um vértice v para um vértice w, então diz-se que v alcança ou atinge w, sendo este alcançável ou atingível de v. Se v alcançar todos os vértices de D, então v é chamado raiz do digrafo. Um digrafo é acíclico quando não possui ciclos. Observe que o grafo subjacente a um digrafo acíclico não é necessariamente acíclico, entretanto, um digrafo de um grafo subjacente acíclico também o será.

Lema 2.7.1. Todo digrafo acíclico possui (pelo menos) uma fonte e um sumidouro.

Seja $D(V,E)$ um digrafo acíclico. Denomina-se fechamento transitivo de D o maior digrafo $D_f(V,E_f)$ que preserva a alcançabilidade de D. Isto é, para todo $v,w \in V$, se v alcança w em D, então $(v,w) \in E_f$. Analogamente, a redução transitiva de D é o menor digrafo $D_r(V,E_r)$ que preserva a alcançabilidade de D. Isto é, se $(v,w) \in E_r$, então v não alcança w em $D-(v,w)$. A redução transitiva é também conhecida como diagrama de Hasse.

Um conjunto parcialmente ordenado ou ordenação parcial $(S, \leq)$ é um conjunto não vazio S e uma relação binária $\leq$ em S, satisfazendo às seguintes propriedades:

(i) $s_1 \leq s_1$ para $s_1 \in S$ ($\leq$ é reflexiva).

(ii) $s_1 \leq s_2$ e $s_2 \leq s_1 \Rightarrow s_1 = s_2$, para $s_1, s_2 \in S$ ($\leq$ é antissimétrica).

(iii) $s_1 \leq s_2$ e $s_2 \leq s_3 \Rightarrow s_1 \leq s_3$, para $s_1, s_2, s_3 \in S$ ($\leq$ é transitiva).

Um conjunto parcialmente ordenado $(S, \leq)$ pode ser também caracterizado pelo par $(S, <)$, em que $<$ é uma relação binária em S definida por $s_1 < s_2 \Leftrightarrow s_1 \leq s_2$ e $s_1 \neq s_2$. A relação $<$ satisfaz:

(i) $s_1 \not< s_1$ para $s_1 \in S$ ($<$ é irreflexiva).

(ii) $s_1 < s_2 \Rightarrow s_2 \not< s_1$, para $s_1, s_2 \in S$ ($<$ é assimétrica).

(iii) $s_1 < s_2$ e $s_2 < s_3 \Rightarrow s_1 < s_3$, para $s_1, s_2, s_3 \in S$ ($<$ é transitiva).

Sej $D_f(V, E_f)$ o fechamento transitivo de um digrafo acíclico $D(V, E)$. Pode-se constatar que E_f é uma relação irreflexiva, assimétrica e transitiva. Consequentemente, fechamentos transitivos de digrafos acíclicos e conjuntos parcialmente ordenados são conceitos equivalentes. Isto é, $P(S, <)$ caracteriza um conjunto parcialmente ordenado se, e somente se, o digrafo $D_f(S, <)$ for o fechamento transitivo de algum digrafo acíclico D. Diz-se então que D induz o conjunto parcialmente ordenado $P(S, <)$. Observe que $s_1 < s_2$ em P se, e somente se, existir um caminho em D de s_1 para s_2. Utiliza-se então a notação $v < w$ para indicar que v alcança w no digrafo acíclico $D(V, E)$, sendo $v, w \in V$.

Finalmente, uma árvore direcionada enraizada é um digrafo D tal que exatamente um vértice r possui grau de entrada nulo, enquanto para todos os demais vértices o grau de entrada é igual a um. É imediato observar que D é acíclico com exatamente uma raiz r. Além disso, o grafo subjacente T de D é uma árvore. Por isso é usual aplicar a nomenclatura de árvores, ou também arborescências, a essa classe de digrafos.

2.8 Fluxo máximo em redes

O conteúdo a seguir é dedicado ao estudo do fluxo máximo em redes, que é um tópico fundamental para a solução de diversos problemas de áreas diferentes, principalmente em Otimização Combinatória. Uma

grande variedade de problemas podem ser resolvidos mediante sua transformação em um caso de fluxo máximo em redes. Além disso, os algoritmos de fluxo máximo constituem exemplos didáticos em complexidade computacional. Serão apresentadas as definições e propriedades básicas, seguidas da apresentação do Teorema do Fluxo Máximo-Corte Mínimo, fundamental na teoria. Em seguida são descritos os principais algoritmos para a resolução do Problema do Fluxo Máximo, sendo o primeiro deles com complexidade exponencial, enquanto os demais são polinomiais com complexidades respectivamente decrescentes, segundo a sequência de apresentação.

2.9 O Problema do Fluxo Máximo

Uma rede é um digrafo $D(V,E)$ em que a cada aresta $e \in E$ está associado um número real positivo $c(e)$ denominado capacidade da aresta e. Suponha que D possua dois vértices especiais e distintos $s,t \in V$ chamados origem e destino, respectivamente, com as seguintes propriedades: o primeiro é uma fonte que alcança todos os vértices, enquanto o destino é um sumidouro alcançado também por todos. Um fluxo f de s a t em D é uma função que a cada aresta $e \in E$ associa um número real não negativo $f(e)$ satisfazendo às seguintes condições:

(i) $0 \leq f(e) \leq c(e)$, para toda aresta $e \in E$.

(ii) $\sum_{w_1} f(w_1,v) = \sum_{w_2} f(v,w_2)$, para todo vértice $v \neq s,t$.

A primeira condição acima simplesmente indica que o fluxo em cada aresta não ultrapassa o valor de sua capacidade. A segunda significa que o fluxo se conserva em cada vértice $v \neq s,t$. Isto é, o somatório dos fluxos das arestas convergentes a v é igual ao das divergentes de v. Esse somatório é denominado valor do fluxo em v. Por analogia, o somatório dos fluxos das arestas divergentes de s e o das convergentes a t são o valor do fluxo em s e o do fluxo em t, respectivamente. O valor do fluxo na origem é denominado valor do fluxo na rede D e denotado por $f(D)$.

A figura 2.20(a) ilustra um fluxo em uma rede. A capacidade e o fluxo em cada aresta estão indicados pelo par de números correspondentes, com o segundo entre parênteses. Por exemplo, a aresta (v_2,v_3) possui capacidade 2 e fluxo 1. O valor do fluxo no vértice v_2 é 3, no destino t é 4 e na rede também é 4. A situação da rede da figura 2.20(b) não corresponde a um fluxo, pois o vértice v_4 não satisfaz à condição (ii) acima (há um total igual a 3 de fluxo convergente a v_4 e um total 4

divergente). De um modo geral, se os valores $f(e)$ não obedecerem às condições da definição acima, a atribuição f será chamada fluxo ilegal.

O fluxo em uma rede possui certa analogia com o escoamento de água de uma origem s a um destino t, através de uma rede de tubulação. A capacidade de cada aresta corresponde à vazão máxima de água através da tubulação correspondente. O escoamento de água obedece às condições (i) e (ii) acima. Contudo, satisfaz também a restrições adicionais.

Naturalmente, o valor do fluxo em uma rede pode variar de um mínimo igual a zero até um certo máximo. Por exemplo, o valor do fluxo na figura 2.20(c) é igual a 7, o qual é máximo para a sua rede. O Problema do Fluxo Máximo consiste em, dada uma rede, determinar seu fluxo máximo.

Seja f um fluxo em uma rede $D(V,E)$. Uma aresta $e \in E$ é saturada quando $f(e) = c(e)$. Um vértice $v \in V$ é saturado quando todas as arestas convergentes a v ou todas divergentes de v estão saturadas. No fluxo da figura 2.20(c), a aresta (v_2,t) e o vértice v_4 estão saturados. Um fluxo é maximal quando todo caminho de s a t em D contém alguma aresta saturada. Isto é, o valor de um fluxo maximal não pode ser aumentado simplesmente por acréscimos de fluxos em algumas arestas.

Naturalmente, todo fluxo máximo é maximal. Contudo, a recíproca não é necessariamente verdadeira. Por exemplo, no fluxo da figura 2.21(a), as arestas (s,v_1), (v_1,v_4) e (v_4,t) estão saturadas, bem como os vértices v_1 e v_4. Esse fluxo é maximal, pois todo caminho de s a t contém uma dessas arestas (ou um desses vértices). Observe que ele não é máximo, pois possui valor 1 enquanto o da figura 2.21(b) possui valor 2.

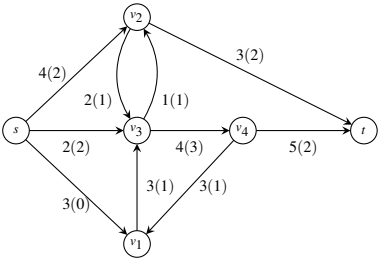
Seja $S \subseteq V$ um subconjunto de vértices tal que $s \in S$ e $t \notin S$. Denote $\overline{S} = V - S$. Um corte $(S,\overline{S})$ em D é o subconjunto das arestas de D que possuem uma extremidade em S e outra em $\overline{S}$. Assim sendo, todo caminho da origem s ao destino t em D contém alguma aresta de $(S,\overline{S})$. Sejam:

$$(S,\overline{S})^+ = \{(v,w) \in E | v \in S \text{ e } w \in \overline{S}\}$$
$$(S,\overline{S})^- = \{(v,w) \in E | v \in \overline{S} \text{ e } w \in S\}.$$

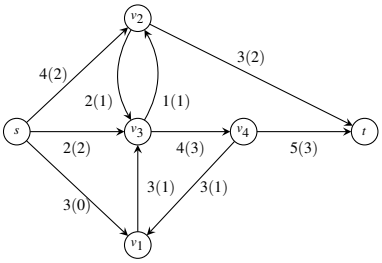
Define-se capacidade $c(S,\overline{S})$ do corte $(S,\overline{S})$ como o somatório das capacidades das arestas de $(S,\overline{S})^+$. Observe que o corte $(S,\overline{S})$ inclui as arestas de $(S,\overline{S})^-$, mas essas não são utilizadas no cálculo de sua capacidade. Um corte mínimo é aquele que possui capacidade mínima.

Seja f um fluxo e $(S,\overline{S})$ um corte em D. Então, $f(S,\overline{S})$ é o fluxo no corte (S,S), sendo definido como a diferença:

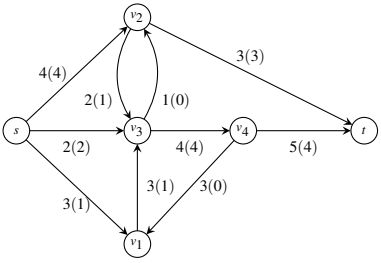
$$f(S,\overline{S}) = \sum_{e \in (S,\overline{S})^+} f(e) - \sum_{e \in (S,\overline{S})^-} f(e)$$



(a) Um fluxo.

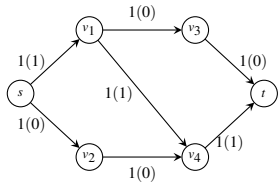


(b) Um fluxo ilegal.

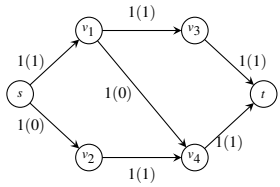


(c) Um fluxo máximo.

Figura 2.20: Exemplo de fluxo em redes.



(a)



(b)

Figura 2.21: Fluxos (a) maximal e (b) máximo.

Observe que, em geral, $c(S,\overline{S}) \neq c(\overline{S},S)$ e $f(S,\overline{S}) \neq f(\overline{S},S)$. Por exemplo, na rede da figura 2.20(a), com $S = s, v_2, v_3$ obtém-se $\overline{S} = v_1, v_4, t$ e o corte $(S,\overline{S}) = (s,v_1),(v_1,v_3),(v_3,v_4),(v_2,t)$, no qual:
$(S,\overline{S})^+ = (s,v_1),(v_3,v_4),(v_2,t)$,
$(S,\overline{S})^- = (v_1,v_3)$,
$c(S,\overline{S}) = c(s,v_1)+c(v_3,v_4)+c(v_2,t) = 10$ e
$f(S,\overline{S}) = f(s,v_1)+f(v_3,v_4)+f(v_2,t)-f(v_1,v_3) =$
$f(S,\overline{S})^+ - f(S,\overline{S})^- = 4$.

Observe que o valor do fluxo em uma rede é igual ao seu valor no corte $(s,V-s)$. Na realidade, esse fato é ainda mais geral. O valor do fluxo em uma rede pode ser medido em qualquer corte, como indica o lema seguinte.

Lema 2.9.1. Seja f um fluxo em uma rede D e $(S,\overline{S})$ um corte em D. Então, $f(S,\overline{S}) = f(D)$.

Prova. Indução no tamanho de S. Se $|S| = 1$, então $S = \{s\}$ e $f(S,\overline{S}) = f(D)$ por definição. Para $|S| > 1$ escolha algum $v \in S$, $v \neq s$. Sejam (v,S) e (S,v) os conjuntos das arestas de E com uma extremidade em v e outra em S, dirigidas de v para S e de S para v, respectivamente. Denote $S' = S - v$. Sejam $f(v,S)$ e $f(S,v)$ os somatórios dos fluxos nas

arestas de (v,S) e (S,v), respectivamente. Observe que $f(S,\overline{S}) = f(\overline{S'},\overline{S'}) + f(S,v) - f(v,S) + f(\overline{S},v) - f(v,\overline{S})$. Pela condição (ii) da definição de fluxo, $f(v,S) + f(v,\overline{S}) = f(S,v) + f(\overline{S},v)$. Logo, $f(S,\overline{S}) = f(S',\overline{S'})$. Pela hipótese de indução, $f(S',\overline{S'}) = f(D)$, o que completa a prova.

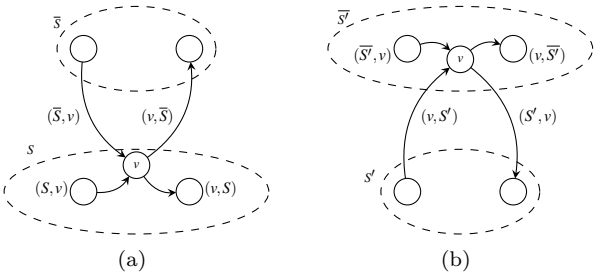


Figura 2.22: Prova do lema 2.9.1.

□

Note que, em particular, $f(V-\{t\},\{t\}) = f(\{s\},V-\{s\}) = t(D)$.

2.9.1 O Teorema do Fluxo Máximo – Corte Mínimo

Nesta seção serão apresentadas as condições que permitem caracterizar um fluxo máximo. Essa caracterização será obtida com o Teorema do Fluxo Máximo – Corte Mínimo apresentado no final desta parte.

Seja f um fluxo em uma rede D. O objetivo, no momento, é aumentar o valor de f, se possível. Cada aresta e pode receber um adicional de fluxo $\leq c(e) - f(e)$, o que talvez produza um aumento no valor de f. Uma aresta e tal que $c(e) - f(e) > 0$ denomina-se aresta direta. É possível também que f não seja máximo e simultaneamente não seja possível aumentar f unicamente através de incrementos de fluxo em arestas diretas. Um fluxo maximal mas não máximo é um exemplo desta última afirmativa, pois neste caso não há caminho de s a t através unicamente de arestas diretas. Em consequência, há situações em que a única forma de aumentar f consiste em, além de incrementar o fluxo em algumas arestas, decrementá-lo em outras. Por exemplo, para aumentar o valor do fluxo na rede da figura 2.20(b), torna-se necessário também decrementá-lo na aresta (v_1,v_4). Naturalmente, uma aresta e pode receber um decremento de fluxo positivo $\leq f(e)$. Se $f(e) > 0$, então e é denominada aresta contrária. Na rede da figura 2.20(a), (s,v_2) é aresta direta e contrária, enquanto (s,v_1) é direta e (s,v_3) é contrária.

Dados f e $D(V,E)$, define-se a rede residual $D'(f)$ como aquela em que o conjunto de vértices coincide com o de D e cujas arestas são obtidas pela seguinte construção: "Se (v,w) é aresta direta de D, então (v,w) é aresta de D', também chamada direta e com capacidade $c'(v,w) = c(v,w) - f(v,w)$. Se (v,w) é aresta contrária de D, então (w,v) é aresta de D', também chamada contrária e com capacidade $c'(w,v) = f(v,w)$".

Em outras palavras, as capacidades das arestas de D' representam as possíveis variações de fluxo que as arestas de D podem sofrer, com o direcionamento de cada aresta indicando sua variação positiva ou negativa. Como exemplo, as redes das figuras 2.23(a) e 2.23(b) são residuais respectivamente dos fluxos nas redes das figuras 2.21 e 2.20(a), respectivamente. As capacidades das arestas estão indicadas nas novas figuras.

Um caminho de s a t na rede residual $D'(f)$ é denominado aumentante (ou caminho de acréscimo de fluxo) para f. Por exemplo, o caminho s, v_2, v_4, v_1, v_3, t na rede da figura 2.23(a) é aumentante para o fluxo da figura 2.21.

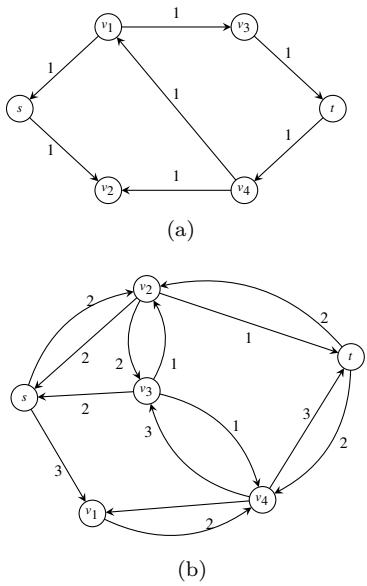


Figura 2.23: Redes residuais.

O lema 2.9.2 seguinte descreve uma aplicação do conceito de rede residual ao Problema do Fluxo Máximo.

Lema 2.9.2. Seja f um fluxo em uma rede $D(V,E)$ e D' a rede residual

correspondente. Suponha que exista em D' um caminho aumentante $v_1, \ldots, v_k$ da origem $s = v_1$ ao destino $t = v_k$. Então $f(D)$ pode ser aumentado de um valor

$$F' = min\{c'(v_j, v_{j+1}) | 1 \leq j < k\}.$$

Prova. A construção seguinte obtém em D um novo fluxo de valor $f(D) + F'$. Para $1 \leq j < k$, se (v_j, v_{j+1}) é aresta direta, incrementar $f(v_j, v_{j+1})$ de F'. Se (v_j, v_{j+1}) é contrária, decrementar $f(v_{j+1}, v_j)$ de F'. Em ambos os casos, o fluxo no vértice v_{j+1} recebeu F' unidades adicionais. Além disso, como F' é a menor dentre as capacidades das arestas do caminho aumentante, assegura-se que os novos fluxos nas arestas também satisfazem às condições da definição. Ou seja, a nova situação é também um fluxo f'. Pelo corte $(s, V - s)$, conclui-se que $f'(D) = f(D) + [f'(s, v_2) - f(s, v_2)] = f(D) + F'$. □

Como exemplo, na rede residual (figura 2.23(a)) do fluxo da figura 2.21(a), s, v_2, v_4, v_1, v_3, t é um caminho aumentante no qual o valor mínimo entre as capacidades de suas arestas é 1. As arestas (s, v_2), (v_2, v_4), (v_1, v_3) e $(v - 3, t)$ são diretas, o que permite incrementar de 1 o fluxo em cada uma delas. A aresta (v_4, v_1) é contrária, o que permite decrementar de 1 em (v_1, v_4). Com essas alterações, a rede da figura 2.21(a) se transforma na da figura 2.21(b). Como exemplo adicional, considere a rede residual da figura 2.23(b), correspondente ao fluxo da figura 2.20(a). No caminho aumentante $s, v1, v4, t$, a capacidade mínima é 1. Assim sendo, pode-se incrementar de 1 o fluxo nas arestas (s, v_1) e (v_4, t) e diminuí-lo de 1 em (v_4, v_1). Isso aumenta em uma unidade o valor do fluxo da figura 2.20(a). Se todo caminho entre a origem e o destino em uma rede D contiver uma certa aresta, naturalmente o valor de qualquer fluxo em D não pode ultrapassar a capacidade $c(e)$. Isto é, essa aresta atua de forma semelhante a um "gargalo"para o fluxo. De uma forma mais geral, o valor de qualquer fluxo f em D não pode ultrapassar o valor de qualquer corte $(S, \overline{S})$, pois $f(D) = f(S, \overline{S}) = \sum_{e \in (S, \overline{S})^+} f(e) - \sum_{e \in (S, \overline{S})^-} f(e) \leq \sum_{e \in (S, \overline{S})^+} c(e) = c(S, \overline{S})$. Ou seja, o fluxo máximo em uma rede não pode ultrapassar a capacidade de seu corte mínimo.

Deste modo, é enunciado o teorema 2.9.1 seguinte, fundamental em Teoria de Fluxo em Redes, que afirma que esse limite superior é atingível (fluxo máximo de corte mínimo).

Teorema 2.9.1. O valor do fluxo máximo em uma rede D é igual à capacidade do corte mínimo de D.

Prova. Seja f um fluxo máximo em D. Pelo observado acima, $f(D) \leq c_{min}$, em que c_{min} é a capacidade do corte mínimo. Suponha $f(D) < c_{min}$ e seja D' a rede residual de f. Há duas possibilidades:

Caso 1: Existe caminho aumentante para f. Essa situação contradiz o lema 2.9.1.

Caso 2: Não existe caminho aumentante. Seja S o conjunto de vértices alcançáveis em D' a partir de s. Naturalmente, $s \in S$ e $t \in S'$. Caso contrário, se $t \in S$ haveria caminho aumentante. Então

(i) D' não possui aresta direta de S para $\overline{S'}$. Isto é, $f(e) = c(e)$, para cada $e \in (S,\overline{S})^+$.

(ii) D' não possui aresta contrária de S para $\overline{S}$. Isto é, $f(e) = 0$, para cada $e \in (S,\overline{S})^-$. Logo, $f(D) = f(S,\overline{S})\sum_{e\in(S,\overline{S})^+} f(e) - \sum_{e\in(S,\overline{S})^-} f(e) = c(S,\overline{S})$, o que contradiz $f(D) < c_{min}$.

Logo, $f(D) = c_{min}$. □

Os corolários 2.9.1 e 2.9.2 seguintes são consequências diretas do teorema 2.9.1.

Corolário 2.9.1. Sejam $(S,\overline{S'})$ um corte e f um fluxo máximo em uma rede D. Então $(S,\overline{S'})$ é mínimo se, e somente se,

(i) toda aresta $e \in (S,\overline{S})^+$ estiver saturada, e

(ii) toda aresta $e \in (S,\overline{S})^-$ satisfizer $f(e) = 0$.

Corolário 2.9.2. Um fluxo f em uma rede D é máximo se, e somente se, não existir caminho aumentante para f.

No exemplo da figura 2.21(b), (s,v_1,v_2,v_3,v_4,t) é um corte mínimo de capacidade 2. Logo, o fluxo máximo dessa rede possui valor 2 (figura 2.21(a)). Na figura 2.20(a), o corte (s,v_1,v_3,v_2,v_4,t) é mínimo e possui capacidade 7. Isso permite concluir que o fluxo da figura 2.20(c) é máximo.

2.9.2 Um primeiro algoritmo

Seja $D(V,E)$ uma rede em que cada aresta $e \in E$ possui capacidade $c(e)$ inteira positiva. Nesta seção formula-se um primeiro algoritmo para determinar o fluxo máximo numa rede D.

Algoritmo 2.1: Fluxo máximo em uma rede (Ford e Fulkerson)

Dados: rede $D(V,E)$, com capacidades $c(e)$ inteiras e positivas, para cada $e \in E$, origem $s \in V$ e destino $t \in V$
$F := 0$
para cada $e \in E$ faça
  $f(e) := 0$
  construir a rede residual $D'(f)$
enquanto existir caminho $v_1, \ldots, v_k$ de $s = v_1$ a $t = v_k$ em D' faça
  $F := min\{c'(v_j, v_{j+1}), |1 \leq j < k\}$
  para cada $j = 1, \ldots, k-1$ faça
    se (v_j, v_{j+1}) é aresta direta então
      $f(v_j, v_{j+1}) := (v_j, v_{j+1}) + F'$
    senão
      $f(v_j, v_{j+1}) := (v_j, v_{j+1}) - F'$
  $F := F + F'$ construir a rede residual $D'(f)$

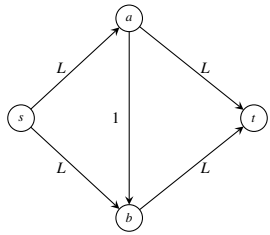


Figura 2.24: Um exemplo de pior caso para o algoritmo 2.1.

A prova do lema 2.9.1 é construtiva, fornecendo, juntamente com o teorema 2.9.1, o seguinte algoritmo:

No passo inicial, define-se $f(e) := 0$ para cada aresta e de D. O valor do fluxo F é, portanto, nulo. No passo geral, constrói-se a rede residual D' de D. Se houver caminho aumentante em D, então F pode ser incrementado, conforme o lema 2.9.1. Repete-se o processo. Se o caminho acima não existir, o fluxo é máximo. Ou seja, como as capacidades das arestas são números inteiros, os incrementos também o são. Portanto, o valor do fluxo se mantém inteiro no processo.

Deste modo, após um número finito de incrementos, F atinge o valor máximo igual à capacidade do corte mínimo de D. Para calcular a

complexidade desse algoritmo é necessário estimar o número de iterações do bloco. Contudo, esse número pode depender do valor F do fluxo máximo. Assim sendo, em um pior caso F pode ser exponencial no tamanho dos incrementos F'. Ou seja, o número de iterações é $\Omega(F)$. Em cada iteração deve-se construir a rede residual D' e um caminho aumentante. Essas operações podem ser realizadas em tempo $O(m)$. O passo inicial também requer $O(m)$ operações. Logo, a complexidade do algoritmo é $O(m(F+1))$. Observe que F pode ser exponencial em m. A rede da figura 13 ilustra um pior caso, em que o valor do fluxo máximo é $2L$. Se os caminhos aumentantes s,a,b,t *e* s,b,a,t forem alternadamente escolhidos pelo algoritmo, então o valor do incremento de fluxo F' é sempre igual a 1. Deste modo, serão necessárias $2L$ iterações, em que L pode ser arbitrariamente grande.

2.9.3 Um algoritmo $O(nm^2)$

Dada uma rede D e um fluxo f em D, o algoritmo 2.1 da seção anterior iterativamente constrói a rede residual $D'(f)$ e procura um caminho aumentante, o qual, se existir, é utilizado para incrementar o valor do fluxo. Não há restrição à escolha do caminho aumentante, isto é, qualquer um pode ser considerado.

Deste modo, será agora apresentado um critério de escolha de tais caminhos, devido a Edmonds e Karp. Esse critério, quando incorporado ao algoritmo 2.1 reduz sensivelmente a sua complexidade. O critério adotado é o seguinte:

"Escolher sempre o caminho aumentante de menor comprimento, isto é, com menor número de arestas."

A justificativa da adoção dessa estratégia é dada pelo lema 2.9.3 seguinte.

Lema 2.9.3. Seja f_1 um fluxo em uma rede D e k o comprimento do menor caminho aumentante p_1 para f_1. Seja f_2 o fluxo obtido em D pelo aumento de f_1 através de p_1, conforme o lema 2.9.1. Então o comprimento do menor caminho aumentante p_2 para f_2, se houver, é $\geq k$.

Prova. Comparando as redes residuais $D'(f_1)$ e $D'(f_2)$, verifica-se que: (i) arestas não pertencentes a p_1 aparecem igualmente nas duas redes, (ii) arestas de p_1 com capacidade mínima em p_1 não figuram em $D'(f_2)$ e (iii) arestas diretas (v,w) de p_1 com $f_1(v,w)=0$ implicam a introdução de arestas contrárias (w,v) em $D'(f_2)$. Portanto, as únicas arestas em $D'(f_2)$ mas não em $D'(f_1)$ são as de (iii). Se p_2 possui comprimento

$< k$, então, necessariamente utiliza arestas de (iii), o que contradiz p_1 ser de comprimento k (isto é, se (v,w) é parte do menor caminho p_1, a introdução em D de arestas do tipo (w,v) não pode produzir caminhos de comprimento menor do que de p_1). □

Neste caso, a única modificação proposta em relação ao algoritmo 2.1 é a forma de se obter o caminho aumentante. Na descrição da seção 2.9.2 anterior, este é arbitrário, enquanto a nova estratégia o define como o de menor comprimento. Pelo lema 2.9.3, os comprimentos dos caminhos aumentantes escolhidos segundo o critério acima formam uma sequência não decrescente. Para cada k, $1 \leq k < n$, existem não mais do que $O(m)$ caminhos aumentantes, pois, em cada tal caminho p, pelo menos uma aresta de p pode ser saturada (a de menor capacidade corrente), o que impossibilita a sua utilização posterior em outro caminho do mesmo comprimento k. Cada menor caminho aumentante pode ser encontrado em $O(m)$ passos, utilizando busca em largura. Logo, a complexidade de todo o algoritmo é $O(nm^2)$. Observe que essa alteração simples introduzida no algoritmo 2.1 é suficiente para transformá-lo em um processo polinomial. A argumentação acima mostra que o algoritmo termina em $O(nm^2)$ passos. Não se considerou que as capacidades das arestas são números inteiros. Isso significa que o processo pode ser aplicado mesmo se essa condição não for satisfeita. Ou seja, no algoritmo da presente seção, bem como nos subsequentes, admitem-se capacidades arbitrárias não negativas.

2.9.4 Um algoritmo $O(n^2m)$

Seja f um fluxo em uma rede $D(V,E)$. O algoritmo da seção anterior iterativamente procura o caminho aumentante para f com menor comprimento. Isto é, o caminho p da origem s ao destino t com menor número de arestas na rede residual $D'(f)$.

Para melhor examinar o problema de encontrar tal caminho, pode-se considerar a sub-rede $D^*(f)$ de $D'(f)$, a qual contém somente os vértices e arestas de $D'(f)$ que podem figurar em p. Por exemplo, um vértice cuja distância a s é maior do que a de s a t obviamente não pode pertencer a p. Da mesma forma, se v_1 e v_2 são vértices tais que a distância de s a v_1 é menor ou igual à de s a v_2, então uma aresta (v_2, v_1) também não aparece em p. Como consequência, a rede $D^*(f)$ é acíclica e somente contém arestas entre vértices localizados em níveis contíguos na árvore da largura T de raiz s. Por extensão, o nível de um vértice v na árvore T é chamado nível de v na rede $D^*(f)$. Por sua vez, $D^*(f)$ é denominado rede de camadas para f.

O algoritmo 2.2 a seguir constrói uma rede de camadas $D^*(f)$, a partir da rede residual $D'(f)$.

Algoritmo 2.2: Construção da rede de camadas $D^*(f)$

```
Dados: rede residual D'(f) efetuar uma busca em largura B em
  D' de raiz s
T := 0 árvore de largura obtida por B
para cada (v,w) ∈ D' faça
    se nvel(v) ≥ nvel(w) em T então
        eliminar (v,w)
    seja v_1,...,v_n a sequência de vértices de D' em ordem não
      crescente de seus níveis em T
para cada j = 1,...,n faça
    se v_j ≠ t e grau saída (v_j) = 0 então
        eliminar v_j
```

O algoritmo acima tem complexidade $O(m)$. A sua correção decorre do fato de que numa busca em largura não existe aresta (v,w) tal que $nvel(v) < nvel(w) - 1$.

Observe que, se f for máximo, não há caminho em D' de s a t. Consequentemente t não será incluído em $D^*(f)$. Logo, o digrafo de camadas $D^*(f)$ não existirá, pois todos os seus vértices serão eliminados no processo.

O algoritmo de determinação do fluxo máximo pode ser reformulado em termos do digrafo de camadas. Naturalmente, a procura de um caminho aumentante p pode ser realizada na rede de camadas D^* em vez de na rede residual D'. Suponha que tenha sido encontrado um caminho p, da origem ao destino em $D^*(f)$, e seja F' a capacidade mínima entre as arestas de p. A estratégia inicial consistia em aumentar de F' o fluxo f em D, obtendo o fluxo aumentado f', o qual seria utilizado para construir $D^*(f')$, repetindo-se o processo. Como alternativa, tenta-se obter $D^*(f')$ diretamente de $D^*(f)$. Para tanto, basta percorrer p em $D^*(f)$, redefinindo as capacidades $c'(e)$ das arestas. Isto é, para cada aresta e de p, efetuar:

se e é aresta direta, então $c'(e) := c'(e) - F'$,

eliminando e de $D^*(f)$ caso $c'(e)$ tenha decrescido para zero.

Caso contrário (quando e é aresta contrária),

$c'(e) : c'(e) + F'$.

O procedimento acima atualiza a rede de camadas $D^*(f)$ em tempo $O(k)$, em que $k < n$ é a distância de s a t em $D^*(f)$. Contudo, esse método somente pode gerar redes $D^*(f')$ caso a distância entre s e t em $D^*(f')$ se mantenha igual a k. Isto é, se p foi o último caminho aumentante do comprimento k, o procedimento acima não pode ser aplicado, pois geraria uma rede de camadas desconexa. Nesse caso, obtém-se $D^*(f')$ a partir da definição, utilizando o algoritmo 2.2.

Observe que o número de vezes em que a rede de camadas D^* deve ser construída pelo algoritmo 2.2 é $O(n)$, isto é, no máximo uma vez para cada comprimento $k, 1 \leq k \leq n-1$. Nas demais vezes, obtém-se D^* pelo procedimento acima, mais eficiente do que a aplicação do algoritmo 2.2. Conforme mencionado, o novo método pode ser empregado sempre que houver, na rede de camadas atualizada $D^*(f')$, um caminho aumentante do mesmo comprimento k obtido na rede anterior $D^*(f)$, isto é, enquanto o fluxo em D^*, definido pelos caminhos aumentantes do mesmo comprimento k, não for maximal. Com isso, pode-se transformar o problema de determinar um fluxo máximo em uma rede arbitrária D em $O(n)$ problemas de determinar um fluxo maximal em uma rede de camadas D^*. Este último é certamente mais simples. A formulação seguinte descreve a nova estratégia. Ao final do processo, f é um fluxo máximo de valor F em D. A complexidade do algoritmo 2.3 é basicamente igual a n vezes a complexidade do processo de obter um fluxo maximal em D^*.

Considere agora o problema de determinar um fluxo maximal f^* de valor F^* na rede de camadas D^* com origem s e destino t, sendo k a distância de s a t em D^*. A capacidade de cada aresta e em D^* será denotada $c^*(e)$.

O processo mais acima descrito para atualização direta de $D^*(f)$ obtém, na realidade, o desejado fluxo maximal. Isto é, no passo inicial define-se $f^*(e) := 0$, para cada aresta e de D^*. No passo geral, escolhe-se um caminho arbitrário p de s a t. Seja F' a capacidade mínima entre as arestas p. Para cada aresta e de p efetua-se $c^*(e) := c^*(e) - F'$, eliminando e se $c^*(e)$ decresceu para zero. Repete-se o processo. Caso não haja caminho de s a t, f^* é maximal.

A escolha do caminho $v_1, ..., v_{k+1}$ de $s = v_1$ a $t = v_{k+1}$ pode ser feita de modo que, para cada $j, 1 \leq j \leq k$, o vértice v_{k+1} seja o primeiro da lista $A(v_j)$. Como qualquer aresta divergente de v_j conduz sempre a um nível mais próximo de t, o método está correto.

Algoritmo 2.3: Fluxo máximo em uma rede (Dinic)

```
Dados: rede D(V,E), cada aresta com capacidade real positiva
  origem s ∈ V e destino t ∈ V
F := 0
para cada e ∈ E faça
    f(e) := 0
    construir a rede de camadas D*(f) pelo algoritmo 2.2
enquanto existir D*(f) faça
    obter o fluxo maximal f* de valor F* em D*
    para cada e ∈ rede D* faça
        se e é aresta direta de D então
            f(v_j, v_{j+1}) := (v_j, v_{j+1}) + F'
        senão
            f(v_j, v_{j+1}) := (v_j, v_{j+1}) − F'
    F := F + F'
    construir D*(f) pelo algoritmo 2.2
```

Cada caminho p é obtido, portanto, em tempo $O(n)$. Como pode haver $O(m)$ caminhos até a separação de s e t, o processo acima encontra um fluxo maximal em D^* após $O(nm)$ passos. Consequentemente a complexidade do algoritmo 2.3 para determinar um fluxo máximo em D é $O(n^2m)$.

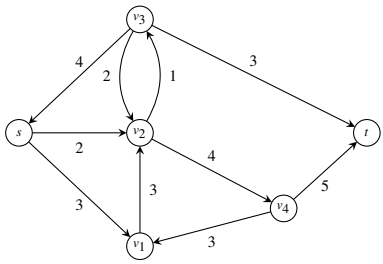
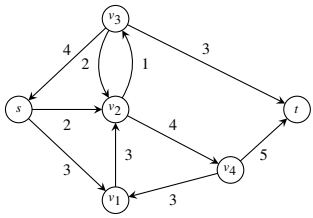


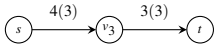
Figura 2.25: Entrada para o exemplo da figura 2.26.

As próximas figuras ilustram um exemplo do algoritmo 2.3 aplicado sobre a rede da figura 2.25. O fluxo inicial é escolhido como de valor 0. A rede residual, inicial, apareceu na figura 2.26(a), a rede de camadas correspondente na figura 2.26(b), e o fluxo maximal, de valor 3, na figura 2.26(c). Observe que os caminhos mínimos na rede (Figura 2.26(c)) possuem comprimento 2. A rede de camadas atualizada se encontra

na figura 2.27(a), gerando a rede de camadas da figura 2.27(b), cujos caminhos mínimos possuem comprimento 3, e o fluxo maximal valor 2. O processo prossegue, obtendo comprimento de caminho mínimo até 4, o qual aparece na figura 2.28(c). O valor do fluxo máximo é a soma dos valores dos fluxos maximais obtidos, igual a $3+2+2=7$.



(a) Rede residual D'



(b) Rede de camadas D^*

s, v_3, t

(c) Caminhos aumentantes

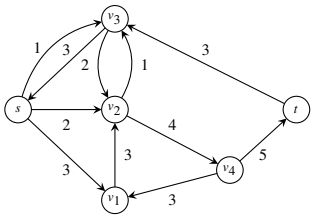
Figura 2.26: 1ª iteração. Fluxo maximal 3.

2.9.5 Um algoritmo $O(n^3)$

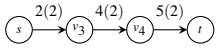
Nesta seção será apresentado um algoritmo de complexidade cúbica no número de vértices para se determinar o fluxo máximo. Inicialmente, será apresentado um algoritmo para determinar um fluxo maximal em uma rede de camadas, proposto por Malhotra, Pramodh, Kumar e Maheshwari. Esse algoritmo possui complexidade $O(n^2)$. Isso permite obter o fluxo máximo em uma rede em $O(n^3)$ passos, conforme já mencionado.

Na seção 2.9.4 anterior, um fluxo maximal foi obtido eliminando-se uma aresta da rede em cada iteração. No algoritmo descrito abaixo, elimina-se um vértice em cada passo.

Para cada vértice v da rede de camadas D^*, define-se capacidade de v como $min\{\sum_w c^*(v,w), \sum_w c^*(v,w)\}$, onde $c^*(w,s) = c^*(t,w) = \infty$. Ou seja, a capacidade de v é o menor dentre os somatórios das capacidades c^* das arestas divergentes e convergentes a v, respectivamente. Esse valor exprime o máximo de fluxo que pode passar através de v.



(a) Rede residual D'



(b) Rede de camadas D^*

s, v_2, v_4, t

(c) Caminhos aumentantes

Figura 2.27: 2ª iteração. Fluxo maximal 2.

Seja v_c o vértice de menor capacidade em D^*. Existe um fluxo f_c^* que satura v_c em D^* e cujo valor é igual à capacidade mínima $c^*(vc)$. A construção desse fluxo está indicada mais adiante. Obtido f_c^*, decrementa-se $c^*(e)$ de $f_c^*(e)$, e incrementa-se $f^*(e)$ de $f_c^*(e)$, para cada aresta e de D^* com $f_c^*(e) \neq 0$. Se v_c é igual a s ou t, o processo termina, pois um fluxo que satura a origem ou o destino é obviamente maximal. Considere então $v_c \neq s,t$. Elimina-se v_c de D^*, pois esse vértice está saturado. Da mesma forma, é eliminada de D^* cada aresta cuja capacidade tornou-se nula. Em seguida são eliminados todos os vértices

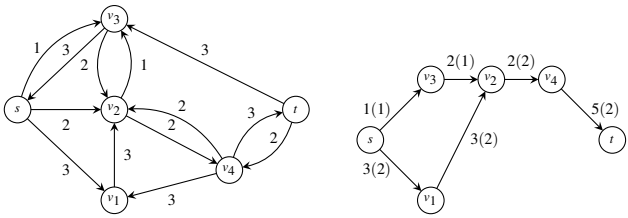
(i) $w \neq t$ com grau saída$(w) = 0$; e

(ii) $w \neq s$ com grau entrada$(w) = 0$.

Deve-se repetir esse passo de eliminação até que os graus de entrada e saída de cada vértice $\neq s,t$ sejam ambos diferentes de zero. Se s ou t foi eliminado, o algoritmo também se encerra, pois o fluxo é maximal. Caso contrário, deve-se repetir todo o processo escolhendo o novo vértice v_c de capacidade mínima na rede corrente D^* e assim por diante.

Em seguida, descrevemos um método para obter o fluxo f_c^* de valor igual a $c^*(v_c)$ e que sature v_c. Considere inicialmente o trecho de v_c a t . O fluxo f_c^* é enviado de v_c até t com a observação do seguinte critério:

"Para cada vértice v de D^* permite-se no máximo uma aresta divergente (v,w) satisfazendo $0 < f_c^*(v,w) < c^*(v,w)$".



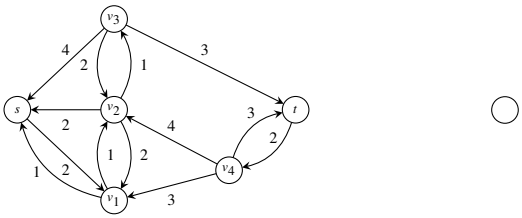
(a) Rede Residual D' (b) Rede de camadas D^*

$$s, v_3, v_2, v_4, t$$

$$s, v_1, v_2, v_4, t$$

(c) Caminhos aumentantes

Figura 2.28: 3ª iteração. Fluxo maximal 2.



(a) Rede Residual D' (b) Rede de camadas D^*

Figura 2.29: 4ª iteração. Fluxo maximal: $3+2+2=7$.

Ou seja, cada aresta considerada deve ser utilizada até a sua capacidade máxima, sempre que possível. Isso implica existir no máximo uma aresta parcialmente saturada divergente de cada vértice.

Seja $F'_c(v)$ o valor total do fluxo convergente ao vértice v. No passo inicial, $F'_c(v_c) = c^*(v_c)$ e $F'_c(v) = 0$ se $v \neq v_c$. O processo termina quando $F'_c(v_t)$ alcançar o valor $c^*(v_c)$ e $F'_c(v) = 0$ se $v \neq v_t$.
O fluxo de saída do vértice v é definido da seguinte maneira: seja (v,w) a primeira aresta divergente de v, na ordem considerada, isto é, w é o primeiro vértice de $A(v)$. Logo,

$$\text{se } F'_c(v) \leq c^*(v,w) \text{ então } f'_c(v,w) := F'_c(v)$$
$$\text{caso contrário } f'_c(v,w) := c^*(v,w)$$
$$F'_c(v) := F'_c(v) - c^*(v,w)$$
$$w := \text{vértice sucessor de } w \text{ em } A(v),$$

repetir o processo.

Para o processo acima de definição do fluxo de saída de cada v, os vértices devem ser considerados em uma ordem topológica. Ou seja, nenhum vértice pode ser considerado sem que os seus antecessores na rede em D^* tenham sido.

Para o trecho de s a v_c o método é similar. Exceto que o fluxo se desenvolve da origem vc ao destino s na rede simétrica de D^* (isto é, invertendo-se a direção de cada aresta de D^*).

Para a determinação da complexidade do algoritmo, observe que para cada vértice v, se q arestas (v,w) foram manipuladas no processo, então pelo menos $q-1$ foram eliminadas. Se o custo da eliminação das arestas for contabilizado em separado, cada vértice v manipulou no máximo uma aresta. Assim sendo, a determinação de cada f_c^* é realizada em $O(n)$ passos. Esse procedimento deve ser repetido no máximo uma vez para cada vértice. Logo, a complexidade é $O(n^2)$ mais o custo da eliminação das arestas. Cada aresta foi eliminada uma vez no máximo. Logo, o custo total de eliminação é $O(m)$, o que mantém $O(n^2)$ como a complexidade do algoritmo de determinação do fluxo maximal em D^*, ou seja, um algoritmo de complexidade $O(n^3)$ para encontrar o fluxo máximo em D.

Finalmente, para o algoritmo 2.1, que determina o fluxo máximo, é apresentado o pseudocódigo a seguir com alguns detalhes implementacionais importantes.

2.10 Notas sobre este capítulo

Este capítulo foi fortemente baseado no livro [202], de Szwarcfiter (2018).

A literatura sobre grafos é relativamente vasta. Há diversos textos de caráter geral, tais como [16, 17, 25, 31, 41, 60, 113, 171, 214]. Entre os textos mais recentes, destacamos [32, 62, 26, 211]. Os seguintes livros abordam temas mais específicos: entre outros, [204] discorre sobre conectividade; [171, 187], sobre o problema das quatro cores; [36, 114], sobre grafos direcionados. [47, 70, 106] tratam de grafos sob enfoque algorítmico. [1] é uma coletânea de aplicações de grafos em uma diversidade de outras áreas. [147] apresenta problemas e exercícios, com sugestões e soluções. [34], por sua vez, estudam as classes de grafos. [197], as representações em grafos. [61], a decomposição em grafos. [148] os emparelhamentos. A literatura brasileira em grafos inclui os textos de [11, 84, 24, 150]. Um texto sobre igualdade min-max em grafos, em língua portuguesa, é o [74]. Os trabalhos pioneiros em árvores foram [130, 69,

Algoritmo 2.4: Pseudocódigo com detalhes de implementação do fluxo máximo.

```
1  def: FluxoMaximo(D);
2  F := 0
3  para cada (v,u) ∈ D.E() faça
4      uv.e.f = 0
5      Dlin = ObterRedeResidual(D)
6      s,t = D.ObterVdeRotulo("s"), D.ObterVdeRotulo("t")
7      P = Busca(Dlin , s,t)
8  enquanto (P) > 0 faça
9      Flin = min([uv.e.r para uv em P])
10     para cada j ∈ P faça
11         se P[j].e.direta então
12             P[j].e.eD.f = P[j].e.eD.f+Flin
13         senão
14             P[j].e.eD.f = P[j].e.eD.f-Flin
15     F := F+Flin
16     Dlin = ObterRedeResidual(D)
17     P = Busca(Dlin , s,t)
18 def: ObterRedeResidual(D);
19 Dlin = GrafoListaAdj(orientado=True);
20 Dlin.DefinirN(D.n,D.RotulosV)
21 para cada (v,uv) ∈ D.E() faça
22     uv = uv.e
23     se uv.c − uv.f > 0 então
24         e = Dlin.AdicionarAresta(D.ObterRotuloV(uv.v1),
             D.ObterRotuloV(uv.v2));
25         e.r = uv.c − uv.f;
26         e.direta = True;
27         e.eD = uv;
28     se uv.f > 0 então
29         e = Dlin.AdicionarAresta(D.ObterRotuloV(uv.v2),
             D.ObterRotuloV(uv.v1));
30         e.r = uv.f;
31         e.direta = False;
32         e.eD = uv;
33 Retorne Dlin;
```

124], bem como os resultados sobre enumeração de árvores de [43]. A importância das árvores para algoritmos é ressaltada em [132].

Os trabalhos fundamentais em Teoria de Fluxo Máximo são [80, 81, 82]. Os algoritmos apresentados foram propostos por [67, 63, 158]. Há outros algoritmos eficientes para o Problema do Fluxo Máximo, tais como: [128], complexidade $O(n^3)$; [45], complexidade $O(n^2\sqrt{m})$; [85], complexidade $O(n^{\frac{5}{3}}m^{\frac{2}{3}})$; [120, 86] , complexidades $O(nmlog^2n)$; [196], complexidade $O(nmlogn)$. Uma variação importante dos algoritmos de fluxo máximo apresentados neste capítulo foi descrita por [103], com implementação eficiente por Goldberg [101]. Para o Problema de Fluxo Máximo, há o livro [131]. Uma visão histórica do trabalho de Ford e Fulkerson no Problema do Fluxo Máximo foi descrita em [191]. Um artigo tutorial indicado de fluxo máximo é o [102].

3

Modelagem com variáveis 0-1

Ana Flávia Uzeda Macambira
Virginia Costa
Nelson Maculan
Gérard Plateau
Viviane Cátia Köhler

3.1 Introdução

Apresentaremos neste capítulo um pouco da arte de modelar diferentes problemas de otimização utilizando variáveis $0-1$ através de exemplos. Outros exemplos podem também ser encontrados em [44, 217, 21].

3.2 Transformação de um modelo de otimização não linear mista em um problema de otimização linear mista

A técnica de linearização é muito importante, pois permite que sejam resolvidos problemas equivalentes ou, em outros casos, aproximados, de forma a obter um limitante (inferior ou superior, dependendo do problema), informação esta que já propicia um ponto de partida para a resolução do problema original, como pode ser visto em [154, 21, 217].

Exemplo 1.

$$
\begin{aligned}
\text{(P): Minimizar } & z = 3x_1x_2x_3^4 - 2\ \min\{x_4,x_5\} - 5x_1x_2 - 3|x_4 - x_5| - 4x_3x_5 + 4x_6, && (3.1)\\
\text{sujeito a: } & 7x_1 + 4x_2 + 2\max\{x_4,x_5\} \leq 40, && (3.2)\\
& 3x_1x_2 + 2x_1x_3 + 4x_2x_3 \geq 2, && (3.3)\\
& x_1x_5 + x_2x_4 + x_6 \geq 30, && (3.4)\\
& x_j \in \{0,1\}, \quad j = 1,2,3; \quad 0 \leq x_j \leq 15, \quad j = 4,5, && (3.5)\\
& x_6 \in \{0,7,16,28\}. && (3.6)
\end{aligned}
$$

O problema (P) definido em (3.1)-(3.6), não linear inteiro misto (PNLIM), pode ser reescrito de forma relaxada (PR), ou seja, com suas variáveis inteiras agora pertencentes a conjuntos contínuos.

$$
\begin{aligned}
\text{(PR): Minimizar } & z = 3x_1x_2x_3^4 - 2\ \min\{x_4,x_5\} - 5x_1x_2 - 3|x_4 - x_5| - 4x_3x_5 + 4x_6, && (3.7)\\
\text{sujeito a: } & 7x_1 + 4x_2 + 2\max\{x_4,x_5\} \leq 40, && (3.8)\\
& 3x_1x_2 + 2x_1x_3 + 4x_2x_3 \geq 2, && (3.9)\\
& x_1x_5 + x_2x_4 + x_6 \geq 30, && (3.10)\\
& 0 \leq x_j \leq 1, \quad j = 1,2,3; \quad 0 \leq x_j \leq 15, \quad j = 4,5, && (3.11)\\
& 0 \leq x_6 \leq 28. && (3.12)
\end{aligned}
$$

Pode-se imaginar que o problema (PR) seja de muito mais fácil solução em relação a (P) pela relaxação das restrições de integralidade das variáveis, porém, num contexto geral isso não é verdade, pois (PR) continua não linear, não convexo e não diferenciável, características que se traduzem em dificuldades para a sua resolução. A maneira de se resolver (P) é contornar a não linearidade do problema, obtendo-se um problema linear inteiro misto (PLIM). Abaixo, apresenta-se passo a passo o procedimento utilizado para a linearização de (P) e logo após é apresentado o modelo (PLIM), resultado do processo de linearização.
Passo 1:
$\min\{x_4,x_5\} = u_1$, e considerando-se $\lambda \in \{0,1\}$:
$-15(1-\lambda) \leq x_5 - x_4 \leq 15\lambda$, têm-se dois casos:

- $x_5 - x_4 \leq 0 \ \rightarrow \lambda = 0, \ \ -15 \leq x_5 - x_4 \leq 0 \ \rightarrow x_5 \leq x_4;$
- $x_5 - x_4 \geq 0 \ \rightarrow \lambda = 1, \ \ 0 \leq x_5 - x_4 \leq 15 \rightarrow x_4 \leq x_5.$

Pode-se observar que, se $\lambda = 0 \rightarrow x_5 \leq x_4$, e se $\lambda = 1 \rightarrow x_4 \leq x_5$.
Agora define-se:

$$
\begin{aligned}
-15(1-\lambda) + x_4 \leq u_1 \leq x_4 + 15(1-\lambda), \text{ e}\\
-15\lambda + x_5 \leq u_1 \leq x_5 + 15\lambda.
\end{aligned}
$$

Passo 2:

$\max\{x_4, x_5\} = u_2$, onde

$$\begin{aligned} -15(1-\lambda) + x_5 \leq u_2 \leq x_5 + 15(1-\lambda), \text{ e} \\ -15\lambda + x_4 \leq u_2 \leq x_4 + 15\lambda. \end{aligned}$$

Passo 3:
$|x_4 - x_5| = u_3$, e considera-se $\beta \in \{0,1\}$, tal que

$$\begin{aligned} -15(1-\beta) \leq x_4 - x_5 \leq 15\beta, \\ -30(1-\beta) + (x_4 - x_5) \leq u_3 \leq (x_4 - x_5) + 30(1-\beta), \\ -30\beta + (x_5 - x_4) \leq u_3 \leq (x_5 - x_4) + 30\beta. \end{aligned}$$

Passo 4:

$$x_3^4 = x_3 \in \{0,1\}.$$

Passo 5:

$$\begin{aligned} x_1 x_2 x_3 = u_{123}, \quad u_{123} \geq 0, \text{ tal que:} \\ u_{123} \leq x_1, \ u_{123} \leq x_2, \ u_{123} \leq x_3, \\ x_1 + x_2 + x_3 - 2 \leq u_{123}. \end{aligned}$$

Passo 6:

$$\begin{aligned} x_1 x_2 = u_{12}, \quad u_{12} \geq 0, \text{ tal que:} \\ u_{12} \leq x_1, \ u_{12} \leq x_2, \ x_1 + x_2 - 1 \leq u_{12}. \end{aligned}$$

Passo 7:

$$\begin{aligned} x_1 x_3 = u_{13}, \quad u_{13} \geq 0, \text{ tal que:} \\ u_{13} \leq x_1, \ u_{13} \leq x_3, \ x_1 + x_3 - 1 \leq u_{13}. \end{aligned}$$

Passo 8:

$$\begin{aligned} x_2 x_3 = u_{23}, \quad u_{23} \geq 0, \text{ tal que:} \\ u_{23} \leq x_2, \ u_{23} \leq x_3, \ x_2 + x_3 - 1 \leq u_{23}. \end{aligned}$$

Passo 9:

$$x_3x_5 = u_{35}, \quad 15x_3 \geq u_{35}, \quad u_{35} \geq 0, \quad \text{tal que:}$$
$$-15(1-x_3)+x_5 \leq u_{35} \leq x_5+15(1-x_3).$$

Passo 10:

$$x_1x_5 = u_{15}, \; 15x_1 \geq u_{15}, \quad u_{15} \geq 0, \quad \text{tal que:}$$
$$-15(1-x_1)+x_5 \leq u_{15} \leq x_5+15(1-x_1).$$

Passo 11:

$$x_2x_4 = u_{24}, \; 15x_2 \geq u_{24}, \quad u_{24} \geq 0, \quad \text{tal que:}$$
$$-15(1-x_2)+x_4 \leq u_{24} \leq x_4+15(1-x_2).$$

Passo 12:

$$x_6 = 7\gamma_1+16\gamma_2+28\gamma_3, \quad \text{tal que:}$$
$$\gamma_1+\gamma_2+\gamma_3 \leq 1.$$

Apresentamos o (PLIM) equivalente ao problema (3.1)-(3.6).

$$
\begin{aligned}
\text{(PLIM):}\quad \text{Minimizar } & z = 3u_{123} - 2u_1 - 5u_{12} - 3u_3 - 4u_{35} + 4(7\gamma_1 + 16\gamma_2 + 28\gamma_3), && (3.13)\\
\text{sujeito a: } & 7x_1 + 4x_2 + 2u_2 \leq 40, && (3.14)\\
& 3u_{12} + 2u_{13} + 4u_{23} \geq 2, && (3.15)\\
& u_{15} + u_{24} + 7\gamma_1 + 16\gamma_2 + 28\gamma_3 \geq 30, && (3.16)\\
& x_5 - x_4 \geq -15(1-\lambda), && (3.17)\\
& x_5 - x_4 \leq 15\lambda, && (3.18)\\
& u_1 \geq -15(1-\lambda) + x_4, && (3.19)\\
& u_1 \leq x_4 + 15(1-\lambda), && (3.20)\\
& u_1 \geq -15\lambda + x_5, && (3.21)\\
& u_1 \leq x_5 + 15\lambda, && (3.22)\\
& u_2 \geq -15(1-\lambda) + x_5, && (3.23)\\
& u_2 \leq x_5 + 15(1-\lambda), && (3.24)\\
& u_2 \geq -15\lambda + x_4, && (3.25)\\
& u_2 \leq x_4 + 15\lambda, && (3.26)\\
& x_4 - x_5 \geq -15(1-\beta), && (3.27)\\
& x_4 - x_5 \leq 15\beta, && (3.28)\\
& u_3 \geq -30(1-\beta) + (x_4 - x_5), && (3.29)\\
& u_3 \leq (x_4 - x_5) + 30(1-\beta), && (3.30)\\
& u_3 \geq -30\beta + (x_5 - x_4), && (3.31)\\
& u_3 \leq (x_5 - x_4) + 30\beta, && (3.32)\\
& x_1 \geq u_{123}, && (3.33)\\
& x_2 \geq u_{123}, && (3.34)\\
& x_3 \geq u_{123}, && (3.35)\\
& x_1 + x_2 + x_3 - 2 \leq u_{123}, && (3.36)\\
& x_1 \geq u_{12}, && (3.37)\\
& x_2 \geq u_{12}, && (3.38)\\
& x_1 + x_2 - 1 \leq u_{12}, && (3.39)\\
& x_1 \geq u_{13}, && (3.40)\\
& x_3 \geq u_{13}, && (3.41)\\
& x_1 + x_3 - 1 \leq u_{13}, && (3.42)\\
& x_2 \geq u_{23}, && (3.43)\\
& x_3 \geq u_{23}, && (3.44)\\
& x_2 + x_3 - 1 \leq u_{23}, && (3.45)\\
& 15x_3 \geq u_{35}, && (3.46)\\
& -15(1-x_3) + x_5 \leq u_{35}, && (3.47)\\
& x_5 + 15(1-x_3) \geq u_{35}. && (3.48)\\
& 15x_1 \geq u_{15}, && (3.49)\\
& -15(1-x_1) + x_5 \leq u_{15}. && (3.50)\\
& x_5 + 15(1-x_1) \geq u_{15}, && (3.51)\\
& 15x_2 \geq u_{24}, && (3.52)\\
& -15(1-x_2) + x_4 \leq u_{24}, && (3.53)\\
& x_4 + 15(1-x_2) \geq u_{24}, && (3.54)\\
& \gamma_1 + \gamma_2 + \gamma_3 \leq 1, && (3.55)\\
& u_1,\ u_2,\ u_3,\ u_{123},\ u_{12},\ u_{13},\ u_{15},\ u_{23},\ u_{24},\ u_{35} \geq 0, && \\
& x_j \in \{0,1\}, \quad j = 1,2,3, \quad 0 \leq x_j \leq 15, \quad j = 4,5, && \\
& \lambda \in \{0,1\},\ \beta \in \{0,1\},\ \gamma_i \in \{0,1\}, \quad i = 1,2,3. && (3.56)
\end{aligned}
$$

Parece até mágica quando se passa de um problema de otimização (P) com algumas variáveis restritas a serem inteiras e outras reais não negativas para o problema (PLIM). Ao se utilizarem novas variáveis 0-1 (α, β, γ) e técnicas de modelagem matemática associadas à otimização combinatória, transforma-se o problema (P) no problema (PLIM) de otimização linear inteira mista (algumas variáveis 0-1 e outras contínuas não negativas), que pode ser resolvido utilizando-se técnicas poliédricas e de enumeração implícita conhecidas e implementadas nos melhores softwares de otimização linear mista (XPRESS, CPLEX, GUROBI, etc.). No caso de (PLIM), foi utilizada a linguagem de modelagem matemática MOSEL associada ao XPRESS [219], apresentada abaixo.

```
model ModelName
uses "mmxprs";
declarations
u123, u1, u2, u3, u12, u13, u23, u35, u15, u24, x6:mpvar
x:array(1..5) of mpvar
lambda, beta:mpvar
gama:array(1..3) of mpvar
Objective:linctr
end-declarations
forall(i in 1..3) x(i) is_binary
forall(i in 1..3) gama(i) is_binary
lambda is_binary
beta is_binary
 7*x(1)+4*x(2)+2*u2 <= 40
3*u12+2*u13+4*u23 >= 2
u15 + u24 + 7*gama(1) + 16*gama(2) + 28*gama(3) >= 30
-15*(1-lambda) <= x(5)-x(4)
x(5)-x(4) <= 15*lambda
-15*(1-lambda)+x(4) <= u1
u1 <= x(4)+15*(1-lambda)
-15*lambda+x(5) <= u1
u1 <= x(5)+15*lambda
-15*(1-lambda)+x(5) <= u2
u2 <= x(5)+15*(1-lambda)
-15*lambda+x(4) <= u2
u2 <= x(4)+15*lambda
-15*(1-beta) <= x(4)-x(5)
x(4)-x(5) <= 15*beta
-30*(1-beta)+(x(4)-x(5)) <= u3
u3 <= (x(4)-x(5))+30*(1-beta)
-30*beta+(x(5)-x(4)) <= u3
u3 <= (x(5)-x(4))+30*beta
u123 <= x(1)
u123 <= x(2)
u123 <= x(3)
x(1)+x(2)+x(3)-2 <= u123
u12 <= x(1)
u12 <= x(2)
x(1)+x(2)-1 <= u12
u13 <= x(1)
```

```
u13 <= x(3)
x(1)+x(3)-1 <= u13
u23 <= x(2)
u23 <= x(3)
u35 <= 15*x(3)
-15*(1-x(3))+x(5) <= u35
u35 <= x(5)+15*(1-x(3))
x(2)+x(3)-1 <= u23
15*x(1) >= u15
-15*(1-x(1))+x(5) <= u15
x(5)+15*(1-x(1)) >= u15
15*x(2) >= u24
-15*(1-x(2))+x(4) <= u24
x(4)+15*(1-x(2)) >= u24
gama(1)+gama(2)+gama(3) <= 1
x6=7*gama(1) + 16*gama(2) + 28*gama(3)
x(4) <= 15
x(5) <= 15
obj:=3*u123-2*u1-5*u12-3*u3-4*u35+4*(7*gama(1) + 16*gama(2) + 28*gama(3))
minimize(obj)
end-model
```

A solução para o problema (3.13) - (3.56) é:
$x_1 = x_2 = x_3 = u_{12} = u_{13} = u_{23} = u_{123} = \lambda = \gamma_1 = 1,$
$x_4 = u_{24} = 8.5, \quad x_5 = u_{15} = u_{35} = 14.5, \quad u_3 = 6, \quad x_6 = 7\gamma_1 = 7 \times 1 = 7.$
$\beta = \gamma_2 = \gamma_3 = 0$
F.O. = -67.

3.3 Utilização de variáveis $0-1$ na solução de equações diofantinas

Exemplo 2.
Desejamos encontrar soluções para a seguinte equação:

$$80k^2 - 66k - 5q^2 + 3q + 9120 = 0,$$

onde $k \in \mathbb{Z}_+,\ 0 \leq k \leq 255 = 2^8 - 1$ e $q \in \mathbb{Z}_+,\ 0 \leq q \leq 1023 = 2^{10} - 1$.

Definimos $v_j \in \{0,1\}$ para $j = 0,1,2,\ldots,7$ e $w_j \in \{0,1\}$ para $j = 0,1,2,\ldots,9$.
Assim, podemos escrever:

$$k = v_0 + 2v_1 + 4v_2 + 8v_3 + 16v_4 + 32v_5 + 64v_6 + 128v_7$$

$k^2 = v_0 + 4v_1 + 16v_2 + 64v_3 + 256v_4 + 1024v_5 + 4096v_6 + 16384v_7 + 4v_0v_1 + 8v_0v_2 + 16v_0v_3 + 32v_0v_4 + 64v_0v_5 + 128v_0v_6 + 256v_0v_7 + 16v_1v_2 + 32v_1v_3 +$

$64v_1v_4+128v_1v_5+256v_1v_6+512v_1v_7+64v_2v_3+128v_2v_4+256v_2v_5+512v_2v_6+$
$1024v_2v_7+256v_3v_4+512v_3v_5+1024v_3v_6+2048v_3v_7+1024v_4v_5+2048v_4v_6+$
$4096v_4v_7+4096v_5v_6+8192v_5v_7+16384v_6v_7$

$q=w_0+2w_1+4w_2+8w_3+16w_4+32w_5+64w_6+128w_7+256w_8+512w_9$

$q^2=w_0+4w_1+16w_2+64w_3+256w_4+1024w_5+4096w_6+16384w_7+65536w_8+$
$262144w_9+4w_0w_1+8w_0w_2+16w_0w_3+32w_0w_4+64w_0w_5+128w_0w_6+256w_0w_7+$
$512w_0w_8+1024w_0w_9+16w_1w_2+32w_1w_3+64w_1w_4+128w_1w_5+256w_1w_6+$
$512w_1w_7+1024w_1w_8+2048w_1w_9+64w_2w_3+128w_2w_4+256w_2w_5+512w_2w_6+$
$1024w_2w_7+2048w_2w_8+4096w_2w_9+256w_3w_4+512w_3w_5+1024w_3w_6+2048w_3w_7+$
$4096w_3w_8+8192w_3w_9+1024w_4w_5+2048w_4w_6+4096w_4w_7+8192w_4w_8+$
$16384w_4w_9+4096w_5w_6+8192w_5w_7+16384w_5w_8+32768w_5w_9+16384w_6w_7+$
$32768w_6w_8+65536w_6w_9+65536w_7w_8+131072w_7w_9+262144w_8w_9.$

Fazemos $v_{ij}=v_iv_j$, com $v_{ij}\geq 0$, e usamos os passos da linearização para colocarmos o modelo todo com variáveis binárias, ou seja:
$v_{ij}\leq v_i$
$v_{ij}\leq v_j$
$v_{ij}\geq v_i+v_j-1$
$0\leq i\leq j\leq 7.$
De forma análoga, fazemos $w_{ij}=w_iw_j$, com $w_{ij}\geq 0$, e usamos os passos da linearização para colocarmos o modelo todo com variáveis binárias, ou seja:
$w_{ij}\leq w_i$
$w_{ij}\leq w_j$
$w_{ij}\geq w_i+w_j-1$
$0\leq i\leq j\leq 9.$
Fazendo isso, teremos no modelo as variáveis
$v_i\in\{0,1\},\ i=0,\ldots,7,$
$w_i\in\{0,1\},\ i=0,\ldots,9,$
$v_{ij}\in[0,1],\ 0\leq i\leq j\leq 7$ e
$w_{ij}\in[0,1], 0\leq i\leq j\leq 9.$
Esse modelo escrito na linguagem Xpress-Mosel é apresentado abaixo.

```
model ModelName
uses "mmxprs";
declarations
I1 = 0..7
I2 = 0..9
v:array(I1) of mpvar
w:array(I2) of mpvar
V: array(I1,I1)of mpvar
W: array(I2,I2) of mpvar
k:mpvar
k2:mpvar
q:mpvar
q2:mpvar
end-declarations
forall (i in I1) v(i) is_binary
```

```
forall (i in I2) w(i) is_binary
k is_integer
q is_integer
k >= 0
k <= 255
q >= 0
q <= 1023
forall(i in I1, j in I1 | i<j) create(V(i,j))
forall(i in I2, j in I2 | i<j) create(W(i,j))
k=sum (i in I1)2^i*v(i)
q=sum (i in I2)2^i*w(i)
forall(i in I1, j in I1 | i < j) do
V(i,j)<=v(i)
V(i,j)<=v(j)
V(i,j)>=v(i)+v(j)-1
end-do
forall(i in I2, j in I2 | i < j) do
W(i,j)<=w(i)
W(i,j)<=w(j)
W(i,j)>=w(i)+w(j)-1
end-do
k2=sum (i in I1)4^i*v(i) + sum(i in I1, j in I1 | i<j)2^(i+j+1)*V(i,j)
q2=sum (i in I2)4^i*w(i) + sum(i in I2, j in I2 | i<j)2^(i+j+1)*W(i,j)
80*k2-66*k-5*q2+3*q+9120=0
Objective:=k+q
maximize(Objective)
end-model
```

A solução para esse exemplo é: $k = 97$, $q = 389$, $k^2 = 9409$, $q^2 = 151321$.
O valor das variáveis binárias v e w é:
$v(0) = v(5) = v(6) = w(0) = w(2) = w(7) = w(8) = 1$
$v(1) = v(2) = v(3) = v(4) = v(7) = w(1) = w(3) = w(4) = w(5) = w(6) = w(9) = 0$.

As soluções para as variáveis $V(i, j)$ são:
$V(0,5) = V(0,6) = V(5,6) = 1$ e todas as outras com valor 0.

As soluções para as variáveis $W(i, j)$ são:
$W(0,2) = W(0,7) = W(0,8) = W(2,7) = W(2,8) = W(7,8) = 1$ e todas as outras com valor 0.

3.4 Transformando problemas de programação linear inteira em problemas de programação linear 0-1

Uma grande quantidade de problemas de otimização combinatória é modelada como problemas de programação linear 0-1. Alguns exemplos de problemas modelados como 0-1 são: Caixeiro Viajante, Set Partitioning, Set Covering, Set Packing, Safe Set e Árvores de Steiner. Vários problemas com aplicações na indústria também são modelados como problemas 0-1 e alguns exemplos são: Facility Location, Fixed Charge Cost, Scheduling e Network Design.

É importante observar que podem existir diferentes formulações para o mesmo problema, porém uma boa formulação $0-1$ é de extrema importância para que o problema seja resolvido.

Supondo que as variáveis inteiras em um problema de programação linear inteira sejam limitadas em valor, é possível transformá-lo em um problema de programação linear 0-1. Ver também [153].

Exemplo 3.
Dado o seguinte Problema da Mochila:

$$
\begin{aligned}
(PM): \quad & \text{Maximizar } z = 16x_1 + 24x_2 + 7x_3 + 2x_4, && (3.57)\\
& \text{sujeito a: } 4x_1 + 7x_2 + 3x_3 + x_4 \leq 27, && (3.58)\\
& x_j \geq 0, \quad j = 1,2,3,4; && \\
& x_j \ \text{ inteiro}, \quad j = 1,2,3,4. && (3.59)
\end{aligned}
$$

Pode-se observar que a restrição da mochila impõe os seguintes limites sobre as variáveis: $0 \leq x_1 \leq 6$, $0 \leq x_2 \leq 3$, $0 \leq x_3 \leq 9$ e $0 \leq x_4 \leq 27$. Para a resolução de (PM), uma ideia é considerar

$$x_1 = \sum_{i=1}^{6} \lambda_i, \quad \lambda_i \in \{0,1\}, \quad i = 1,\ 2, \ldots,\ 6.$$

A mesma ideia é aplicada para x_2,x_3 e x_4. Desta forma, são utilizadas 6+3+9+27=45 variáveis binárias no modelo do problema linear 0-1 equivalente a (PM).

Agora suponha $x_k \in \mathbb{Z}$, $p_k \in \mathbb{Z}_+$, $0 \leq x_k \leq p_k$, uma vez que o valor de p varia de acordo com o valor máximo que cada variável x pode assumir. Uma outra ideia consiste em encontrar $q_k \in \mathbb{Z}_+$, tal que $2^{q_k+1} \leq p_k < 2^{q_k+2}$ e executar o algoritmo 3.1, de forma a reescrever cada variável $x_k \in \mathbb{Z}$, $k = 1 \ldots n$ em função de variáveis λ_m, onde $\lambda_m \in \{0,1\}$. Utilizando

Algoritmo 3.1: Variáveis para formulação $0-1$

Entrada: n, p_k, q_k
Saída: Variáveis de PM reescritas $0-1$ utilizando menor quantidade de variáveis

```
início
    m = 1
    para k = 1...n faça
        x_k = 0
        para i = 0...q_k faça
            x_k = x_k + 2^i λ_m
            m = m + 1
        fim
        x_k = x_k + [p_k − (2^(q+1) − 1)]λ_m
        m = m + 1
    fim
fim
```

o algoritmo 3.1, as variáveis $x_1,...,x_4$ serão reescritas como:

$$\begin{aligned} x_1 &= \lambda_1 + 2\lambda_2 + 3\lambda_3 \\ x_2 &= \lambda_4 + 2\lambda_5 \\ x_3 &= \lambda_6 + 2\lambda_7 + 4\lambda_8 + 2\lambda_9 \\ x_4 &= \lambda_{10} + 2\lambda_{11} + 4\lambda_{12} + 8\lambda_{13} + 12\lambda_{14} \\ \lambda_m &\in \{0,1\}, \quad m = 1,2,...,14. \end{aligned}$$

Pode-se então escrever o modelo do problema linear $0-1$ equivalente ao Problema da Mochila com quatorze variáveis binárias, como apresentado a seguir.

$$\begin{aligned} (PM-BIN): \quad \text{Maximizar } & z_0 = 16\lambda_1 + 32\lambda_2 + 48\lambda_3 + 24\lambda_4 + 48\lambda_5 + 7\lambda_6 + 14\lambda_7 \\ & + 28\lambda_8 + 14\lambda_9 + 2\lambda_{10} + 4\lambda_{11} + 8\lambda_{12} + 16\lambda_{13} + 24\lambda_{14}, \quad (3.60) \\ \text{sujeito a: } & 4\lambda_1 + 8\lambda_2 + 12\lambda_3 + 7\lambda_4 + 14\lambda_5 + 3\lambda_6 + 6\lambda_7 + 12\lambda_8 \\ & + 6\lambda_9 + \lambda_{10} + 2\lambda_{11} + 4\lambda_{12} + 8\lambda_{13} + 12\lambda_{14} \leq 27, \quad (3.61) \\ & \lambda_m \in \{0,1\}, \quad m = 1,\ldots,14. \quad (3.62) \end{aligned}$$

O modelo (3.60)-(3.62) é equivalente ao modelo (3.57)-(3.59) e utiliza o menor número possível de variáveis binárias. Segue abaixo o modelo na linguagem Xpress-Mosel.

```
model ModelName
uses "mmxprs";
declarations
```

```
var = 1..14
lambda: array(var) of mpvar
constr:array(var)  of integer
obj_func:array(var) of integer
end-declarations
obj_func::[16, 32, 48, 24, 48, 7, 14, 28, 14, 2, 4, 8, 16, 24]
constr::[4, 8, 12, 7, 14, 3, 6, 12, 6, 1, 2, 4, 8, 12]
forall(i in var) lambda(i) is_binary
sum(i in var) constr(i)*lambda(i)<=27
mochila:=sum(i in var) obj_func(i)*lambda(i)
maximize(mochila)
end-model
```

A solução encontrada é $x_2 = x_3 = x_4 = 1$, $x_1 = x_5 = x_6 = x_7 = x_8 = x_9 = x_{10} = x_{11} = x_{12} = x_{13} = x_{14} = 0$. Em algumas aplicações, a variável x pode ter que assumir valores discretos ou contínuos, por exemplo: $x \in X = \{2\} \cup [5,7] \cup \{10\} \cup [15,+\infty)$. Por razões práticas, supomos que x seja limitado por um valor suficientemente grande representado por M, que será incluído no modelo. Desta forma, a variável x pode ser escrita como uma função de variáveis binárias:

$$2\lambda_1 + 5\lambda_2 + 10\lambda_3 + 15\lambda_4 \leq x \leq 2\lambda_1 + 7\lambda_2 + 10\lambda_3 + M\lambda_4,$$
$$\sum_{i=1}^{4} \lambda_i = 1, \quad \lambda_i \in \{0,1\}, \quad i = 1,...,4.$$

Se $\lambda_1 = 1$, significa que $2 \leq x \leq 2$. Se $\lambda_2 = 1$, tem-se que $5 \leq x \leq 7$. Da mesma forma, se $\lambda_3 = 1$, tem-se que $10 \leq x \leq 10$, e, por fim, se $\lambda_4 = 1$, então $15 \leq x \leq M$.

Um grande número de problemas de otimização pode ser reescrito com a utilização de variáveis binárias, portanto essas formulações são uma ferramenta poderosa. Infelizmente não há certeza de que todas as formulações com variáveis binárias podem ser resolvidas em um tempo computacional viável para a sua aplicação, mesmo com o uso de um computador potente, pois o número de soluções binárias (viáveis ou inviáveis) pode ser enorme. Para exemplificar, suponha um problema de programação linear 0-1 com cem variáveis binárias. Nesse contexto, o número de soluções viáveis pode ser $O(2^{100})$. O leitor pode averiguar quanto tempo seria necessário para que um computador potente inspecionasse a viabilidade destas 2^{100} soluções de um problema de programação linear 0-1. Felizmente grandes problemas de programação linear 0-1 podem ser resolvidos com a utilização do branch-and-bound clássico proposto por softwares comerciais, como Xpress, Cplex e Gurobi, para a resolução de problemas lineares inteiros mistos. Problemas lineares inteiros mistos com 150 mil variáveis, sendo 2 mil binárias e 130 mil contínuas e 45 mil restrições lineares já foram resolvidos, veja em [59].

3.5 Uma aplicação relacionada à formulação de um problema de scheduling de processadores não relacionados com restrições de precedência

Esse problema, com maiores detalhes, foi publicado em [156]. Seja $T = \{t_1, \ldots, t_n\}$ um conjunto de tarefas parcialmente ordenadas e $P = \{p_1, \ldots, p_n\}$ um conjunto de processadores. Também é dado um grafo de precedência $G = (T, A)$ sem circuitos e direcionado, associado ao conjunto de tarefas $(t_i, t_j) \in A$ se, e somente se, a tarefa t_i dever ser executada imediatamente antes da tarefa t_j. Cada tarefa deve ser atribuída a exatamente um processador, no qual é totalmente executada sem interrupção. Para cada tarefa $t_j \in T$ e para cada processador $p_k \in P$, denota-se por d_{jk} o tempo total de processamento da tarefa t_j atribuída ao processador p_k. O problema de sequenciamento (sheduling) de tarefas com restrições de precedência consiste em encontrar uma atribuição das tarefeas em T aos processadores em P, tal que o tempo total de processamento seja mínimo (makespan). Esse problema é NP-difícil [88].
Para facilitar a notação, são definidos os seguintes conjuntos: $N = \{1, 2, ..., n\}$ associado a T e $M = \{1, ..., m\}$ associado a P. Seja $\Gamma(j) = \{i \in N \mid (t_i, t_j) \in A\}$, isto é, o conjunto dos índices das tarefas que antecedem imediatamente a tarefa t_j. Caso não haja nenhuma tarefa antecedendo a tarefa t_j, teremos $\Gamma(j) = \emptyset$. Pode-se supor que cada processador trabalhe em turnos, e que o número de turnos é menor ou igual ao número total de tarefas, podendo ser zero, isto é, nunca ser utilizado na solução.

Definição das variáveis:
$y^s_{jk} \in \{0, 1\}$, isto é, $y^s_{jk} = 1$, se a tarefa t_j é realizada no processador p_k, durante o seu turno s, e igual a 0 caso contrário.
Para cada tarefa t_j, define-se uma variável $wj \geq 0$ que fornece o tempo de início de sua execução.
O problema pode ser formulado da seguinte maneira:

$$\text{(Scheduling):} \quad \text{Minimizar } z, \tag{3.63}$$

sujeito a:

$$\sum_{k=1}^{m} \sum_{s=1}^{n} y^s_{jk} = 1, \ j \in N, \tag{3.64}$$

$$\sum_{j=1}^{n} y_{jk}^{1} \leq 1, \ k \in M, \tag{3.65}$$

$$\sum_{j=1}^{n} y_{jk}^{s} \leq \sum_{j=1}^{n} y_{jk}^{s-1}, \ k \in M, \ s = 2,3,...,n, \tag{3.66}$$

$$w_j \geq w_i + \sum_{k=1}^{m} \sum_{s=1}^{n} d_{ik} y_{ik}^{s}, \ i \in \Gamma(j), \ j \in N, \tag{3.67}$$

$$w_j \geq w_i + d_{ik} - \alpha[2 - (y_{ik}^{s} + \sum_{r=s+1}^{n} y_{jk}^{r})], k \in M, \ s = 1,2,...,n-1, \ i \neq j, (i,j) \in N \times N, \tag{3.68}$$

$$z \geq w_j + \sum_{k=1}^{m} \sum_{s=1}^{n} d_{jk} y_{jk}^{s}, \ j \in N, \tag{3.69}$$

$$y_{ij}^{s} \in \{0,1\}, \ j \in N, \ k \in M, \ s = 1,2,...,n; \ w_j \geq 0, \ j \in N, \tag{3.70}$$

em que α é uma cota superior para o valor ótimo de z; por exemplo, basta tomarmos α igual à soma dos n maiores valores de d_{ik}.

O modelo de otimização linear misto (variáveis inteiras e contínuas) acima fornece, caso consiga ser resolvido num tempo razoável, o problema de sequenciamento aqui apresentado. As equações (3.64) garantem que cada tarefa será realizada em um único processador. As restrições (3.65) e (3.66) servem para dizer que cada processador não pode realizar simultaneamente duas tarefas. As primeiras (3.65) significam que em cada processador haverá no máximo uma primeira tarefa a ser executada; as segundas (3.66) significam que, se uma tarefa for realizada no turno s ($s \geq 2$) do processador, então este processador deverá ter realizado uma outra tarefa no turno $s-1$. As desigualdades (3.67) representam as condições de precedência: uma tarefa só poderá ser executada quando todas aquelas que a antecedem o forem. Quando $(i,j) \notin A$, ou $(j,i) \notin A$, então as restrições (3.68) serão consideradas: expressam o fato de que a tarefa t_j deve começar pelo menos após d_{ik} unidades de tempo após t_i, no processador p_k , isto é, $y_{ik}^{s} = \sum_{r=s+1}^{n} y_{jk}^{r} = 1$, para algum $s = 1,2....,n-1$.

No exemplo abaixo é apresentado um modelo de scheduling composto por cinco tarefas e dois processadores. A ordem de precedência das tarefas é dada pela figura 3.1. Os valores das durações das tarefas em cada processador são dados a seguir: $d_{11} = 3$, $d_{12} = 2$, $d_{21} = 1$, $d_{22} = 4$, $d_{31} = 5$, $d_{32} = 3$, $d_{41} = 2$, $d_{42} = 3$, $d_{51} = 4$, $d_{52} = 5$. O valor 21 será atribuído à constante α.

Exemplo 4.

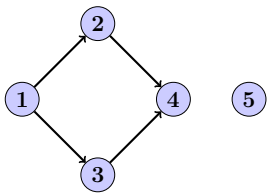


Figura 3.1: Grafo de precedências.

$$\text{(Scheduling):} \quad \text{Minimizar } z, \tag{3.71}$$

sujeito a:

$$y_{11}^1+y_{11}^2+y_{11}^3+y_{11}^4+y_{11}^5+y_{12}^1+y_{12}^2+y_{12}^3+y_{12}^4+y_{12}^5=1, \tag{3.72}$$
$$y_{21}^1+y_{21}^2+y_{21}^3+y_{21}^4+y_{21}^5+y_{22}^1+y_{22}^2+y_{22}^3+y_{22}^4+y_{22}^5=1, \tag{3.73}$$
$$y_{31}^1+y_{31}^2+y_{31}^3+y_{31}^4+y_{31}^5+y_{32}^1+y_{32}^2+y_{32}^3+y_{32}^4+y_{32}^5=1, \tag{3.74}$$
$$y_{41}^1+y_{41}^2+y_{41}^3+y_{41}^4+y_{41}^5+y_{42}^1+y_{42}^2+y_{42}^3+y_{42}^4+y_{42}^5=1, \tag{3.75}$$
$$y_{51}^1+y_{51}^2+y_{51}^3+y_{51}^4+y_{51}^5+y_{52}^1+y_{52}^2+y_{52}^3+y_{52}^4+y_{52}^5=1, \tag{3.76}$$
$$y_{11}^1+y_{21}^1+y_{31}^1+y_{41}^1+y_{51}^1, \leq 1 \tag{3.77}$$
$$y_{12}^1+y_{22}^1+y_{32}^1+y_{42}^1+y_{52}^1, \leq 1 \tag{3.78}$$
$$y_{11}^2+y_{21}^2+y_{31}^2+y_{41}^2+y_{51}^2, \leq 1 \tag{3.79}$$
$$y_{12}^2+y_{22}^2+y_{32}^2+y_{42}^2+y_{52}^2, \leq 1 \tag{3.80}$$
$$y_{11}^3+y_{21}^3+y_{31}^3+y_{41}^3+y_{51}^3, \leq 1 \tag{3.81}$$
$$y_{12}^3+y_{22}^3+y_{32}^3+y_{42}^3+y_{52}^3, \leq 1 \tag{3.82}$$
$$y_{11}^4+y_{21}^4+y_{31}^4+y_{41}^4+y_{51}^4, \leq 1 \tag{3.83}$$
$$y_{12}^4+y_{22}^4+y_{32}^4+y_{42}^4+y_{52}^4, \leq 1 \tag{3.84}$$
$$y_{11}^5+y_{21}^5+y_{31}^5+y_{41}^5+y_{51}^5, \leq 1 \tag{3.85}$$
$$y_{12}^5+y_{22}^5+y_{32}^5+y_{42}^5+y_{52}^5, \leq 1 \tag{3.86}$$
$$y_{11}^2+y_{21}^2+y_{31}^2+y_{41}^2+y_{51}^2 \leq y_{11}^1+y_{21}^1+y_{31}^1+y_{41}^1+y_{51}^1, \tag{3.87}$$
$$y_{11}^3+y_{21}^3+y_{31}^3+y_{41}^3+y_{51}^3 \leq y_{11}^2+y_{21}^2+y_{31}^2+y_{41}^2+y_{51}^2, \tag{3.88}$$
$$y_{11}^4+y_{21}^4+y_{31}^4+y_{41}^4+y_{51}^4 \leq y_{11}^3+y_{21}^3+y_{31}^3+y_{41}^3+y_{51}^3, \tag{3.89}$$

$$y_{11}^5+y_{21}^5+y_{31}^5+y_{41}^5+y_{51}^5 \leq y_{11}^4+y_{21}^4+y_{31}^4+y_{41}^4+y_{51}^4, \tag{3.90}$$
$$y_{12}^2+y_{22}^2+y_{32}^2+y_{42}^2+y_{52}^2 \leq y_{12}^1+y_{22}^1+y_{32}^1+y_{42}^1+y_{52}^1, \tag{3.91}$$
$$y_{12}^3+y_{22}^3+y_{32}^3+y_{42}^3+y_{52}^3 \leq y_{12}^2+y_{22}^2+y_{32}^2+y_{42}^2+y_{52}^2, \tag{3.92}$$
$$y_{12}^4+y_{22}^4+y_{32}^4+y_{42}^4+y_{52}^4 \leq y_{12}^3+y_{21}^3+y_{31}^3+y_{41}^3+y_{51}^3, \tag{3.93}$$
$$y_{12}^5+y_{22}^5+y_{32}^5+y_{42}^5+y_{52}^5 \leq y_{12}^4+y_{22}^4+y_{32}^4+y_{42}^4+y_{52}^4, \tag{3.94}$$
$$w_2 \geq w_1+3y_{11}^1+3y_{11}^2+3y_{11}^3+3y_{11}^4+2y_{12}^1+2y_{12}^2+2y_{12}^3+2y_{12}^4, \tag{3.95}$$
$$w_3 \geq w_1+3y_{11}^1+3y_{11}^2+3y_{11}^3+3y_{11}^4+2y_{12}^1+2y_{12}^2+2y_{12}^3+2y_{12}^4, \tag{3.96}$$
$$w_4 \geq w_2+y_{21}^1+y_{21}^2+y_{21}^3+y_{21}^4+4y_{22}^1+4y_{22}^2+4y_{22}^3+4y_{22}^4, \tag{3.97}$$
$$w_4 \geq w_3+5y_{31}^1+5y_{31}^2+5y_{31}^3+5y_{31}^4+3y_{32}^1+3y_{32}^2+3y_{32}^3+3y_{32}^4, \tag{3.98}$$
$$w_2 \geq w_1+3-21[2-(y_{11}^1+y_{21}^2+y_{21}^3+y_{21}^4+y_{21}^5)], \tag{3.99}$$
$$w_2 \geq w_1+3-21[2-(y_{11}^2+y_{21}^3+y_{21}^4+y_{21}^5)], \tag{3.100}$$
$$w_2 \geq w_1+3-21[2-(y_{11}^3+y_{21}^4+y_{21}^5)], \tag{3.101}$$
$$w_2 \geq w_1+3-21[2-(y_{11}^4+y_{21}^5)], \tag{3.102}$$
$$w_2 \geq w_1+2-21[2-(y_{12}^1+y_{22}^2+y_{22}^3+y_{22}^4+y_{22}^5)], \tag{3.103}$$
$$w_2 \geq w_1+2-21[2-(y_{12}^2+y_{22}^3+y_{22}^4+y_{22}^5)], \tag{3.104}$$
$$w_2 \geq w_1+2-21[2-(y_{12}^3+y_{22}^4+y_{22}^5)], \tag{3.105}$$
$$w_2 \geq w_1+2-21[2-(y_{12}^4+y_{22}^5)], \tag{3.106}$$
$$w_3 \geq w_1+3-21[2-(y_{11}^1+y_{31}^2+y_{31}^3+y_{31}^4+y_{31}^5)], \tag{3.107}$$
$$w_3 \geq w_1+3-21[2-(y_{11}^2+y_{31}^3+y_{31}^4+y_{31}^5)], \tag{3.108}$$
$$w_3 \geq w_1+3-21[2-(y_{11}^3+y_{31}^4+y_{31}^5)], \tag{3.109}$$
$$w_3 \geq w_1+3-21[2-(y_{11}^4+y_{31}^5)], \tag{3.110}$$
$$w_3 \geq w_1+2-21[2-(y_{12}^1+y_{32}^2+y_{32}^3+y_{32}^4+y_{32}^5)], \tag{3.111}$$
$$w_3 \geq w_1+2-21[2-(y_{12}^2+y_{32}^3+y_{32}^4+y_{32}^5)], \tag{3.112}$$
$$w_3 \geq w_1+2-21[2-(y_{12}^3+y_{32}^4+y_{32}^5)], \tag{3.113}$$
$$w_3 \geq w_1+2-21[2-(y_{12}^4+y_{32}^5)], \tag{3.114}$$
$$w_4 \geq w_1+3-21[2-(y_{11}^1+y_{41}^2+y_{41}^3+y_{41}^4+y_{41}^5)], \tag{3.115}$$
$$w_4 \geq w_1+3-21[2-(y_{11}^2+y_{41}^3+y_{41}^4+y_{41}^5)], \tag{3.116}$$
$$w_4 \geq w_1+3-21[2-(y_{11}^3+y_{41}^4+y_{41}^5)], \tag{3.117}$$
$$w_4 \geq w_1+3-21[2-(y_{11}^4+y_{41}^5)], \tag{3.118}$$

$$w_4 \geq w_1 + 2 - 21[2 - (y_{12}^1 + y_{42}^2 + y_{42}^3 + y_{42}^4 + y_{42}^5)], \tag{3.119}$$
$$w_4 \geq w_1 + 2 - 21[2 - (y_{12}^2 + y_{42}^3 + y_{42}^4 + y_{42}^5)], \tag{3.120}$$
$$w_4 \geq w_1 + 2 - 21[2 - (y_{12}^3 + y_{42}^4 + y_{42}^5)], \tag{3.121}$$
$$w_4 \geq w_1 + 2 - 21[2 - (y_{12}^4 + y_{42}^5)], \tag{3.122}$$
$$w_5 \geq w_1 + 4 - 21[2 - (y_{11}^1 + y_{51}^2 + y_{51}^3 + y_{51}^4 + y_{51}^5)], \tag{3.123}$$
$$w_5 \geq w_1 + 4 - 21[2 - (y_{11}^2 + y_{51}^3 + y_{51}^4 + y_{51}^5)], \tag{3.124}$$
$$w_5 \geq w_1 + 4 - 21[2 - (y_{11}^3 + y_{51}^4 + y_{51}^5)], \tag{3.125}$$
$$w_5 \geq w_1 + 4 - 21[2 - (y_{11}^4 + y_{51}^5)], \tag{3.126}$$
$$w_5 \geq w_1 + 5 - 21[2 - (y_{12}^1 + y_{52}^2 + y_{52}^3 + y_{52}^4 + y_{52}^5)], \tag{3.127}$$
$$w_5 \geq w_1 + 5 - 21[2 - (y_{12}^2 + y_{52}^3 + y_{52}^4 + y_{52}^5)], \tag{3.128}$$
$$w_5 \geq w_1 + 5 - 21[2 - (y_{12}^3 + y_{52}^4 + y_{52}^5)], \tag{3.129}$$
$$w_5 \geq w_1 + 5 - 21[2 - (y_{12}^4 + y_{52}^5)], \tag{3.130}$$
$$w_3 \geq w_2 + 1 - 21[(2 - (y_{21}^1 + y_{31}^2 + y_{31}^3 + y_{31}^4 + y_{31}^5)], \tag{3.131}$$
$$w_3 \geq w_2 + 1 - 21[(2 - (y_{21}^2 + y_{31}^3 + y_{31}^4 + y_{31}^5)], \tag{3.132}$$
$$w_3 \geq w_2 + 1 - 21[(2 - (y_{21}^3 + y_{31}^4 + y_{31}^5)], \tag{3.133}$$
$$w_3 \geq w_2 + 1 - 21[(2 - (y_{21}^4 + y_{31}^5)], \tag{3.134}$$
$$w_3 \geq w_2 + 4 - 21[(2 - (y_{22}^1 + y_{32}^2 + y_{32}^3 + y_{32}^4 + y_{32}^5)], \tag{3.135}$$
$$w_3 \geq w_2 + 4 - 21[(2 - (y_{22}^2 + y_{32}^3 + y_{32}^4 + y_{32}^5)], \tag{3.136}$$
$$w_3 \geq w_2 + 4 - 21[(2 - (y_{22}^3 + y_{32}^4 + y_{32}^5)], \tag{3.137}$$
$$w_3 \geq w_2 + 4 - 21[(2 - (y_{22}^4 + y_{32}^5)], \tag{3.138}$$
$$w_4 \geq w_2 + 1 - 21[2 - (y_{21}^1 + y_{41}^2 + y_{41}^3 + y_{41}^4 + y_{41}^5)], \tag{3.139}$$
$$w_4 \geq w_2 + 1 - 21[2 - (y_{21}^2 + y_{41}^3 + y_{41}^4 + y_{41}^5)], \tag{3.140}$$
$$w_4 \geq w_2 + 1 - 21[2 - (y_{21}^3 + y_{41}^4 + y_{41}^5)], \tag{3.141}$$
$$w_4 \geq w_2 + 1 - 21[2 - (y_{21}^4 + y_{41}^5)], \tag{3.142}$$
$$w_4 \geq w_2 + 4 - 21[2 - (y_{22}^1 + y_{42}^2 + y_{42}^3 + y_{42}^4 + y_{42}^5)], \tag{3.143}$$
$$w_4 \geq w_2 + 4 - 21[2 - (y_{22}^2 + y_{42}^3 + y_{42}^4 + y_{42}^5)], \tag{3.144}$$
$$w_4 \geq w_2 + 4 - 21[2 - (y_{22}^3 + y_{42}^4 + y_{42}^5)], \tag{3.145}$$
$$w_4 \geq w_2 + 4 - 21[2 - (y_{22}^4 + y_{42}^5)], \tag{3.146}$$
$$w_5 \geq w_2 + 4 - 21[2 - (y_{21}^1 + y_{51}^2 + y_{51}^3 + y_{51}^4 + y_{51}^5)], \tag{3.147}$$

$$w_5 \geq w_2 + 4 - 21[2 - (y_{21}^2 + y_{51}^3 + y_{51}^4 + y_{51}^5)], \tag{3.148}$$
$$w_5 \geq w_2 + 4 - 21[2 - (y_{21}^3 + y_{51}^4 + y_{51}^5)], \tag{3.149}$$
$$w_5 \geq w_2 + 4 - 21[2 - (y_{21}^4 + y_{51}^5)], \tag{3.150}$$
$$w_5 \geq w_2 + 5 - 21[2 - (y_{22}^1 + y_{52}^2 + y_{52}^3 + y_{52}^4 + y_{52}^5)], \tag{3.151}$$
$$w_5 \geq w_2 + 5 - 21[2 - (y_{22}^2 + y_{52}^3 + y_{52}^4 + y_{52}^5)], \tag{3.152}$$
$$w_5 \geq w_2 + 5 - 21[2 - (y_{22}^3 + y_{52}^4 + y_{52}^5)], \tag{3.153}$$
$$w_5 \geq w_2 + 5 - 21[2 - (y_{22}^4 + y_{52}^5)], \tag{3.154}$$
$$w_4 \geq w_3 + 5 - 21[2 - (y_{31}^1 + y_{41}^2 + y_{41}^3 + y_{41}^4 + y_{41}^5)], \tag{3.155}$$
$$w_4 \geq w_3 + 5 - 21[2 - (y_{31}^2 + y_{41}^3 + y_{41}^4 + y_{41}^5)], \tag{3.156}$$
$$w_4 \geq w_3 + 5 - 21[2 - (y_{31}^3 + y_{41}^4 + y_{41}^5)], \tag{3.157}$$
$$w_4 \geq w_3 + 5 - 21[2 - (y_{31}^4 + y_{41}^5)], \tag{3.158}$$
$$w_4 \geq w_3 + 3 - 21[2 - (y_{32}^1 + y_{42}^2 + y_{42}^3 + y_{42}^4 + y_{42}^5)], \tag{3.159}$$
$$w_4 \geq w_3 + 3 - 21[2 - (y_{32}^2 + y_{42}^3 + y_{42}^4 + y_{42}^5)], \tag{3.160}$$
$$w_4 \geq w_3 + 3 - 21[2 - (y_{32}^3 + y_{42}^4 + y_{42}^5)], \tag{3.161}$$
$$w_4 \geq w_3 + 3 - 21[2 - (y_{32}^4 + y_{42}^5)], \tag{3.162}$$
$$w_5 \geq w_3 + 4 - 21[2 - (y_{31}^1 + y_{51}^2 + y_{51}^3 + y_{51}^4 + y_{51}^5)], \tag{3.163}$$
$$w_5 \geq w_3 + 4 - 21[2 - (y_{31}^2 + y_{51}^3 + y_{51}^4 + y_{51}^5)], \tag{3.164}$$
$$w_5 \geq w_3 + 4 - 21[2 - (y_{31}^3 + y_{51}^4 + y_{51}^5)], \tag{3.165}$$
$$w_5 \geq w_3 + 4 - 21[2 - (y_{31}^4 + y_{51}^5)], \tag{3.166}$$
$$w_5 \geq w_3 + 5 - 21[2 - (y_{32}^1 + y_{52}^2 + y_{52}^3 + y_{52}^4 + y_{52}^5)], \tag{3.167}$$
$$w_5 \geq w_3 + 5 - 21[2 - (y_{32}^2 + y_{52}^3 + y_{52}^4 + y_{52}^5)], \tag{3.168}$$
$$w_5 \geq w_3 + 5 - 21[2 - (y_{32}^3 + y_{52}^4 + y_{52}^5)], \tag{3.169}$$
$$w_5 \geq w_3 + 5 - 21[2 - (y_{32}^4 + y_{52}^5)], \tag{3.170}$$
$$w_5 \geq w_4 + 4 - 21[2 - (y_{41}^1 + y_{51}^2 + y_{51}^3 + y_{51}^4 + y_{51}^5)], \tag{3.171}$$
$$w_5 \geq w_4 + 4 - 21[2 - (y_{41}^2 + y_{51}^3 + y_{51}^4 + y_{51}^5)], \tag{3.172}$$
$$w_5 \geq w_4 + 4 - 21[2 - (y_{41}^3 + y_{51}^4 + y_{51}^5)], \tag{3.173}$$
$$w_5 \geq w_4 + 4 - 21[2 - (y_{41}^4 + y_{51}^5)], \tag{3.174}$$
$$w_5 \geq w_4 + 5 - 21[2 - (y_{42}^1 + y_{52}^2 + y_{52}^3 + y_{52}^4 + y_{52}^5)], \tag{3.175}$$
$$w_5 \geq w_4 + 5 - 21[2 - (y_{42}^2 + y_{52}^3 + y_{52}^4 + y_{52}^5)], \tag{3.176}$$

$$w_5 \geq w_4 + 5 - 21[2 - (y_{42}^3 + y_{52}^4 + y_{52}^5)], \tag{3.177}$$

$$w_5 \geq w_4 + 5 - 21[2 - (y_{42}^4 + y_{52}^5)], \tag{3.178}$$

$$w_1 \geq w_2 + 1 - 21[2 - (y_{21}^1 + y_{11}^2 + y_{11}^3 + y_{11}^4 + y_{11}^5)], \tag{3.179}$$

$$w_1 \geq w_2 + 1 - 21[2 - (y_{21}^2 + y_{11}^3 + y_{11}^4 + y_{11}^5)], \tag{3.180}$$

$$w_1 \geq w_2 + 1 - 21[2 - (y_{21}^3 + y_{11}^4 + y_{11}^5)], \tag{3.181}$$

$$w_1 \geq w_2 + 1 - 21[2 - (y_{21}^4 + y_{11}^5)], \tag{3.182}$$

$$w_1 \geq w_2 + 4 - 21[2 - (y_{22}^1 + y_{12}^2 + y_{12}^3 + y_{12}^4 + y_{12}^5)], \tag{3.183}$$

$$w_1 \geq w_2 + 4 - 21[2 - (y_{22}^2 + y_{12}^3 + y_{12}^4 + y_{12}^5)], \tag{3.184}$$

$$w_1 \geq w_2 + 4 - 21[2 - (y_{22}^3 + y_{12}^4 + y_{12}^5)], \tag{3.185}$$

$$w_1 \geq w_2 + 4 - 21[2 - (y_{22}^4 + y_{12}^5)], \tag{3.186}$$

$$w_1 \geq w_3 + 5 - 21[2 - (y_{31}^1 + y_{11}^2 + y_{11}^3 + y_{11}^4 + y_{11}^5)], \tag{3.187}$$

$$w_1 \geq w_3 + 5 - 21[2 - (y_{31}^2 + y_{11}^3 + y_{11}^4 + y_{11}^5)], \tag{3.188}$$

$$w_1 \geq w_3 + 5 - 21[2 - (y_{31}^3 + y_{11}^4 + y_{11}^5)], \tag{3.189}$$

$$w_1 \geq w_3 + 5 - 21[2 - (y_{31}^4 + y_{11}^5)], \tag{3.190}$$

$$w_1 \geq w_3 + 3 - 21[2 - (y_{32}^1 + y_{12}^2 + y_{12}^3 + y_{12}^4 + y_{12}^5)], \tag{3.191}$$

$$w_1 \geq w_3 + 3 - 21[2 - (y_{32}^2 + y_{12}^3 + y_{12}^4 + y_{12}^5)], \tag{3.192}$$

$$w_1 \geq w_3 + 3 - 16[2 - (y_{32}^3 + y_{12}^4 + y_{12}^5)], \tag{3.193}$$

$$w_1 \geq w_3 + 3 - 16[2 - (y_{32}^4 + y_{12}^5)], \tag{3.194}$$

$$w_1 \geq w_4 + 2 - 21[2 - (y_{41}^1 + y_{11}^2 + y_{11}^3 + y_{11}^4 + y_{11}^5)], \tag{3.195}$$

$$w_1 \geq w_4 + 2 - 21[2 - (y_{41}^2 + y_{11}^3 + y_{11}^4 + y_{11}^5)], \tag{3.196}$$

$$w_1 \geq w_4 + 2 - 21[2 - (y_{41}^3 + y_{11}^4 + y_{11}^5)], \tag{3.197}$$

$$w_1 \geq w_4 + 2 - 21[2 - (y_{41}^4 + y_{11}^5)], \tag{3.198}$$

$$w_1 \geq w_4 + 3 - 16[2 - (y_{42}^1 + y_{12}^2 + y_{12}^3 + y_{12}^4 + y_{12}^5)], \tag{3.199}$$

$$w_1 \geq w_4 + 3 - 21[2 - (y_{42}^2 + y_{12}^3 + y_{12}^4 + y_{12}^5)], \tag{3.200}$$

$$w_1 \geq w_4 + 3 - 21[2 - (y_{42}^3 + y_{12}^4 + y_{12}^5)], \tag{3.201}$$

$$w_1 \geq w_4 + 3 - 21[2 - (y_{42}^4 + y_{12}^5)], \tag{3.202}$$

$$w_1 \geq w_5 + 4 - 21[2 - (y_{51}^1 + y_{11}^2 + y_{11}^3 + y_{11}^4 + y_{11}^5)], \tag{3.203}$$

$$w_1 \geq w_5 + 4 - 21[2 - (y_{51}^2 + y_{11}^3 + y_{11}^4 + y_{11}^5)], \tag{3.204}$$

$$w_1 \geq w_5 + 4 - 21[2 - (y_{51}^3 + y_{11}^4 + y_{11}^5)], \tag{3.205}$$

$$w_1 \geq w_5 + 4 - 21[2 - (y_{51}^4 + y_{11}^5)], \tag{3.206}$$
$$w_1 \geq w_5 + 5 - 21[2 - (y_{52}^1 + y_{12}^2 + y_{12}^3 + y_{12}^4 + y_{12}^5)], \tag{3.207}$$
$$w_1 \geq w_5 + 5 - 21[2 - (y_{52}^2 + y_{12}^3 + y_{12}^4 + y_{12}^5)], \tag{3.208}$$
$$w_1 \geq w_5 + 5 - 21[2 - (y_{52}^3 + y_{12}^4 + y_{12}^5)], \tag{3.209}$$
$$w_1 \geq w_5 + 5 - 21[2 - (y_{52}^4 + y_{12}^5)], \tag{3.210}$$
$$w_2 \geq w_3 + 5 - 21[2 - (y_{31}^1 + y_{21}^2 + y_{21}^3 + y_{21}^4 + y_{21}^5)], \tag{3.211}$$
$$w_2 \geq w_3 + 5 - 21[2 - (y_{31}^2 + y_{21}^3 + y_{21}^4 + y_{21}^5)], \tag{3.212}$$
$$w_2 \geq w_3 + 5 - 21[2 - (y_{31}^3 + y_{21}^4 + y_{21}^5)], \tag{3.213}$$
$$w_2 \geq w_3 + 5 - 21[2 - (y_{31}^4 + y_{21}^5)], \tag{3.214}$$
$$w_2 \geq w_3 + 3 - 21[2 - (y_{32}^1 + y_{22}^2 + y_{22}^3 + y_{22}^4 + y_{22}^5)], \tag{3.215}$$
$$w_2 \geq w_3 + 3 - 21[2 - (y_{32}^2 + y_{22}^3 + y_{22}^4 + y_{22}^5)], \tag{3.216}$$
$$w_2 \geq w_3 + 3 - 21[2 - (y_{32}^3 + y_{22}^4 + y_{22}^5)], \tag{3.217}$$
$$w_2 \geq w_3 + 3 - 21[2 - (y_{32}^4 + y_{22}^5)], \tag{3.218}$$
$$w_2 \geq w_4 + 2 - 21[2 - (y_{41}^1 + y_{21}^2 + y_{21}^3 + y_{21}^4 + y_{21}^5)], \tag{3.219}$$
$$w_2 \geq w_4 + 2 - 21[2 - (y_{41}^2 + y_{21}^3 + y_{21}^4 + y_{21}^5)], \tag{3.220}$$
$$w_2 \geq w_4 + 2 - 21[2 - (y_{41}^3 + y_{21}^4 + y_{21}^5)], \tag{3.221}$$
$$w_2 \geq w_4 + 2 - 21[2 - (y_{41}^4 + y_{21}^5)], \tag{3.222}$$
$$w_2 \geq w_4 + 3 - 21[2 - (y_{42}^1 + y_{22}^2 + y_{22}^3 + y_{22}^4 + y_{22}^5)], \tag{3.223}$$
$$w_2 \geq w_4 + 3 - 21[2 - (y_{42}^2 + y_{22}^3 + y_{22}^4 + y_{22}^5)], \tag{3.224}$$
$$w_2 \geq w_4 + 3 - 21[2 - (y_{42}^3 + y_{22}^4 + y_{22}^5)], \tag{3.225}$$
$$w_2 \geq w_4 + 3 - 21[2 - (y_{42}^4 + y_{22}^5)], \tag{3.226}$$
$$w_2 \geq w_5 + 4 - 21[2 - (y_{51}^1 + y_{21}^2 + y_{21}^3 + y_{21}^4 + y_{21}^5)], \tag{3.227}$$
$$w_2 \geq w_5 + 4 - 21[2 - (y_{51}^2 + y_{21}^3 + y_{21}^4 + y_{21}^5)], \tag{3.228}$$
$$w_2 \geq w_5 + 4 - 21[2 - (y_{51}^3 + y_{21}^4 + y_{21}^5)], \tag{3.229}$$
$$w_2 \geq w_5 + 4 - 21[2 - (y_{51}^4 + y_{21}^5)], \tag{3.230}$$
$$w_2 \geq w_5 + 5 - 21[2 - (y_{52}^1 + y_{22}^2 + y_{22}^3 + y_{22}^4 + y_{22}^5)], \tag{3.231}$$
$$w_2 \geq w_5 + 5 - 21[2 - (y_{52}^2 + y_{22}^3 + y_{22}^4 + y_{22}^5)], \tag{3.232}$$
$$w_2 \geq w_5 + 5 - 21[2 - (y_{52}^3 + y_{22}^4 + y_{22}^5)], \tag{3.233}$$
$$w_2 \geq w_5 + 5 - 21[2 - (y_{52}^4 + y_{22}^5)], \tag{3.234}$$

$$w_3 \geq w_4 + 2 - 21[2 - (y^1_{41} + y^2_{31} + y^3_{31} + y^4_{31} + y^5_{31})], \tag{3.235}$$

$$w_3 \geq w_4 + 2 - 21[2 - (y^2_{41} + y^3_{31} + y^4_{31} + y^5_{31})], \tag{3.236}$$

$$w_3 \geq w_4 + 2 - 21[2 - (y^3_{41} + y^4_{31} + y^5_{31})], \tag{3.237}$$

$$w_3 \geq w_4 + 2 - 21[2 - (y^4_{41} + y^5_{31})], \tag{3.238}$$

$$w_3 \geq w_4 + 3 - 21[2 - (y^1_{42} + y^2_{32} + y^3_{32} + y^4_{32} + y^5_{32})], \tag{3.239}$$

$$w_3 \geq w_4 + 3 - 21[2 - (y^2_{42} + y^3_{32} + y^4_{32} + y^5_{32})], \tag{3.240}$$

$$w_3 \geq w_4 + 3 - 21[2 - (y^3_{42} + y^4_{32} + y^5_{32})], \tag{3.241}$$

$$w_3 \geq w_4 + 3 - 21[2 - (y^4_{42} + y^5_{32})], \tag{3.242}$$

$$w_3 \geq w_5 + 4 - 21[2 - (y^1_{51} + y^2_{31} + y^3_{31} + y^4_{31} + y^5_{31})], \tag{3.243}$$

$$w_3 \geq w_5 + 4 - 21[2 - (y^2_{51} + y^3_{31} + y^4_{31} + y^5_{31})], \tag{3.244}$$

$$w_3 \geq w_5 + 4 - 21[2 - (y^3_{51} + y^4_{31} + y^5_{31})], \tag{3.245}$$

$$w_3 \geq w_5 + 4 - 21[2 - (y^4_{51} + y^5_{31})], \tag{3.246}$$

$$w_3 \geq w_5 + 5 - 21[2 - (y^1_{52} + y^2_{32} + y^3_{32} + y^4_{32} + y^5_{32})], \tag{3.247}$$

$$w_3 \geq w_5 + 5 - 21[2 - (y^2_{52} + y^3_{32} + y^4_{32} + y^5_{32})], \tag{3.248}$$

$$w_3 \geq w_5 + 5 - 21[2 - (y^3_{52} + y^4_{32} + y^5_{32})], \tag{3.249}$$

$$w_3 \geq w_5 + 5 - 21[2 - (y^4_{52} + y^5_{32})], \tag{3.250}$$

$$w_4 \geq w_5 + 4 - 21[2 - (y^1_{51} + y^2_{41} + y^3_{41} + y^4_{41} + y^5_{41})], \tag{3.251}$$

$$w_4 \geq w_5 + 4 - 21[2 - (y^2_{51} + y^3_{41} + y^4_{41} + y^5_{41})], \tag{3.252}$$

$$w_4 \geq w_5 + 4 - 21[2 - (y^3_{51} + y^4_{41} + y^5_{41})], \tag{3.253}$$

$$w_4 \geq w_5 + 4 - 21[2 - (y^4_{51} + y^5_{41})], \tag{3.254}$$

$$w_4 \geq w_5 + 5 - 21[2 - (y^1_{52} + y^2_{42} + y^3_{42} + y^4_{42} + y^5_{42})], \tag{3.255}$$

$$w_4 \geq w_5 + 5 - 21[2 - (y^2_{52} + y^3_{42} + y^4_{42} + y^5_{42})], \tag{3.256}$$

$$w_4 \geq w_5 + 5 - 21[2 - (y^3_{52} + y^4_{42} + y^5_{42})], \tag{3.257}$$

$$w_4 \geq w_5 + 5 - 21[2 - (y^4_{52} + y^5_{42})], \tag{3.258}$$

$$z \geq w_1 + 3y^1_{11} + 3y^2_{11} + 3y^3_{11} + 3y^4_{11} + 3y^5_{11} + 2y^1_{12} + 2y^2_{12} + 2y^3_{12} + 2y^4_{12} + 2y^5_{12}, \tag{3.259}$$

$$z \geq w_2 + y^1_{21} + y^2_{21} + y^3_{21} + y^4_{21} + y^5_{21} + 4y^1_{22} + 4y^2_{22} + 4y^3_{22} + 4y^4_{22} + 4y^5_{22}, \tag{3.260}$$

$$z \geq w_3 + 5y^1_{31} + 5y^2_{31} + 5y^3_{31} + 5y^4_{31} + 5y^5_{31} + 3y^1_{32} + 3y^2_{32} + 3y^3_{32} + 3y^4_{32} + 3y^5_{32}, \tag{3.261}$$

$$z \geq w_4 + 2y_{41}^1 + 2y_{41}^2 + 2y_{41}^3 + 2y_{41}^4 + 2y_{41}^5 + 3y_{42}^1 + 3y_{42}^2 + 3y_{42}^3 + 3y_{42}^4 + 3y_{42}^5, \tag{3.262}$$

$$z \geq w_5 + 4y_{51}^1 + 4y_{51}^2 + 4y_{51}^3 + 4y_{51}^4 + 4y_{51}^5 + 5y_{52}^1 + 5y_{52}^2 + 5y_{52}^3 + 5y_{52}^4 + 5y_{52}^5. \tag{3.263}$$

A seguir, o modelo relativo ao exemplo na linguagem Xpress-Mosel é apresentado.

```
model ModelName
uses "mmxprs";
declarations
ntarefa=5
nturno=5
tarefa = 1..ntarefa
processador = 1..2
turno = 1..nturno
nos = range
arcos: dynamic array(nos,nos) of integer
d: array(tarefa,processador) of real
w: array(tarefa) of mpvar
y: array(tarefa, processador,turno) of mpvar
z: mpvar
end-declarations
initializations from 'scheduling.txt'
  arcos  d
 end-initializations
forall (j in tarefa) forall(k in processador)forall (s in turno) y(j,k,s) is_binary
forall (j in tarefa) do
sum(s in turno,k in processador)y(j,k,s)=1
end-do
forall(k in processador, l in turno) do
sum(j in tarefa) y(j,k,l) <=1
end-do
forall(k in processador, s in 2..nturno) do
sum(j in tarefa)y(j,k,s) <= sum(j in tarefa) y(j,k,s-1)
end-do
forall(i,j in tarefa | exists(arcos(i,j))) do
 w(j) >= w(i)+ sum(k in processador)sum(s in turno)d(i,k)*y(i,k,s)
end-do
forall (i in tarefa)do
alfa+=max(j in processador) d(i,j)
end-do
forall(i,j in tarefa| i<>j, k in processador, s in 1..nturno-1) do
  w(j) >= w(i)+d(i,k)-alfa*(2-(y(i,k,s)+sum(r in (s+1)..nturno)y(j,k,r)))
end-do
forall(j in tarefa) do
z >= w(j)+sum(k in processador, s in turno)d(j,k)*y(j,k,s)
end-do
minimize(z)
end-model
```

As informações lidas do arquivo "scheduling.txt" são:
arcos:[(1 2) 1
(1 3) 1
(2 4) 1
(3 4) 1]
d:[3 2
1 4
5 3
2 3
4 5]

A solução encontrada é: $y_{11}^1 = y_{11}^2 = y_{11}^3 = y_{11}^4 = y_{11}^5 = 0$,
$y_{12}^1 = 1,\ y_{12}^2 = y_{12}^3 = y_{12}^4 = y_{12}^5 = 0$,
$y_{21}^1 = 0,\ y_{21}^2 = 1,\ y_{21}^3 = y_{21}^4 = y_{21}^5 = 0$,
$y_{22}^1 = y_{22}^2 = y_{22}^3 = y_{22}^4 = y_{22}^5 = 0$,
$y_{31}^1 = y_{31}^2 = y_{31}^3 = y_{31}^4 = y_{31}^5 = 0$,
$y_{32}^1 = 0,\ y_{32}^2 = 1,\ y_{32}^3 = y_{32}^4 = y_{32}^5 = 0$,
$y_{41}^1 = y_{41}^2 = 0,\ y_{41}^3 = 1,\ y_{41}^4 = y_{41}^5 = 0$,
$y_{42}^1 = y_{42}^2 = y_{42}^3 = y_{42}^4 = y_{42}^5 = 0$,
$y_{51}^1 = 1,\ y_{51}^2 = y_{51}^3 = y_{51}^4 = y_{51}^5 = 0$,
$y_{52}^1 = y_{52}^2 = y_{52}^3 = y_{52}^4 = y_{52}^5 = 0$,
$w_1 = 0,\ w_2 = 4,\ w_3 = 2,\ w_4 = 5,\ w_5 = 0$.
O valor da função-objetivo é 7. A figura 3.2 ilustra a solução encontrada para os processadores P1 e P2 respectivamente. Os valores de 0 a 7 indicam o tempo utilizado por cada tarefa em cada processador.

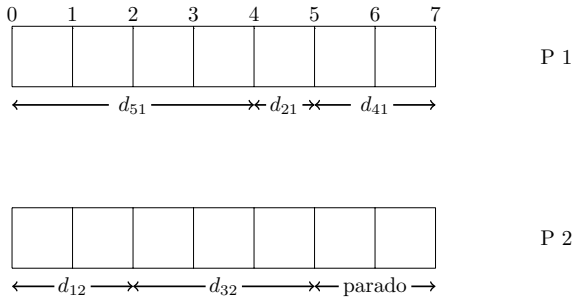


Figura 3.2: Ilustração da solução obtida.

3.6 Transformação de um modelo de otimização não linear mista com função-objetivo como a soma de frações 0-1 em um problema de otimização linear mista

Observamos que alguns problemas possuem função-objetivo não linear, enquanto as restrições são todas lineares. A não linearidade encontrada na função-objetivo é a soma de frações (SF), conforme exemplo a seguir.

Exemplo 5.

$$\text{(SF):} \quad \text{Maximizar } z = \frac{5x_1}{(1+3x_2)} + \frac{3x_3}{(7+6x_5)} + x_4, \tag{3.264}$$

$$\text{sujeito a: } x_1 + x_3 + x_4 \leq 2, \tag{3.265}$$

$$x_2 + x_5 \geq 1, \tag{3.266}$$

$$x_j \in \{0,1\}, \quad j = 1,2,3,4,5. \tag{3.267}$$

Nos artigos [133] e [134] temos exemplo de outras funções-objetivo como soma de frações. Uma sugestão para contornar a não linearidade neste caso de soma de frações é realizar a linearização da mesma, conforme comparações propostas em [22].

Neste caso não é necessário utilizar métodos de resolução de problemas não lineares, pois é possível contornar essa não linearidade com acréscimo de uma nova variável e realizar a linearização do produto de duas variáveis inteiras.

Considere as variáveis

$$y_1 = \frac{1}{1+3x_2}$$

e

$$y_2 = \frac{1}{7+6x_5}.$$

Fazemos a linearização da fração transformando o quociente em produto e o modelo será modificado e denominado MSF. Lembrando que o denominador de todas as frações deve ser diferente de zero para que a linearização possa ser realizada.

Exemplo 6.

$$
\begin{aligned}
\text{(MSF):} \quad & \text{Maximizar } z = 5x_1y_1 + 3y_2x_3 + x_4, && (3.268)\\
& \text{sujeito a: } (3.265), (3.266), && (3.269)\\
& y_1 + 3y_1x_2 = 1, && (3.270)\\
& 7y_2 + 6y_2x_5 = 1, && (3.271)\\
& 0 < y_i \leq 1 \quad i = 1,2; && (3.272)\\
& 0 \leq z_i \leq 1 \quad i = 1,2; && (3.273)\\
& x_j \in \{0,1\}, \quad j = 1,2,3,4,5. && (3.274)
\end{aligned}
$$

O problema (SF) definido em (3.264)-(3.267), não linear inteiro misto (PNLIM), pode ser reescrito de forma relaxada (RSF) como problema linear inteiro misto (PLIM), acrescentando-se as novas variáveis t_1, t_2, z_1, z_2, que tratam os produtos de variáveis existentes em MSF,

$$t_1 = x_1y_1,$$

$$t_2 = x_3y_2,$$

$$z_1 = y_1x_2,$$

$$z_2 = y_2x_5,$$

e acrescentando-se as restrições para a linearização dos produtos, tanto da função-objetivo como das restrições.

Exemplo 7.

$$
\begin{aligned}
\text{(RSF):} \quad & \text{Maximizar } z = 5t_1 + 3t_2 + x_4, && (3.275)\\
& \text{sujeito a: } (3.265), (3.266) && (3.276)\\
& y_1 + 3z_1 = 1, && (3.277)\\
& 7y_2 + 6z_2 = 1, && (3.278)\\
& z_1 \leq y_1, && (3.279)\\
& z_1 \leq x_2, && (3.280)\\
& z_1 \geq y_1 + x_2 - 1, && (3.281)\\
& z_2 \leq y_2, && (3.282)\\
& z_2 \leq x_5, && (3.283)\\
& z_2 \geq y_2 + x_5 - 1, && (3.284)\\
& z_1 \leq y_1, && (3.285)\\
& z_1 \leq x_2, && (3.286)\\
& z_1 \geq y_1 + x_2 - 1, && (3.287)\\
& z_2 \leq y_2, && (3.288)\\
& z_2 \leq x_5, && (3.289)\\
& z_2 \geq y_2 + x_5 - 1, && (3.290)\\
& t_1 \leq y_1, && (3.291)\\
& t_1 \leq x_1, && (3.292)\\
& t_1 \geq y_1 + x_1 - 1 && (3.293)\\
& t_2 \leq y_2, && (3.294)\\
& t_2 \leq x_3, && (3.295)\\
& t_2 \geq y_2 + x_3 - 1, && (3.296)\\
& 0 < y_i \leq 1 \quad i = 1,2; && (3.297)\\
& 0 \leq z_i \leq 1 \quad i = 1,2; && (3.298)\\
& 0 \leq t_i \leq 1 \quad i = 1,2; && (3.299)\\
& x_j \in \{0,1\}, \quad j = 1,2,3,4,5. && (3.300)
\end{aligned}
$$

```
model ModelName
uses "mmxprs"
declarations
  x: array(1..5) of mpvar
  y: array(1..2) of mpvar
  z:array(1..2) of mpvar
  t: array(1..2) of mpvar
end-declarations
x(1)+x(3)+x(4)<=2
x(2)+x(5)>=1
z(1)<=y(1)
z(1)<=x(2)
z(1)>=y(1)+x(2)-1
z(2)<=y(2)
```

```
z(2)<=x(5)
z(2)>=y(2)+x(5)-1
t(1)<=y(1)
t(1)<=x(1)
t(1)>=x(1)+y(1)-1
t(2)<=y(2)
t(2)<=x(3)
t(2)>=y(2)+x(3)-1
y(1)+3*z(1)=1
7*y(2)+6*z(2)=1
forall (i in 1..5) x(i) is_binary
!objective function
obj:= 5*t(1)+(3*t(2))+x(4)
maximize(obj)
end-model
```

Cuja solução é: $x_1 = x_4 = x_5 = 1,\ x_2 = x_3 = 0,\ y_1 = 1,\ y_2 = 0.0769231,\ z_1 = 0,$ $z_2 = 0.0769231,\ t_1 = 1,\ t_2 = 0.$ O valor ótimo da função-objetivo é 6.

3.7 Conclusões

Neste capítulo apresentamos algumas modelagens triviais e outras menos triviais no uso de variáveis 0-1 para problemas de otimização. No entanto, os problemas de otimização combinatória NP-hard em grafos, tais como o do Caixiero Viajante, Árvore de Steiner e o do caminho mínimo entre dois vértices distintos de um grafo em presença de circuitos negativos, foram deixados para o próximo capítulo deste livro.

4

Problemas NP-difíceis em Otimização em grafos

Virginia Costa
Ana Flávia Uzeda Macambira
Laura Bahiense
Laura Patuzzi
Nelson Maculan

Existem problemas de programação linear inteira mista que exigem um árduo trabalho de implementação e um grande esforço computacional para a sua resolução, mesmo para instâncias pequenas. Softwares comerciais de otimização, tais como o CPLEX, Gurobi, XPRESS e outros, não podem ser utilizados diretamente, devido a modelagens que apresentam um número exponencial de restrições. Essa característica está presente no Problema do Caixeiro Viajante (TSP – Travelling Salesman Problem) e também no Problema da Árvore de Steiner em grafos. Para esses problemas, porém, existem outras abordagens, baseadas em programação inteira mista com variáveis binárias, que fazem uso de um número polinomial de restrições e variáveis.

Neste capítulo, serão discutidas algumas abordagens de programação inteira mista para problemas de otimização em grafos, tais como: ciclos, árvores e caminhos. Alguns desses problemas são considerados NP-difíceis, e serão exemplificados através de pequenas instâncias. Desta forma, este capítulo está organizado da seguinte maneira: na seção 1 apresentamos uma formulação matemática para o problema de encontrar um subgrafo conexo. Na seção 2 tratamos do problema de encontrar um caminho elementar entre um vértice origem e um vértice destino. Na seção 3, abordamos o Problema dos Ciclos Elementares. O Problema do Caixeiro Viajante é apresentado na seção 4, e a seção 5 é dedicada aos problemas relacionados a árvores: Árvores Geradoras e Árvores de Steiner.

4.1 Subgrafo conexo de G

Seja $G = (V,E)$ um grafo de vértices $i,\ j \in V$ e arestas $\{i,j\} \in E$, com $i < j$. Considere um vetor x, cujas componentes $x_i \in \{0,1\}$, para $i \in V$, são associadas aos vértices de um subgrafo de G, de tal forma que, se $x_i = 1$, então o vértice i pertence ao subgrafo, caso contrário, $x_i = 0$. Considere também um vetor y de componentes $y_{ij} \in \{0,1\}$, onde $\{i,j\} \in E$. Cada componente y_{ij} está associada a uma aresta $\{i,j\}$ do subgrafo de G, sendo $y_{ij} = 1$ quando a aresta pertencer ao subgrafo e 0, caso contrário. Os vetores x e y definem o subgrafo, fornecendo informações sobre seus vértices e arestas.

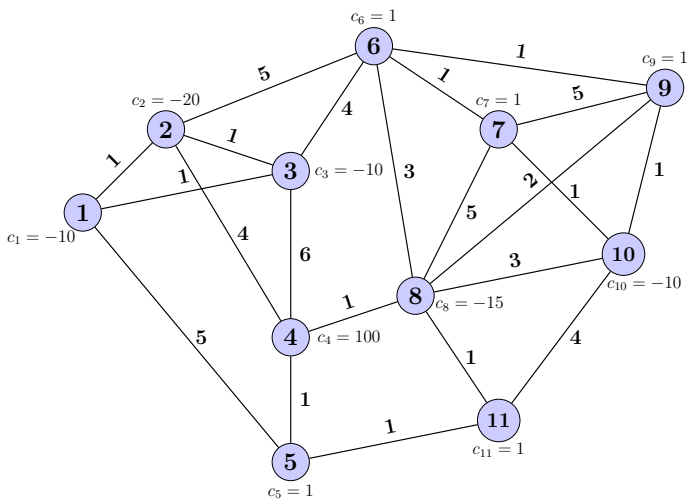
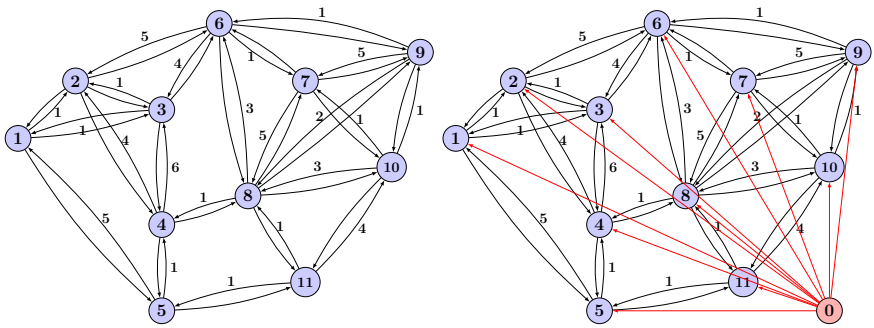


Figura 4.1: Exemplo de um grafo não orientado $G = (V,E)$ com onze vértices.

$G_d = (V,A)$, onde $A = \{(i,j),(j,i) \mid \{i,j\} \in E\ \}$ é o conjunto dos arcos. Definiremos também um novo grafo orientado $G_d^0 = (V_0,A_0)$, onde $V_0 = V \cup \{0\}$, $E_0 = E \cup \{\{0,j\} | j \in V\}$, $A_0 = A \cup \{(0,j) |\ j \in V\}$ (ver figura 4.2). Consideraremos $\Gamma^+(i) = \{j \mid (i,j) \in A_0\}$ e $\Gamma^-(i) = \{j \mid (j,i) \in A_0\}$. O vértice 0 será utilizado como origem para um problema de fluxos não simultâneos [107]. Utilizaremos a variável $z_{ij}^k \geq 0$, que representa o fluxo no arco $(i,j) \in A_0$ que sai do vértice 0 para o vértice $k \in V$. Considerando essa abordagem, temos a seguinte formulação matemática proposta em [155]:



(a) $G_d = (V, A)$. (b) $G_d^0 = (V_0, A_0)$.

Figura 4.2: Adicionando um vértice 0 ao grafo G_d e o conjunto de arcos $\{(0, j) \mid j \in V\}$.

$$\sum_{j \in \Gamma^+(0)} z_{0j}^k - x_k = 0, \qquad k \in V \tag{4.1}$$

$$\sum_{j \in \Gamma^+(i)} z_{ij}^k - \sum_{j \in \Gamma^-(i)} z_{ji}^k = 0, \qquad i \in V - \{k\},\ k \in V \tag{4.2}$$

$$\sum_{j \in \Gamma^+(k)} z_{kj}^k - \sum_{j \in \Gamma^-(k)} z_{jk}^k + x_k = 0, \qquad k \in V \tag{4.3}$$

$$z_{ij}^k \le y_{ij} \ \text{ e } \ z_{ji}^k \le y_{ij}, \qquad \{i, j\} \in E,\ k \in V \tag{4.4}$$

$$y_{ij} \le x_i \ \text{ e } \ y_{ij} \le x_j, \qquad \{i, j\} \in E \tag{4.5}$$

$$\sum_{j \in V} y_{0j} = 1 \tag{4.6}$$

$$z_{ij}^k \ge 0, \qquad (i, j) \in A_0,\ k \in V \tag{4.7}$$

$$y_{ij} \in \{0, 1\},\ x_k \in \{0, 1\}, \qquad \{i, j\} \in E_0, k \in V. \tag{4.8}$$

Todos os vetores x e y que satisfazem as restrições de (4.1) a (4.8) estão associados a um subgrafo conexo de G. Como um subgrafo conexo deve ter pelo menos dois vértices, temos que incluir as restrições

$$\sum_{\{i,j\} \in E} y_{ij} \ge 1 \ \text{ ou } \sum_{i \in V} x_i \ge 2. \tag{4.9}$$

Sobre as soluções viáveis de (4.1) a (4.9), deveremos fazer um comentário importante. Sabemos que as variáveis x_i e y_{ij} pertencerão sempre ao conjunto $\{0, 1\}$. No entanto, como ficariam os valores das variáveis $z_{ij}^k \ge 0$?

Consideremos o sistema de $(|V|+1)(|V|-1)$ equações lineares simultâneas (4.1), (4.2) e (4.3), o qual para cada k pode ser escrito da seguinte maneira $Mz^k = b^k$, e as demais restrições envolvendo z^k sob a seguinte forma $0 \le z^k \le d^k$, em que z^k é o vetor das variáveis z_{ij}^k. É sabido que a matriz M é conhecida como a matriz vértice-arco para representar o grafo G_d^0, logo o seu posto é igual a $|V|$. M^k é também uma matriz

totalmente unimodular, isto é, todas as suas submatrizes formadas pela interseção de p linhas com p colunas têm determinante no conjunto $\{-1,0,1\}$, inclusive para $p=|V|$. Sabemos que $d^k \in \{-1,0,1\}^{(|V|+1)}$, assim sendo toda solução básica de $Mz^k = b^k$ será inteira. Como $0 \leq z^k \leq d^k$, teremos nesse caso que $z^k \in \{0,1\}^{(|V|+1)}$.

Associamos a cada vértice $i \in V$ um custo c_i e a cada aresta $\{i,j\} \in E$ um custo w_{ij}.
Para a obtenção de um grafo conexo G_{SC} de custo total mínimo, subgrafo de G, resolveremos o seguinte problema de programação inteira mista, em que os valores de c_i, $i \in V$ e de $w_{ij} \in E$ estão na figura 4.1:

$$(\mathrm{P}_1): \quad \text{Minimizar } f(x,y) = \sum_{i \in V} c_i x_i + \sum_{\{i,j\} \in E} w_{ij} y_{ij}, \tag{4.10}$$

sujeito às restrições de (4.1) a (4.9).

As restrições (4.1) - (4.9) de P_1 são aplicadas a cada vértice $k \in V$. A restrição (4.1) se refere aos fluxos que saem do vértice de origem 0 com destino a cada vértice j. Essa restrição impõe que a soma dos fluxos que saem do vértice 0 deva ser igual a 1, caso a variável x_k assuma esse valor. A restrição (4.2) nos mostra que, para os demais vértices $i \in V$, tais que $i \neq k$, deve haver a conservação de fluxo, isto é, a quantidade de fluxo que entra em i deve ser igual à quantidade de fluxo que sai desse vértice. Já na restrição (4.3), que se refere ao vértice k, temos que o somatório de todo o fluxo que entra nesse vértice deve ser maior que o somatório de todo o fluxo que sai, garantindo assim a conexidade a um de seus vértices adjacentes. A restrição (4.4) se refere à capacidade de cada aresta $\{i,j\} \in E_0$, e a restrição (4.5) é responsável por remover as arestas ligadas a vértices retirados do subgrafo de G. A restrição (4.6) impede a existência de mais de uma aresta ligada a 0, a fim de evitar que tenhamos um subgrafo não conexo ao removermos esse vértice, que foi inserido como um auxiliar para essa modelagem matemática.

Para ilustrarmos a formulação matemática P_1 proposta por [155], considere o grafo G visto na figura 4.1. A partir desse grafo, vamos obter o grafo G_d, direcionando suas arestas conforme a figura 4.2(a). Para utilizarmos as restrições de (4.1) a (4.9), é preciso incluir um vértice de origem 0. A inclusão desse vértice pode ser vista na figura 4.2(b), que mostra o grafo G_d^0. Na figura 4.1, podemos observar os pesos w_{ij} escritos próximos às arestas, e, próximo a cada vértice, temos as componentes do vetor c. Por exemplo, para a aresta $\{2,6\}$, temos $w_{26} = 5$ e, para o vértice 2, temos que $c_2 = -20$. Considerando os pesos expressos na figura 4.1 e implementando P_1 em linguagem Xpress-Mosel [219], obtemos o resultado visto na figura 4.3, em que $f(x,y) = -53$ é o valor da função-objetivo (4.10).

4.2 Caminhos elementares

Seja p um inteiro positivo maior que zero, considere um caminho elementar entre dois vértices distintos $v_0, v_p \in V$ em um grafo G, que contenha uma sequência de $p+1$ vértices, $\{v_0, v_1, v_2, v_3, \ldots, v_p\} \subseteq V$, formando p arestas consecutivas, $\{\{v_0,v_1\},\{v_1,v_2\},\{v_2,v_3\},\ldots,\{v_{p-1},v_p$ E. Podemos encontrar esse caminho, utilizando as restrições de (4.1) a (4.4) e de (4.6) a (4.8), e substituindo a restrição (4.5) pelas equações a seguir:

$$\sum_{u \in E(l)} y_u = 1, \qquad l \in \{v_0, v_p\}, \tag{4.11}$$

$$\sum_{u \in E(l)} y_u = 2x_l, \qquad l \in V - \{v_0, v_p\}, \tag{4.12}$$

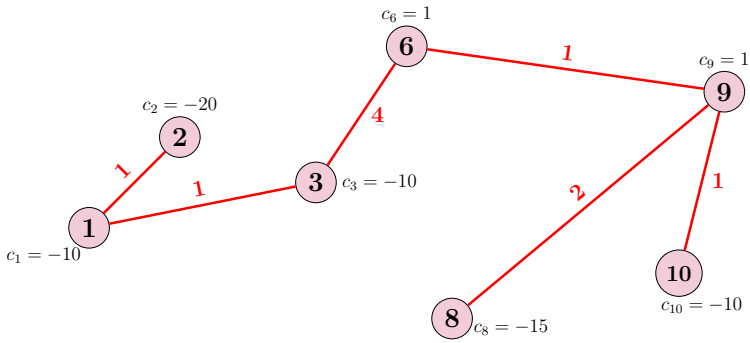


Figura 4.3: Grafo conexo G_{SC}, subgrafo de G, obtido a partir de P_1.

em que $E(l)$ é o conjunto das arestas $u = \{v_i, v_j\}$, tal que $i, j \in \{0, 1, 2, \ldots, p\}$ e $v_i = l$ ou $v_j = l$. Assim, todos os vetores x e y que satisfazem as restrições de (4.1) a (4.4), de (4.6) a (4.8), (4.11) e (4.12) estão associados a um caminho elementar de G entre os vértices v_0 e v_p. É importante observar que as componentes x_{v_0} e x_{v_p} de x, referentes aos vértices v_0 e v_p de G, devem ser iguais a 1 obrigatoriamente.

Observando o grafo G da figura 4.1, é possível traçar caminhos elementares entre os vértices 2 e 7, por exemplo. As figuras 4.4(a) e 4.4(b) mostram dois caminhos possíveis entre esses dois vértices. Note que, em ambas as imagens, os vértices extremos, ou seja, $v_0 = 2$ e $v_p = 7$, possuem grau igual a 1. Essa condição é imposta na equação (4.11). Já os vértices intermediários pertencentes ao caminho possuem grau 2, conforme a equação (4.12). O que diferencia uma figura da outra é a quantidade de vértices intermediários e, consequentemente, a quantidade p de arestas.

Para o caso em que $v_0 = 2$ e $v_p = 7$, podemos escrever a equação (4.11) como o conjunto de equações abaixo:

$$\begin{aligned} y_{12} + y_{23} + y_{24} + y_{26} &= 1, \\ y_{67} + y_{78} + y_{79} + y_{7,10} &= 1. \end{aligned}$$

Note que, em cada uma dessas duas últimas equações, somente uma componente y_u, $u \in E(l)$ e $l \in \{2, 7\}$, poderá ter valor igual a 1, o que representa a existência de apenas uma aresta ligada a cada extremidade.

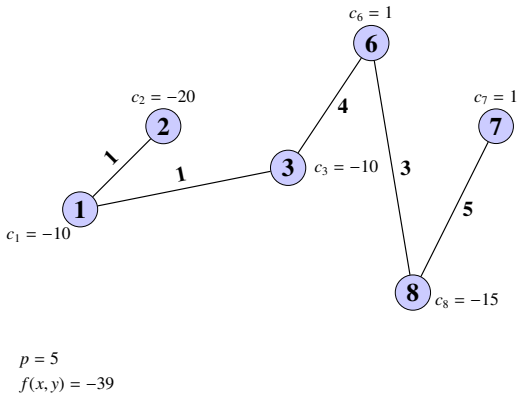
Os vértices intermediários são tratados pela equação (4.12). Vejamos um exemplo em que $l \in V - \{2, 7\}$, tal que $l = 3$. Neste caso, a equação (4.12) é escrita como segue:

$$y_{23} + y_{34} + y_{36} = 2x_3.$$

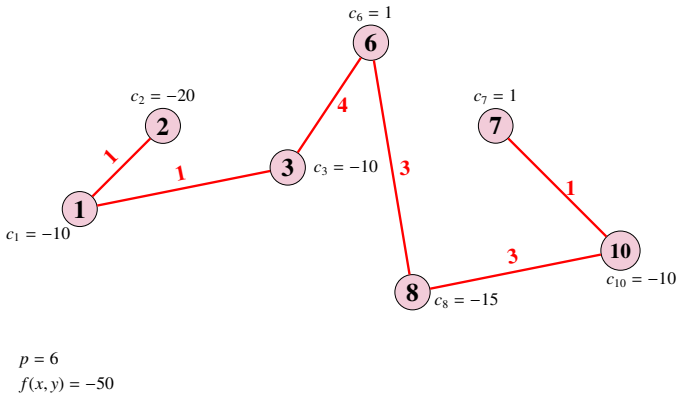
Note que, para $x_3 = 0$, temos $y_{23} + y_{34} + y_{36} = 0$ e, consequentemente, $y_{23} = y_{34} = y_{36} = 0$. Mas, para $x_3 = 1$, temos

$$y_{23} + y_{34} + y_{36} = 2 \Rightarrow \begin{cases} y_{23} = 0 \text{ e } y_{34} = y_{36} = 1 \text{ ou} \\ y_{34} = 0 \text{ e } y_{23} = y_{36} = 1 \text{ ou} \\ y_{36} = 0 \text{ e } y_{23} = y_{34} = 1, \end{cases}$$

ou seja, devemos ter duas componentes y_u, $u \in E(3)$, iguais a 1. Sendo cada variável y_u associada à permanência da aresta u no caminho elementar, então temos que o vértice 3 deve ter obrigatoriamente grau 2, por não estar nas extremidades do caminho.



(a) Solução viável.



(b) Solução ótima.

Figura 4.4: Soluções viáveis para o problema P_2, obtidas a partir do grafo G, considerando $v_0 = 2$ e $v_p = 7$.

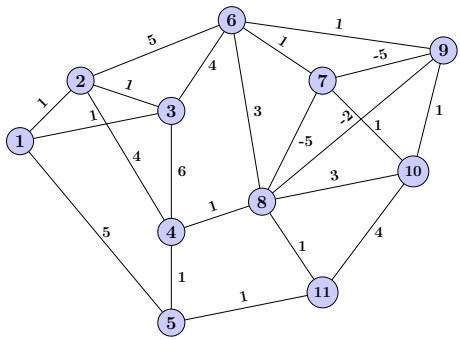


Figura 4.5: Grafo $\bar{G} = (V,E)$ obtido a partir de G. Neste grafo, consideramos $c_i = 0$, para $i \in \{1,2,\ldots,11\}$, $w_{78} = w_{87} = -5$, $w_{79} = w_{97} = -5$ e $w_{89} = w_{98} = -2$.

Utilizando as equações (4.11) e (4.12), juntamente com algumas restrições vistas na seção 4.1, com os dados da figura 4.2(b), podemos escrever o seguinte problema de programação inteira mista para a obtenção de um caminho elementar mínimo:

$$(\mathrm{P}_2): \quad \text{Minimizar } f(x,y) = \sum_{i \in V} c_i x_i + \sum_{\{i,j\} \in E} w_{ij} y_{ij}, \tag{4.13}$$

sujeito às restrições de (4.1) a (4.4), de (4.6) a (4.8), (4.11) e (4.12).

Resolvendo-se P_2 para o grafo G, considerando o caso $v_0 = 2$ e $v_p = 7$, obtemos o caminho elementar visto na figura 4.4(b) como solução ótima, em que o valor da função-objetivo (4.13) é igual a -50. É importante observar que, por haver pesos negativos relacionados aos vértices, não é possível o uso do algoritmo de Dijkstra para a obtenção de caminhos mínimos no grafo G.

4.3 Ciclos elementares

Na seção 4.1, vimos como garantir que um subgrafo de G seja conexo. Na seção 4.2, foi visto como se obter um caminho elementar entre dois vértices v_0 e v_p com p arestas. Vejamos, agora, como se obterem ciclos elementares com p arestas, $3 \leq p \leq |E|$. Para tal, considere as seguinte restrições:

$$\sum_{\{i,j\} \in E} y_{ij} = 2x_i, \; i \in V, \tag{4.14}$$

$$\sum_{i \in V} x_i = p., \tag{4.15}$$

A inclusão da equação (4.14) ou da equação (4.15) no lugar da restrição (4.5), vista na seção 4.1, garante que cada vértice do subgrafo de G terá grau igual a 2, condição necessária para a ocorrência de ciclos. Se compararmos as equações (4.12) e (4.14), notaremos que a segunda foi obtida a partir da primeira, considerando-se

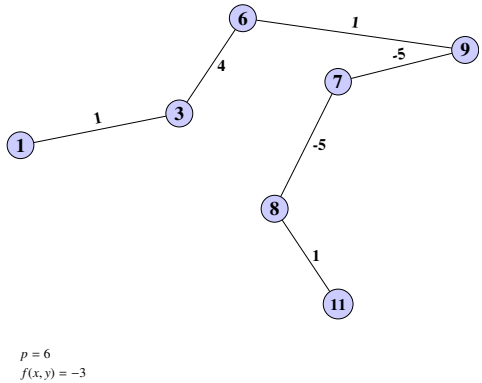


Figura 4.6: Solução obtida pela resolução de P_2 a partir do grafo $\bar{G}$ (figura 4.5).

$v_0 = v_p$. Neste caso, podemos escrever um problema de programação inteira mista da seguinte forma:

$$(P_3): \quad \text{Minimizar } f(x,y) = \sum_{i \in V} c_i x_i + \sum_{\{i,j\} \in E} w_{ij} y_{ij}, \tag{4.16}$$

sujeito às restrições de (4.1) a (4.4), de (4.6) a (4.8) e de (4.14).

Ao resolvermos P_3, utilizando as informações disponíveis no grafo G da figura 4.1, obtemos o ciclo elementar da figura 4.7. Note que foram removidos os vértices 4 e 9, e todos os vértices presentes no ciclo possuem grau igual a 2. O valor apurado para função-objetivo foi -43. A mesma ideia embutida em P_3 pode ser utilizada na obtenção de ciclos hamiltonianos, mediante algumas modificações que serão vistas a seguir.

4.3.1 Ciclos Hamiltonianos

Em um ciclo hamiltoniano, há a presença de todos os vértices de G. Desta forma, tendo em vista as restrições de (4.1) a (4.4) e de (4.6) a (4.14), percebemos que as componentes x_i do vetor x, associadas a cada vértice $i \in V$, recebem todas o valor 1. Também não há a necessidade de acrescentarmos o vértice 0 como origem, pois, como estamos em um ciclo, podemos utilizar qualquer vértice para esse fim. Usaremos,

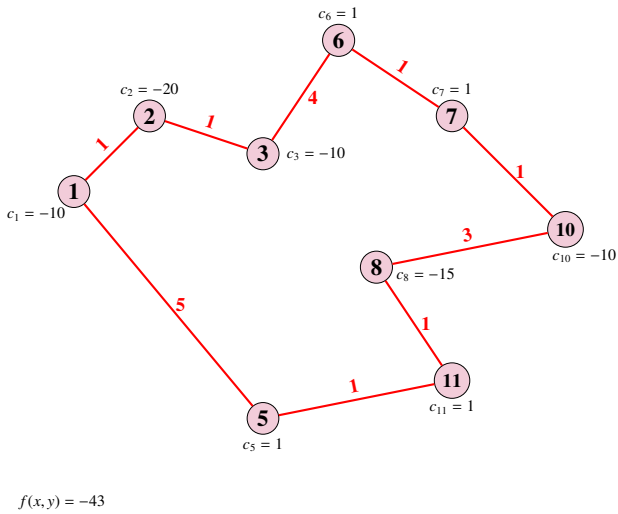


Figura 4.7: Ciclo elementar obtido a partir da solução de P_3, utilizando-se o grafo G, para $p = 9$.

então, o vértice 1. Assim, podemos reescrever essas restrições como segue:

$$\sum_{j \in \Gamma^+(1)} z^k_{1j} - \sum_{j \in \Gamma^-(1)} z^k_{j1} = 1, \qquad k \in V - \{1\} \tag{4.17}$$

$$\sum_{j \in \Gamma^+(i)} z^k_{ij} - \sum_{j \in \Gamma^-(i)} z^k_{ji} = 0, \qquad i \in V - \{1, k\},\ k \in V - \{1\} \tag{4.18}$$

$$\sum_{j \in \Gamma^+(k)} z^k_{kj} - \sum_{j \in \Gamma^-(k)} z^k_{jk} = -1, \qquad k \in V - \{1\} \tag{4.19}$$

$$z^k_{ij} \leqslant y_{ij} \text{ e } z^k_{ji} \leqslant y_{ij}, \qquad \{i, j\} \in E,\ k \in V - \{1\} \tag{4.20}$$

$$\sum_{\{i,j\} \in E} y_{ij} = 2, \qquad i \in V \tag{4.21}$$

$$z^k_{ij} \geqslant 0, \qquad (i, j) \in A,\ k \in V - \{1\} \tag{4.22}$$

$$y_{ij} \in \{0, 1\}, \qquad \{i, j\} \in E. \tag{4.23}$$

Note que todos os vetores y que satisfazem as restrições de (4.17) a (4.23) estão associados a um ciclo hamiltoniano de G. A seguir, na seção 4.4, com o Problema do Caixeiro Viajante, serão vistos mais detalhes sobre o uso dessas restrições na obtenção de um ciclo hamiltoniano de tamanho mínimo.

4.4 Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) pode ser definido da seguinte forma: seja G um grafo conexo, orientado ou não, e Vo conjunto de seus vértices. Seja

$W = (w_{ij}) \in \mathbb{R}^{|V|\times|V|}$ uma matriz cujos elementos representam pesos associados a cada aresta ou arco existente entre os vértices $i \in V$ e $j \in V$ de G. Deseja-se determinar o ciclo hamiltoniano cujo somatório dos pesos de suas arestas ou arcos seja mínimo.

Quando G é um grafo não orientado, temos que $w_{ij} = w_{ji}$ e o PCV é denominado simétrico. Caso contrário, o PCV é denominado assimétrico. Se a cada vértice $i \in V$ estiver associado um ponto w_i no espaço euclidiano, tal que $w_{ij} = ||w_i - w_j||$, e a desigualdade triangular

$$w_{ij} + w_{jk} \geq w_{ik}, \forall i, j, k \in V$$

é satisfeita, dizemos que o PCV é euclidiano.

Em 1954, Dantzig, Fulkerson e Johnson propuseram uma formulação de programação linear inteira com variáveis binárias para a resolução do PCV. A formulação matemática é a seguinte:

$$(\text{PCV}_1): \quad \text{Minimizar } f(y) = \sum_{(i,j)\in A} w_{ij} y_{ij}, \tag{4.24}$$

sujeito a:

$$\sum_{j\in V,\ j\neq i} y_{ij} = 1\ , \qquad \forall i \in V,\ (i,j) \in A \tag{4.25}$$

$$\sum_{i\in V,\ i\neq j} y_{ij} = 1\ , \qquad \forall j \in V,\ (i,j) \in A \tag{4.26}$$

$$\sum_{(i,j)\in S\times S,\ i\neq j} y_{ij} \leq |S| - 1, \qquad \forall S \subset V,\ 2 \leq |S| \leq n-1 \tag{4.27}$$

$$y_{ij} \in \{0,1\}\ , \qquad \forall (i,j) \in A,\ i \neq j. \tag{4.28}$$

Em PCV_1, podemos perceber que se trata de uma formulação matemática para problemas assimétricos, num grafo (orientado) completo e sem laços. As restrições (4.25) impõem que cada vértice será origem de apenas um arco, enquanto as restrições (4.26) impõem que cada vértice será o destino de apenas um arco. Sendo assim, cada vértice i terá apenas dois arcos conectados e consequentemente terá grau igual a 2. As restrições (4.27) são responsáveis pela eliminação de subciclos.

Para problemas simétricos, as restrições (4.25) e (4.26) podem ser substituídas pelas equações (4.21) e o problema PCV_1 pode ser reescrito da seguinte forma:

$$(\text{PCV}_2): \quad \text{Minimizar } f(y) = \sum_{\{i,j\}\in E} w_{ij} y_{ij}, \tag{4.29}$$

sujeito a:

$$\sum_{\{i,j\}\in E(i)} y_{ij} = 2\ , \qquad \forall i \in V, \tag{4.30}$$

$$\sum_{\{i,j\}\in E(S)} y_{ij} \leq |S| - 1, \qquad \forall S \subset V,\ 3 \leq |S| \leq n-1, \tag{4.31}$$

$$y_{ij} \in \{0,1\}\ , \qquad \forall \{i,j\} \in E, \tag{4.32}$$

em que $E(S)$ é o conjunto das arestas com ambas as extremidades em $S \subset V$, e $E(i)$ é o conjunto de todas as arestas com uma extremidade em $i \in V$.

Como em ciclos hamiltonianos há a inclusão de todos os vértices do grafo, não usaremos, nesta seção, as componentes c_i, $i \in V$, vistas no grafo G dessa figura. Note que, sem considerarmos a restrição (4.31), temos na figura 4.8 uma solução viável, obtida a

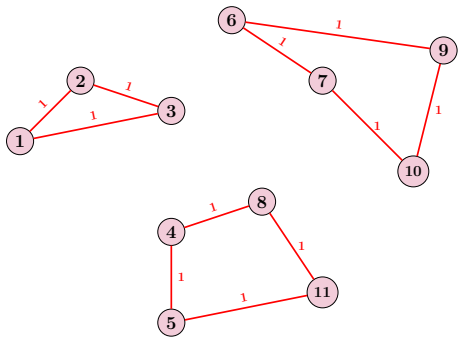


Figura 4.8: Solução viável obtida a partir do grafo G da figura 4.1 para o PCV_2, excluindo a restrição (4.31). Observamos aqui a ocorrência de subciclos.

partir do grafo G, para o PCV_2. Nessa solução, não há um ciclo hamiltoniano. O que se nota é a ocorrência de pequenos subciclos. As restrições (4.31) eliminam dentro de cada subconjunto $S \subset V$ a formação de subciclos. Desta forma, ao considerarmos as restrições (4.31), a solução da figura 4.8 deixa de ser viável, pois, para $S = \{1,2,3\}$, por exemplo, teríamos $|S| = 3$ e

$$\sum_{\{i,j\} \in E(S)} y_{ij} \leq 2 \quad \Rightarrow \; y_{12} + y_{13} + y_{23} \leq 2,$$

o que significa que pelo menos uma componente y_{ij}, $i, j \in \{1,2,3\}$, deve ser igual a zero.

Sendo $n = |V|$, é importante observar que o número de restrições em (4.27) é igual a $2^n - n - 2$. Notemos que 2^n é o número total de subconjuntos de V, do qual subtraímos n subconjuntos unitários além do conjunto vazio e V. Este último é removido para que não haja a exclusão de todas as soluções viáveis. Portanto, em (4.27) temos um número exponencial de restrições. Em (4.31), para a contagem do número dessas restrições, subtraímos do número de restrições de (4.27) a quantia $[n(n-1)]/2$. Assim, retiramos os conjuntos de dois elementos, pois não é necessária a inclusão de restrições do tipo $y_{ij} \leq 1$, visto que $y_{ij} \in \{0,1\}$.

Em 1960, Miller, Tucker e Zemlin, reduziram o número de restrições de eliminação de subciclos, escrevendo uma formulação matemática que utiliza n variáveis arbitrárias $u_i \in \mathbb{R}$, $i \in \{1,\ldots,n\}$. Nessa formulação matemática, a restrição (4.27) é substituída por

$$u_i - u_j + n \cdot y_{ij} \leq n - 1 \qquad i \in \{1,\ldots,n\}, j \in \{2,\ldots,n\}, \; (i \neq j). \tag{4.33}$$

Em (4.33), em se tratando de problemas assimétricos, o número total de restrições é igual ao número de elementos de A, ou seja, $|A|$, que pode ser igual, no máximo, à quantidade de arranjos $A_{n,2} = n(n-1) = n^2 - n$, em casos de grafos completos. Portanto, temos para essa formulação matemática, um número polinomial de restrições de eliminação de subciclos. Em casos de problemas simétricos, temos que a quantidade de restrições (4.33) é dada pelo número de elementos de E, ou seja, $|E|$, e pode chagar no máximo ao número de combinações $C_{n,2} = [n(n-1)]/2$.

A formulação matemática proposta por [167] pode ser vista abaixo, no PCV_3.

$$(\text{PCV}_3): \quad \text{Minimizar } f(y) = \sum_{(i,j)\in A} w_{ij} y_{ij}, \tag{4.34}$$

sujeito a:

$$\sum_{j\in V,\ j\neq i} y_{ij} = 1\ , \qquad \forall i \in V, \tag{4.35}$$

$$\sum_{i\in V,\ i\neq j} y_{ij} = 1\ , \qquad \forall j \in V, \tag{4.36}$$

$$u_i - u_j + n \cdot y_{ij} \leq n-1, \qquad i \in V,\ j \in V - \{1\},\ i \neq j, \tag{4.37}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A,\ i \neq j, \tag{4.38}$$

$$u_i \geq 0, \qquad i \in V. \tag{4.39}$$

Para analisarmos o uso de (4.33), considere um subciclo contendo $k-1$ vértices e o conjunto $\{v_1, v_2, v_3, \ldots, v_k\}$, em que $v_1 = v_k$ e cada v_i recebe um vértice do subciclo em questão. Em (4.33), temos $k-1$ restrições da seguinte forma:

$$\begin{array}{rcl} u_{v_1} - u_{v_2} & \leq & -1 \\ u_{v_2} - u_{v_3} & \leq & -1 \\ u_{v_3} - u_{v_4} & \leq & -1 \\ & \vdots & \\ u_{v_{k-1}} - u_{v_k} & \leq & -1. \end{array}$$

Ao somarmos essas $k-1$ restições, obtemos $u_{v_1} - u_{v_k} \leq -1(k-1)$, isto é, $u_{v_1} - u_{v_k} \leq 1-k$. Como $v_1 = v_k$, temos que $u_{v_k} - u_{v_k} = 0 \leq 1-k$, o que é uma contradição, visto que $1 < k$. Portanto, a ocorrência de subciclos faz com que as restrições de (4.33) não sejam satisfeitas. Para exemplificar melhor esse fato, usaremos a situação descrita na figura 4.8. Assim, vamos considerar o subconjunto $S \subset V$ tal que $S = \{4,5,8,11\}$. Neste caso, temos $k = 5$, $v_1 = v_k = 4$ e atribuiremos um elemento de S aos demais v_i, ou seja, $v_2 = 5$, $v_3 = 11$ e $v_4 = 8$. Podemos, então, desenvolver (4.33) da seguinte forma:

$$\left.\begin{array}{rcl} u_4 - u_5 & \leq & -1 \\ u_5 - u_{11} & \leq & -1 \\ u_{11} - u_8 & \leq & -1 \\ u_8 - u_4 & \leq & -1 \end{array}\right\} u_4 - u_4 = 0 \leq -4 \text{ (contradição)}.$$

Logo, o resultado expresso na figura 4.8 não seria uma solução viável de PCV_3. Segundo [167], se as componentes y_{ij} correspondem de fato a um percurso, as variáveis u_{v_i}, $1 \leq i \leq k$, podem ser ajustadas de forma a receber a ordem que os vértices serão visitados dentro desse percurso. Por exemplo, se o vértice 4 for o segundo vértice a ser visitado, então $u_4 = u_{v_2} = 2$ e assim por diante.

Tendo em vista o PCV_3, note que, sem perda de generalidade, podemos reescrever (4.34) da seguinte forma:

$$\min\ f(y) = \sum_{(i,j)\in A} w_{ij} y_{ij} + \left\{ \min \sum_{i\in V} u_i \right\}.$$

Observe que, se considerarmos que cada u_i receberá a ordem em que o vértice i será visitado dentro do percurso, temos que u_i pode ser ajustado de tal forma que

$u_i \in \{0,1,2,\ldots,n-1\}$. Isso significa que, ao chegarmos na solução ótima do PCV$_3$, $\sum_{i\in V} u_i = [n(n-1)]/2$ e, portanto, uma constante.

Temos, então, um subproblema de minimização de PCV$_3$, cuja função-objetivo é $f(u) = \sum_{i\in V} u_i$, e que está sujeito às restrições (4.37) e (4.39). Multiplicando (4.37) por -1, esse subproblema pode ser escrito, em sua forma matricial, como:

$$\min\ f(u) = \sum_{i\in V} u_i, \tag{4.40}$$

sujeito a:

$$\mathcal{A}u \geq 1 - n(1-y_{ij}), \qquad i \in \{1,\ldots,n\}, \quad j \in \{2,\ldots,n\},\ (i \neq j) \tag{4.41}$$

$$u_i \geq 0, \qquad i \in \{1,\ldots,n\}, \tag{4.42}$$

em que a matriz $\mathcal{A}$, com n^2-2n+1 linhas e n colunas, é tal que

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_1 \\ \mathcal{A}_2 \\ \vdots \\ \mathcal{A}_k \\ \vdots \\ \mathcal{A}_n \end{bmatrix},$$

em que $\mathcal{A}_1$, com $n-1$ linhas, é tal que

$$\mathcal{A}_1 = [\ -\mathbf{1}\ \ \mathbf{I}\ ],$$

em que $\mathbf{1}$ é o vetor-coluna com todos os elementos iguais a 1 e $\mathbf{I}$ é a matriz identidade com $n-1$ colunas.

Cada $\mathcal{A}_k$, com $n-2$ linhas, é dada por

$$\mathcal{A}_k = \begin{bmatrix} 0 & 1 & \cdots & 0 & -1 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & -1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & -1 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & -1 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & -1 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix},$$

ou seja, os elementos a_{ij} de $\mathcal{A}_k$ são dados da seguinte forma:

$$a_{ij} = \begin{cases} -1, & \text{se } j = k, \\ 1, & \text{se } j < k \text{ e } j = i+1, \\ 1, & \text{se } j > k \text{ e } j = i+2, \\ 0, & \text{para os demais elementos.} \end{cases}$$

A partir do subproblema de PCV$_3$, podemos obter o seguinte problema dual:

$$\max\ f(z) = z^\top[\mathbf{1} - n(\mathbf{1} - y)], \tag{4.43}$$

sujeito a:

$$z^\top \mathcal{A} \leq \mathbf{1} \tag{4.44}$$

$$z_{ij} \geq 0, \qquad i \in V,\ j \in V - \{1\}, \tag{4.45}$$

em que $\mathbf{1}$ é o vetor-coluna com $n^2 - 2n + 1$ componentes iguais a 1 e z o vetor-coluna de componentes z_{ij} que representam as variáveis duais. Desenvolvendo as operações em (4.43) e em (4.44), podemos reescrever o problema dual acima da seguinte forma:

$$\max\ f(z) = \sum_{i=1}^{n} \sum_{j=2}^{n} [1 - n(1 - y_{ij})] z_{ij}, \quad (i \neq j), \tag{4.46}$$

sujeito a:

$$-\sum_{j=2}^{n} z_{1j} \leq 1 \tag{4.47}$$

$$\sum_{j=1}^{n} z_{ji} - \sum_{j=2}^{n} z_{ij} \leq 1, \qquad i \in V - \{1\},\ i \neq j \tag{4.48}$$

$$z_{ij} \geq 0, \qquad i \in V,\ j \in V - \{1\}. \tag{4.49}$$

Em (4.46), se $y_{ij} = 1$, temos que $[1 - n(1 - y_{ij})] z_{ij} = z_{ij}$, e se $y_{ij} = 0$, temos que $z_{ij} = 0$, pois não há a presença do arco $(i, j) \in A$. Logo, (4.46) pode ser reescrita como

$$\max\ f(z) = \sum_{i=1}^{n} \sum_{j=2}^{n} z_{ij}, \quad (i \neq j). \tag{4.50}$$

O problema dual, visto em (4.46)–(4.50), é utilizado por Gavish e Graves, em [91], para elaborar uma formulação matemática para a solução do PCV, baseada na ideia de fluxo em redes. Nessa formulação, z_{ij} representa o fluxo no arco $(i, j) \in A$ e sua capacidade máxima é dada pela restrição

$$z_{ij} \leq (n - 1) \cdot y_{ij}. \tag{4.51}$$

Assim, o PCV$_3$ pode ser reescrito, com a substituição de seu subproblema, visto em (4.40)–(4.42), pelo seu dual, visto em (4.46)– (4.50), juntamente com a restrição (4.51).

$$(\text{PCV}_{3\text{a}}): \quad \text{Minimizar } f(y) = \sum_{(i,j) \in A} w_{ij} y_{ij}, \tag{4.52}$$

sujeito a:

$$\sum_{j \in V,\ j \neq i} y_{ij} = 1\ , \qquad \forall i \in V, \tag{4.53}$$

$$\sum_{i \in V,\ i \neq j} y_{ij} = 1\ , \qquad \forall j \in V, \tag{4.54}$$

$$-\sum_{j=2}^{n} z_{1j} \leq 1, \tag{4.55}$$

$$\sum_{j=1}^{n} z_{ji} - \sum_{j=2}^{n} z_{ij} \leq 1, \qquad i \in V - \{1\},\ i \neq j, \tag{4.56}$$

$$z_{ij} \leq (n - 1) y_{ij}, \qquad i \in V,\ j \in V - \{1\}, \tag{4.57}$$

$$y_{ij} \in \{0, 1\}, \qquad \forall (i, j) \in A,\ i \neq j, \tag{4.58}$$

$$z_{ij} \geq 0, \qquad i \in V,\ j \in V - \{1\}. \tag{4.59}$$

Pelo Teorema da Dualidade Forte, [154], temos que a solução ótima do problema primal é igual à solução ótima do problema dual. Desta forma, podemos afirmar que, ao atingir o ótimo, (4.50) tem o mesmo valor que (4.40) e, portanto, é constante. Neste caso, não há necessidade de colocar (4.50) na função objetivo de PCV_{3a}. Note que a restrição (4.55) se torna sem efeito na presença de (4.57) e, portanto, pode ser removida. O PCV_{3a} é bastante parecido com a formulação matemática proposta por Gavish e Graves, no ano de 1978, em [91], que pode ser vista no PCV_4 logo abaixo. Nessa mesma referência é provado que o PCV_4 equivale ao PCV.

$$(\text{PCV}_4): \quad \text{Minimizar } f(y) = \sum_{(i,j)\in A} w_{ij} y_{ij}, \tag{4.60}$$

sujeito a:

$$\sum_{j\in V,\ j\neq i} y_{ij} = 1\ , \qquad \forall i \in V, \tag{4.61}$$

$$\sum_{i\in V,\ i\neq j} y_{ij} = 1\ , \qquad \forall j \in V, \tag{4.62}$$

$$\sum_{j=1}^{n} z_{ji} - \sum_{j=2}^{n} z_{ij} = 1, \qquad i \in V - \{1\},\ i \neq j, \tag{4.63}$$

$$z_{ij} \leq (n-1) y_{ij}, \qquad i \in V,\ j \in V - \{1\}, \tag{4.64}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A,\ i \neq j, \tag{4.65}$$

$$z_{ij} \geq 0, \qquad i \in V,\ j \in V - \{1\}. \tag{4.66}$$

A formulação matemática vista em PCV_4 apresenta $n-1$ restrições em (4.63) e $2n^2 - 3n + 1$ restrições em (4.64), totalizando um número polinomial de restrições de eliminação de subciclos igual a $2n^2 - 2n$. Vejamos agora como essa formulação poderia excluir a solução encontrada na figura 4.8. Observando a figura em questão, verificamos que as componentes de y iguais a 1 são: y_{12}, y_{13}, y_{23}, y_{45}, y_{48}, $y_{5,11}$, y_{67}, y_{69}, $y_{7,10}$, $y_{8,11}$, $y_{9,10}$. As demais componentes são iguais a zero. Sendo assim, por (4.64), podemos dizer que $z_{ij} = 0$ para $y_{ij} = 0$. Como o problema da figura 4.8 é simétrico, podemos afirmar também que $z_{ji} = 0$ para $y_{ij} = 0$. Assim, vamos considerar o subconjunto $S = \{6,7,9,10\}$ de V, que se refere ao subciclo encontrado no canto superior à direita dessa figura. Neste caso, então, somente z_{67}, z_{76}, z_{69}, z_{96}, $z_{7,10}$, $z_{10,7}$, $z_{9,10}$, $z_{10,9}$ podem ser diferentes de zero. Considerando esse fato, por (4.65), temos o seguinte sistema de equações:

$$\begin{aligned}
z_{76} + z_{96} - z_{67} - z_{69} &= 1, \\
z_{67} + z_{10,7} - z_{76} - z_{7,10} &= 1, \\
z_{9,10} + z_{7,10} - z_{10,9} - z_{10,7} &= 1, \\
z_{69} + z_{10,9} - z_{96} - z_{9,10} &= 1.
\end{aligned}$$

Esse sistema é impossível. Logo, a solução da figura 4.8 não seria viável para o PCV 4. Alguns autores incluem, ainda, em PCV_4, a restrição

$$\sum_{j=2}^{n} z_{1j} = n - 1,$$

o que completa a formulação matemática de fluxo em rede, em que o somatório dos lados direitos das restrições deve ser igual a zero, veja [217] pág 41.

Em 2003, Maculan, Plateau e Lisser, em [155], propuseram uma formulação matemática que também se utiliza de uma variável que representa um fluxo entre os arcos ou arestas. Essa formulação, vista em PCV_5, é baseada na ideia de fluxos não simultâneos e contém as restrições vistas em (4.17) até (4.23), para obtenção de um ciclo hamiltoniano. Assim, a seguinte formulação matemática pode ser escrita para resolver o PCV:

$$(\text{PCV}_5): \quad \text{Minimizar } f(y) = \sum_{\{i,j\}\in E} w_{ij} y_{ij}, \tag{4.67}$$

sujeito às restrições de
(4.17) a (4.23).

O PCV_5 é usado em caso de problemas simétricos. Para problemas assimétricos, a restrição (4.21) é substituída pelas restrições

$$\sum_{j\in V,\ j\neq i} y_{ij} = 1\ , \qquad \forall i \in V,$$
$$\sum_{i\in V,\ i\neq j} y_{ij} = 1\ , \qquad \forall j \in V,$$

e as restrições (4.20) são substituídas por

$$z_{ij}^k \leqslant y_{ij}, \qquad (i,j)\in A,\ k\in V-\{1\}.$$

Diferente do PCV_4, o PCV_5 apresenta uma variável de fluxo tridimensional z_{ij}^k, onde $k \in V-\{1\}$, que é utilizada na obtenção de um caminho mínimo do vértice 1 ao vértice k. Assim, o ciclo hamiltoniano de menor custo seria uma mesclagem de todos os caminhos mínimos obtidos de 1 até k. Na figura 4.9, vemos a obtenção de cada caminho mínino para cada $k \in V-\{1\}$. É importante observar que, embora a figura 4.9 mostre cada caminho separadamente, estes não são obtidos um de cada vez, mas, sim, através da resolução de um único problema de programação linear. Ou seja, a figura 4.9, de (a) a (l), representa a solução ótima de um único problema aplicado ao grafo da figura 4.1, cuja formulação matemática podemos ver no PCV_5. Desta forma, a figura 4.9, de (a) a (g), informa os valores das variáveis z_{ij}^k. Apesar da não obrigatoriedade de integralidade, vide restrições (4.20), temos que as variáveis $z_{ij}^k \in \{0,1\}$ no ótimo. Uma interpretação para esse fato seria: $z_{ij}^k = 1$, caso haja fluxo do vértice i ao vértice j no caminho de 1 a k. Por exemplo, na figura 4.9(d), existe uma aresta ativa (em vermelho) de 2 a 4, no menor caminho de 1 a 5. Isso significa que a variável $z_{24}^5 = 1$.

Podemos observar, nesta última formulação matemática, que há um número polinomial de restrições. Temos $n^2 - n$ restrições de (4.17) a (4.19) e $2|E|(n-1)$ restrições do tipo (4.20), que são utilizadas na eliminação de subciclos em PCV_5.

4.5 Árvores

Um subgrafo de G é chamado árvore se for conexo e não apresentar ciclos. Desta forma, considere p vértices de G e as equações a seguir:

$$\sum_{i\in V} x_i = p, \tag{4.68}$$

$$\sum_{\{i,j\}\in E} y_{ij} = p-1, \tag{4.69}$$

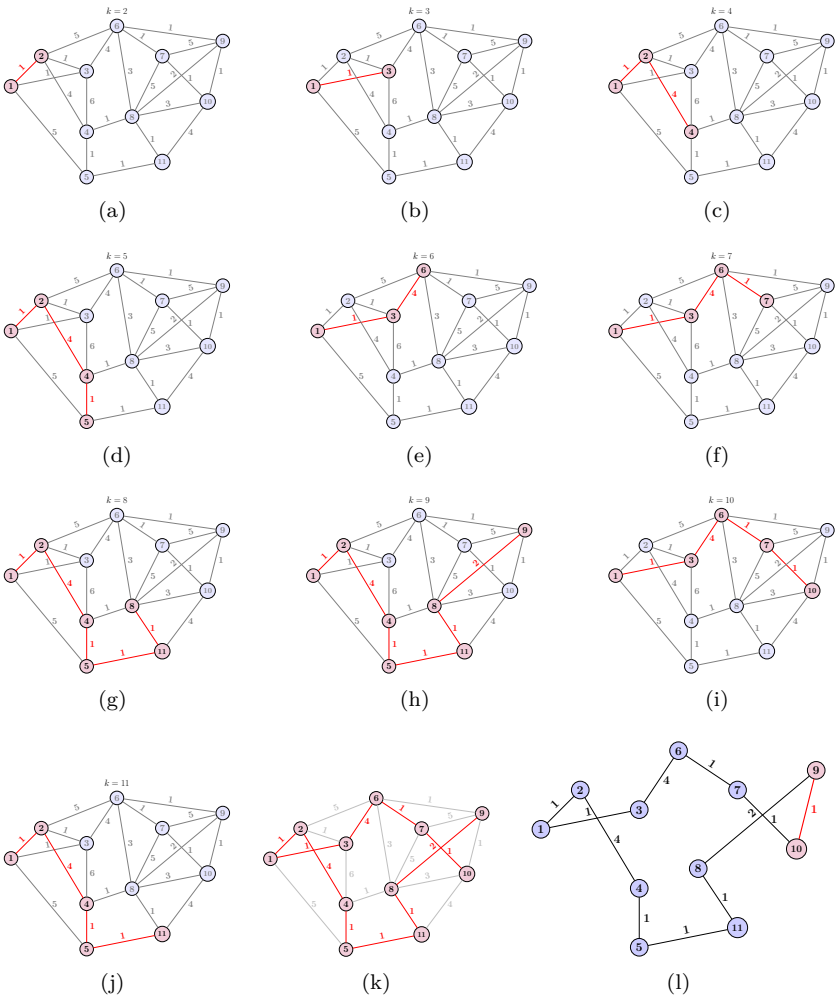


Figura 4.9: Conjunto de caminhos de 1 a k, de onde se obtém um ciclo hamiltoniano de menor custo. A figura 4.9(k) representa a mesclagem de todos os caminhos obtidos. Note que a presença da aresta $\{9, 10\}$, na figura 4.9(l), se deve à restrição (4.21). A figura 4.9(l) representa a solução ótima do PCV_5 para o grafo da figura 4.1, considerando-se todas as componentes c_i do vetor c iguais a zero.

onde $p \leq |V|$. Todos os vetores x e y que satisfazem às restrições de (4.1) a (4.4), de (4.6) a (4.8), (4.68) e (4.69) estão associados a uma subárvore de G com p vértices.

4.5.1 Árvores geradoras

Seja $G = (V,E)$ um grafo conexo, definimos $T = (V,E_1)$ um subgrafo conexo e sem ciclos de G. O subgrafo T é, por definição, uma árvore geradora de G, assim sendo $p = |V|$. Podemos obter uma formulação matemática para a obtenção de uma árvore geradora utilizando as restrições vistas na seção 4.3, usadas na obtenção de ciclos hamiltonianos. Para isso substituiremos a restrição (4.21) pela equação (4.69). Assim, temos a seguinte formulação:

$$\sum_{j\in\Gamma^+(1)} z^k_{1j} - \sum_{j\in\Gamma^-(1)} z^k_{j1} = 1, \qquad k \in V - \{1\} \tag{4.70}$$

$$\sum_{j\in\Gamma^+(i)} z^k_{ij} - \sum_{j\in\Gamma^-(i)} z^k_{ji} = 0, \qquad i \in V - \{1,k\},\ k \in V - \{1\} \tag{4.71}$$

$$\sum_{j\in\Gamma^+(k)} z^k_{kj} - \sum_{j\in\Gamma^-(k)} z^k_{jk} = -1, \qquad k \in V - \{1\} \tag{4.72}$$

$$z^k_{ij} \leqslant y_{ij} \text{ e } z^k_{ji} \leqslant y_{ij}, \qquad \{i,j\} \in E,\ k \in V - \{1\} \tag{4.73}$$

$$\sum_{\{i,j\}\in E} y_{ij} = n - 1, \qquad i, j \in V \tag{4.74}$$

$$z^k_{ij} \geqslant 0, \qquad (i,j) \in A,\ k \in V - \{1\} \tag{4.75}$$

$$y_{ij} \in \{0,1\}, \qquad \{i,j\} \in E, \tag{4.76}$$

onde $n = |V|$. Um vetor y que satisfaz às restrições de (4.70) a (4.76) está associado a uma árvore geradora de G.

4.5.2 Problema da Árvore de Steiner em grafos

Dado um grafo conexo e não orientado G com pesos não negativos atribuídos às arestas, e seja $\overline{V} \subset V$ o conjunto de vértices obrigatórios. O Problema da Árvore Mínima de Steiner em grafos consiste em encontrar uma árvore de peso mínimo que contenha todos os vértices do conjunto $\overline{V}$ utilizando, quando necessário, vértices pertencentes ao conjunto $V - \overline{V}$. Os vértices pertencentes a $V - \overline{V}$ que foram utilizados na obtenção dessa árvore são denominados vértices de Steiner. A formulação matemática para esse problema é apresentada a seguir, onde $s \in \overline{V}$.

$$(\text{PAS}_1): \quad \min\ f(y) = \sum_{\{i,j\}\in E} w_{ij} y_{ij}, \tag{4.77}$$

sujeito a:

$$\sum_{j\in\Gamma^+(s)} z^k_{sj} - 1 = 0, \qquad k \in \overline{V} - \{s\} \tag{4.78}$$

$$\sum_{j\in\Gamma^+(i)} z^k_{ij} - \sum_{j\in\Gamma^-(i)} z^k_{ji} = 0, \qquad i \in V - \{s,k\},\ k \in \overline{V} - \{s\} \tag{4.79}$$

$$\sum_{j\in\Gamma^+(k)} z^k_{kj} - \sum_{j\in\Gamma^-(k)} z^k_{jk} + 1 = 0, \qquad k \in \overline{V} - \{s\} \tag{4.80}$$

$$z^k_{ij} \le y_{ij} \ \text{ e } \ z^k_{ji} \le y_{ij}, \qquad \{i,j\} \in E,\ k \in \overline{V} - \{s\} \tag{4.81}$$

$$z^k_{ij} \ge 0, \qquad (i,j) \in A,\ k \in \overline{V} - \{s\} \tag{4.82}$$

$$y_{ij} \in \{0,1\}, \qquad \{i,j\} \in E. \tag{4.83}$$

A função-objetivo (4.77) minimiza o peso da Árvore de Steiner. As equações (4.78), (4.79) e (4.80) representam a conservação de fluxo para o vértice s, escolhido para ser a origem do fluxo, para os vértices não terminais e para os vértices terminais, respectivamente. As restrições (4.81) permitem um fluxo não nulo z^k_{ij} ou z^k_{ji} passar por um arco (i,j) pertencente ao conjunto de arcos A somente se a aresta $\{i,j\}$ pertencente ao conjunto de arestas E estiver incluída na solução. As restrições (4.82) e (4.83), respectivamente, definem como contínuas as variáveis de fluxo z^k_{ij} e como binárias as variáveis y_{ij} relativas às arestas.

A formulação matemática vista em (4.77) - (4.83) é voltada para problemas não orientados, isto é, em que o valor da variável y_{ij} é o mesmo para os arcos (i,j) e (j,i). De acordo com [7], podemos ainda substituir a restrição (4.81) por

$$z^k_{ij} + z^k_{ji} \le y_{ij},\ \{i,j\} \in E,\ k \in \overline{V} - \{s\}. \tag{4.84}$$

A desigualdade (4.84) significa que o total de fluxos simultâneos em sentidos contrários referentes a uma aresta $\{i,j\} \in E$ não deve ultrapassar a capacidade dessa aresta. A formulação matemática apresentada a seguir também descreve um fluxo não orientado.

$$(\text{PAS}_2): \quad \min\ f(y) = \sum_{\{i,j\}\in E} w_{ij} y_{ij}, \tag{4.85}$$

sujeito a:

$$\sum_{j\in\Gamma^+(s)} z^k_{sj} - 1 = 0, \qquad k \in \overline{V} - \{s\} \tag{4.86}$$

$$\sum_{j\in\Gamma^+(i)} z^k_{ij} - \sum_{j\in\Gamma^-(i)} z^k_{ji} = 0, \qquad i \in V - \{s,k\},\ k \in \overline{V} - \{s\} \tag{4.87}$$

$$\sum_{j\in\Gamma^+(k)} z^k_{kj} - \sum_{j\in\Gamma^-(k)} z^k_{jk} + 1 = 0, \qquad k \in \overline{V} - \{s\} \tag{4.88}$$

$$z^k_{ij} + z^k_{ji} \le y_{ij}, \qquad \{i,j\} \in E,\ k \in \overline{V} - \{s\} \tag{4.89}$$

$$z^k_{ij} \ge 0, \qquad (i,j) \in A,\ k \in \overline{V} - \{s\} \tag{4.90}$$

$$y_{ij} \in \{0,1\}, \qquad \{i,j\} \in E. \tag{4.91}$$

Na próxima formulação matemática a ser apresentada, as variáveis $y_{ij} \in \{0,1\}$ serão associadas aos arcos $(i,j) \in A$.

$$(\text{PAS}_3): \quad \min\ f(y) = \sum_{(i,j)\in A} w_{ij} y_{ij}, \tag{4.92}$$

sujeito a:

$$\sum_{j\in\Gamma^+(s)} z^k_{sj} - 1 = 0, \qquad k \in \overline{V} - \{s\} \tag{4.93}$$

$$\sum_{j\in\Gamma^+(i)} z^k_{ij} - \sum_{j\in\Gamma^-(i)} z^k_{ji} = 0, \qquad i \in V - \{s,k\},\ k \in \overline{V} - \{s\} \tag{4.94}$$

$$\sum_{j\in\Gamma^+(k)} z^k_{kj} - \sum_{j\in\Gamma^-(k)} z^k_{jk} + 1 = 0, \qquad k \in \overline{V} - \{s\} \tag{4.95}$$

$$z^k_{ij} \le y_{ij}, \qquad (i,j) \in A,\ k \in V - \{s\} \tag{4.96}$$

$$z^k_{ij} \ge 0, \qquad (i,j) \in A,\ k \in \overline{V} - \{s\} \tag{4.97}$$

$$y_{ij} \in \{0,1\}, \qquad (i,j) \in A. \tag{4.98}$$

Observe que tanto a função-objetivo de PAS_3 como as suas restrições (4.93)-(4.98) têm finalidades semelhantes às de PAS_1 e PAS_2, com a diferença de explicitarem a orientação do fluxo.

Para encontrarmos as relaxações lineares dos problemas PAS_1 e PAS_2, trocamos $y_{ij} \in \{0,1\},\ \{i,j\} \in E$ por $0 \le y_{ij} \le 1$. A relaxação linear do problema PAS_3 é obtida ao substituirmos $y_{ij} \in \{0,1\},\ (i,j) \in A$ por $0 \le y_{ij} \le 1$.
Denominemos essa relaxação de $\text{RPAS}_i,\ i = 1,2,3$. Resultados teóricos sobre as relaxações lineares de PAS_1, PAS_2 e PAS_3 podem ser vistos em [7], assim como os resultados computacionais que serão apresentados a seguir para as referidas formulações. As instâncias b, c, d e e utilizadas foram obtidas da biblioteca STP OR-Library (J.E. Beasley), que disponibiliza 78 instâncias de problemas em grafos não orientados (simétricos), das quais 48 foram executadas com sucesso e as 30 restantes excederam a capacidade de memória do computador utilizado. Os testes foram executados com a utilização de um computador Intel Xeon^{TM} Processador (3.07 GHz, 48GB RAM), com sistema operacional Ubuntu 12.04 LTS. Para a modelagem, utilizou-se AMPL versão 20150721 (Linux 64 bits), e para a resolução o software IBM ILOG CPLEX Optimization Studio Interactive Optimizer 12.6.0.0.

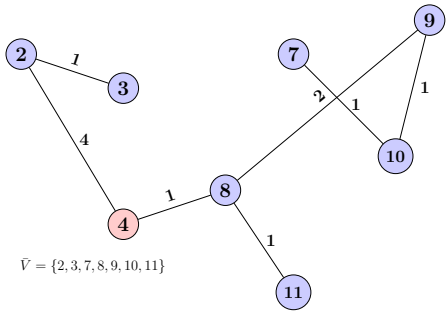
Podemos sempre transformar esses problemas em problemas em grafos orientados, como foi feito no início deste capítulo para transformar G em G_d. A aresta $\{i,j\}$ será substituída por dois arcos: (i,j) e (j,i). As variáveis y_{ij} agora serão definidas para $(i,j) \in A$, em que A é formado por todos os arcos que substituíram as arestas de E. Os pesos serão da forma $w_{ij} = w_{ji}$. Dobramos o número de variáveis bivalentes ($\{0,1\}$). Seja $val(.)$ o valor ótimo da função-objetivo do problema (.) . Como foi relembrado em [7], temos que

$$val(\text{RPAS}_1) \le \text{val}(\text{RPAS}_2) \le \text{val}(\text{RPAS}_3).$$

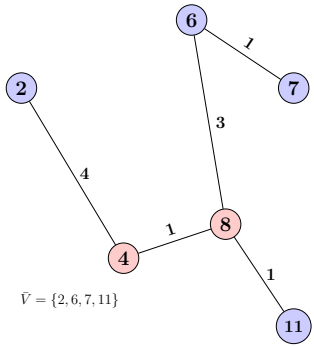
Logo, a relaxação $val((\text{PAS}_3))$ será a mais promissora, poderá oferecer uma melhor cota inferior para o problema (PAS).
As formulações PAS_3 e RPAS_3 encontraram a solução inteira ótima para 47 das 48 instâncias, e a instância $e7$ foi resolvida somente pela formulação PAS_3. Em todas as tabelas a seguir, os tempos em negrito indicam qual formulação foi mais rápida.

A tabela 4.1 mostra que o PAS_3 Relaxado foi mais rápido do que o PAS_3 em 55,56% das instâncias. Podemos ver nas tabelas 4.2, 4.3 e 4.4 que a formulação PAS_3 Relaxado teve um desempenho melhor do que na tabela 4.1, encontrando a solução ótima em menor tempo do que PAS_3 para, respectivamente, 63,16%, 100% e 75% das instâncias.

Lembremos que essa comparação pelo tempo não tem muito sentido, pois a resolução dos problemas RPAS_3 é feita sem nenhum pré-tratamento e não visa à busca de uma solução inteira. Já a solução dos problemas PAS_3 visa a encontrar uma solução inteira, para isso o CPLEX possui técnicas de programação inteira, que serão



(a)



(b)

Figura 4.10: Nas figuras 4.10(a) e 4.10(b), vemos duas árvores mínimas de Steiner obtidas como solução da formulação (4.77)–(4.83), utilizando-se como entrada o grafo G da figura 4.1. Os vértices em vermelho representam vértices de Steiner.

estudadas neste livro e que são utilizadas antes mesmo da busca de uma solução do problema relaxado.

O importante é verificar que em apenas um dos exemplos solucionados a relaxação não forneceu valores inteiros para os y_{ij}, $(i, j) \in A$. Nessa biblioteca, todos os custos w_{ij}, $(i, j) \in A$, são positivos e inteiros. No exemplo $c18$ forneceu $val(RPAS_3) = 112.21$ e $val(PAS_3) = 113$. No entanto, $val(RPAS_3)$ é uma cota inferior para $val(PAS_3)$, logo, o menor inteiro maior ou igual a $112,21$, que é 113, o será.

Instâncias b	Solução ótima	Tempo PAS_3 (seg)	Tempo $RPAS_3$ (seg)
b1	82	0.029816	0.011383
b2	83	0.017348	0.015156
b3	138	0.030536	0.030129
b4	59	0.025302	0.026025
b5	61	0.026231	0.027734
b6	122	0.123923	0.124025
b7	111	0.044627	0.030505
b8	104	0.034261	0.043912
b9	220	0.088182	0.085621
b10	86	0.087335	0.055728
b11	88	0.101192	0.103585
b12	174	0.481558	0.454020
b13	165	0.445830	0.438060
b14	235	0.111448	0.107215
b15	318	0.145138	0.149599
b16	127	0.128929	0.131864
b17	131	0.202319	0.203120
b18	218	0.605989	0.599645

Tabela 4.1: Resultados computacionais das formulações PAS_3 e PAS_3 Relaxado para instâncias STP OR-Library *b*.

Instâncias c	Solução ótima	Tempo STP_3 (seg)	Tempo $RSTP_3$ (seg)
c1	85	0.14	0.06
c2	144	0.49	0.29
c3	754	5.66	6.80
c4	1079	23.60	22.98
c5	1579	60.81	71.32
c6	55	0.69	0.18
c7	102	1.49	1.44
c8	509	149.86	205.77
c9	707	1,004.31	2,218.25
c10	1093	1,865.51	575.52
c11	32	2.85	1.52
c12	46	15.91	7.62
c13	258	4,856.99	8,440.30
c14	323	4,130.90	10,246.80
c15	556	7,076.49	119,248.00
c16	11	12.74	5.94
c17	18	57.15	39.41
c18	113 ($LSTP_3$=112.21)	48,009.80	31,724.10
c19	146	361,601.00	110,588.00

Tabela 4.2: Resultados computacionais das formulações PAS_3 e PAS_3 Relaxado para instâncias STP OR-Library *c*.

Instâncias d	Solução ótima	Tempo STP_3 (seg)	Tempo $RSTP_3$ (seg)
d1	106	1.3730	0.2899
d2	220	1.9243	0.3405
d6	67	6.1254	1.2607
d7	103	11.7520	2.7777
d11	29	8.1311	5.7592
d12	42	49.7962	37.5852
d16	13	30.6025	22.1697

Tabela 4.3: Resultados computacionais das formulações PAS_3 e $RPAS_3$ para instâncias STP OR-Library *d*.

Instâncias e	Solução ótima	Tempo STP_3 (seg)	Tempo $LSTP_3$ (seg)
e1	11	1.0574	0.5481
e2	214	3.6418	1.7713
e6	73	10.0968	4.0827
e7	145	84.6429	Falta de memória

Tabela 4.4: Resultados computacionais das formulações PAS_3 e PAS_3 Relaxado para instâncias STP OR-Library *e*.

5

Geração de Colunas em Otimização Inteira

Luidi Simonetti
Nelson Maculan
Ana Flávia Uzeda Macambira
Pedro Henrique González

5.1 Geração de Colunas em Programação Linear

Considere o problema de programação linear (PL) definido no capítulo 1. A região viável $X = \{x \in \mathbb{R}^n : Ax = b,\ x \geq 0\}$, de (PL), é um espaço convexo limitado ou não. Caso tenhamos um espaço ilimitado, a região viável do problema pode ser descrita por uma combinação convexa de vértices (pontos extremos) e raios extremos, conforme ilustrado na figura 5.1.

Sendo $\mathcal{V} = \{v_1, v_2, ..., v_p\}$ o conjunto de vértices e $\mathcal{R} = \{r_1, r_2, ..., r_q\}$ o conjunto de raios, de acordo com o Teorema de Minkowiski [116, 154], podemos representar matematicamente $x \in X$ como:

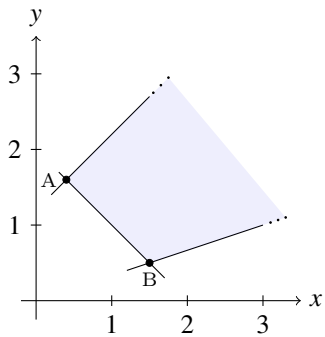
$$x = \sum_{v_j \in \mathcal{V}} \lambda_j v_j + \sum_{r_i \in \mathcal{R}} \mu_i r_i, \qquad \sum_{j=1}^{p} \lambda_j = 1 \text{ e } \mu_i \geq 0,\ i = 1, ..., q. \tag{5.1}$$

Escrevendo x como na equação (5.1), já estamos satisfazendo $Ax = b$ e $x \geq 0$. Caso o espaço convexo seja limitado, podemos representar (PL) utilizando apenas o conjunto de vértices.

$$(PL1): \text{Minimizar} \quad z = c^\top (\sum_{v_j \in \mathcal{V}} \lambda_j v_j), \tag{5.2}$$

$$\text{sujeito a:} \sum_{j=1}^{p} \lambda_j = 1, \tag{5.3}$$

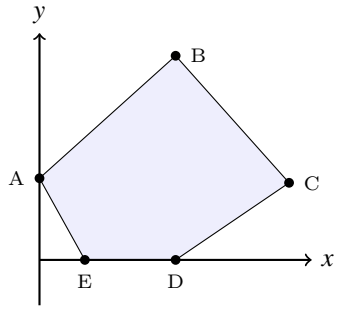
$$\lambda_j \geq 0, \quad j = 1, 2, ..., p. \tag{5.4}$$



A figura ao lado ilustra uma região viável ilimitada. Um dos raios extremos tem base no ponto A, de coordenadas (0.4,1.6) e direção (t, t+1.2), e o outro raio extremo tem base no ponto B, de coordenadas (1.5,0.5) e direção (t, 0.33335t).

Figura 5.1: Raios extremos.

Observe que as restrições $Ax = b$ e $x \geq 0$ não necessitam aparecer em (PL1), pois a representação das variáveis x, como visto na equação (5.1), já descreve a região viável definida por elas. Uma representação geométrica de espaço convexo limitado pode ser vista na figura 5.2.



A figura ao lado representa um espaço convexo limitado. Os pontos A, B, C, D e E são seus pontos extremos. Com eles podemos representar qualquer ponto viável (região cinza).

Figura 5.2: Região viável limitada.

Quando o espaço convexo é ilimitado, a representação se faz com a utilização de vértices e raios extremos, como apresentado a seguir.

$$(PL2): \text{Minimizar} \quad z = c^\top \Big(\sum_{v_j \in \mathcal{V}} \lambda_j v_j + \sum_{r_i \in \mathcal{R}} \mu_i r_i \Big), \tag{5.5}$$

$$\text{sujeito a:} \sum_{j=1}^{p} \lambda_j = 1, \tag{5.6}$$

$$\lambda_j \geq 0, \quad j = 1, 2, ..., p, \tag{5.7}$$

$$\mu_i \geq 0, \quad i = 1, 2, ..., q. \tag{5.8}$$

Vale ressaltar que (PL1) e (PL2) são problemas equivalentes a (PL), sendo a forma de representação da região viável deste último a única diferença existente entre eles.

As reformulações (PL1) (5.2)-(5.4) e (PL2) (5.5)-(5.8) não são muito interessantes para representar todo o conjunto de restrições, pois nestes casos não é necessário resolver (PL1) e (PL2). Para (PL1), pelo Teorema dos Pontos Extremos, em teoria bastaria inspecionarmos cada vértice para encontrar aquele que minimiza o valor da função-objetivo, ou seja, o ponto ótimo. No caso de (PL2), além de, em tese, termos de inspecionar cada vértice, teríamos que verificar a existência de alguma direção de decréscimo (raio extremo), e, no caso de a encontrarmos, o problema seria ilimitado. No entanto, vale ressaltar que o número de vértices pode ser de ordem exponencial, o que impossibilitaria a inspeção dos vértices um a um.

É mais interessante representar apenas parte do conjunto de restrições como vértices e raios extremos e deixar a outra parte com a representação original. Formalizando, suponha que dividamos $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ e $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$. Os critérios de escolha para a divisão da matriz A serão discutidos ao longo deste capítulo (ver também [12]). Assim, temos que $Ax = b$ pode ser escrito como:

$$
\begin{aligned}
(PL') : \text{Minimizar} \quad & z = c^\top x, && (5.9)\\
\text{sujeito a:} \quad & A_1 x = b_1, && (5.10)\\
& A_2 x = b_2, && (5.11)\\
& x \geq 0. && (5.12)
\end{aligned}
$$

Para simplificarmos, consideraremos que a região viável determinada por $X_1 = \{x \in \mathbb{R}^n : A_1 x = b_1,\ x \geq 0\}$ seja um politipo. A figura 5.3 ilustra a divisão do conjunto de restrições, onde a região cinza representa as restrições $A_2 x = b_2$ e os pontos de A a E são os pontos extremos de X_1.

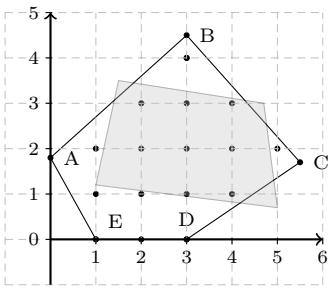


Figura 5.3: Divisão do conjunto de restrições.

Representando o conjunto de restrições (5.10) como uma combinação convexa de vértices, temos:

$$
\begin{aligned}
(PL1') : \text{Minimizar} \quad & z = c^\top \Big(\sum_{v_j \in \mathcal{V}} \lambda_j v_j \Big),\\
\text{sujeito a:} \quad & A_2 \Big(\sum_{v_j \in \mathcal{V}} \lambda_j v_j \Big) = b_2,\\
& \sum_{j=1}^{p} \lambda_j = 1,\\
& \lambda_j \geq 0, \quad j = 1, 2, ..., p,
\end{aligned}
$$

ou ainda

$$(PL1') : \text{Minimizar} \quad z = \sum_{v_j \in \mathcal{V}} (c^\top v_j)\lambda_j, \tag{5.13}$$

$$\text{sujeito a:} \quad \sum_{v_j \in \mathcal{V}} (A_2 v_j)\lambda_j = b_2, \tag{5.14}$$

$$\sum_{j=1}^{p} \lambda_j = 1, \tag{5.15}$$

$$\lambda_j \geq 0, \quad j = 1, 2, ..., p. \tag{5.16}$$

A representação apresentada em (PL1') (5.13)-(5.16) é a reformulação Dantzig-Wolfe [58] para o problema (PL') (5.9)-(5.12).

Exemplo 1.

$$\text{Minimizar} \quad -x_1 - x_2,$$

$$\text{sujeito a:} \quad x_1 - x_2 \leq 2, \tag{5.17}$$

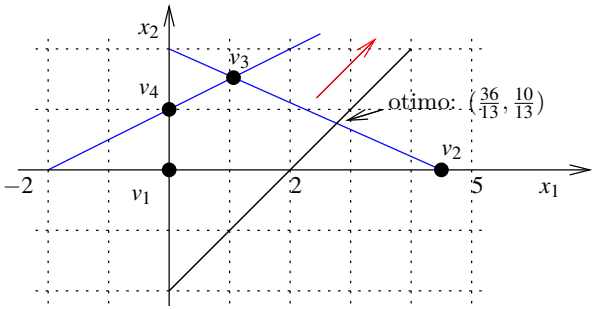
$$4x_1 + 9x_2 \leq 18, \tag{5.18}$$

$$-2x_1 + 4x_2 \leq 4, \tag{5.19}$$

$$x_1, x_2 \geq 0.$$

$A_2 x = b_2$ é representado pela restrição (5.17) e $A_1 x = b_1$, pelas restrições (5.18) e (5.19). Os pontos extremos do politopo $\{x \in \mathbb{R}^2_+ \mid A_1 x = b_1\}$ são:

$$v_1 = (0,0), \quad v_2 = (\frac{9}{2}, 0), \quad v_3 = (\frac{18}{17}, \frac{26}{17}) \quad \text{e } v_4 = (0,1).$$



Sendo assim, reescrevendo o problema em função dos vértices, temos que todo ponto do politopo é dado por:

$$(x_1, x_2) = (0,0)\lambda_1 + (\frac{9}{2}, 0)\lambda_2 + (\frac{18}{17}, \frac{26}{17})\lambda_3 + (0,1)\lambda_4,$$

com $\lambda_i \geq 0$ para todo $i \in \{1,2,3,4\}$ e $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$. Como $A_2 = [1, -1]$, temos que $A_2 v_i$ é dado por:

$$A_2 v_1 = [1,-1][0,0]^\top = 0, \quad A_2 v_2 = [1,-1][\frac{9}{2}, 0]^\top = \frac{9}{2}$$

$$A_2 v_3 = [1,-1][\frac{18}{17}, \frac{26}{17}]^\top = -\frac{8}{17}, \quad A_2 v_4 = [1,-1][0,1]^\top = -1$$

Para os coeficientes da função-objetivo, temos:

$$c^\top v_1 = [-1,-1][0,0]^\top = 0, \quad c^\top v_2 = [-1,-1][\frac{9}{2}, 0]^\top = -\frac{9}{2}$$

$$c^\top v_3 = [-1,-1][\frac{18}{17},\frac{26}{17}]^\top = -\frac{44}{17}, \quad c^\top v_4 = [-1,-1][0,1]^\top = -1$$

Como resultado das reformulações feitas acima, temos o modelo:

$$\begin{aligned} \text{Minimizar} \quad & 0\lambda_1 - \frac{9}{2}\lambda_2 - \frac{44}{17}\lambda_3 - \lambda_4, \\ \text{sujeito a:} \quad & 0\lambda_1 + \frac{9}{2}\lambda_2 - \frac{8}{17}\lambda_3 - \lambda_4 \leq 2, \\ & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1, \\ & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0. \end{aligned}$$

A solução ótima do modelo é:
$\lambda_1 = \lambda_4 = 0$, $\lambda_2 = 0.497$ e $\lambda_3 = 0.503$ e o valor ótimo da função-objetivo é -3.538. A solução ótima do problema original é:
$x_1^* = \frac{36}{13} \simeq 2.769$ e $x_2^* = \frac{10}{13} \simeq 0.769$. Portanto, temos que (x_1^*, x_2^*) não é ponto extremo de $\{x \in \mathbb{R}^2_+ \mid A_1 x = b_1\}$.

Em (PL1'), descrito em (5.13)-(5.16), as restrições (5.14) são chamadas restrições acopladoras. Se A_1 tem dimensões $m_1 \times n$ e A_2 é uma matriz $m_2 \times n$, então o problema (PL') (5.9)-(5.12) possui $m_1 - 1$ restrições a mais do que (PL1'), que é a reformulação Dantzig-Wolfe para (PL'). Por outro lado, o número de variáveis em (PL1') é igual ao número de pontos extremos de $\{x \in \mathbb{R}^n_+ \ : \ A_2 x = b_2\}$, o que significa possivelmente um número exponencial de variáveis.

Suponha que desejemos resolver um problema de programação linear com um número de variáveis muito grande ou que tenhamos reformulado (PL') (5.9)-(5.12) utilizando vértices, vide (PL1') (5.13)-(5.16), ou vértices e raios. Se utilizarmos o algoritmo Simplex para resolver o problema em questão, o passo relativo ao cálculo dos custos reduzidos de todas as variáveis não básicas em cada iteração será altamente custoso pela peculiaridade do problema ou pelo número muito grande de variáveis. A melhor forma de resolver esse tipo de problema é aplicando uma das técnicas de geração de colunas.

As técnicas de geração automática de colunas em programação linear foram apresentadas primeiramente nos trabalhos [58], [97], e [98]. Nessas técnicas, as colunas não básicas não são inspecionadas uma a uma, e sim de forma implícita. Para essa inspeção utilizamos algum algoritmo que tenta encontrar alguma coluna com custo reduzido negativo. Se não existir nenhuma coluna com custo reduzido negativo, o algoritmo não retorna nada. O fato de não haver mais nenhuma coluna capaz de contribuir para a melhoria do valor da função-objetivo do problema reformulado significa que a solução ótima do problema original foi alcançada, mesmo sem a necessidade de gerar todas as colunas de forma explícita. As colunas básicas e as colunas não básicas que foram inspecionadas e acrescentadas ao problema constituem uma pequena parte do conjunto de todas as colunas do problema original e fazem parte do que chamamos de problema mestre restrito (PMR). O algoritmo utilizado para inspecionar implicitamente as colunas do problema original é denominado problema auxiliar (PA).

Considere $\mathcal{V}' \subset \mathcal{V}$ um subconjunto de colunas do problema original que inclui, no mínimo, um conjunto de variáveis básicas relacionadas a uma solução básica viável

do problema original. Sendo assim, temos o problema mestre restrito

$$(PMR): \text{Minimizar} \quad \bar{z} = \sum_{v_j \in \mathcal{V}'} (c^\top v_j)\lambda_j, \tag{5.20}$$

$$\text{sujeito a:} \quad \sum_{v_j \in \mathcal{V}'} (A_2 v_j)\lambda_j = b_2, \tag{5.21}$$

$$\sum_{j: v_j \in \mathcal{V}'} \lambda_j = 1, \tag{5.22}$$

$$v_j \geq 0, \quad v_j \in \mathcal{V}'. \tag{5.23}$$

A escolha desse subconjunto de colunas pode ser feito com colunas artificiais, heurísticas ou soluções "warm start". Um ponto importante é que o PMR gera um limite superior, (UB), de valor $\bar{z}$, para a solução ótima do problema original z^*. Logo, $\bar{z} \geq z^*$.

O problema auxiliar tem como objetivo encontrar a coluna $v \in \mathcal{V} \setminus \mathcal{V}'$ com o menor custo reduzido possível. Sejam μ e σ as variáveis duais referentes às restrições (5.21) e (5.22), respectivamente. Então, o custo reduzido da coluna v pode ser escrito como $(c^\top - \mu A_2)v - \sigma$. Se a coluna com o menor custo reduzido for negativa, adicionamos a nova coluna ao problema mestre restrito; caso contrário, provamos que não existe nenhuma coluna com custo reduzido negativo e, sendo assim, resolvemos o problema original. Logo, podemos escrever o problema auxiliar como:

$$(PA): \text{Minimizar} \quad cr = (c^\top - \mu A_2)v - \sigma, \tag{5.24}$$

$$\text{sujeito a:} \quad A_1 v = b_1, \tag{5.25}$$

$$v \geq 0. \tag{5.26}$$

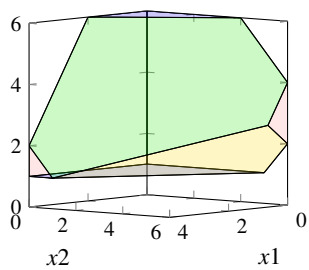
Seja cr^* o valor da solução ótima de (PA). Se $cr^* \geq 0$, temos $\bar{z} = z^*$. Caso contrário, achamos uma nova coluna para adicionar ao PMR. Como $\sum_{j: v_j \in \mathcal{V}} \lambda_j = 1$, podemos estimar um limite inferior (LB) para z^* em cada iteração:

$$\bar{z} + cr^* \leq z^* \leq \bar{z} \tag{5.27}$$

Vale ressaltar que no exemplo 2, temos um problema de maximização, portanto a cada iteração teremos um limite superior (UB) para o valor ótimo do problema original, obtido quando calculamos $\bar{z} + cr^*$.

Exemplo 2.

$$\begin{aligned} \text{Maximizar} \quad & 2x_1 + 4x_2 + x_3, \\ \text{sujeito a:} \quad & 2x_1 + x_2 + x_3 \leq 10, \\ & x_1 + x_2 - x_3 \leq 4, \\ & 0 \leq x_1 \leq 4, \ 0 \leq x_2 \leq 6, \ 1 \leq x_3 \leq 6. \end{aligned}$$



Problema mestre:

$$\text{Maximizar} \quad z = \sum_{j=1}^{p_k} (c^\top v_j)\lambda_j, \tag{5.28}$$

$$\text{sujeito a:} \quad \sum_{j=1}^{p_k} (A_{2_1} v_j)\lambda_j \leq 10, \tag{5.29}$$

$$\sum_{j=1}^{p_k} (A_{2_2} v_j)\lambda_j \leq 4, \tag{5.30}$$

$$\sum_{j=1}^{p_k} \lambda_j = 1. \tag{5.31}$$

Considere μ_1, μ_2 e σ as variáveis duais relativas às restrições 5.29, 5.30 e 5.31 respectivamente.

Problema auxiliar:

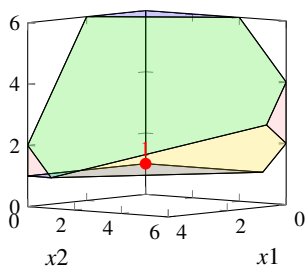
$$\begin{aligned} \text{Maximizar} \quad & cr = (2-2\mu_1-\mu_2)x_1 + (4-\mu_1-\mu_2)x_2 + (1-\mu_1+\mu_2)x_3 - \sigma, \\ \text{sujeito a:} \quad & 0 \leq x_1 \leq 4,\ 0 \leq x_2 \leq 6,\ 1 \leq x_3 \leq 6. \end{aligned}$$

A cada iteração temos que $\bar{z}$ é o valor ótimo da função-objetivo do problema mestre restrito e cr^* é o valor ótimo da função-objetivo do problema auxiliar. A cada iteração, apresentaremos os valores de $\bar{z}$, λ_i, μ_1, μ_2 e σ relativos ao problema mestre restrito e os valores de x_1, x_2, x_3 relativos ao problema auxiliar, além do valor do UB.

Primeira iteração

Seja a coluna 1 dada por $x_1 = x_2 = 0$, $x_3 = 1$. Essa coluna corresponde ao vértice indicado pelo ponto na figura abaixo.
Problema mestre para a coluna 1:

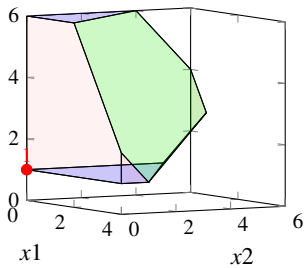
$$\begin{aligned} \text{Maximizar} \quad & z = 1\lambda_1, \\ \text{sujeito a:} \quad & 1\lambda_1 \leq 10, \\ & -1\lambda_1 \leq 4, \\ & \lambda_1 = 1, \end{aligned}$$



$\bar{z} = 1$, $\lambda_1 = 1$ e $\mu_1 = \mu_2 = \sigma = 0$.

Problema auxiliar:

$$\begin{aligned} \text{Maximizar} \quad & cr = 2x_1 + 4x_2 + 1x_3, \\ \text{sujeito a: } & 0 \le x_1 \le 4,\ 0 \le x_2 \le 6,\ 1 \le x_3 \le 6, \end{aligned}$$

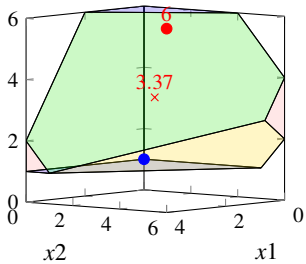


$cr^* = 38$, $x_1 = 4$, $x_2 = x_3 = 6$. UB $= 1 + 38 = 39$. Logo, resolvendo o problema auxiliar para a coluna 1, chegamos à coluna 2, $v_2^\top = (4,6,6)$, de forma que os coeficientes para λ_2 na função-objetivo e nas restrições do problema mestre da segunda iteração serão, respectivamente:

$$c^\top v_2 = (2,4,1)^\top \begin{pmatrix} 4 \\ 6 \\ 6 \end{pmatrix} = 38,\ A_{2_1} v_2 = (2,1,1)^\top \begin{pmatrix} 4 \\ 6 \\ 6 \end{pmatrix} = 20,\ A_{2_2} v_2 = (1,1,-1)^\top \begin{pmatrix} 4 \\ 6 \\ 6 \end{pmatrix} = 4.$$

Segunda iteração

$$\begin{aligned} \text{Maximizar} \quad & z = 1\lambda_1 + 38\lambda_2, \\ \text{sujeito a:} \quad & 1\lambda_1 + 20\lambda_2 \le 10, \\ & -1\lambda_1 + 4\lambda_2 \le 4, \\ & \lambda_1 + \lambda_2 = 1, \end{aligned}$$



$\bar{z} = 18.526316$, $\lambda_1 = 0.526316$, $\lambda_2 = 0.473684$,
$\mu_1 = 1.947368$, $\mu_2 = 0$, $\sigma = -0.947368$.

Problema auxiliar:

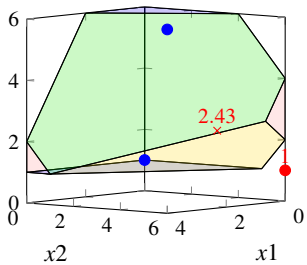
$$\begin{aligned} \text{Maximizar} \quad & cr = -1.8904736x_1 + 2.052632x_2 - 0.947368x_3 + 0.947368, \\ \text{sujeito a: } & 0 \le x_1 \le 4,\ 0 \le x_2 \le 6,\ 1 \le x_3 \le 6, \end{aligned}$$

$cr^* = 12.315792$, $x_1 = 0$, $x_2 = 6$, $x_3 = 1$.
UB $= 18.526316 + 12.315792 = 30.842108$.
Temos então que a coluna 3 é dada por: $v_3^\top = (0,6,1)$, e os coeficientes de λ_3 na função-objetivo e nas restrições do problema mestre da terceira iteração serão, respectivamente:

$$c^\top v_3 = (2,4,1)^\top \begin{pmatrix} 0 \\ 6 \\ 1 \end{pmatrix} = 25,\ A_{2_1} v_3 = (2,1,1)^\top \begin{pmatrix} 0 \\ 6 \\ 1 \end{pmatrix} = 7,\ A_{2_2} v_3 = (1,1,-1)^\top \begin{pmatrix} 0 \\ 6 \\ 1 \end{pmatrix} = 5.$$

Terceira iteração

$$\begin{aligned} \text{Maximizar} \quad z &= 1\lambda_1 + 38\lambda_2 + 25\lambda_3, \\ \text{sujeito a:} \quad 1\lambda_1 + 20\lambda_2 + 7\lambda_3 &\leq 10, \\ -1\lambda_1 + 4\lambda_2 + 5\lambda_3 &\leq 4, \\ \lambda_1 + \lambda_2 + \lambda_3 &= 1, \end{aligned}$$



$\bar{z} = 25.857143$, $\lambda_1 = 0.119048$, $\lambda_2 = 0.285714$, $\lambda_3 = 0.595238$,
$\mu_1 = 1.214286$, $\mu_2 = 2.785714$ e $\sigma = 2.571429$.

Problema auxiliar:

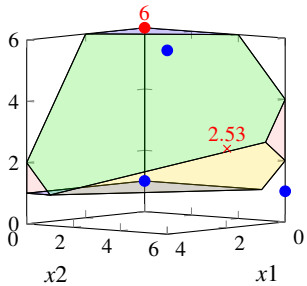
$$\begin{aligned} \text{Maximizar} \quad cr &= -3.214286x_1 + 2.571428x_3 - 2.571429, \\ \text{sujeito a: } 0 \leq x_1 \leq 4,\ 0 &\leq x_2 \leq 6,\ 1 \leq x_3 \leq 6, \end{aligned}$$

$cr^* = 12.857139$, $x_1 = x_2 = 0$, $x_3 = 6$.
UB: 38.714282.
Desta forma, temos que a coluna 4 é dada por $v_4^\top = (0,0,6)$, e os coeficientes de λ_4 na função-objetivo e nas restrições do problema mestre da quarta iteração serão, respectivamente:

$$c^\top v_4 = (2,4,1)^\top \begin{pmatrix} 0 \\ 0 \\ 6 \end{pmatrix} = 6,\ A_{2_1} v_4 = (2,1,1)^\top \begin{pmatrix} 0 \\ 0 \\ 6 \end{pmatrix} = 6,\ A_{2_2} v_4 = (1,1,-1)^\top \begin{pmatrix} 0 \\ 0 \\ 6 \end{pmatrix} = -6.$$

Quarta iteração

$$\begin{aligned} \text{Maximizar} \quad z &= 1\lambda_1 + 38\lambda_2 + 25\lambda_3 + 6\lambda_4, \\ \text{sujeito a:} \quad 1\lambda_1 + 20\lambda_2 + 7\lambda_3 + 6\lambda_4 &\leq 10, \\ -1\lambda_1 + 4\lambda_2 + 5\lambda_3 - 6\lambda_4 &\leq 4, \\ \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 &= 1, \end{aligned}$$



$\bar{z} = 26.75$, $\lambda_1 = 0.0$, $\lambda_2 = 0.236111$, $\lambda_3 = 0.694444$, $\lambda_4 = 0.069444$,
$\mu_1 = 1.125$, $\mu_2 = 1.625$ e $\sigma = 9.0$.

Problema auxiliar:

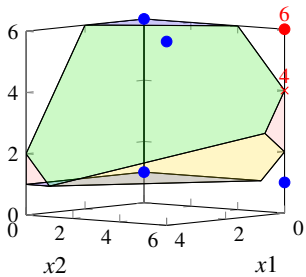
$$\begin{aligned} \text{Maximizar} \quad cr &= -1.875x_1 + 1.25x_2 + 1.5x_3 - 9, \\ \text{sujeito a: } 0 \leq x_1 \leq 4,\ 0 &\leq x_2 \leq 6,\ 1 \leq x_3 \leq 6, \end{aligned}$$

$cr^* = 7.5$, $x_1 = 0$, $x_2 = 6$, $x_3 = 6$,
UB: 34.25.
Desta forma, temos que a coluna 5 é dada por $v_4^\top = (0,6,6)$, e os coeficientes para λ_5 na função-objetivo e nas restrições do problema mestre a seguir serão, respectivamente:

$$c^\top v_4 = (2,4,1)^\top \begin{pmatrix} 0 \\ 6 \\ 6 \end{pmatrix} = 30,\ A_{2_1} v_4 = (2,1,1)^\top \begin{pmatrix} 0 \\ 6 \\ 6 \end{pmatrix} = 12,\ A_{2_2} v_4 = (1,1,-1)^\top \begin{pmatrix} 0 \\ 6 \\ 6 \end{pmatrix} = 0.$$

Quinta iteração

$$\begin{aligned} \text{Maximizar } z = 1\lambda_1 + 38\lambda_2 + 25\lambda_3 + 6\lambda_4 + 30\lambda_5, \\ \text{sujeito a: } 1\lambda_1 + 20\lambda_2 + 7\lambda_3 + 6\lambda_4 + 12\lambda_5 \leq 10, \\ -1\lambda_1 + 4\lambda_2 + 5\lambda_3 - 6\lambda_4 + 0\lambda_5 \leq 4, \\ \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 1, \end{aligned}$$



$\bar{z} = 28.0,\ \lambda_1 = 0.0,\ \lambda_2 = 0.0,\ \lambda_3 = 0.4,\ \lambda_4 = 0.0,\ \lambda_5 = 0.6,$
$\mu_1 = 1.0,\ \mu_2 = 0.0$ e $\sigma = 18.0$.

Problema auxiliar:

$$\begin{aligned} \text{Maximizar } \ cr = 3x_2 - 18, \\ \text{sujeito a: } 0 \leq x_1 \leq 4,\ 0 \leq x_2 \leq 6,\ 1 \leq x_3 \leq 6, \end{aligned}$$

$cr* = 0,\ x_1 = 0,\ x_2 = 6,\ x_3 = 1.$
UB: 28.0.

Como UB $= \bar{z} = 28.0$, encontramos a solução ótima.

5.1.1 Decomposição Dantzig-Wolfe

A geração de colunas pode explorar algumas características do problema que não seriam fáceis de explorar se ele fosse resolvido diretamente. Um exemplo disso é quando A_1 tem uma estrutura de bloco diagonal.

$$A_1 = \begin{bmatrix} A_1^1 & & & \\ & A_1^2 & & \\ & & \ddots & \\ & & & A_1^k \end{bmatrix}, b_1 = \begin{bmatrix} b_1^1 \\ b_1^2 \\ \vdots \\ b_1^k \end{bmatrix}.$$

Nessa situação, o problema auxiliar pode ser resolvido separadamente: $cr^{i*} = \min\{(c^{i\top} - \mu^k A_2^i)v^i - \sigma^i \mid A_1^i v^i = b_1^i, v^i \geq 0\}$, $i \in \{1, \ldots, k\}$. O algoritmo termina quando $cr^{i*} \geq 0$ para todos $i \in \{1, \ldots, k\}$, caso contrário pelo menos uma coluna é adicionada ao PMR. Como feito anteriormente, podemos estimar um limite inferior para z^* em cada iteração:

$$\bar{z} + \sum_{i=1}^{k} cr^{i*} \leq z^* \leq \bar{z}.$$

No exemplo 2, A_1 é constituído por $0 \leq x_1 \leq 4,\ 0 \leq x_2 \leq 6$ e $1 \leq x_3 \leq 6$, e tem uma estrutura de bloco diagonal. Seria possível resolver três subproblemas auxiliares, já que a decisão de uma variável não impacta na outra.

5.2 Geração de Colunas em Programação Inteira

Neste caso, estamos interessados em gerar colunas viáveis que contenham apenas soluções inteiras para as variáveis. Com esse objetivo resolvemos o problema relaxado e aplicamos a técnica de branch-and-bound. Nesta seção, apresentaremos duas formas tradicionais de resolver problemas de programação inteira usando a geração de colunas. Abordagens utilizadas para o branch-and-bound serão apresentadas na seção 5.2.3.

5.2.1 Geração de colunas do tipo Dantzig-Wolfe

Desenvolveremos nesta seção o método de geração de colunas para um problema de programação linear $0-1$ apresentado em [58].

$$(P): \text{Minimizar} \quad z = c^\top x, \tag{5.32}$$

$$\text{sujeito a:} \quad A_1 x = b_1, \tag{5.33}$$

$$A_2 x = b_2, \tag{5.34}$$

$$x \in \mathbb{B}^n, \tag{5.35}$$

onde $A_1 \in \mathbb{R}^{m_1 \times n}$, $A_2 \in \mathbb{R}^{m_2 \times n}$, $b_1 \in \mathbb{R}^{m_1}$ e $b_2 \in \mathbb{R}^{m_2}$. Seja $(\overline{P})$ o problema obtido quando removemos a restrição (6.4), ou seja, sua relaxação linear.

Podemos aplicar a (P) uma reformulação do tipo Dantizg-Wolfe, como apresentado na seção 5.1. Vamos escolher as restrições (6.2) e (6.4) para fazermos a nossa reformulação Dantizg-Wolfe. Observe que a restrição (6.4) de integralidade das variáveis será considerada na reformulação de Dantzig-Wolfe. Para simplificar, suponha que a região viável definida pela restrição (6.2) seja compacta, o que eliminaria a necessidade do uso de raios extremos. Desta forma, podemos escrever o espaço viável definido pelas restrições (6.2) e (6.4) como uma combinação linear de vértices extremos. Seja $X = \{x \in \mathbb{B}^n \ | A_1 x = b_1\}$ e $\mathcal{V} = \{v_1, v_2, ..., v_p\}$ os pontos extremos do conjunto X, com cardinalidade $p = |\mathcal{V}|$. Observe que $v_i \in \mathbb{B}^n$.

Todos os pontos de X podem ser escritos da seguinte maneira:

$$x = \sum_{v_j \in \mathcal{V}} \lambda_j v_j, \quad \sum_{j=1}^{p} \lambda_j = 1, \quad v_j \in \{0,1\}^n, \ j = 1, 2, ..., p. \tag{5.36}$$

Logo, (P) pode ser também escrito como:

$$(P'): \text{Minimizar} \quad z = \sum_{v_j \in \mathcal{V}} (c^\top v_j) \lambda_j, \tag{5.37}$$

$$\text{sujeito a:} \quad \sum_{v_j \in \mathcal{V}} (A_2 v_j) \lambda_j = b_2, \tag{5.38}$$

$$\sum_{j=1}^{p} \lambda_j = 1, \tag{5.39}$$

$$\lambda_j \in \mathbb{B}, \quad j = 1, 2, ..., p. \tag{5.40}$$

Como feito anteriormente, seja $(\overline{P}')$ a relaxação linear de (P').

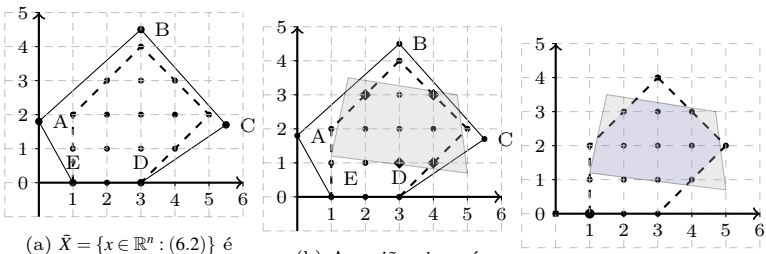
Seja val$(\bullet)$ o valor ótimo da função-objetivo do problema de minimização $(\bullet)$. Logo, temos: val $(\overline{P}') \leq$ val (P), o que nos diz que val $(\overline{P}')$ é uma cota inferior para o valor ótimo de (P). No problema (P') definido em (5.37)-(5.40), as colunas geradas são da forma

$$\begin{pmatrix} A_2 v_j \\ 1 \end{pmatrix} \tag{5.41}$$

Para resolver $(\overline{P}')$ podemos utilizar as técnicas de geração de colunas em otimização linear, como já apresentado em 5.1. Observação importante:

$$\text{val } (\overline{P}) \leq \text{val } (\overline{P}') \leq \text{val } (P)$$

Essa característica da reformulação por geração de colunas nos garante que o valor da relaxação linear de (P') é sempre maior ou igual ao valor da relaxação linear da formulação original (P). Isso é demonstrado no capítulo 11 do livro Otimização linear, referenciado em [154] e em [217]. Nestas duas referências citadas, a demonstração também nos informa que a geração de colunas obtém o mesmo valor da relaxação lagrangiana equivalente. Podemos entender isso através da figura 5.4.



(a) $\bar{X} = \{x \in \mathbb{R}^n : (6.2)\}$ é representada pela região definida pela linha contínua e X é representada pela região definida pela linha tracejada.

(b) A região cinza é definida por $\mathcal{Q} = \{x \in \mathbb{R}^n : (6.3)\}$.

(c) A região em azul, $(\overline{P}')$, é a interseção de X e $\mathcal{Q}$.

Figura 5.4: Melhora da relaxação linear da geração de colunas.

Uma vez solucionado $(\overline{P}')$, teremos gerado p_k colunas $(p_k \leq p)$ do tipo (5.41). Na prática, p_k é bem menor do que p. Se na solução $(\overline{P}')$ obtivermos todos os λ_j, $j = 1, 2, ..., p_k$ inteiros, então encontramos uma solução ótima de (P). Caso contrário, temos que utilizar técnicas de branch-and-pricing. Observe que de modo geral não é necessário a integralidade das variáveis λ. A solução de $(\overline{P}')$ é solução ótima de (P) se a combinação dos pontos extremos representa um ponto x inteiro.

Seja (P_k') o problema de minimização linear inteira formada só pelas colunas geradas para resolver $(\overline{P}')$.

$$(P_k') : \text{Minimizar} \quad z = \sum_{j=1}^{p_k} (c^\top v_j)\lambda_j, \tag{5.42}$$

$$\text{sujeito a:} \quad \sum_{j=1}^{p_k} (A_2 v_j)\lambda_j = b_2, \tag{5.43}$$

$$\sum_{j=1}^{p_k} \lambda_j = 1, \tag{5.44}$$

$$\lambda_j \in \mathbb{B}, \ j = 1, ..., p_k. \tag{5.45}$$

Temos que:

$$\text{val } (\bar{P}') \leq \text{val } (P) \leq \text{val } (P_k').$$

5.2.2 Geração de colunas do tipo Gilmore-Gomory

As principais referências relacionadas a esse assunto são [97], [98]. Alguns problemas importantes em programação linear inteira podem ser colocados sob a seguinte forma:

$$(P): \text{Minimizar} \quad z = \sum_{j=1}^{p} c_j \lambda_j, \tag{5.46}$$

$$\text{sujeito a:} \quad \sum_{j=1}^{p} a_j \lambda_j = b, \tag{5.47}$$

$$\lambda_j \geq 0, \ j = 1, 2, ..., p, \tag{5.48}$$

$$\lambda_j \text{ inteiro } j = 1, 2, ..., p, \tag{5.49}$$

onde $a_j \in \mathcal{C} \subseteq \mathbb{Z}_+^m$, $|\mathcal{C}| = p$, $b \in \mathbb{Z}_+^m$, e $c_j = f(a_j) \in \mathbb{R}$. Muitas vezes o conjunto $\mathcal{C}$ é definido na literatura como $\mathcal{C} \subseteq \{0,1\}^m$. O número de variáveis inteiras do problema (P), (5.46)-(5.49), pode ser muito grande. Por exemplo, se cada $a_j \in \{0,1\}^m$ representar uma árvore geradora de um grafo completo com $|V|$ nós, $p = |V|^{(|V|-2)}$. Note que nesse caso a formulação original já possui um número muito grande de variáveis.

A ideia é começarmos a resolver a relaxação linear $(\overline{P})$ de (P), isto é, eliminarmos as restrições de integralidade (5.49), considerando apenas um número bem menor de colunas geradas, formando uma solução básica viável. Assim como na geração de colunas do tipo Dantzig-Wolfe, definimos o problema mestre restrito $(\overline{P}')$ de $(\overline{P})$. Para resolver $(\overline{P}')$ utilizaremos as técnicas de geração de colunas em otimização linear, como já apresentado em 5.1.

Seja $B = (a_1 \ a_2 \ ...a_m)$, uma base primal viável de $(\overline{P})$. Temos que verificar se essa solução básica é ótima. Consideremos $u^\top = c_B^\top B^{-1}$, onde $c_B = (c_1 \ c_2, ..., c_m)^\top$. Para garantir que a solução básica viável associada à matriz B seja ótima, devemos verificar que $\bar{c}_j = f(a_j) - u^\top a_j \geq 0, \ j = 1, 2, \ldots, p$, ou seja, custo reduzido não negativo. Para isso basta calcularmos

$$(GE): \quad \min\{f(a) - u^\top a \mid a \in \mathcal{C}\}.$$

Se val $(GE) \geq 0$, a matriz B está associada a uma solução ótima de $(\overline{P})$. Se val $(GE) = f(a^*) - u^\top a^* < 0$, colocaremos a^* na base, substituindo uma das colunas de B. O método é repetido até que val $(GE) \geq 0$.

Para a obtenção da solução ótima de (GE) temos que resolver um problema de otimização combinatória. Quando f for uma função não linear, o problema poderá ser muito difícil. Vamos supor que $f(a_j) = d^\top a_j + d_0, \ j = 1, 2, ...p,$ onde $d = (d_1 \ d_2 \ ...d_m)^\top \in \mathbb{R}^m, \ d_0 \in \mathbb{R}$. Nas aplicações do problema estudado por Gilmore-Gomory [97], [98], $d = 0, \ d_0 = 1$; logo $f(a_j) = 1, \ j = 1, 2, ..., p$. Em [157] foi considerado $d_0 = 0$ e $f(a_j) = d^\top a_j, \ j = 1, 2, ..., p$.

Se a solução ótima de $(\overline{P})$ for inteira, encontraremos uma solução ótima também para (P); caso contrário, teremos que utilizar métodos de enumeração implícita com geração de colunas, como já dito na seção anterior e detalhado a seguir.

5.2.3 Ramificação e cortes

Depois de resolvermos a relaxação linear da geração de colunas ((5.37)-(5.40) ou ((5.46)-(5.49)), pode ocorrer de não obtermos valores inteiros para todas as variáveis λ. No caso da reformulação Dantizg-Wolfe, as variáveis λ podem não representar uma solução inteira (variáveis x) no problema original. Para lidarmos com essas situações devemos fazer uma enumeração implícita (branch-and-bound) ou utilizarmos planos

de corte, ver capitulo 7. Na prática, mesmo no caso da enumeração, podemos ver a ramificação (branch) como a adição de um plano de corte.

A utilização mais comum do método branch-and-bound acontece quando temos uma solução x_i fracionária e adicionamos uma restrição $x_i \leq \lfloor x_i \rfloor$ em um ramo e no outro ramo $x_i \geq \lceil x_i \rceil$. O uso do método branch-and-bound na geração de colunas é chamado de branch-and-price. No branch-and-price, tanto para o caso da obtenção de valores de λ não inteiros como para o caso da reformulação Dantzig-Wolfe, quando as variáveis λ não representam uma solução inteira no problema original, pelo menos uma restrição deve ser adicionada à geração de colunas. Sendo assim, temos duas opções para fazer isso: adicionar essa restrição no problema mestre ou no problema auxiliar.

Se adicionarmos novas restrições no problema mestre restrito, o custo reduzido das colunas mudará, já que a coluna do problema mestre agora tem pelo menos uma nova linha. Sendo assim, o problema auxiliar desse problema mestre restrito terá a sua função-objetivo modificada, o que pode determinar a mudança do tipo de problema com que estamos lidando. Exemplo: suponha que o problema auxiliar seja um Problema da Mochila. O problema mestre pode ter uma restrição por item (p. ex. vértice). Caso adicionemos uma restrição referente à combinação de dois itens (p. ex. aresta), o custo reduzido só muda se na coluna estiverem os dois itens juntos, ou seja, agora temos o Problema da Mochila Quadrática. Portanto, o problema auxiliar que antes era linear, com a adição de uma restrição no problema mestre, tornou-se um problema não linear quadrático. Agora um algoritmo para o Problema da Mochila não resolve mais esse problema auxiliar.

No caso de adicionarmos as restrições no problema auxiliar, também temos que remover as colunas já adicionadas ao problema mestre restrito que não respeitam as novas restrições. Da mesma forma que a primeira opção, essas novas restrições podem mudar muito a estrutura do problema auxiliar.

Como já dito anteriormente, o uso mais comum do método branch-and-bound ocorre quando se deseja encontrar um valor inteiro para uma variável fracionária x_i. Na geração de colunas, essa forma de utilização do método branch-and-price não é interessante, porque em um ramo teríamos $\lambda_i = 1$ e no outro $\lambda_i = 0$. Além disso, esse procedimento gera uma árvore de enumeração muito desbalanceada. Isso ocorre porque no lado em que forçamos uma coluna na solução estamos fixando diversos valores de x ao valor dessa coluna, normalmente todo o vetor x. No outro ramo, estaríamos proibindo esses valores no vetor x, o que normalmente é muito difícil de fazer, ou seja, estaríamos proibindo que essa coluna fosse gerada novamente. Se quisermos fazer a ramificação sobre a variável original x_i, o mais indicado é adicionar as restrições ao problema auxiliar conforme mencionado anteriormente. Entretanto, existem outras formas de ramificar mais indicadas para a geração de colunas ([174, 206]).

5.2.4 Estratégias de aceleração

Um ponto importante para tornar a geração de colunas eficiente é a utilização de técnicas de aceleração. A seguir apresentaremos algumas estratégias para alcançarmos a eficiência da geração de colunas.

Como já vimos no início deste capítulo, podemos particionar as restrições do problema original (PL') (5.9)-(5.12), escolher um dos conjuntos para representar como uma combinação convexa de vértices e raios extremos e reescrever o problema (PL1') (5.13)-(5.16). Também como já vimos anteriormente, ao considerarmos um subconjunto de colunas do problema (PL1') (5.13)-(5.16), temos o problema mestre restrito (PMR) (5.20)-(5.23) e o problema auxiliar (PA) (5.24)-(5.26). Observe que tudo de-

pende de como foi feita a escolha do subconjunto de restrições representado como a combinação de vértices e raios extremos, pois dessa escolha obtemos o (PMR) e (PA) e, a partir dele, obtemos o (PA) - ver figura 5.4. Portanto, podemos escolher resolver um problema auxiliar difícil ou podemos tentar encontrar um problema auxiliar mais "fraco", porém que seja computacionalmente mais eficiente. Isso deve levar a um limite inferior pior para o PMR, contudo melhor que o modelo original (variáveis x).

Uma outra forma de acelerar a resolução do problema auxiliar seria buscar uma coluna com custo reduzido negativo ao invés de buscar a coluna de custo reduzido mais negativo. Neste caso, podemos usar heurísticas ou colunas já geradas e que não estão presentes no (PMR). No caso de vários problemas auxiliares (ver subseção 5.1.1), pode ser benéfico resolver só um subconjunto desses problema em cada iteração.

Uma das estratégias mais utilizadas é a adição de múltiplas colunas por iteração. Todas as colunas adicionadas devem ter custo reduzido negativo, mas não necessariamente o mínimo. Isso geralmente leva a uma diminuição do número de iterações da geração de colunas e a um aumento do tempo da resolução do Simplex do (PMR) em cada iteração. Eventualmente resolvemos o problema auxiliar usando programação dinâmica em vez de programação matemática. Nesse caso, a geração de múltiplas colunas por iteração é natural por ser intrínseca ao método.

Remover colunas não mais relevantes também é uma tática utilizada, principalmente para colunas não básicas que possuem custo reduzido maior que um certo valor. Ver mais detalhes em [149].

Da mesma forma que no método de plano de cortes, podemos interromper a geração de colunas antes de obtermos a relaxação linear quando percebemos que as iterações não estão mais impactando a relaxação linear. Nesse caso, devemos começar a enumeração (branch) sem usar a relaxação linear do PRM como um limite dual.

Uma observação importante é que as variáveis duais não têm um comportamento suave durante a geração de colunas e oscilam muito. No início da geração de colunas, o (PMR) possui pouca informação útil para o dual e, como consequência, colunas ruins podem ser geradas. Normalmente as colunas básicas no (PMR) final são geradas nas últimas iterações. Uma maneira de controlar esse comportamento é controlar as variáveis duais usando limites inferiores e superiores para elas. Exemplos de estratégias de estabilização são a Box Step [159, 160] e a de Neame [174].

6

Dualidade em Otimização Linear Inteira

Nelson Maculan
Ana Flávia Uzeda Macambira
Luidi Simonetti
Abilio Lucena

6.1 Introdução

Suponhamos que dispuséssemos de uma formulação linear inteira para um problema de otimização *NP*-difícil e que gostaríamos de transformá-la para gerar limitantes duais para o valor de suas soluções ótimas. A relaxação lagrangiana, a ser aqui discutida, é uma das principais técnicas disponíveis para alcançar tal objetivo.

Para facilitar a apresentação, assumimos que nossa formulação pode ser particionada em dois conjuntos de restrições, em que a eliminação de todas as restrições contidas em um deles, em particular, resulta na formulação de um problema "mais fácil de se resolver". Numa relaxação lagrangiana, ao invés de simplesmente eliminarmos tais restrições, associamos "penalidades válidas", também conhecidas como "multiplicadores de Lagrange", a cada uma delas e as levamos à função-objetivo. Ou seja, as restrições são "dualizadas lagrangianemente", dando origem a um "subproblema lagrangiano" (SL). Resolvido esse SL, como veremos, obteremos um limitante dual como o que especificamos acima.

No cenário que acabamos de descrever, limitantes duais para um problema *NP*-difícil são obtidos ao se resolver um problema pertencente à classe *P* ou de menor complexidade. Notamos que essa mesma situação também se aplica ao caso específico das simples relaxações lineares, isto é, quando as restrições de integralidade não são levadas em consideração. Aqui nos restaria, é verdade, identificar o melhor conjunto de penalidades a utilizar ou, de forma equivalente, formular um SL que resulte no melhor limitante dual possível. Além de indicar como fazer isso, mostraremos também

quão forte é esse limitante, comparando-o àquele fornecido pela "relaxação linear" de nossa formulação. Finalmente, mostraremos também como os limitantes duais aqui descritos podem ser utilizados para resolver um problema *NP*-difícil de forma exata.

Neste capítulo, que se baseia em grande parte no capítulo 11 do livro [154], vamos eventualmente ser pouco precisos com a notação, referindo-nos a um "problema de otimização" e a uma "formulação para aquele problema" como se tivessem o mesmo significado. Isso é feito, mais uma vez, para facilitar a exposição. Notemos, no entanto, que podem existir diferentes formulações lineares inteiras para um mesmo problema. Notemos também que relaxações lagrangianas são aplicadas a formulações de problemas, e não diretamente aos problemas propriamente ditos. Duas formulações distintas podem levar a diferentes limitantes duais para um mesmo problema.

6.2 Relaxação lagrangiana

Seja o problema de otimização linear inteira *NP*-difícil formulado como:

$$(PLI): \text{Minimizar} \quad z = c^\top x, \tag{6.1}$$

$$\text{sujeito a:} \quad Ax \leq b, \tag{6.2}$$

$$Gx \leq d, \tag{6.3}$$

$$x \in \{0,1\}^n, \tag{6.4}$$

onde $c^\top \in R^n,\ x \in R^n,\ A \in R^{m\times n},\ b \in R^m,\ G \in R^{p\times n},\ d \in R^p$.

Associemos às restrições (6.2) o vetor $u = (u_1\ u_2\ \ldots\ u_m)^\top \geq 0$, conhecido como o vetor dos multiplicadores de Lagrange. Suponhamos que as restrições (6.2), (6.3), (6.4) formem um conjunto não vazio. Sendo assim, para $u \geq 0$ podemos escrever:

$$L(u) = \min\left\{\ c^\top x + u^\top(Ax-b)\ \ |\ \ Gx \leq d,\ x \in \{0,1\}\ \right\}. \tag{6.5}$$

O problema de otimização (6.5) é denominado uma relaxação lagrangiana de (PLI).

Propriedade 1. $L(u) \leq val(PLI)$, em que $u \geq 0$ e $val(\bullet)$ representa o valor da função-objetivo no ótimo do problema $(\bullet)$

Demonstração. Suponhamos que x^* seja uma solução ótima de (PLI), isto é, $val(PLI) = c^\top x^*$. Sabemos também que $L(u) \leq c^\top x^* + u^\top(Ax^* - b)$, pois $u \geq 0$ e $Ax^* - b \leq 0$ implicando $u^\top(Ax^* - b) \leq 0$. Assim sendo:

$$L(u) \leq c^\top x^* + u^\top(Ax^* - b) \leq c^\top x^* = val(PLI).$$

□

A propriedade 1 nos diz que, ao resolvermos $L(u)$ encontramos uma cota inferior para $val(PLI)$. Quanto maior o valor da cota encontrada, mais nos aproximamos do valor $val(PLI)$. Desta forma, visando à busca da maior cota possível, consideramos o seguinte problema:

$$(D): \text{Maximizar} \quad L(u), \tag{6.6}$$

$$\text{sujeito a:} \quad u \geq 0. \tag{6.7}$$

(D) é considerado um problema dual de (PLI). Se u^* for uma solução ótima de (D), teremos $val(D) = L(u^*)$. Geralmente $val(D) < val(PLI)$, e dizemos que $val(LIP) - val(D)$ mede o salto primal-dual (gap).

6.3 Resolução do problema dual

6.3.1 Técnica de geração de colunas

O problema (D) pode ser escrito sob a forma de um problema de otimização linear:

$$(D): \text{Maximizar} \quad L(u) = w, \tag{6.8}$$

$$\text{sujeito a:} \quad w \leq c^\top x^k + u^\top (Ax^k - b), \; k = 1,2,...,K, \tag{6.9}$$

$$u \geq 0, \tag{6.10}$$

onde $\{x^1, x^2, ..., x^K\} = \{\; x \in \{0,1\}^n \mid Gx \leq d \;\}$. Devemos notar que w e u são as incógnitas de (D); os vetores x^k de componentes $0-1$ são soluções viáveis de (6.2) e (6.3), supostamente conhecidos. Poderíamos pensar em resolver o problema (D) logo acima por técnicas de geração de linhas. O problema (D) também pode ser escrito do seguinte modo:

$$(D): \text{Maximizar} \quad L(u) = w, \tag{6.11}$$

$$\text{sujeito a:} \quad w - u^\top (Ax^k - b) \leq c^\top x^k, \; k = 1,2,...,K, \tag{6.12}$$

$$u \geq 0. \tag{6.13}$$

No entanto, optamos por trabalhar com o seu problema dual abaixo.

$$(DD): \text{Minimizar} \quad \sum_{k=1}^{K} (c^\top x^k)\lambda_k, \tag{6.14}$$

$$\text{sujeito a:} \quad \sum_{k=1}^{K} \lambda_k = 1, \tag{6.15}$$

$$-\sum_{k=1}^{K} (Ax^k - b)\lambda_k \geq 0, \tag{6.16}$$

$$\lambda_k \geq 0, \; k = 1,2,...,K, \tag{6.17}$$

em que $\lambda_k, \; k = 1,2,...,K$ representam as variáveis duais. A igualdade (6.15) implica que (DD) possa também ser escrito da seguinte maneira:

$$(DD): \text{Minimizar} \quad \sum_{k=1}^{K} (c^\top x^k)\lambda_k, \tag{6.18}$$

$$\text{sujeito a:} \quad \sum_{k=1}^{K} \lambda_k = 1, \tag{6.19}$$

$$\sum_{k=1}^{K} (Ax^k)\lambda_k \leq b, \tag{6.20}$$

$$\lambda_k \geq 0, \; k = 1,2,...,K. \tag{6.21}$$

Observação: Se substituirmos no modelo acima as restrições (6.21) por $\lambda_k \in \{0,1\}$, $k = 1,2,...,K$, denominaremos esse problema (DDD), que é equivalente a (PLI). Por quê?

Para resolvermos o problema (6.18)-(6.21), introduziremos um vetor $s = (s_1 \; s_2 ... s_m)^\top \geq 0$, cujas componentes são as variáveis de folga das restrições (6.20), obtendo

$$(DD): \text{Minimizar} \quad \sum_{k=1}^{K} (c^\top x^k)\lambda_k, \tag{6.22}$$

$$\text{sujeito a:} \quad \sum_{k=1}^{K} \lambda_k = 1, \tag{6.23}$$

$$\sum_{k=1}^{K} (Ax^k)\lambda_k + s = b, \tag{6.24}$$

$$\lambda_k \geq 0, \; k = 1, 2, ..., K, \tag{6.25}$$

$$s \geq 0. \tag{6.26}$$

Usaremos o algoritmo Simplex para resolver (DD) sob a forma (6.22)-(6.26). As colunas $\binom{1}{Ax^k}$ serão geradas pelas técnicas de geração de colunas utilizadas em Otimização linear (Dantzig-Wolfe).

Supomos que conheçamos uma matriz quadrada $B \in R^{(m+1)\times(m+1)}$, inversível formada por colunas $\binom{1}{Ax^k}$ associadas às variáveis λ_k e $\binom{0}{e_i}$ associadas às variáveis s_i, em que e_i é um vetor da básica canônica do eixo i em R^m, isto é, $e_i = (0\ 0...0\ 1\ 0...0\ 0)^\top$, em que todas as componentes são nulas exceto a $i-$ésima, que é igual a 1, tal que:

$$B^{-1}\begin{pmatrix} 1 \\ b \end{pmatrix} \geq 0.$$

Logo, B está associada a uma solução básica viável de (6.22)-(6.26). Será que B também está associada a uma solução ótima de (6.22)-(6.26)?

Seja q_B o vetor-linha associado aos coeficientes das variáveis básicas na função-objetivo, consideramos $v = q_B B^{-1}$, cujas coordenadas são os multiplicadores do Simplex. Podemos escrever $v = (v_0\ v^1)$, onde $v_0 \in R$ e $v^1 \in R^m$. Notamos que v_0 está associado à restrição (6.23). O custo reduzido associado a uma coluna da forma $\binom{0}{e_i}$ é

$$0 - v_i^1 = 0 - v\begin{pmatrix} 0 \\ e_i \end{pmatrix}.$$

Se $v_i^1 > 0$, a coluna $\binom{0}{e_i}$ entrará na base no lugar de outra coluna após a operação de pivoteamento e teremos uma nova matriz B. Caso não haja v_i^1, $i = 1, 2, ..., m$ positivo, passaremos para o cálculo do custo reduzido relativo à coluna $\binom{1}{Ax^k}$, que será

$$\bar{q}_k = c^\top x^k - v\begin{pmatrix} 1 \\ Ax^k \end{pmatrix} = c^\top x^k - v^1 Ax^k - v_0 = (c^\top - v^1 A)x^k - v_0.$$

Desejamos obter um k para o qual $\bar{q}_k < 0$. Para isso teremos que resolver o seguinte problema auxiliar:

$$(PA): \text{Minimizar} \quad (c^\top - v^1 A)x - v_0, \tag{6.27}$$

$$\text{sujeito a:} \quad Gx \leq d, \tag{6.28}$$

$$x \in \{0, 1\}^n. \tag{6.29}$$

Seja x^* uma solução ótima de (PA), logo $val(PA) = (c^\top - v^1 A)x^* - v_0$. Se $val(PA) \geq 0$, a solução básica associada à matriz B será uma ótima. Caso contrário, isto é, $val(PA) < 0$, a coluna $\binom{1}{Ax^*}$ entrará na base, substituindo outra coluna, e teremos uma nova matriz B.

Lembremos que $val(D) = val(DD)$.

Quando substituímos em (PLI) a restrição (6.4) por $x \in [0, 1]^n$, temos o problema $(LPLI)$, que será uma relaxação contínua de (PLI), cuja implicação é que $val(LPLI) \leq val(PLI)$.

Consideremos o conjunto de pontos que satisfaça as restrições (6.3) e (6.4), isto é, $X = \{\ x \in \{0,1\}^n \mid Gx \leq d\ \} \neq \phi$. Sejam $\bar{X} = \{\ x \in [0,1]^n \mid Gx \leq d\ \}$ e $V(\bar{X})$ o conjunto dos vértices de $\bar{X}$. É fácil verificar que todo ponto de X é um vértice do politopo $\bar{X}$, no entanto, podemos ter vértices de $\bar{X}$ que não pertencem ao conjunto X. Isto é, $X \subseteq V(\bar{X})$. Se no lugar de X usássemos o conjunto $V(\bar{X})$ para aplicarmos as técnicas de geração de colunas, estaríamos resolvendo o problema ($LPLI$). Daí podemos concluir que $val(LPLI) \leq val(D)$.

Quando $X = V(\bar{X})$ diremos que a relaxação lagrangiana possui a propriedade de integralidade. Neste caso o problema (PA) será resolvido pelo algoritmo do Simplex, em que $x \in \{0,1\}^n$ será substituído por $x \in [0,1]^n$. Sabemos também que nesse caso $val(LPLI) = val(D)$.

Apresentaremos um pequeno exemplo a seguir.

$$
\begin{aligned}
(PLI): \text{Minimizar} \quad & z = -6x_1 - 8x_2 + 5x_3, && (6.30)\\
\text{sujeito a:} \quad & -4x_1 - 4x_2 - x_3 \leq -2, && (6.31)\\
& 2x_2 - 2x_3 \leq 1, && (6.32)\\
& -2x_2 + 2x_3 \leq 1, && (6.33)\\
& 2x_1 + 2x_2 + 2x_3 \leq 5, && (6.34)\\
& x_j \in \{0,1\},\ \ j = 1,2,3. && (6.35)
\end{aligned}
$$

Associamos a variável $u \in R_+$ à restrição (6.31). Escrevemos o seu dual lagrangiano:

$$
\begin{aligned}
(D): \text{Maximizar} \quad & L(u) = w, && (6.36)\\
\text{sujeito a:} & && \\
& w \leq -6x_1^k - 8x_2^k + 5x_3^k + (-4x_1^k - 4x_2^k - x_3^k + 2)u,\ \ k = 1,2,3, && (6.37)\\
& u \geq 0, && (6.38)
\end{aligned}
$$

em que $x^1 = (0\ 0\ 0)$, $x^2 = (1\ 0\ 0)$ e $x^3 = (0\ 1\ 1)$ são todas as soluções viáveis do conjunto formado pelas restrições (6.32), (6.33), (6.34) e (6.35). Logo, (D) pode ser escrito da seguinte maneira:

$$
\begin{aligned}
(D): \text{Maximizar} \quad & L(u) = w, && (6.39)\\
\text{sujeito a:} \quad & w \leq 2u, && (6.40)\\
& w \leq -6 - 2u, && (6.41)\\
& w \leq -3 - 3u, && (6.42)\\
& u \geq 0. && (6.43)
\end{aligned}
$$

O problema de otimização linear (D), descrito por (6.39)-(6.43), possui apenas duas variáveis e sua solução é encontrada facilmente: $u^* = 0$ e $w^* = -6 \rightarrow val(D) = -6$. A solução do problema ($LPLI$), isto é, a relaxação contínua de (PLI), nos dá também um problema de otimização linear, no qual obtemos $val(LPLI) = -11,5$. Verificamos que $val(LPLI) < val(D) \leq val(PLI)$. Essa relaxação lagrangiana (D) não possui a propriedade da integralidade.

6.3.2 Método utilizando subgradientes

Definiremos a seguir um subgradiente para uma função côncava e forneceremos um método específico para resolver o problema (D), descrito em (6.6) e (6.7).

$L(u)$ é uma função côncava e afim por partes.

Definição: Um vetor $\gamma \in R^n$ é um subgradiente de uma função côncava $f : R^n \rightarrow R$, no ponto $x^0 \in R^n$, se $f(x) \leq f(x^0) + \gamma^\top (x - x^0)$ para todo $x \in R^n$.

Definição: O subdiferencial $\partial f(x^0)$ de f em x^0 é o conjunto de todos os subgradientes de f em x^0.

Observação: Se f for diferenciável em x^0, então $\partial f(x^0) = \{\nabla f(x^0)\}$, em que $\nabla f(x^0)$ representa o gradiente de f em x^0.

Podemos notar também que $\partial L(u) \neq \phi$, para $u \geq 0$.

Propriedades do subdiferencial: $\partial f(x^0) \neq \phi$ é um conjunto convexo e fechado.

Podemos também demonstrar que, se $f : R^n \to R$ é uma função côncava, então x^0 maximiza f se e somente se $0 \in \partial f(x^0)$.

Um subgradiente de $L(u)$ no ponto $\bar{u}$ é $A\bar{x} - b$, tal que $L(\bar{u}) = c^\top \bar{x} + \bar{u}^\top (A\bar{x} - b)$. Para verificarmos esse resultado:
$L(u^0) + (u - u^0)^\top (A\bar{x} - b) = c^\top \bar{x} + {u^0}^\top (A\bar{x} - b) + (u - u^0)^\top (A\bar{x} - b)$,
ou ainda
$L(u^0) + (u - u^0)^\top (A\bar{x} - b) = c^\top \bar{x} + u^\top (A\bar{x} - b) \geq L(u)$, para $u \geq 0$.

Observação: A direção de um subgradiente γ pode não ser de subida.

Método de otimização usando subgradientes para resolver (D)

Para resolvermos o problema (D), descrito por (6.6)-(6.7), temos que encontrar o máximo de $L(u)$ tal que $u \geq 0$. Uma ideia do processo iterativo é descrito a seguir:

i. Começamos com um dado $u^0 \geq 0$, podendo ser igual ao vetor nulo 0.

ii. Cada iteração será da forma $u^{k+1} = u^k + \alpha_k \gamma^k$, em que $\alpha_k > 0$ será o passo e γ^k um subgradiente de L no ponto u^k.

iii. Para o cálculo de um subgradiente de L no ponto u^k, obteremos $\bar{x}$ uma solução de $L(u^k) = \min_{x \in \{0,1\}} \{c^\top x + {u^k}^\top (Ax - b) \mid Gx \leq d\}$.

iv. Tomaremos $\gamma^k = A\bar{x} - b$. Se $\bar{x}$ for a única solução, implica que L é diferenciável em u^k.

Para que possamos garantir a convergência do método, a escolha de α_k deve satisfazer às duas condições abaixo:

$$lim_{k \to \infty} \ \alpha_k = 0$$

e

$$lim_{s \to \infty} \sum_{k=1}^{s} \alpha_k = \infty.$$

Por exemplo, $\alpha_k = \frac{1}{k}, \ k \geq 1$.
Na resolução dos problemas duais lagrangianos utilizamos [115]:

$$\alpha_k = \rho_k \frac{UB - LB}{||\gamma_k||^2}, \ \rho_k \in (0, 2], \tag{6.44}$$

onde $LB \leq val(PLI) \leq UB$.

Para maiores detalhes sobre os métodos de otimização com o uso de subgradientes, ver [115] e [175].

6.4 Uso da relaxação lagrangiana num método de enumeração implícita

O trabalho de Fisher [78] é uma referência para esta seção.

Em problemas de otimização combinatória, os métodos duais baseados nas relaxações lagrangianas visam a fornecer cotas inferiores, LB, para $val(PLI)$, pois $LB =$

$val(D) \leq val(PLI)$. No entanto, quando se busca uma coluna ou um subgradiente, dependendo do método, podemos encontrar valores para x que também satisfaçam às restrições $Ax \leq b$. Neste caso obtemos cotas superiores, UB, para $val(PLI)$. Essas cotas têm sido utilizadas de maneira muito eficiente nos métodos de enumeração implícita.

A resolução do dual (D) de (PLI), para uma determinada relaxação lagrangiana, não nos fornece, obrigatoriamente, uma solução ótima x^* para (PLI). Podemos obter nos problemas de otimização auxiliares para gerar colunas ou para encontrar vetores subgradientes $\bar{x} \in \{0,1\}^n$, viáveis para (PLI), fornecendo-nos cotas superiores $UB = c^\top \bar{x} \geq val(PLI)$. No entanto, poderemos desenvolver métodos de enumeração implícita do tipo branch-and-bound e suas extensões.

6.4.1 Métodos de branch-and-bound

Apresentaremos a seguir a versão proposta por Dakin em 1965 [55] para os métodos de enumeração implícita do tipo branch-and-bound.

Seja o problema de otimização linear inteira mista:

$$(PLIM): \text{Minimizar} \quad z = c_1^\top x + c_2^\top y, \tag{6.45}$$

$$\text{sujeito a:} \quad A_1 x + A_2 y = b, \tag{6.46}$$

$$x \in [0, \beta]^p, \tag{6.47}$$

$$y \geq 0, \tag{6.48}$$

$$x \text{ inteiro}, \tag{6.49}$$

onde $c_1 \in R^p,\ c_2 \in R^q,\ x^p \in R^p,\ y \in R^q,\ b \in R^m, A_1 \in R^{m \times p},\ A_2 \in R^{m \times q}$ e $\beta \in Z,\ \beta \geq 1$. A restrição (6.47) indica que cada componente $x_j, j = 1, 2, ..., p$ do vetor x está no intervalo $[0, \beta]$.

Quando não considerarmos a restrição (6.49), teremos uma relaxação contínua de $(PLIM)$, denominada $(PLIM_0)$, que é um problema de otimização linear que pode ser resolvido pelo algoritmo do Simplex.

Caso as restrições do problema $(PLIM_0)$ formem um conjunto vazio, então o conjunto das restrições de $(PLIM)$ também será vazio.

Vamos supor que tenhamos o conhecimento de LB e UB, tais que $LB \leq val(PLIM) \leq UB$. Caso não tenhamos esses valores, suporemos que $LB = -\infty$ e $UP = +\infty$.

Consideremos que x^0 e y^0 forneçam uma solução ótima de $(PLIM_0)$. Se x^0 possuir todas as suas componentes inteiras, então x^0 e y^0 fornecerão uma solução ótima de $(PLIM)$. Caso contrário, isto é, existirá pelo menos uma componente x_l^0 que não pertencerá ao conjunto dos números inteiros. Faremos $LB = val(PLIM_0)$ e construiremos dois novos problemas de otimização linear a partir de $(PLIM_0)$: ao primeiro $(PLIM_{01})$ acrescentaremos a restrição $x_l \leq \lfloor x_l^0 \rfloor$ às restrições de $(PLIM_0)$ e ao segundo $(PLIM_{02})$, a restrição $x_l \geq \lfloor x_l^0 \rfloor + 1$ às restrições de $(PLIM_0)$.

Lembremos que $\lfloor \alpha \rfloor$ representa o maior inteiro menor ou igual a $\alpha \in R$. Exemplos $\lfloor \pi \rfloor = 3,\ \lfloor 7.3 \rfloor = 7,\ \lfloor -2.3 \rfloor = -3$.

É interessante observar que:

i. se $(PLIM_{01})$ não for vazio, no seu ótimo, obteremos uma solução $x^1,\ y^1$, cujo valor para $x_l^1 = \lfloor x_l^0 \rfloor$;

ii. se $(PLIM_{02})$ não for vazio, no seu ótimo, obteremos uma solução $x^2,\ y^2$, cujo valor para $x_l^2 = \lfloor x_l^0 \rfloor + 1$.

Sabemos que:
$(PLIM_0) \leq val(PLIM_{01}) \leq val(PLIM)$ e
$(PLIM_0) \leq val(PLIM_{02}) \leq val(PLIM)$.

Diremos que os problemas $(PLIM_{01})$ e $(PLIM_{02})$ são os sucessores do problema $(PLIM_0)$. E o problema $(PLIM_0)$ será fechado.

As operações a seguir serão realizadas para todos os problemas não fechados.

i. O problema $(PLIM_{0i})$ nos fornece uma solução ótima em que o vetor x possui todas as componentes inteiras, logo teremos uma solução viável para $(PLIM)$. Se $val(PLIM_{0i}) < UB$, atualizaremos o seu valor $UB = val(PLIM_{0i})$. Esse problema então será fechado, isto é, não terá sucessores;

ii. O problema $(PLIM_{0i})$ possui seu conjunto de soluções vazio, logo não tem sucessores e será fechado;

iii. O problema $(PLIM_{0i})$ possui uma solução, tal que $val(PLIM_{0i}) \geq UB$ será fechado, pois os valores das soluções ótimas de suas sucessões serão também maiores ou iguais a UB;

iv. Entre os problemas não fechados escolheremos aquele com o menor valor da função-objetivo e formaremos seus dois problemas sucessores.

Na realidade desenvolvemos uma árvore binária para a enumeração implícita.

Podemos observar que existe um problema inicial, $(PLIM_0)$, que dá origem a dois subproblemas, $(PLIM_{01})$ e $(PLIM_{02})$. Cada um desses dois subproblemas ou será fechado ou gerará mais dois subproblemas, e assim sucessivamente, dando origem a uma árvore binária, em que cada subproblema é um vértice. Dizemos que essa é uma árvore para a enumeração implícita porque, como visto nas operações acima, alguns vértices serão fechados antes de gerarem subproblemas. Desta forma, nem todos os possíveis vértices serão gerados.

Quando todos os problemas que formaram essa árvore binária estiverem fechados, verificaremos se o motivo de todos os fechamentos foi o fato de os problemas serem vazios. Neste caso, $(PLIM)$ é vazio. Caso contrário, uma solução ótima está associada ao problema que forneceu o UB final.

As restrições (6.47) e (6.49) possuem um número finito de soluções; assim sendo, o método de branch-and-bound termina sempre. No entanto, as cotas superiores e inferiores influenciarão muito a performance dos métodos de enumeração implícita. Supondo que $\beta = 1$ e $p = 100$, as restrições (6.47) e (6.49) terão 2^{100} soluções possíveis. Usando logaritmo em base 10, verificamos que 2^{100} é igual a um número α inteiro em base decimal com 31 algarismos, ou seja, $10^{30} < \alpha < 10^{31}$.

6.4.2 Um exemplo ilustrando a resolução do dual lagrangiano

Tomemos o seguinte problema de otimização inteira:

$$(PLI): \text{Minimizar} \quad z = -5x_1 - 6x_2 - 3x_3, \tag{6.50}$$

$$\text{sujeito a:} \quad 3x_1 + 4x_2 + 2x_3 \leq 4, \tag{6.51}$$

$$x_1 + x_2 + x_3 \leq 2, \tag{6.52}$$

$$x_1 + x_3 \leq 1, \tag{6.53}$$

$$x_j \in \{0,1\} \;\; j = 1,2,3. \tag{6.54}$$

Associaremos à restrição (6.51) a variável $u_1 \geq 0$ e à restrição (6.52) a variável $u_2 \geq 0$. Sejam $x = (x_1 \; x_2 \; x_3)^\top$ e $X = \{x_1 + x_3 \leq 1 | \; x \in \{0,1\}^3\}$.

$$L(u_1, u_2) = \min\{-5x_1 - 6x_2 - 3x_3 + (3x_1 + 4x_2 + 2x_3 - 4)u_1 + (x_1 + x_2 + x_3 - 2)u_2 \;\; | x \in X\}$$

O problema dual para essa relaxação lagrangiana será:

$$(DPLI): \max L(u_1,u_2), \quad \text{sujeito a: } u_1 \geq 0,\ u_2 \geq 0.$$

Iniciaremos o problema com uma solução viável para $(DPLI)$, $u_1^0 = u_2^0 = 0$.

$$L(u_1^0,u_2^0) = \min\{-5x_1 - 6x_2 - 3x_3 \mid x \in X\}$$

Verificamos, facilmente, que $x_1^0 = x_2^0 = 1, x_3^0 = 0$ fornecerá $L(u_1^0,u_2^0) = L(0,0) = -5\times 1 - 6\times 1 - 3\ \times 0 = -11$, que será uma cota inferior para $val(DPLI)$. Faremos $LB = -11$.

Verificamos também que a solução $x_1^0 = x_2^0 = 1,\ x_3^0 = 0$ não satisfaz à restrição (6.31). No entanto, a solução $x_1^1 = 1,\ x_2^0 = x_3^0 = 0$ é uma solução viável de (PLI), logo $val(PLI) \leq -5$. Tomaremos $UB = -5$.

Etapa 1: Encontrar u^1, tal que $u^1 = u^0 + \alpha_0\gamma^0$.
Passaremos a calcular um subgradiente de L no ponto $u^0 = (u_1^0\ u_2^0) = (0\ 0)$.

$$\gamma^0 = \begin{pmatrix}\gamma_1^0\\ \gamma_2^0\end{pmatrix} = \begin{pmatrix}3x_1^0 + 4x_2^0 + 3x_3^0 - 4\\ x_1^0 + x_2^0 + x_3^0 - 2\end{pmatrix} = \begin{pmatrix}3\\ 0\end{pmatrix}.$$

O passo α_0 será calculado como está em (6.44), isto é:

$$\alpha_0 = \rho_0 \frac{UB - LB}{||\gamma^0||^2},\ \rho_0 \in (0,2].$$

Temos que $||\gamma^0||^2 = 3^2 + 0^2 = 9$, e escolhemos $\rho_0 = 2$.

$$\alpha_0 = 2\frac{-5-(-11)}{9} = \frac{4}{3},$$

$$u^1 = \begin{pmatrix}u_1^1\\ u_2^1\end{pmatrix} = \begin{pmatrix}u_1^0\\ u_2^0\end{pmatrix} + \frac{4}{3}\begin{pmatrix}\gamma_1^0\\ \gamma_2^0\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix} + \frac{4}{3}\begin{pmatrix}3\\ 0\end{pmatrix} = \begin{pmatrix}4\\ 0\end{pmatrix}.$$

Isto é, $u_1^1 = 4$ e $u_2^1 = 0$. Caso houvesse $u_i^j < 0$, tomaríamos $u_i^j := \max\{0,\ u_i^j\}$.

$$L(u^1) = \min\{-5x_1 - 6x_2 - 3x_3 + (3x_1 + 4x_2 + 2x_3 - 4)u_1^1 + (x_1 + x_2 + x_3 - 2)u_2^1 \ |x \in X\}$$

ou ainda

$$L(u^1) = \min\{-5x_1 - 6x_2 - 3x_3 + (3x_1 + 4x_2 + 2x_3 - 4)4 \ |x \in X\}$$
$$L(u^1) = \min\{7x_1 + 10x_2 + 5x_3 - 16 \ |x \in X\}$$

$L(u^1) = -16$, para $x_1^2 = x_2^2 = x_3^2 = 0$. Como $L(u^1) < LB$, a cota inferior não será modificada. A solução $x_1^2 = x_2^2 = x_3^2 = 0$ não é viável para (PLI), assim também não modificaremos a cota superior UB.

Etapa 2:
$u^2 = u^1 + \alpha_1\gamma^1$.

$$\gamma^1 = \begin{pmatrix}\gamma_1^1\\ \gamma_2^1\end{pmatrix} = \begin{pmatrix}3x_1^1 + 4x_2^1 + 3x_3^1 - 4\\ x_1^1 + x_2^1 + x_3^1 - 2\end{pmatrix} = \begin{pmatrix}-4\\ -2\end{pmatrix}.$$

Temos que $||\gamma^1||^2 = (-4)^2 + (-2)^2 = 20$, e escolhemos $\rho_1 = 2$.

$$\alpha_1 = \rho_1 \frac{UB - LB}{||\gamma^1||^2},\ \rho_1 \in (0,2].$$

$$\alpha_1 = 2\frac{-5-(-11)}{20} = \frac{3}{5}.$$

$u_1^2 = \max\{0, 4 + \frac{3}{5}\times(-4)\} = \frac{8}{5}$ e $u_2^2 = \max\{0, 0 + \frac{3}{5}\times(-2)\} = 0$.

$$L(u^2) = \min\{-5x_1 - 6x_2 - 3x_3 + (3x_1 + 4x_2 + 2x_3 - 4)u_1^2 + (x_1 + x_2 + x_3 - 2)u_2^2 \ |x \in X\}.$$

$$L(u^2) = \min\{-5x_1 - 6x_2 - 3x_3 + (3x_1 + 4x_2 + 2x_3 - 4)\frac{8}{5} \ |x \in X\}$$

ou ainda

$$L(u^2) = \min\{-5x_1 - 6x_2 - 3x_3 + (3x_1 + 4x_2 + 2x_3 - 4)\frac{8}{5} \ |x \in X\}.$$

$$L(u^2) = \min\{-\frac{1}{5}x_1 + \frac{2}{5}x_2 + \frac{1}{5}x_3 - \frac{32}{5} \quad | \ x \in X\}.$$

$L(u^2)$ é encontrado para $x_1^2 = 1,\ x_2^2 = x_3^2 = 0$, fornecendo $L(u^2) = -\frac{33}{5} = -6,6$. Para a próxima etapa faríamos $LB = -6,6$. Lembremos que todos os coeficientes da função-objetivo de (PLI) são inteiros, nesse caso consideraremos $LB = -6$.

É interessante notar que a função-objetivo de $(DPLI)$ começou com o valor -11, aumentou para -16 e, finalmente, cresceu até $-6,6$.

No estágio atual sabemos que $-6 \leq val(PLI) \leq -5$. Será que existe uma solução viável de (PLI) tal que $val(PLI) = -6$?

Para isso verificaremos se existe $x = (x_1\ x_2\ x_3)^\top \in X$, tal que $-5x_1 - 6x_2 - 3x_3 = -6$. De fato, fazendo $x_1 = 0,\ x_2 = 1,\ x_3 = 0$, temos que $x \in X$ e obtemos $val(PLI) = -6$.

No nosso exemplo conseguimos encontrar uma solução ótima para (PLI), resolvendo o seu dual lagrangiano $(DPLI)$. Nem sempre isso é possível. Muitas vezes encontramos o máximo de $(DPLI)$, no qual o UB é muito distante do $LB = val(DPLI)$. Nesse caso essas cotas são utilizadas nos métodos de enumeração implícita, tal como branch-and-bound.

7

Combinatória Poliédrica

Manoel Campêlo

7.1 Introdução

Em poucas palavras, combinatória poliédrica refere-se ao estudo de poliedros associados a problemas de otimização combinatória. Esses problemas são definidos em termos de uma coleção $\mathcal{F}$ de subconjuntos de um conjunto universo finito, a cujos elementos atribuímos um valor. O poliedro associado ao problema é dado pela envoltória convexa dos vetores característicos de cada conjunto em $\mathcal{F}$.

Mais precisamente, tais vetores característicos podem ser descritos em geral pelo conjunto de pontos inteiros (ou melhor, binários) em um poliedro $P \subseteq \mathbb{R}^n$, em que n é o número de elementos do universo. Assim, usualmente modelamos um problema de otimização combinatória (POC) como um problema de programação linear 0-1 da forma: $\max c^\top x$ s.a $x \in P \cap \mathbb{B}^n$, em que o vetor $c \in \mathbb{R}^n$ representa o valor dos elementos do universo. Aqui, vamos considerar problemas mais gerais, em que substituímos $\mathbb{B}^n$ por $\mathbb{Z}^n$, ou seja,

$$\max c^\top x \text{ s.a } x \in P \cap \mathbb{Z}^n. \tag{7.1}$$

Como veremos adiante, podemos substituir, em (7.1), o conjunto discreto $P \cap \mathbb{Z}^n$ por sua envoltória convexa, que, por sua vez, descreve um poliedro (associado a (7.1)). Dessa maneira, (7.1) pode ser reformulado como um problema de programação linear (PPL).

O objetivo final deste capítulo é descrever parcial ou totalmente tal poliedro. Para atingi-lo, apresentamos duas caracterizações de poliedro na seção 7.3. Uma delas estabelece quais desigualdades lineares são necessárias e suficientes para descrever um poliedro; a outra caracteriza poliedro como um conjunto gerado através de um número finito de pontos e direções. Enquanto a partir desta deduzimos que (7.1) equivale de fato a um PPL, aquela será primordial para caracterizar tal PPL. Esses dois resultados, essenciais nesse contexto, serão objeto da seção 7.4. Os fundamentos teóricos para esse desenvolvimento compõem a seção 7.2, em que apresentamos conceitos e propriedades associados a conjuntos afins e convexos. Na seção 7.5, apresentamos condições necessárias e/ou suficientes para determinar se um poliedro é inteiro, o que nos possibilita reconhecer quando temos uma descrição completa do po-

liedro associado a (7.1). Finalmente, na seção 7.6 sugerimos como usar efetivamente essa equivalência entre (7.1) e um PPL, visando a sua resolução, através do método de planos de corte.

O conteúdo aqui apresentado é, em sua maior parte, composto por resultados clássicos, que podem ser encontrados (possivelmente com outra apresentação ou em outro encadeamento) em excelentes livros de otimização combinatória ou Teoria Poliédrica [170, 190, 188, 216, 223] ou compilações específicas sobre combinatória poliédrica [2, 3, 40, 76, 93, 179, 180, 192]. Bibliografias comentadas podem ser encontradas em [4, 110].

Procuramos construir um texto autocontido, valorizando detalhes de como utilizar de forma prática os resultados teóricos apresentados. Supomos que o leitor possua conhecimento básico de álgebra linear e programação linear, especialmente dualidade linear. Alertamos ainda que faremos uso frequente do Lema de Farkas.

7.2 Conceitos e propriedades preliminares

7.2.1 Conjuntos afins, cônicos e convexos

Usando as operações de soma vetorial e multiplicação por escalar, podemos combinar vetores para gerar um outro. As seguintes formas de combinação são frequentemente usadas neste texto.

Definição 7.2.1. Dados $S = \{x^1, x^2, \ldots, x^p\} \subset \mathbb{R}^n$ e $\alpha_1, \alpha_2, \ldots, \alpha_p \in \mathbb{R}$, dizemos que o vetor $x = \sum_{i=1}^p \alpha_i x^i$ é uma:

i) combinação linear de S;
ii) combinação afim de S, se $\sum_{i=1}^p \alpha_i = 1$;
iii) combinação cônica de S, se $\alpha_i \geq 0$, para todo $i \in \{1, 2, \ldots, p\}$;
iv) combinação convexa de S, se $\sum_{i=1}^p \alpha_i = 1$ e $\alpha_i \geq 0$, para todo $i \in \{1, 2, \ldots, p\}$, ou seja, se a combinação é afim e cônica.

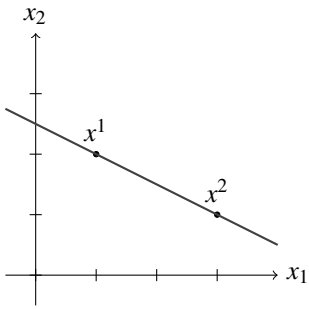
É comum também considerarmos o conjunto obtido por todas as possíveis combinações de um mesmo tipo.

Definição 7.2.2. O fecho linear (resp. fecho afim, fecho cônico, fecho convexo) de um conjunto não vazio $S \subseteq \mathbb{R}^n$, denotado por $\text{span}(S)$ (resp. $\text{afim}(S)$, $\text{cone}(S)$, $\text{conv}(S)$), é o conjunto de todas as combinações lineares (resp. afins, cônicas, convexas) de seus subconjuntos finitos. Adicionalmente, $\text{span}(\emptyset) = \text{cone}(\emptyset) = \{0\}$ e $\text{afim}(\emptyset) = \text{conv}(\emptyset) = \emptyset$. O termo fecho é comumente substituído por envoltória.

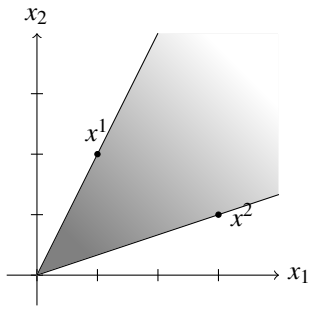
A figura 7.1 mostra os fechos afim, cônico e convexo de dois pontos x^1 e x^2. Note que x é combinação afim de x^1 e x^2 se $x = (1-\alpha)x^1 + \alpha x^2 = x^1 + \alpha(x^2 - x^1)$, para algum $\alpha \in \mathbb{R}$. Logo, o fecho afim de x^1 e x^2 é a reta passando por esses dois pontos (figura 7.1(a)). Adicionalmente, para uma combinação convexa, devemos ter $\alpha \in [0,1]$, de modo que o fecho convexo de x^1 e x^2 se reduz ao segmento de reta entre esses dois pontos (figura 7.1(c)). Por outro lado, observe que $\{\alpha x : \alpha \geq 0\}$ é a semirreta começando na origem e contendo x. Assim, o fecho cônico de x^1 e x^2 é formado pelas semirretas $\{\alpha_1 x^1 : \alpha_1 \geq 0\}$ e $\{\alpha_2 x^2 : \alpha_2 \geq 0\}$, bem como pela soma de quaisquer dois pontos sobre tais semirretas (figura 7.1(b)).

Exercício 7.2.1. Sejam $S, R \subseteq \mathbb{R}^n$. Mostre que:

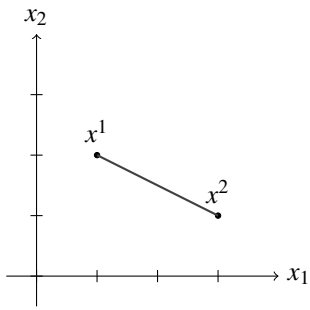
i) $S \subseteq \text{conv}(S) \subseteq \text{afim}(S)$ e $S \subseteq \text{conv}(S) \subseteq \text{cone}(S)$;
ii) $\text{afim}(R \cap S) \subseteq \text{afim}(R) \cap \text{afim}(S)$, $\text{cone}(R \cap S) \subseteq \text{cone}(R) \cap \text{cone}(S)$, $\text{conv}(R \cap S) \subseteq \text{conv}(R) \cap \text{conv}(S)$.



(a) Fecho afim.



(b) Fecho cônico.



(c) Fecho convexo.

Figura 7.1: Fechos de dois pontos.

Associados às combinações, definimos os seguintes tipos de conjuntos:

Definição 7.2.3. Um conjunto $S \subseteq \mathbb{R}^n$ é dito um subespaço linear (resp. conjunto afim, cone, conjunto convexo) se $S = \text{span}(S)$ (resp. $S = \text{afim}(S)$, $S = \text{cone}(S)$, $S = \text{conv}(S)$). Dizemos ainda que $\text{span}(S)$ (resp. $\text{afim}(S)$, $\text{cone}(S)$, $\text{conv}(S)$) é o subespaço linear (resp. conjunto afim, cone, conjunto convexo) gerado por S.

Exercício 7.2.2. Mostre que as seguintes assertivas são equivalentes:
i) $C = \text{afim}(S)$;
ii) C é o menor conjunto afim contendo S;
iii) C é a interseção de todos os conjuntos afins contendo S.
Mostre essas mesmas equivalências com respeito a conjunto convexo.

Considerando que subespaços lineares são conjuntos bastante explorados em cursos de álgebra linear, concentramo-nos aqui em conjuntos afins, convexos e cônicos. Começamos redefinindo esses conjuntos através de combinações de até dois pontos.

Proposição 7.2.1. Para $S \subseteq \mathbb{R}^n$, $S \neq \emptyset$, as seguintes caracterizações são válidas:
i) S é conjunto afim se, e somente se, para todos $x, y \in S$ e $\alpha \in \mathbb{R}$, $(1-\alpha)x + \alpha y \in S$.
ii) S é conjunto convexo se, e somente se, para todos $x, y \in S$ e $\alpha \in [0,1]$, $(1-\alpha)x + \alpha y \in S$.
iii) S é um cone se, e somente se, S é convexo e, para todos $x \in S$ e $\alpha \geq 0$, $\alpha x \in S$.

Prova: Para mostrar (i), primeiro suponha que S seja afim, isto é, $S = \text{afim}(S)$. Pela definição 7.2.2, dados $x, y \in S$ e $\alpha \in \mathbb{R}$, temos que $(1-\alpha)x + \alpha y \in \text{afim}(S) = S$. Agora, suponha que

$$(1-\alpha)x + \alpha y \in S, \forall x, y \in S, \forall \alpha \in \mathbb{R}. \tag{7.2}$$

Como $S \subseteq \text{afim}(S)$, resta mostrar a inclusão inversa ou ainda que $\text{afim}(S') \subseteq S$, para todo subconjunto finito $S' \subseteq S$. O resultado vale trivialmente se $|S'| = 1$. Suponha-o também verdadeiro para todo $S' \subseteq S$ com $|S'| \leq s$. Considere agora que $|S'| = s+1$, com $S' = \{x^0, \ldots, x^s\}$, $s \geq 1$. Seja $x \in \text{afim}(S')$, isto é, $x = \alpha_0 x^0 + \sum_{i=1}^s \alpha_i x^i$, com $\sum_{i=0}^s \alpha_i = 1$. Queremos mostrar que $x \in S$. Como $s \geq 1$, existe algum multiplicador, digamos α_0, diferente de 1. Temos então $x = \alpha_0 x^0 + (1-\alpha_0)y$, onde $y = \sum_{i=1}^s \frac{\alpha_i}{1-\alpha_0} x^i \in \text{afim}(S' \setminus \{x^0\})$. Pela hipótese indutiva, $y \in S$. Finalmente, por (7.2), obtemos $x \in S$.

A demonstração de (ii) é similar e será deixada como exercício.

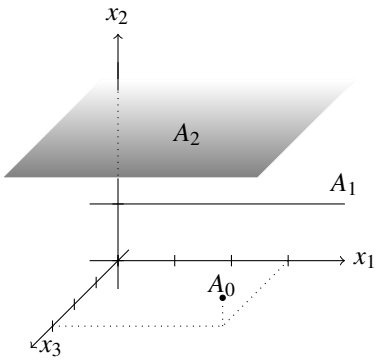
Com respeito a (iii), note que a implicação direta decorre trivialmente da definição de cone. Admita agora que $S = \text{conv}(S)$ e $\alpha x \in S$, para todo $x \in S$ e $\alpha \geq 0$ (em particular $0 \in S$). Queremos mostrar que $\text{cone}(S) \subseteq S$, posto que $S \subseteq \text{cone}(S)$. Seja $x \in \text{cone}(S)$, isto é, $x = \sum_{i=1}^p \alpha_i x^i$ com $x^i \in S$ e $\alpha_i \geq 0$, para todo $i \in \{1, \ldots, p\}$. Defina $\alpha = \sum_{i=1}^p \alpha_i$. Caso $\alpha = 0$, temos $x = 0 \in S$. Caso contrário, temos que $\frac{x}{\alpha} = \sum_{i=1}^p \frac{\alpha_i}{\alpha} x^i \in \text{conv}(S) = S$. Logo, $x = \alpha \frac{x}{\alpha} \in S$. Com isso, concluímos que $\text{cone}(S) \subseteq S$. ∎

A caracterização acima fornece uma visão geométrica desses conjuntos. Dados $x, y \in \mathbb{R}^n$, os pontos da forma $(1-\alpha)x + \alpha y$ definem a reta passando por x e y, quando $\alpha \in \mathbb{R}$, ou o segmento de reta entre x e y, quando $\alpha \in [0,1]$. Já os pontos da forma αx, $\alpha \geq 0$, definem a semirreta começando na origem e contendo x.

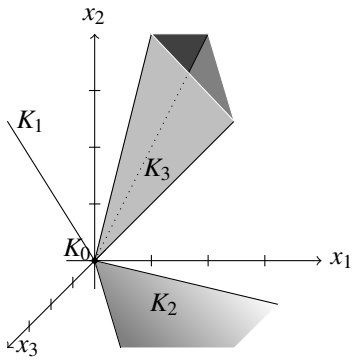
A seguir, vamos relacionar conjunto afim com subespaço linear e sistema de equações lineares. Relembre que, dados $S, R \subseteq \mathbb{R}^n$ e $\lambda \in \mathbb{R}$, $\lambda S = \{\lambda x : x \in S\}$, $S + R = \{x + y : x \in S, y \in R\}$ (soma de Minkowski) e $S - R = S + (-R)$. Quando $R = \{y\}$, simplificamos a notação $S + R$ como $S + y$, que define o transladado de S por y. S e $S + y$ são ditos conjuntos paralelos.

Exercício 7.2.3. Sejam $S \subseteq \mathbb{R}^n$ conjunto afim e $x, y \in S$. Mostre que $S - x = S - y = S - S$.

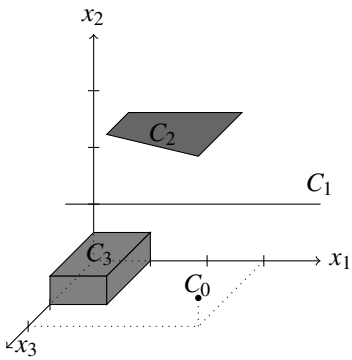
Lema 7.2.1. Para $S \subseteq \mathbb{R}^n$ e $y \in S$, $\text{afim}(S) - y = \text{span}(S - y)$.



(a) A_0, A_1 e A_2 são conjuntos afins.



(b) K_0, K_1, K_2 e K_3 são cones.



(c) C_0, C_1, C_2 e C_3 são convexos.

Figura 7.2: Exemplos de conjuntos afins, cones e conjuntos convexos.

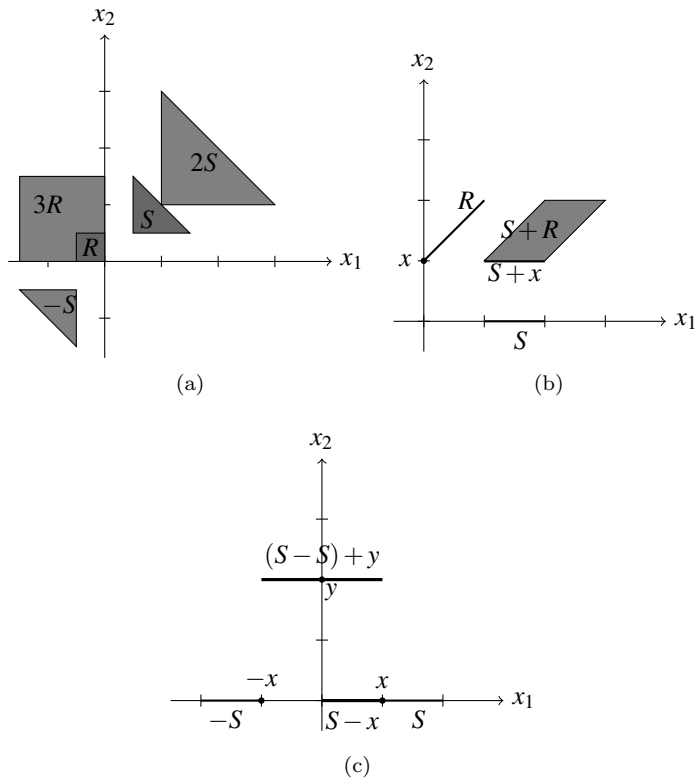


Figura 7.3: Multiplicação por escalar e soma de Minkowski.

Prova: Seja $x \in \text{afim}(S) - y$, ou seja, $x = (\sum_{i=1}^{p} \alpha_i x^i) - y$, com $\sum_{i=1}^{p} \alpha_i = 1$ e $x^i \in S$, para todo $i \in \{1, \ldots, p\}$. Logo, $x = \sum_{i=1}^{p} \alpha_i (x^i - y) \in \text{span}(S - y)$. Reciprocamente, seja $x \in \text{span}(S - y)$, ou seja, $x = \sum_{i=1}^{p} \alpha_i (x^i - y) = \sum_{i=1}^{p} \alpha_i x^i + (1 - \sum_{i=1}^{p} \alpha_i) y - y$, com $y \in S$ e $x^i \in S$, para todo $i \in \{1, \ldots, p\}$. Logo, $x \in \text{afim}(S) - y$. ∎

Corolário 7.2.1. Seja $S \subseteq \mathbb{R}^n$ e $y \in S$. S é um conjunto afim se, e somente se, o transladado $S - y$ é um subespaço linear. Em particular, S é um conjunto afim e $0 \in S$ se, e somente se, S é um subespaço linear.

Os resultados acima mostram que conjunto afim é o translado de um subespaço linear e que este subespaço paralelo é único. Outra caracterização, em termos de sistemas lineares, é dada abaixo, em que nos valemos do fato de que o complemento ortogonal a um subespaço linear $S \subseteq \mathbb{R}^n$ é o subespaço linear $S^\perp = \{y \in \mathbb{R}^n : x^\top y = 0, \forall x \in S\}$ e que $(S^\perp)^\perp = S$.

Teorema 7.2.1. Se $b \in \mathbb{R}^m$ e $A \in \mathbb{R}^{m \times n}$, então o conjunto $C = \{x \in \mathbb{R}^n : Ax = b\}$ é afim. Por outro lado, se $C \subseteq \mathbb{R}^n$ é um conjunto afim, então existem $b \in \mathbb{R}^m$ e $A \in \mathbb{R}^{m \times n}$ tais

que $C = \{x \in \mathbb{R}^n : Ax = b\}$.

Prova: Sejam $x, y \in C = \{x \in \mathbb{R}^n : Ax = b\}$ e $\alpha \in \mathbb{R}$. Então, $A((1-\alpha)x + \alpha y) = (1-\alpha)b + \alpha b = b$, implicando que C seja afim. Agora, mostramos a recíproca. Se $C = \emptyset$ ou $\mathbb{R}^n$, o resultado é direto. Do contrário, seja $S = C - \bar{x}$, $\bar{x} \in C$, o subespaço linear paralelo a C. Seja $\{x^1, x^2, \ldots, x^m\}$ uma base para o subespaço $S^\perp$, ortogonal a S. Portanto, $S = (S^\perp)^\perp = \{y \in \mathbb{R}^n : (x^i)^\top y = 0, i \in \{1, 2, \ldots, m\}\}$. Seja $A \in \mathbb{R}^{m \times n}$ a matriz cuja i-ésima linha é $(x^i)^\top$. Logo, $S = \{y \in \mathbb{R}^n : Ay = 0\}$. Com isso, temos que $C = S + \bar{x} = \{y + \bar{x} \in \mathbb{R}^n : Ay = 0\} = \{x \in \mathbb{R}^n : Ax = A\bar{x}\}$. O resultado segue, tomando $b = A\bar{x}$. ■

Como vimos, todo cone é um conjunto convexo. Por outro lado, a todo conjunto convexo estão associados cones que definimos a seguir.

Definição 7.2.4. Seja $C \subseteq \mathbb{R}^n$ um conjunto convexo não vazio. Uma direção de recessão de C é um vetor $d \in \mathbb{R}^n \setminus \{0\}$ tal que $x + \lambda d \in C$, para todo $x \in C$ e $\lambda \geq 0$. O cone de recessão de C, denotado por $\mathrm{rec}(C)$, é o conjunto de todas as direções de recessão de C junto com 0.

Podemos mostrar que, se existe $\bar{x} \in C$ tal que $\bar{x} + \lambda d \in C$, para todo $\lambda \geq 0$, então o mesmo acontece para todo $x \in C$ [186, teorema 8.3], quando C é convexo, não vazio e fechado. Nesse caso, podemos, alternativamente, definir uma direção de recessão d considerando apenas a existência de um ponto $\bar{x}$ tal que a semirreta $\bar{x} + \lambda d$, $\lambda \geq 0$, pertença ao conjunto.

Na figura 7.4, exibimos um conjunto convexo e seu cone de recessão. Para os convexos da figura 7.2(c), temos $\mathrm{rec}(C_0) = \mathrm{rec}(C_2) = \mathrm{rec}(C_3) = \emptyset$ e $\mathrm{rec}(C_1)$ igual aos pontos do eixo x_1.

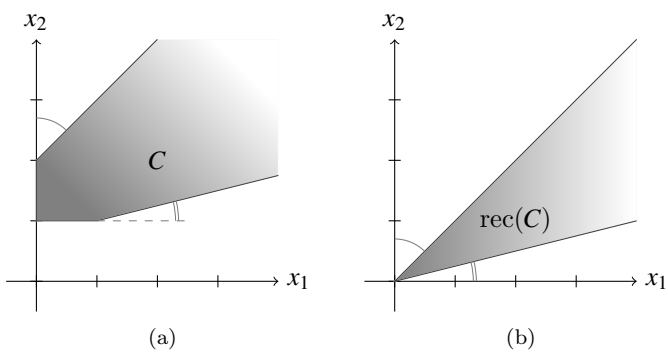


Figura 7.4: Exemplo de conjunto convexo e seu cone de recessão.

Quando ambos d e $-d$ pertencem a $\mathrm{rec}(C)$, tais vetores compõem o espaço de linearidade de C, o maior subespaço linear contido em C.

Definição 7.2.5. O espaço de linearidade de um conjunto convexo não vazio $C \subseteq \mathbb{R}^n$, denotado por $\mathrm{lin}(C)$, é composto pelos vetores $d \in \mathbb{R}^n$ tais que $x + \lambda d \in C$, para todo $x \in C$ e $\lambda \in \mathbb{R}$.

Exercício 7.2.4. Para $C, D \subseteq \mathbb{R}^n$ convexos não vazios, mostre que:

i) $\text{rec}(C)$ é um cone.
ii) $C = C + \text{rec}(C)$.
iii) $\text{rec}(-C) = -\text{rec}(C)$.
iv) Se $C \cap D \neq \emptyset$, então $\text{rec}(C \cap D) = \text{rec}(C) \cap \text{rec}(D)$.
v) $\text{lin}(C) = \text{rec}(C) \cap \text{rec}(-C)$.
vi) C é limitado se, e somente se, $\text{rec}(C) = \{0\}$.

7.2.2 Dimensão e posto

A ideia de dimensão de um conjunto é bastante intuitiva do ponto de vista geométrico. Aqui, formalizamos essa noção através do conceito de posto, ou melhor, de independência (linear ou afim) dos pontos do conjunto. Em termos gerais, dizemos que um ponto x é linear ou afim-independente de um conjunto S se $x \notin \text{span}(S)$ ou $x \notin \text{afim}(S)$, respectivamente. Mais formalmente:

Definição 7.2.6. Um conjunto de vetores $S \subseteq \mathbb{R}^n$ é dito afim-independente (AI) se

$$\forall S' \subseteq S, \text{ finito: } \sum_{x \in S'} \alpha_x x = 0, \sum_{x \in S'} \alpha_x = 0 \Rightarrow \alpha = 0.$$

Caso contrário, é dito afim-dependente (AD). Em particular, S é dito linearmente independente (LI) se

$$\forall S' \subseteq S, \text{ finito: } \sum_{x \in S'} \alpha_x x = 0 \Rightarrow \alpha = 0.$$

Se não é LI, o conjunto é linearmente dependente (LD).

Os conceitos de dependência e independência linear e afim são ilustrados na figura 7.5.

Exercício 7.2.5. Mostre que as seguintes assertivas são equivalentes:
i) $\{x^0, x^1, \ldots, x^p\} \subset \mathbb{R}^n$ é AI;
ii) $\{x^1 - x^0, x^2 - x^0, \ldots, x^p - x^0\} \subset \mathbb{R}^n$ é LI;
iii) $\{(x^0, -1), (x^1, -1), \ldots, (x^p, -1)\} \subset \mathbb{R}^{n+1}$ é LI;

Exercício 7.2.6. Mostre que S é LD (resp. AD) se, e somente se, existe $x \in S$ tal que $x \in \text{span}(S \setminus \{x\})$ (resp. $x \in \text{afim}(S \setminus \{x\})$).

Exercício 7.2.7. Mostre que $\{x^0, x^1, \ldots, x^p\} \subset \mathbb{R}^n$ é AI se, e somente se, para todo $i \in \{1, 2, \ldots, p\}$, existe $(a^i, b_i) \in \mathbb{R}^n \times \mathbb{R}$ tal que $(a^i)^\top x^i \neq b_i$ e $(a^i)^\top x^j = b_i$, para $j < i$.

Definição 7.2.7. O posto (resp. posto-afim) de um conjunto $S \subseteq \mathbb{R}^n$ é a cardinalidade do maior conjunto LI (resp. AI) em S.

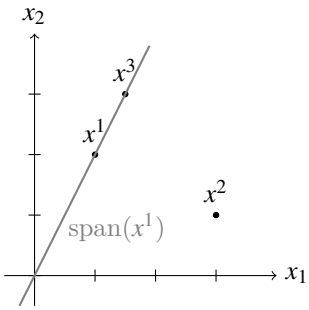
Exercício 7.2.8. Mostre que

$$\text{posto-afim}(S) = \begin{cases} \text{posto}(S) + 1, & \text{se } 0 \in \text{afim}(S) \\ \text{posto}(S), & \text{se } 0 \notin \text{afim}(S) \end{cases}$$

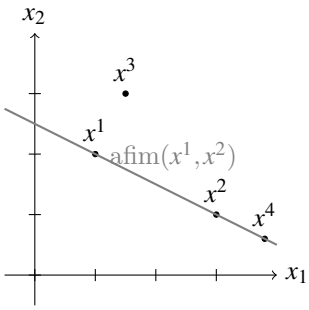
Veja a diferenciação entre os casos na figura 7.6.

Como sabemos, o posto de um subespaço linear define sua dimensão. À luz do corolário 7.2.1, definimos a dimensão de um conjunto afim como a dimensão do subespaço linear paralelo a ele. Em geral:

Definição 7.2.8. A dimensão de um conjunto não vazio $S \subseteq \mathbb{R}^n$, denotada por $\dim(S)$, é o posto do subespaço linear paralelo a $\text{afim}(S)$. Adicionalmente, $\dim(\emptyset) = -1$. Se $\dim(S) = n$, dizemos que S tem dimensão plena.



(a) $\{x^1, x^2\}$ é LI. $\{x^1, x^3\}$ é LD.



(b) $\{x^1, x^2, x^3\}$ é AI. $\{x^1, x^2, x^4\}$ é AD.

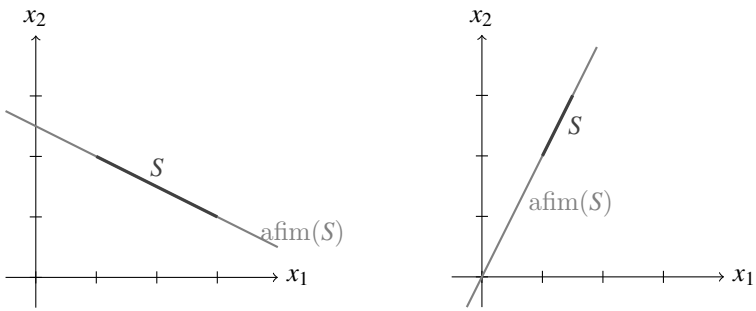
Figura 7.5: Dependência e independência.

A figura 7.2 apresenta conjuntos de diferentes dimensões. Com respeito aos conjuntos afins da figura 7.2(a), temos claramente $\dim(A_i) = i$, para $i \in \{0,1,2\}$. Similarmente, os cones e convexos das figuras 7.2(b) e 7.2(c) são tais que $\dim(K_i) = \dim(C_i) = i$, para $i \in \{0,1,2,3\}$.

Note que $\dim(S) = \dim(\text{afim}(S))$. Na verdade, como $\dim(S) \leq \dim(R)$, para todo $S \subseteq R$, temos que $\dim(S) = \dim(\text{conv}(S)) = \dim(\text{afim}(S))$. Podemos então relacionar a dimensão de um conjunto com seu posto-afim.

Proposição 7.2.2. Seja $S \subseteq \mathbb{R}^n$. Então $\dim(S) = \text{posto-afim}(S) - 1$.

Prova: Podemos considerar $S \neq \emptyset$, do contrário o resultado é trivial por definição. Sejam $d = \dim(S)$, $p = \text{posto-afim}(S)$ e S' o espaço paralelo a $\text{afim}(S)$. Note que $d = \dim(\text{afim}(S)) = \dim(S')$ e $p = \text{posto-afim}(\text{afim}(S))$. Temos também que $S' = \text{afim}(S) - x^0$, com $x^0 \in \text{afim}(S)$. Primeiro, tome o conjunto LI $\{x^1 - x^0, x^2 - x^0, \ldots, x^d - x^0\} \subseteq S'$, onde $x^i \in \text{afim}(S)$, para todo $i \in \{0,1,\ldots,d\}$. Pelo exercício 7.2.5, $\{x^0, x^1, \ldots, x^d\}$ é AI. Logo $p \geq d+1$. Agora, tome o conjunto AI $\{x^1, x^2, \ldots, x^p\} \subseteq S \subseteq \text{afim}(S)$. Pela definição de S' e exercício 7.2.5, $\{x^2 - x^1, x^3 - x^1, \ldots, x^p - x^1\} \subseteq S'$ e é LI. Portanto, $d \geq p-1$. Concluímos que $d = p-1$. ∎



(a) posto-afim(S) = posto(S) = 2. (b) posto-afim(S) = 2 = posto(S) + 1.

Figura 7.6: Posto × posto-afim.

Além disso, a partir da caracterização de conjunto afim através de sistema linear, obtemos outra expressão para a dimensão de um conjunto. Para isso, usamos a clássica igualdade $\dim(\mathcal{N}(A)) + \text{posto}(A) = n$, válida para qualquer matriz $A \in \mathbb{R}^{m\times n}$, em que $\mathcal{N}(A) = \{x \in \mathbb{R}^n : Ax = 0\}$ é o núcleo de A e posto(A) é o maior número de linhas (ou colunas) LI de A.

Proposição 7.2.3. Sejam $S \subseteq \mathbb{R}^n$, $S \neq \emptyset$, e $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^m$ tais que $\text{afim}(S) = \{x \in \mathbb{R}^n : Ax = b\}$. Então $\dim(S) = n - \text{posto}(A)$.

Prova: Seja $x \in S \subseteq \text{afim}(S)$. Note que $y \in \text{afim}(S)$ se, e somente se, $A(y - x) = 0$, ou melhor, $y - x \in \mathcal{N}(A)$. Logo, $\text{afim}(S) = x + \mathcal{N}(A)$ e, então, $\dim(S) = \dim(\mathcal{N}(A)) = n - \text{posto}(A)$. ∎

Na figura 7.6, apresentamos dois conjuntos unidimensionais. Na figura 7.6(a), temos $\text{afim}(S) = \{(x_1, x_2) : x_1 + 2x_2 = 5\}$, enquanto $\text{afim}(S) = \{(x_1, x_2) : -2x_1 + x_2 = 0\}$ na figura 7.6(b). Em ambos os casos, $\dim(S)$ pode ser obtido, alternativamente, pelas proposições 7.2.2 e 7.2.3.

Na verdade, tais proposições são frequentemente usadas conjuntamente para fornecer limites para a dimensão de um conjunto $S \subseteq \mathbb{R}^n$. Se S possui $p + 1$ elementos afim-independentes, então $\dim(S) \geq p$. Por outro lado, se $S \subseteq \{x \in \mathbb{R}^n : Ax = b\}$, então $\dim(S) \leq n - \text{posto}(A)$.

7.2.3 Hiperplanos e semiespaços

Hiperplanos e semiespaços são conjuntos convexos de especial interesse em nosso contexto.

Definição 7.2.9. Um hiperplano é um conjunto afim em $\mathbb{R}^n$ de dimensão $n - 1$.

Teorema 7.2.2. $H \subset \mathbb{R}^n$ é um hiperplano se, e somente se, existem $\pi \in \mathbb{R}^n \backslash \{0\}$ e $\pi_0 \in \mathbb{R}$ tais que $H = \{x \in \mathbb{R}^n : \pi^\top x = \pi_0\} \neq \emptyset$. Mais ainda, a representação de H em termos de (π, π_0) é única, a menos de múltiplos, ou seja, $H = \{x \in \mathbb{R}^n : \lambda^\top x = \lambda_0\}$ implica $(\lambda, \lambda_0) = \alpha(\pi, \pi_0)$ para algum $\alpha \neq 0$.

Prova: Pela proposição 7.2.3 e teorema 7.2.1, temos que H é hiperplano se, e somente se, $H = \{x \in \mathbb{R}^n : Ax = b\} \neq \emptyset$ para um vetor b e uma matriz A com posto 1. Daí segue a primeira parte do enunciado. Também da proposição 7.2.3 deduzimos que, se $x \in H$ satisfaz $\pi^\top x = \pi_0$ e $\lambda^\top x = \lambda_0$, então λ é um múltiplo de π. A mesma relação deve, consequentemente, existir entre λ_0 e π_0. ∎

Corolário 7.2.2. Todo conjunto afim de $\mathbb{R}^n$ é a interseção de um número finito de hiperplanos.

Associados a um hiperplano $H = \{x \in \mathbb{R}^n : \pi^\top x = \pi_0\}$, definimos quatro conjuntos obtidos pela substituição da igualdade em $\pi^\top x = \pi_0$ por uma desigualdade da forma $\geq$, $\leq$, $>$ ou $<$.

Definição 7.2.10. Um semiespaço (fechado) é um conjunto da forma $\{x \in \mathbb{R}^n : \pi^\top x \leq \pi_0\}$, enquanto um semiespaço aberto é da forma $\{x \in \mathbb{R}^n : \pi^\top x < \pi_0\}$, para $\pi \in \mathbb{R}^n \backslash \{0\}$ e $\pi_0 \in \mathbb{R}$.

Definição 7.2.11. Sejam $C, D \subseteq \mathbb{R}^n$ não vazios. Um hiperplano $H = \{x \in \mathbb{R}^n : \pi^\top x = \pi_0\}$ separa C e D se C está contido em um dos semiespaços associados a H e D está contido no outro semiespaço associado a H, isto é,

$$C \subseteq \{x \in \mathbb{R}^n : \pi^\top x \leq \pi_0\} \quad \text{e} \quad D \subseteq \{x \in \mathbb{R}^n : \pi^\top x \geq \pi_0\}.$$

A figura 7.7(a) apresenta um hiperplano $\pi^T x = \pi_0$ e os semiespaços a ele associados. Observe que o vetor π é ortogonal ao hiperplano e "aponta" para o semiespaço $\pi^T x \geq \pi_0$. A figura 7.7(b) mostra que o hiperplano separa o conjunto convexo C do conjunto $\{\bar{x}\}$.

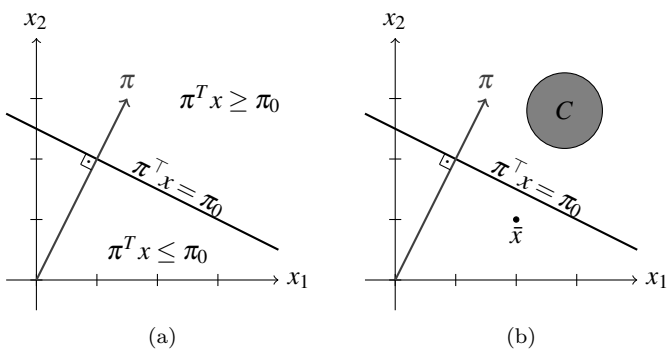


Figura 7.7: Hiperplano, semiespaço e separação.

7.3 Poliedros

7.3.1 Introdução

Nesta seção, definimos nosso principal objeto de interesse, poliedro, e introduzimos uma notação associada.

Definição 7.3.1. Poliedro é um conjunto definido pela interseção de um número finito de semiespaços. Um poliedro limitado é chamado politopo.

Em outras palavras, poliedro é o conjunto solução de um sistema de desigualdades lineares.

Proposição 7.3.1. $P \subseteq \mathbb{R}^n$ é poliedro se, e somente se, existem $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^m$ tais que $P = \{x \in \mathbb{R}^n : Ax \leq b\}$.

Exemplos de poliedros e politopos em $\mathbb{R}^2$ são apresentados na figura 7.8. Além desses, todos os cones e conjuntos convexos das figuras 7.2(b) e 7.2(c) também são poliédricos.

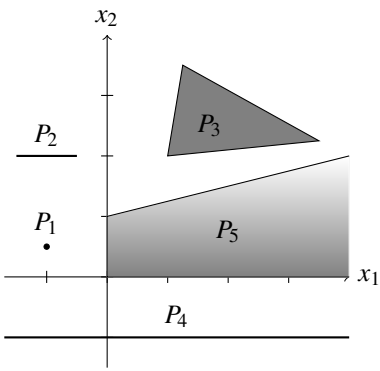


Figura 7.8: Exemplos de poliedros em $\mathbb{R}^2$. Em particular, P_1, P_2 e P_3 são politopos.

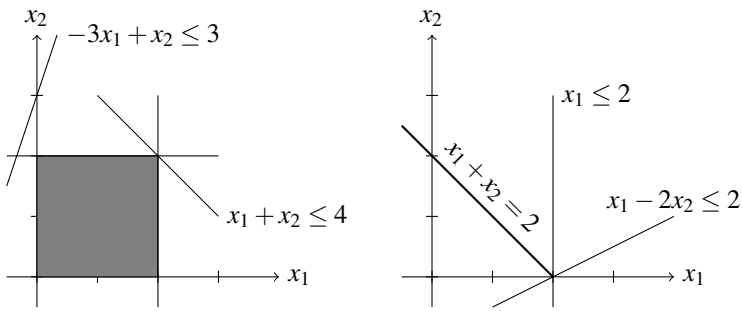
Exercício 7.3.1. Seja $P = \{x \in \mathbb{R}^n : Ax \leq b\} \neq \emptyset$ um poliedro. Mostre que P é convexo, $\text{rec}(P) = \{d \in \mathbb{R}^n : Ad \leq 0\}$ e $\text{lin}(P) = \{d \in \mathbb{R}^n : Ad = 0\}$. Conclua que $\text{lin}(P) = \{0\}$ se, e somente se, $\text{posto}(A) = n$.

Ao longo desta seção, reservamos a letra P para denotar um poliedro em $\mathbb{R}^n$. A matriz (A,b) é chamada uma representação ou descrição do poliedro $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Em alguns momentos vamos usar a notação $P(A,b)$ para expressar claramente a representação do poliedro P. Note que duas ou mais desigualdades em P podem implicar uma ou mais igualdades (por exemplo, $-x_1 - x_2 \leq 0$, $x_1 \leq 0$, $x_2 \leq 0$ implicam $x_1 = 0$, $x_2 = 0$), que são chamadas igualdades implícitas.

Todo poliedro tem infinitas representações. Por exemplo, podemos modificar uma dada representação por:

1. Acréscimo de desigualdades redundantes.

$$
\begin{array}{rrrrrr}
(a) & 0 \leq x_1 \leq 2 & (b) & 0 \leq x_1 \leq 2 & (c) & 0 \leq x_1 \leq 2 \\
 & 0 \leq x_2 \leq 2 & & 0 \leq x_2 \leq 2 & & 0 \leq x_2 \leq 2 \\
 & & & x_1 + x_2 \leq 4 & & -3x_1 + x_2 \leq 3
\end{array}
$$



(a) Desigualdades redundantes. (b) Desigualdades alternativas.

Figura 7.9: Diferentes representações de um poliedro.

2. Combinação de uma desigualdade com igualdades implícitas.

$$(a)\quad \begin{aligned} x_1 + x_2 &\leq 2 \\ -x_1 - x_2 &\leq -2 \\ x_1 &\leq 2 \end{aligned} \qquad (b)\quad \begin{aligned} x_1 + x_2 &\leq 2 \\ -x_1 - x_2 &\leq -2 \\ x_1 - 2x_2 &\leq 2 \quad \Leftrightarrow 3(x_1 \leq 2) - 2(x_1 + x_2 = 2) \end{aligned}$$

Definição 7.3.2. Um poliedro é dito racional se existe uma representação sua (A,b) com todos os elementos racionais.

Notação Dado um poliedro $P(A,b)$, onde $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^m$, vamos denotar por:

- $(a^i)^\top$, b_i: i-ésima linha de A e i-ésima componente de b;
- $M = \{1,2,\ldots,m\}$: índices das desigualdades em $Ax \leq b$;
- $M^= = \{i \in M : \forall x \in P, (a^i)^\top x = b_i\}$: índices das igualdades implícitas;
- $M^\leq = M \backslash M^= = \{i \in M : \exists x \in P, (a^i)^\top x < b_i\}$: índices das desigualdades legítimas;
- $(A^=, b^=)$: submatriz de (A,b) com linhas indexadas por $M^=$, se $M^= \neq \emptyset$ (por conveniência podemos admitir $(A^=, b^=) = (0,0)$, se $M^= = \emptyset$);
- $(A^\leq, b^\leq)$: submatriz de (A,b) com linhas indexadas por $M^\leq$, se $M^\leq \neq \emptyset$ (por conveniência podemos admitir $(A^\leq, b^\leq) = (0,1)$, se $M^\leq = \emptyset$).

Quando desejarmos deixar explícito o poliedro, usaremos $_P$ como subíndice na notação dos conjuntos e submatrizes definidos acima (por exemplo, $M_P^=$ e $(A_P^=, b_P^=)$).

Note que podemos escrever alternativamente $P(A,b) = \{x \in \mathbb{R}^n : A^= x = b^=, A^\leq x \leq b^\leq\}$. Importante observar também que, se $i \in M^\leq$, então $((a^i)^\top, b_i)$ não pode ser escrito como combinação linear das linhas de $(A^=, b^=)$.

7.3.2 Dimensão

Da subseção 7.2.2, sabemos que a dimensão de um conjunto está relacionada com seu posto-afim. Aqui, especializamos a proposição 7.2.3 para poliedros. Para caracterizar afim(P), primeiro obtemos alguns resultados auxiliares sobre pontos internos de P.

Definição 7.3.3. Um ponto $x \in P(A,b)$ é ponto interno se $A^\leq x < b^\leq$ e ponto interior se $Ax < b$. Seja inn(P) o conjunto dos pontos internos de P.

A figura 7.10 ilustra os conceitos de ponto interno e ponto interior em poliedros.

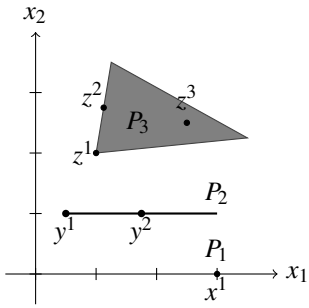


Figura 7.10: Pontos internos: x^1, y^2, z^3. Ponto interior: z^3. Não são internos: y^1, z^1 e z^2. P_1 e P_2 não têm ponto interior.

Proposição 7.3.2. Se $P \neq \emptyset$, então P possui ponto interno.

Prova: Se $M^{\leq} = \emptyset$, todo ponto é interno. Considere então que $M^{\leq} \neq \emptyset$. Para todo $i \in M^{\leq}$ existe $x^i \in P$ tal que $(a^i)^\top x^i < b_i$. Tome $\bar{x} = \frac{1}{|M^{\leq}|}\sum_{i \in M^{\leq}} x^i$. Como P é convexo, $\bar{x} \in P$. Então, para todo $k \in M^{\leq}$, temos que $(a^k)^\top \bar{x} = \frac{1}{|M^{\leq}|}\sum_{i \in M^{\leq}} (a^k)^\top x^i < b_k$, em que a desigualdade segue de $(a^k)^\top x^i \leq b_k$, para todo $i \in M^{\leq}$, e $(a^k)^\top x^k < b_k$. Logo, $\bar{x} \in \operatorname{inn}(P)$. ■

A partir de qualquer ponto interno de $P(A,b)$ e qualquer direção em $\mathcal{N}(A^=)$, podemos obter outro ponto do poliedro, conforme estabelece o lema 7.3.1 abaixo, ilustrado pela figura 7.11.

Lema 7.3.1. Se $\bar{x} \in \operatorname{inn}(P(A,b))$ e $d \in \mathcal{N}(A^=)$, então existe $\varepsilon > 0$ tal que $\bar{x} + \varepsilon d \in P$. Em particular, se $Ad \not\leq 0$, existem $\varepsilon > 0$ e $i \in M^{\leq}$ tais que $\bar{x} + \varepsilon d \in P$ e $(a^i)^\top(\bar{x} + \varepsilon d) = b_i$.

Prova: Existe $\varepsilon \in \mathbb{R}$ tal que

$$0 < \varepsilon \leq \inf_{i \in M^{\leq}} \left\{ \frac{b_i - (a^i)^\top \bar{x}}{(a^i)^\top d} : (a^i)^\top d > 0 \right\}. \tag{7.3}$$

Temos que $A^=(\bar{x} + \varepsilon d) = A^= \bar{x} = b^=$. Considere agora $i \in M^{\leq}$. Se $(a^i)^\top d \leq 0$, então trivialmente $(a^i)^\top(\bar{x} + \varepsilon d) \leq (a^i)^\top \bar{x} \leq b_i$. Caso $(a^i)^\top d > 0$, pela definição de ε temos que

$$(a^i)^\top(\bar{x} + \varepsilon d) \leq (a^i)^\top \bar{x} + \frac{b_i - (a^i)^\top \bar{x}}{(a^i)^\top d}(a^i)^\top d = b_i.$$

Logo, $\bar{x} + \varepsilon d \in P$. Mais ainda, se $Ad \not\leq 0$, o ínfimo em (7.3) é atingido em algum $i \in M^{\leq}$. Tomando ε igual a esse ínfimo, chegamos a $(a^i)^\top(\bar{x} + \varepsilon d) = b_i$. ■

Proposição 7.3.3. Se $P(A,b) \neq \emptyset$, então $\operatorname{afim}(P) = \{x \in \mathbb{R}^n : A^= x = b^=\}$.

Prova: Seja $S = \{x \in \mathbb{R}^n : A^= x = b^=\}$. Como $P \subseteq S$ e S é afim, $\text{afim}(P) \subseteq S$. Tome agora $x \in S$ e $\bar{x} \in \text{inn}(P)$. Note que $x - \bar{x} \in \mathcal{N}(A^=)$. Pelo lema 7.3.1, existe $\varepsilon > 0$ tal que $x' = \bar{x} + \varepsilon(x - \bar{x}) \in P$. Então, $x = \frac{1}{\varepsilon}x' + \frac{\varepsilon - 1}{\varepsilon}\bar{x} \in \text{afim}(P)$. Logo, $S \subseteq \text{afim}(P)$. ∎

Usando a proposição 7.2.3, obtemos diretamente outra expressão para a dimensão de P, além de $\dim(P) = \text{posto-afim}(P) - 1$.

Corolário 7.3.1. Se $P(A,b) \neq \emptyset$, então $\dim(P) = n - \text{posto}(A^=)$.

Corolário 7.3.2. Sobre $P \subseteq \mathbb{R}^n$, as seguintes assertivas são equivalentes:

i) P tem dimensão plena;
ii) P contém $n+1$ pontos afim-independentes;
iii) P tem um ponto interior;
iv) $\text{afim}(P) = \mathbb{R}^n$;
v) $P \subseteq \{x \in \mathbb{R}^n : a^\top x = \alpha\} \Rightarrow a = 0, \alpha = 0$.

Exercício 7.3.2. [170, Pág. 87] Determinamos a dimensão do poliedro P definido pelas sete seguintes desigualdades:

$$\begin{array}{rrr} x_1 + x_2 + x_3 \leq 1 & x_1 + x_3 \leq 1 & -x_1 \leq 0 \\ -x_1 - x_2 - x_3 \leq -1 & x_3 \leq 2 & -x_2 \leq 0 \\ x_1 + 2x_2 + 2x_3 \leq 2 & & \end{array}$$

Note que $(1 \quad 0 \quad 0), (0 \quad 1 \quad 0), (0 \quad 0 \quad 1)$ são pontos AI em P, implicando $\dim(P) \geq 2$. Por outro lado, $x_1 + x_2 + x_3 = 1$ para todo $x \in P$, levando a $\text{posto}(A^=) \geq 1$, ou ainda $\dim(P) \leq 3 - 1 = 2$. Portanto, $\dim(P) = 2$.

Exercício 7.3.3. Dado um grafo $G = (V,E)$ com $V = \{1,2,\ldots,n\}$, defina

$$P = \{x \in \mathbb{R}^n : x_u + x_v \leq 1 \, \forall uv \in E, \; 0 \leq x_u \leq 1 \, \forall u \in V\}.$$

Note que $P \cap \mathbb{Z}^n$ define uma formulação para o problema do conjunto independente máximo em G. Para $u \in \{1,2,\ldots,n\}$, seja $e^u \in \mathbb{B}^n$ tal que $e^u_j = 1$, se $u = j$, e $e^u_j = 0$, caso contrário. O conjunto $\{e^u : u \in V\} \cup \{0\}$ é AI. Logo, $\dim(P) = n$. Qual a dimensão de $\text{conv}(P \cap \mathbb{Z}^n)$?

Exercício 7.3.4 (adaptado de [39]). Seja $D = (V,A)$ um digrafo acíclico com $|V| = n$ e $|A| = m$. Sejam $N^-(v) = \{u : uv \in A\}$ os vértices que originam arcos entrando em v e $S = \{v : N^-(v) = \emptyset\}$ o conjunto de fontes. Considere o poliedro

$$\begin{aligned} P = \{x \in \mathbb{R}^{n+m} : x_{vv} + \sum_{u \in N^-(v)} x_{uv} \geq 1 \quad \forall v \in V, \\ 0 \leq x_{uv} \leq 1 \quad \forall v \in V \quad \forall u \in N^-(v) \cup \{v\}\}. \end{aligned} \tag{7.4}$$

Observe que $P \cap \mathbb{Z}^{n+m}$ modela uma cobertura de V, em que um vértice $v \in V$ pode ser coberto por qualquer vértice em $N^-(v) \cup \{v\}$.

Para $v \in V$, $u \in N^-(v)$, defina $e^{uv} \in \mathbb{B}^{n+m}$ tal que $e^{uv}_{wz} = 1$, se $w = u$ e $z = v$, e $e^{uv}_{wz} = 0$, caso contrário. Considere o conjunto de $m + n - |S| + 1$ pontos dado por

$$X = \sum_{u \in V} e^{uu}, \qquad X^{uv} = X + e^{uv} \;\; \forall (u,v) \in A,$$

$$X^v = X - e^{vv} + e^{r(v)v} \;\; \forall v \in V \setminus S \text{ com } r(v) \in N^-(v).$$

Esses pontos pertencem a P (verifique!). Vamos mostrar que são AI. Observe que a matriz cujas linhas são os vetores $X^{uv} - X \;\; \forall (u,w) \in A$ e $X^v - X \;\; \forall v \in V \setminus S$ tem a forma $\begin{pmatrix} 0 & 0 & I_m \\ 0 & -I_{n-|S|} & \mathrm{J} \end{pmatrix}$, em que as primeiras $|S|$ colunas se referem às componentes $_{ww}$, para

$w \in S$, as $n-|S|$ colunas seguintes se relacionam às componentes $_{ww}$, para $w \in V \setminus S$, e as demais m colunas, às componentes $_{wv}$; I_k é uma matriz identidade de tamanho k, e J é uma matriz em que cada linha tem um único elemento não nulo (e igual a 1). Claramente, esses vetores-linha são LI, mostrando que os vetores que os originaram são AI. Logo, temos $\dim(P) \geq m+n-|S|$. Por outro lado, para todo $v \in S$, (7.4) torna-se $x_{uv} \geq 1$, levando a $x_{vv} = 1$. Portanto, $\text{posto}(A^{=}) \geq |S|$, o que implica $\dim(P) \leq n+m-|S|$. Logo, $\dim(P) = n+m-|S|$.

Exercício 7.3.5. Encontre a dimensão do poliedro obtido substituindo, em (7.4), as desigualdades por igualdades.

7.3.3 Desigualdades válidas e faces

Como vimos, um poliedro pode possuir infinitas representações. Aqui, investigaremos quais delas são minimais. Em outras palavras, queremos saber que desigualdades de uma representação (A,b) são necessárias e suficientes para descrever $P = P(A,b)$ e se tais desigualdades são as mesmas em uma outra representação (A',b') de P. Nesse contexto, são fundamentais os conceitos de desigualdade válida e face, que apresentamos a seguir, e especialmente de desigualdades indutoras de facetas, que discutimos na próxima subseção.

Definição 7.3.4. A desigualdade $\pi^\top x \leq \pi_0$, que denotaremos simplesmente por (π,π_0), é dita uma desigualdade válida para um conjunto C se $\pi^\top x \leq \pi_0$, para todo $x \in C$.

A figura 7.12 apresenta exemplo de uma desigualdade válida para um poliedro P.

Exercício 7.3.6. Sejam (π,π_0) uma desigualdade válida para $S \subseteq \mathbb{R}^n$ e $H = \{x \in \mathbb{R}^n : \pi^\top x = \pi_0\}$. Mostre que (π,π_0) é válida para $\text{conv}(S)$ e que $\text{conv}(S) \cap H = \text{conv}(S \cap H)$.

Definição 7.3.5. Um subconjunto $F \subseteq P$ é dito uma face de P se existe desigualdade válida (π,π_0) tal que $F = \{x \in P : \pi^\top x = \pi_0\}$. A face F é própria se $F \neq \emptyset$ e $F \neq P$. Dizemos que (π,π_0) representa (ou define ou induz) F. Se $F \neq \emptyset$, então dizemos que (π,π_0) suporta P.

Observe que a desigualdade válida apresentada na figura 7.12 induz uma face vazia de P. Faces de dimensão 0 e 1, bem como desigualdades válidas que as definem, são apresentadas nas figuras 7.13(a) e 7.13(b), respectivamente. Em particular, note que F_0 pode ser representada por diferentes desigualdades válidas.

Observe que a desigualdade (π,π_0) ser válida para um poliedro não vazio $P \subseteq \mathbb{R}^n$ equivale a dizer que $P \subseteq \{x \in \mathbb{R}^n : \pi^\top x \leq \pi_0\}$ ou ainda que $\max\{\pi^\top x : x \in P\} \leq \pi_0$. Adicionalmente, F é uma face não vazia de $P \neq \emptyset$ se, e somente se, existe desigualdade válida (π,π_0) tal que $\max\{\pi^\top x : x \in P\} = \pi_0$ e $F = \arg\max\{\pi^\top x : x \in P\}$. Note também que toda face de um poliedro é também poliedral.

Exercício 7.3.7 (Cont. exercício 7.3.2). A desigualdade $(\pi,\pi_0) = (-1,-1,1,1)$ é válida para P, pois $\max\{-x_1 - x_2 + x_3 : x \in P\} = 1 = \pi_0$. Portanto, (π,π_0) descreve uma face F de P. Note que essa desigualdade não pertence à descrição original de P. Encontre outra desigualdade válida que descreva a mesma face F.

Podemos caracterizar desigualdade válida e face de um poliedro como segue.

Proposição 7.3.4. (π,π_0) é uma desigualdade válida para $P(A,b) \neq \emptyset$ se, e somente se, $\pi^\top = u^\top A$ e $\pi_0 \geq u^\top b$ para algum $u \geq 0$. Mais ainda, F é uma face não vazia de P se, e somente se, $F = \{x \in P : u^\top A x = u^\top b\} \neq \emptyset$, para algum $u \geq 0$.

Prova: (π,π_0) é uma desigualdade válida se, e somente se, $\max\{\pi^\top x : x \in P\} \leq \pi_0$. Por dualidade linear, isso é equivalente a $\min\{b^\top u : A^\top u = \pi, u \geq 0\} \leq \pi_0$. Note ainda que o mínimo é igual a π_0 se (π,π_0) define face não vazia. Assim, tomando u como a solução ótima do dual, o resultado segue. ∎

Exercício 7.3.8 (Cont. exercício 7.3.3). Mostre que, se $H \subseteq V$ induz um ciclo em G, então $\sum_{v\in H} x_v \leq |H|/2$ é válida para P. E que podemos substituir $|H|/2$ por $\lfloor |H|/2 \rfloor$ quando consideramos o conjunto $P \cap \mathbb{Z}^n$ em vez de P.

Acima, definimos face de $P(A,b)$ a partir do hiperplano associado a uma (única) desigualdade válida, que pode não pertencer à representação (A,b). Entretanto, podemos defini-la usando a interseção de (vários) hiperplanos relativos a desigualdades de (A,b).

Proposição 7.3.5. F é face não vazia de $P(A,b)$ se, e somente se, $F = \{x \in \mathbb{R}^n : (a^i)^\top x = b_i \, \forall i \in I,\ (a^i)^\top x \leq b_i \, \forall i \in M \setminus I\} \neq \emptyset$, para algum $I \subseteq M$, $I \supseteq M^=$.

Prova: Inicialmente, derivamos um resultado auxiliar. Dado $u \in \mathbb{R}^m$, $u \geq 0$, defina $I_u = M^= \cup \{i \in M : u_i > 0\}$ e $F_u = \{x \in \mathbb{R}^n : (a^i)^\top x = b_i \, \forall i \in I_u, (a^i)^\top x \leq b_i \, \forall i \in M \setminus I_u\}$. Vamos mostrar que $F_u = \{x \in P : u^\top Ax = u^\top b\}$. Trivialmente, temos $F_u \subseteq \{x \in P : u^\top Ax = u^\top b\}$. Reciprocamente, seja $x \in P$ tal que $u^\top(Ax - b) = 0$. Como $Ax \leq b$ e $A^= x = b^=$, devemos ter $(a^i)^\top x = b_i$ para todo $i \in I_u$. Portanto, $x \in F_u$.

Agora suponha que F seja face não vazia. Pela proposição 7.3.4, existe $u \geq 0$ tal que $F = \{x \in P : u^\top Ax = u^\top b\} \neq \emptyset$, ou seja, $F = F_u$. Isso mostra a implicação direta para $I = I_u$. Para a inversa, suponha $F = \{x \in \mathbb{R}^n : (a^i)^\top x = b_i \, \forall i \in I, (a^i)^\top x \leq b_i \, \forall i \in M \setminus I\} \neq \emptyset$, com $I \supseteq M^=$. Então $F = F_u$ para $u \in \mathbb{B}^m$ tal que $u_i = 1$ se, e somente se, $i \in I$. Logo, $F = \{x \in P : u^\top Ax = u^\top b\} \neq \emptyset$. Da proposição 7.3.4, segue que F é face não vazia. ∎

Corolário 7.3.3. Toda face de uma face de P é uma face de P.

Exercício 7.3.9 (Cont. exercício 7.3.8). Apresente uma descrição da face induzida por $\sum_{v\in H} x_v \leq |H|/2$ usando apenas desigualdades da representação de P (algumas transformadas em igualdades).

A partir da proposição 7.3.5, vemos que uma face F de um poliedro $P(A,b)$ tem como representação a matriz (A,b) estendida por $(-(a^i)^\top, -b_i)$, para toda desigualdade $(a^i)^\top x \leq b_i$ que se torna igualdade em F. Por simplicidade, vamos usar $(A_F^=, b_F^=)$ e $(A_F^\leq, b_F^\leq)$ para denotar as igualdades implícitas e desigualdades legítimas de F, respectivamente.

A proposição 7.3.5 mostra também que o número de faces de um poliedro é finito. Além disso, ela pode ser usada para estabelecer as dimensões dessas faces.

Proposição 7.3.6. Se F é face própria de $P \neq \emptyset$, então $\dim(F) \leq \dim(P) - 1$.

Prova: Sendo F própria, podemos concluir da proposição 7.3.5 que $M_F^= \supsetneq M^=$, isto é, que existe $i \in M_F^= \cap M^\leq$. Dado que $((a^i)^\top, b^i)$ não é combinação linear das linhas de $(A^=, b^=)$ e $P \neq \emptyset$, temos $\text{posto}(A_F^=) > \text{posto}(A^=)$. Pelo corolário 7.3.1, $\dim(F) = n - \text{posto}(A_F^=) < n - \text{posto}(A^=) = \dim(P)$. ∎

Proposição 7.3.7. Suponha $P(A,b) \subseteq \mathbb{R}^n$, não vazio. Se $\text{posto}(A) = n-k$, P tem face de dimensão $k, k+1, \ldots, \dim(P)$ e nenhuma face não vazia de dimensão menor que k.

Prova: Seja F uma face não vazia de dimensão mínima em P. Como $\text{posto}(A_F^=) \leq \text{posto}(A) = n-k$, então $\dim(F) = n - \text{posto}(A_F^=) \geq k$. Para mostrar a igualdade, considere $\bar{x} \in \text{inn}(F)$. Vamos mostrar que $F \subseteq \{y \in P : Ay = A\bar{x}\}$. Tome $y \in F$ e $d = \bar{x} - y$. Temos que $d, -d \in \mathcal{N}(A_F^=)$. Pelo lema 7.3.1, derivamos que $Ad \leq 0$. Caso contrário, existiria

$i \in M_F^{\leq}$ tal que $F_i := \{x \in P : A_F^{=} x = b_F^{=}, (a^i)^\top x = b_i\} \neq \emptyset$: um absurdo, pois F_i é face de P e $\dim(F_i) < \dim(F)$. Similarmente, concluímos que $A(-d) \leq 0$. Portanto, $Ad = 0$, ou seja, $Ay = A\bar{x}$, mostrando que $F \subseteq \{y \in P : Ay = A\bar{x}\}$. Isso implica que $\dim(F) \leq n - \text{posto}(A) = k$ e, por conseguinte, $\dim(F) = k$.

Agora, seja $F \neq \emptyset$ uma face tal que $\dim(F) < \dim(P)$. Devemos mostrar que existe face de dimensão $\dim(F)+1$. Temos que $M_F^{=} \supsetneq M^{=}$ (ver proposição 7.3.5) e que $\text{posto}(A_F^{=}) > \text{posto}(A^{=})$. Assim, podemos reduzir o posto de $A_F^{=}$ de uma unidade, removendo um subconjunto de linhas indexadas por $I \subseteq M_F^{=} \setminus M^{=}$, $I \neq \emptyset$. Defina $F' = \{x \in \mathbb{R}^n : (a^i)^\top x = b_i \ \forall i \in M_F^{=} \setminus I, (a^i)^\top x \leq b_i \ \forall i \in M_F^{\leq} \cup I\}$. Note que $F' \neq \emptyset$ (pois $F' \supset F$), $M_F^{=} \setminus I \supseteq M^{=}$ e, pela proposição 7.3.5, F' define face de P. Mais ainda $\dim(F') = \dim(F)+1$, pois $\text{posto}(A_{F'}^{=}) = \text{posto}(A_F^{=}) - 1$. ■

Usando o exercício 7.3.1, caracterizamos os poliedros com faces de dimensão zero.

Corolário 7.3.4. Suponha $P \neq \emptyset$. P tem face de dimensão zero se, e somente se, $\text{lin}(P) = \{0\}$. Em particular, $P \subseteq \mathbb{R}^n_+$ tem face de dimensão zero.

Exercício 7.3.10. Sejam $F \subseteq \mathbb{R}^n$ uma face minimal não vazia de $P(A,b)$ e $\bar{x} \in F$. Mostre que $F = \bar{x} + \text{lin}(P)$ e $\dim(F) = n - \text{posto}(A)$.

7.3.4 Facetas

As faces próprias maximais de P são chamadas facetas ou, precisamente:

Definição 7.3.6. Uma faceta de P é uma face não vazia F tal que $\dim(F) = \dim(P) - 1$.

Como qualquer face, toda faceta é induzida por uma desigualdade válida. Porém, pela proposição 7.3.5, essa desigualdade válida pode sempre ser escolhida entre aquelas da representação original.

Corolário 7.3.5. Seja F face não vazia de $P(A,b)$. F é faceta se, e somente se, existe $k \in M^{\leq}$ tal que $F = \{x \in P : (a^k)^\top x = b_k\}$.

Exercício 7.3.11 (cont. exercício 7.3.2). A desigualdade $(\pi, \pi_0) = (2, -7, 2, 2)$ é válida para P. De fato, $(\pi, \pi_0) = 7(-1,-1,-1,-1) + 9(1,0,1,1)$, ou seja, é uma combinação cônica de duas desigualdades que definem P. Portanto, (π, π_0) descreve uma face F de P. Mais ainda, os pontos $(1,0,0)$ e $(0,0,1)$ pertencem a F, mostrando que $\dim(F) \geq 1 = \dim(P) - 1$. Por outro lado, $(0,1,0) \in P \setminus F$. Logo, $F \subset P$ e, por conseguinte, $\dim(F) \leq \dim(P) - 1$. Temos então que F é uma faceta. Pelo corolário 7.3.5, existe pelo menos uma desigualdade da descrição original de P que representa F. Encontre duas dessas desigualdades.

Por outro lado, duas desigualdades quaisquer que induzem uma mesma faceta de P diferem entre si (a menos de multiplicação por escalar) de uma combinação linear das igualdades de P.

Teorema 7.3.1. Suponha $\dim(P) > 0$. Seja $F = \{x \in P : \pi^\top x = \pi_0\}$ face de $P = P(A,b)$. São equivalentes as assertivas:

i) F é faceta;

ii) $F \neq P$, e $F \subseteq \{x \in P : \lambda^\top x = \lambda_0\} \Rightarrow (\lambda^\top, \lambda_0) = \alpha(\pi^\top, \pi_0) + u^\top(A^{=}, b^{=})$ para $\alpha \in \mathbb{R}$, $u \in \mathbb{R}^{|M^{=}|}$;

iii) $F \neq \emptyset$, $F \neq P$, e $F \subseteq \{x \in P : \lambda^\top x = \lambda_0\} \Rightarrow \lambda^\top = \alpha\pi^\top + u^\top A^{=}$ para $\alpha \in \mathbb{R}$, $u \in \mathbb{R}^{|M^{=}|}$.

Prova: $(i) \Rightarrow (ii)$ Admita que F é faceta, ou seja, $F \neq \emptyset$, $F \neq P$ e $\dim(F) = \dim(P) - 1 = n - (\text{posto}(A^{=}) + 1)$. Então, $\text{posto}(\begin{bmatrix} A^{=} \\ \pi^\top \end{bmatrix}) = \text{posto}(A^{=}) + 1$. Suponha que $F \subseteq \{x \in P :$

$\lambda^\top x = \lambda_0\}$. Então, $F = \{x \in P : \pi^\top x = \pi_0, \lambda^\top x = \lambda_0\}$. Logo, posto$(\begin{bmatrix} A^= \\ \pi^\top \end{bmatrix}) =$ posto$(\begin{bmatrix} A^= \\ \pi^\top \\ \lambda^\top \end{bmatrix})$.
Sendo $F \neq \emptyset$, $(\lambda^\top, \lambda_0) = \alpha(\pi^\top, \pi_0) + u^\top(A^=, b^=)$ para $\alpha \in \mathbb{R}$, $u \in \mathbb{R}^{|M^=|}$.
$(ii) \Rightarrow (i)$ Admita que (ii) é verdadeira. Então, para $i \in M_F^=$, existem $\alpha_i \in \mathbb{R}$ e $u^i \in \mathbb{R}^{|M^=|}$ tais que $((a^i)^\top, b_i) = \alpha_i(\pi^\top, \pi_0) + (u^i)^\top(A^=, b^=)$, pois $F \subseteq \{x \in P : (a^i)^\top x = b_i\}$. Como $\dim(F) < \dim(P)$, ou melhor, posto$(A_F^=) >$ posto$(A^=)$, deve existir $k \in M_F^=$ tal que $\alpha_k \neq 0$. Assim, obtemos $(\pi^\top, \pi_0) = \frac{1}{\alpha_k}((a^k)^\top, b_k) - \frac{(u^k)^\top}{\alpha_k}(A^=, b^=)$, de modo que $((a^i)^\top, b_i) = \frac{\alpha_i}{\alpha_k}((a^k)^\top, b_k) + (u^i - \frac{\alpha_i}{\alpha_k}u^k)^\top(A^=, b^=)$ $\forall i \in M_F^=$. Portanto, $((a^i)^\top, b_i)$ é combinação linear das linhas de $\begin{bmatrix} A^= & b^= \\ (a^k)^\top & b_k \end{bmatrix}$ $\forall i \in M_F^=$, ou seja, posto$(A_F^=) \leq$ posto$(A^=) + 1$. Por conseguinte, $\dim(F) = n -$ posto$(A_F^=) \geq n -$ posto$(A^=) - 1 = \dim(P) - 1 \geq 0$. Como $F \neq P$, concluímos que F é faceta.
$(ii) \Leftrightarrow (iii)$ Trivialmente temos $(ii) \Rightarrow (iii)$, dado que (ii) (ou equivalentemente (i)) implica $F \neq \emptyset$. Suponha agora (iii). Basta mostrar que $\lambda_0 = \alpha\pi_0 + u^\top b^=$. Como $F \neq \emptyset$, então existe $x \in F$. Logo, $\lambda_0 = \lambda^\top x = (\alpha\pi^\top + u^\top A^=)x = \alpha\pi_0 + u^\top b^=$. ■

Corolário 7.3.6. Sejam F faceta de $P(A,b)$ e (π, π_0) uma desigualdade válida para P que representa F. Então (λ, λ_0) é desigualdade válida que representa F se, e somente se, $(\lambda^\top, \lambda_0) = \alpha(\pi^\top, \pi_0) + u^\top(A^=, b^=)$ para $\alpha > 0$ e $u \in \mathbb{R}^{|M^=|}$.

Prova: Suponha que (λ, λ_0) seja desigualdade válida que representa F, ou seja, $F = \{x \in P : \pi^\top x = \pi_0\} = \{x \in P : \lambda^\top x = \lambda_0\}$. Pelo teorema 7.3.1, $(\lambda^\top, \lambda_0) = \alpha(\pi^\top, \pi_0) + u^\top(A^=, b^=)$ para $\alpha \in \mathbb{R}$, $u \in \mathbb{R}^{|M^=|}$. Note que $\alpha \neq 0$; caso contrário, (λ, λ_0) seria uma igualdade de P. Além disso, $\alpha > 0$, pois $\pi^\top x \leq \pi_0$, $\lambda^\top x \leq \lambda_0$ e $A^= x = b^=$, para todo $x \in P$.

Suponha, agora, que $(\lambda^\top, \lambda_0) = \alpha(\pi^\top, \pi_0) + u^\top(A^=, b^=)$ para $\alpha > 0$ e $u \in \mathbb{R}^{|M^=|}$. Note que $\lambda^\top x - \lambda_0 = \alpha(\pi^\top x - \pi_0)$, para todo $x \in P$. Como $\alpha > 0$, (λ, λ_0) é válida para P. Mais ainda, $\{x \in P : \lambda^\top x = \lambda_0\} = \{x \in P : \pi^\top x = \pi_0\} = F$. Logo, (λ, λ_0) representa F. ■

No corolário acima é preciso que $\alpha > 0$ (diferentemente do teorema 7.3.1), pois se deseja que (λ, λ_0) seja desigualdade válida.

7.3.5 Descrição de poliedros via desigualdades

O corolário 7.3.6 sugere a definição de uma relação de equivalência entre as desigualdades válidas de P.

Definição 7.3.7. Duas desigualdades válidas (λ, λ_0) e (π, π_0) são equivalentes com respeito a $P(A,b)$ se

$$(\lambda^\top, \lambda_0) = \alpha(\pi^\top, \pi_0) + u^\top(A^=, b^=), \text{ para algum } \alpha > 0 \text{ e } u \in \mathbb{R}^{|M^=|}.$$

Note que, se (λ, λ_0) e (π, π_0) são desigualdades equivalentes, no máximo uma delas é necessária em uma representação de P. De fato, para todo $x \in P$, $\pi^\top x \leq \pi_0$ se, e somente se, $\lambda^\top x \leq \lambda_0$. Caso (π, π_0) induza faceta, mostramos que exatamente uma (qualquer) das desigualdades equivalentes a ela será necessária.

Lema 7.3.2. Seja F faceta de P. Exatamente uma (qualquer) das desigualdades que representam F é necessária para descrever P.

Prova: Seja $P_F \neq \emptyset$ o poliedro obtido de $P(A,b)$ removendo-se todas as desigualdades que representam F. Vamos mostrar que $P_F \neq P$, ou melhor, $P_F \backslash P \neq \emptyset$. Seja (π, π_0) uma desigualdade válida para P que representa F. Temos que $\pi^\top$ não é combinação

linear das linhas de $A^=$, ou seja, o sistema $(A^=)^\top u = \pi$ não tem solução. Pelo Lema de Farkas, existe $y \in \mathbb{R}^n$ tal que $A^= y = 0$ e $\pi^\top y > 0$. Note que $y \in \mathcal{N}(A^=_{P_F})$ e $F \subseteq P_F$. Seja $\bar{x} \in \operatorname{inn}(F) \subseteq \operatorname{inn}(P_F)$. Pelo lema 7.3.1, existe $\varepsilon > 0$ tal que $\bar{x} + \varepsilon y \in P_F$. Mais ainda, como $\pi^\top y > 0$ e $\pi^\top \bar{x} = \pi_0$, temos $\pi^\top(\bar{x} + \varepsilon y) = \pi_0 + \varepsilon \pi^\top y > \pi_0$. Logo, $\bar{x} + \varepsilon y \in P_F \setminus P$. Portanto, uma (e exatamente uma qualquer) das desigualdades que representam F é necessária para descrever P. ∎

Por outro lado, as desigualdades legítimas que não induzem faceta podem ser removidas de toda representação de P.

Lema 7.3.3. Seja F face de $P \neq \emptyset$ com $\dim(F) < \dim(P) - 1$. Nenhuma desigualdade que representa F é necessária em qualquer representação de P.

Prova: Considere uma desigualdade (π, π_0) que representa F e admita que ela pertence à representação (A, b). Então, $(\pi, \pi_0) = (a^r, b_r)$, para algum $r \in M^\leq$. Defina $P' = \{x \in \mathbb{R}^n : A^= x = b^=, (a^i)^\top x \leq b_i, i \in M^\leq \setminus \{r\}\}$ e suponha, por absurdo, que $P' \neq P(A, b)$. Então existe $\hat{x} \in \mathbb{R}^n$ tal que $A^= \hat{x} = b^=$, $(a^i)^\top \hat{x} \leq b_i$ $\forall i \in M^\leq \setminus \{r\}$ e $(a^r)^\top \hat{x} > b_r$. Vamos mostrar que existe $z \in F$ tal que

$$A^= z = b^=, (a^i)^\top z < b_i, \forall i \in M^\leq \setminus \{r\}, (a^r)^\top z = b_r.$$

Isso resultaria em $M^\leq \setminus \{r\} \subseteq M^\leq_F \subsetneq M^\leq$, ou melhor, $M^=_F = M^= \cup \{r\}$, implicando $\dim(F) = n - \operatorname{posto}(A^=_F) = n - \operatorname{posto}(A^=) - 1 = \dim(P) - 1$, o que contrariaria a hipótese.

Para determinar z, tome $\bar{x} \in \operatorname{inn}(P)$. Temos que $A^= \hat{x} = A^= \bar{x} = b^=$, $(a^r)^\top \hat{x} > b_r > (a^r)^\top \bar{x}$ e $(a^i)^\top \hat{x} \leq b_i$, $(a^i)^\top \bar{x} < b_i$, para todo $i \in M^\leq \setminus \{r\}$. Defina $\varepsilon = \frac{b_r - (a^r)^\top \bar{x}}{(a^r)^\top \hat{x} - (a^r)^\top \bar{x}}$ e note que $\varepsilon \in (0, 1)$. Seja $z = (1 - \varepsilon)\bar{x} + \varepsilon \hat{x}$. Então $A^= z = b^=$ e $(a^r)^\top z = (a^r)^\top \bar{x} + \varepsilon (a^r)^\top (\hat{x} - \bar{x}) = b_r$. Mais ainda, para todo $i \in M^\leq \setminus \{r\}$, $(a^i)^\top z = (1 - \varepsilon)(a^i)^\top \bar{x} + \varepsilon (a^i)^\top \hat{x} < b_i$. ∎

A proposição 7.3.3, o corolário 7.3.1 e os lemas 7.3.2 e 7.3.3 levam-nos a concluir que:

Teorema 7.3.2. Seja $P \subseteq \mathbb{R}^n$, $P \neq \emptyset$. Uma descrição minimal de P é $\{x \in \mathbb{R}^n : (a^i)^\top x = b_i, i \in \{1, \ldots, k\}, (a^i)^\top x \leq b_i, i \in \{k+1, \ldots, k+t\}\}$ se, e somente se:

i) $k = n - \dim(P)$, t é o número de facetas de P,
ii) $\operatorname{afim}(P) = \{x \in \mathbb{R}^n : (a^i)^\top x = b_i, i \in \{1, \ldots, k\}\}$ e
iii) cada faceta de P é $\{x \in P : (a^i)^\top x = b_i\}$, para todo $i \in \{k+1, \ldots, k+t\}$.

Se P tem dimensão plena, P tem descrição minimal única (a menos de múltiplos), em que cada desigualdade induz uma faceta diferente.

Exercício 7.3.12 (cont. exercício 7.3.2). Determine um sistema minimal que descreva P.

7.3.6 Vértices e direções extremas

A descrição de poliedros através de desigualdades válidas é comumente denominada representação externa. Outra forma de descrição, que se chama representação interna, dá-se através de pontos e direções extremas.

Definição 7.3.8. Seja $C \subseteq \mathbb{R}^n$ um conjunto convexo. Um ponto $x \in C$ é um ponto extremo ou vértice de C se $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$, com $x^1, x^2 \in C$, implica que $x^1 = x^2$. Um vetor $d \in \operatorname{rec}(C) \setminus \{0\}$ é uma direção (de recessão) extrema de C se $d = \frac{1}{2}d^1 + \frac{1}{2}d^2$, com $d^1, d^2 \in \operatorname{rec}(C) \setminus \{0\}$, implica que $d^1 = \lambda d^2$ para algum $\lambda \neq 0$.

A figura 7.14 destaca em negrito os vértices e direções extremas de um poliedro P.

Exercício 7.3.13. Se d é uma direção extrema e $d = \frac{1}{2}d^1 + \frac{1}{2}d^2$, com $d^1, d^2 \in \mathrm{rec}(C)$, então $d^1 = \lambda_1 d$ e $d^2 = \lambda_2 d$, para $\lambda_1 \neq 0$ ou $\lambda_2 \neq 0$.

Note que, se d é uma direção de recessão, λd também o será, para todo $\lambda > 0$. Sendo assim, para identificarmos as direções de recessão (em particular extremas) basta considerarmos direções normalizadas.

Vamos caracterizar vértices e direções extremas como faces de P e $\mathrm{rec}(P)$, respectivamente. Consideramos a seguinte notação adicional: dados um poliedro $P(A,b)$ e um ponto $x \in P$, denotamos $(A_x^=, b_x^=)$ a submatriz de (A,b) formada pelas linhas $((a^i)^\top, b_i)$ tais que $(a^i)^\top x = b_i$. Note que $(A_x^=, b_x^=) = (A_P^=, b_P^=)$ se $x \in \mathrm{inn}(P)$.

Lema 7.3.4. Seja $x \in P(A,b)$. Se F é face de menor dimensão de P contendo x, então $x \in \mathrm{inn}(F)$ e, portanto, $(A_x^=, b_x^=) = (A_F^=, b_F^=)$.

Prova: Suponha, pela contrapositiva, que $x \in F \setminus \mathrm{inn}(F)$. Logo, existe $i \in M_F^{\leq}$ tal que $(a^i)^\top x = b_i$. Em outras palavras, x pertence à face $F' := \{y \in P : A_F^= y = b_F^=, (a^i)^\top y = b_i\}$. Portanto, F não é uma face de menor dimensão contendo x. ∎

Proposição 7.3.8. As seguintes equivalências são verdadeiras: (i) x é ponto extremo de P se, e somente se, $\{x\}$ é face de dimensão zero de P; (ii) d é direção extrema de P se, e somente se, $\mathrm{rec}(P) \cap \{\lambda d : \lambda \in \mathbb{R}\}$ é face unidimensional de $\mathrm{rec}(P)$.

Prova: Para mostrar (i), primeiro suponha que x seja ponto extremo de $P(A,b)$ e seja F a menor face de P contendo x. É suficiente mostrar que $F = \{x\}$, ou melhor, $F \subseteq \{x\}$. Tome $y \in F$. Temos que $x \in \mathrm{inn}(F)$ (pelo lema 7.3.4) e $x - y, y - x \in \mathcal{N}(A_F^=)$. Usando o Lema 7.3.1, deduzimos que existe $\varepsilon > 0$ tal que $x^1 = x + \varepsilon(y - x) \in F \subseteq P$ e $x^2 = x - \varepsilon(y - x) \in F \subseteq P$. Como $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$, segue que $x = x^1 = x^2$, implicando $y = x$. Admita agora que $\{x\}$ é face de dimensão zero de $P(A,b) \subseteq \mathbb{R}^n$. Então, $\mathrm{posto}(A_x^=) = n$, de modo que existe $(\tilde{A}, \tilde{b})$, uma submatriz de $(A_x^=, b_x^=)$, tal que $\tilde{A}_{n\times n}$ é invertível e $\tilde{A}x = \tilde{b}$. Tome $x^1, x^2 \in P$ tais que $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$. Então, $\tilde{b} = \tilde{A}x = \frac{1}{2}(\tilde{A}x^1 + \tilde{A}x^2)$. Como $\tilde{A}x^1 \leq \tilde{b}$ e $\tilde{A}x^2 \leq \tilde{b}$, concluímos que $\tilde{A}x^1 = \tilde{A}x^2 = \tilde{b}$, ou seja, $x = x^1 = x^2$. Portanto, x é ponto extremo.

Para mostrar (ii), lembre que $\mathrm{rec}(P) = \{d \in \mathbb{R}^n : Ad \leq 0\}$. Então, para $d \in \mathrm{rec}(P)$, $A_d^=$ denota a submatriz de A formada pelas linhas $(a^i)^\top$ tais que $(a^i)^\top d = 0$.

Suponha que d seja direção extrema de P. Seja F a menor face de $\mathrm{rec}(P)$ contendo d e $F' = \mathrm{rec}(P) \cap \{\lambda d : \lambda \in \mathbb{R}\}$. Como $\dim(F') = 1$, é suficiente mostrar que $F' = F$. Pelo lema 7.3.4, concluímos que $d \in \mathrm{inn}(F)$ e $F = \{y \in \mathrm{rec}(P) : A_d^= y = 0\}$. Claramente, $F' \subseteq F$. Tome agora $y \in F \subseteq \mathrm{rec}(P)$. Como $d - y, y - d \in \mathcal{N}(A_F^=)$, pelo Lema 7.3.1, deduzimos que existe $\varepsilon > 0$ tal que $d^1 = d + \varepsilon(y - d) \in F$ e $d^2 = d - \varepsilon(y - d) \in F$. Dado que $d = \frac{1}{2}d^1 + \frac{1}{2}d^2$, devemos ter $d^1 = \lambda_1 d$ e $d^2 = \lambda_2 d$ (exercício 7.3.13). Portanto, $y = \frac{\lambda_1 + \varepsilon - 1}{\varepsilon} d$, levando a $y \in F'$.

Finalmente, suponha que $F' = \mathrm{rec}(P) \cap \{\lambda d : \lambda \in \mathbb{R}\}$ seja face unidimensional de $\mathrm{rec}(P)$. Então $d \neq 0$, $A_{F'}^= = A_d^=$ e, por conseguinte, $\mathrm{posto}(A_d^=) = n - 1$. Pelo corolário 7.3.1, $\dim(S) = 1$, onde $S = \{y \in \mathbb{R}^n : A_d^= y = 0\}$. Logo, S é o subespaço gerado pelo vetor d, isto é, $S = \{\lambda d : \lambda \in \mathbb{R}\}$. Sejam $d^1, d^2 \in \mathrm{rec}(P) \setminus \{0\}$ tais que $d = \frac{1}{2}d^1 + \frac{1}{2}d^2$. Como $Ad^1 \leq 0$, $Ad^2 \leq 0$ e $0 = A_d^= d = \frac{1}{2}A_d^= d^1 + \frac{1}{2}A_d^= d^2$, devemos ter $A_d^= d^1 = A_d^= d^2 = 0$, ou seja, $d^1, d^2 \in S$. Logo, $d^1 = \lambda_1 d$ e $d^2 = \lambda_2 d$, com $\lambda_1, \lambda_2 \neq 0$. Assim, $d^1 = \lambda d^2$, para $\lambda = \lambda_1/\lambda_2 \neq 0$, mostrando que d é direção extrema. ∎

Corolário 7.3.7. Todo poliedro tem um número finito de vértices e direções extremas (normalizadas).

Exercício 7.3.14. Mostre que, se R é o conjunto das direções extremas normalizadas de P, então $\mathrm{cone}(R) = \mathrm{rec}(P)$.

Note que a face unidimensional de $\mathrm{rec}(P)$ definida por uma direção extrema d, isto é, $\mathrm{rec}(P) \cap \{\lambda d : \lambda \in \mathbb{R}\}$, é igual a $\{\lambda d : \lambda \in \mathbb{R}\}$, se $d \in \mathrm{lin}(P)$, ou igual a $\{\lambda d : \lambda \geq 0\}$, se $d \in \mathrm{rec}(P) \setminus \mathrm{lin}(P)$. Observe também que podemos caracterizar vértice e direção extrema alternativamente como:

Corolário 7.3.8. Para $P(A,b)$, temos que: (i) x é ponto extremo de P se, e somente se, $x \in P$ e $\mathrm{posto}(A_x^=) = n$; (ii) d é direção extrema de P se, e somente se, $d \in \mathrm{rec}(P)$ e $\mathrm{posto}(A_d^=) = n-1$ (nesse caso, as igualdades referem-se à descrição $(A,0)$ de $\mathrm{rec}(P)$).

Proposição 7.3.9. Se P é racional, então seus vértices são racionais e suas direções extremas são múltiplos de vetores inteiros.

Prova: Sendo $P(A,b)$ racional, podemos considerar que (A,b) é racional. Se $\bar{x}$ é vértice, pelo corolário 7.3.8, concluímos que $\bar{x}$ é solução do sistema $A_{\bar{x}}^= x = b_{\bar{x}}^=$, onde $\mathrm{posto}(A_{\bar{x}}^=) = n$. Pela Regra de Cramer, deduzimos que $\bar{x} \in \mathbb{Q}^n$. Considere agora uma direção extrema $\bar{d} \neq 0$ de P. Sem perda de generalidade, podemos supor que $\bar{d}_1 = \pm 1$. Também pelo corolário 7.3.8, temos que $\bar{d}$ é solução de $A_{\bar{d}}^= d = 0$ e $d_1 = \bar{d}_1$, com $\mathrm{posto}(A_{\bar{d}}^=) = n-1$. Concluímos, pois, que $\bar{d}$ é solução de um sistema linear com coeficientes racionais $A'd = b'$, onde $\mathrm{posto}(A') = n$. Novamente, obtemos que $\bar{d} \in \mathbb{Q}^n$ e, por conseguinte, $\bar{d}$ tem um representante $\lambda \bar{d} \in \mathbb{Z}^n$, para $\lambda > 0$. ∎

7.3.7 Descrição de poliedros por vértices e direções extremas

Neste ponto, podemos derivar a representação interna de um poliedro. Por simplicidade, vamos considerar apenas poliedros com vértices. Admitir que um poliedro $P(A,b) \neq \emptyset$ tem vértice equivale a supor $\mathrm{lin}(P) = \{0\}$ ou, ainda, $\mathrm{posto}(A) = n$ (ver exercício 7.3.1 e corolário 7.3.4). Tal hipótese é satisfeita, por exemplo, quando $P \subseteq \mathbb{R}_+^n$, condição que pode ser sempre obtida no caso de programação linear e programação inteira (com uma mudança de variáveis, caso necessário).

O clássico teorema a seguir estabelece que todo poliedro é a soma do fecho convexo de seus vértices com o fecho cônico de suas direções extremas (ver figura 7.15).

Teorema 7.3.3 (Teorema de Minkowski). Suponha $P \neq \emptyset$ e $\mathrm{lin}(P) = \{0\}$. Existem conjuntos finitos Q e R, $Q \neq \emptyset$, tais que $P = \mathrm{conv}(Q) + \mathrm{cone}(R)$. Se P é racional, então $Q \subset \mathbb{Q}^n$ e $R \subset \mathbb{Z}^n$. Em particular, podemos tomar Q e R como os conjuntos de vértices e direções extremas de P, respectivamente.

Prova: Sejam $Q = \{x^k : k \in K\}$ e $R = \{d^j : j \in J\}$ os conjuntos de vértices e direções extremas de P. Pelo corolário 7.3.4 e proposição 7.3.8, $Q \neq \emptyset$. Defina $P' = \mathrm{conv}(Q) + \mathrm{cone}(R)$. Tome $x \in P'$. Então $x = y + d$, onde $y \in P$ (pois P é convexo) e $d \in \mathrm{cone}(R) = \mathrm{rec}(P)$. Logo, $x \in P$, mostrando que $P' \subseteq P$. Suponha, por absurdo, que exista $\bar{y} \in P \setminus P'$. Então não existe $(\lambda, \mu) \in \mathbb{R}_+^{|K|} \times \mathbb{R}_+^{|J|}$ tal que

$$\sum_{k \in K} x^k \lambda_k + \sum_{j \in J} d^j \mu_j = \bar{y} \quad \text{e} \quad \sum_{k \in K} -\lambda_k = -1.$$

Pelo Lema de Farkas, existe $(\pi, \pi_0) \in \mathbb{R}^n \times \mathbb{R}$ tal que:

$$\pi^\top d^j \leq 0 \; \forall j \in J, \quad \pi^\top x^k - \pi_0 \leq 0 \; \forall k \in K \quad \text{e} \quad \pi^\top \bar{y} - \pi_0 > 0. \tag{7.5}$$

O primeiro conjunto de desigualdades em (7.5) leva a $\pi^\top d \leq 0$, para todo $d \in \mathrm{rec}(P)$. Sendo $P = P(A,b)$, temos que $\{d : Ad \leq 0, \pi^\top d > 0\} = \emptyset$. Novamente pelo Lema de Farkas, existe $u \geq 0$ tal que $A^\top u = \pi$. Então, para todo $x \in P$, $\pi^\top x = u^\top A x \leq u^\top b$. Como $P \neq \emptyset$ é um conjunto fechado, isso mostra que existe $\alpha \in \mathbb{R}$ tal que $\max\{\pi^\top x : x \in P\} = \alpha$, ou melhor, que a face $F := \{x \in P : \pi^\top x = \alpha\}$ é não vazia. Como $\mathrm{posto}(A) = n$, pelas

proposições 7.3.7 e 7.3.8, F contém vértice. Posto que todo vértice de F é vértice de P, concluímos que existe $k \in K$ tal que $\max\{\pi^\top x : x \in P\} = \pi^\top x^k$. Isto leva a $\pi^\top \bar{y} \leq \pi^\top x^k$, o que contradiz os dois últimos blocos de desigualdades em (7.5). Portanto, $P \subseteq P'$.

A conclusão que $Q \subset \mathbb{Q}^n$ e $R \subset \mathbb{Z}^n$ quando P é racional advém da proposição 7.3.9. ∎

Quando $L := \text{lin}(P) \neq \{0\}$, podemos obter uma generalização do resultado acima, decompondo $P = L + (P \cap L^\perp)$ e observando que $\text{lin}(P \cap L^\perp) = \{0\}$ (ver por exemplo [190]).

Exercício 7.3.15. Mostre que (π, π_0) é válida para P se, e somente se, $\pi^\top x \leq \pi_0$, para todo x vértice de P, e $\pi^\top d \leq 0$, para todo d direção extrema de P.

Para mostrarmos uma recíproca para o teorema 7.3.3, vamos derivar algumas propriedades sobre projeção de poliedros. Dado um poliedro $P \subseteq \times \mathbb{R}^p$, com pontos descritos pelo par $(x, y) \in \mathbb{R}^n \times \mathbb{R}^p$, definimos $\text{proj}_x(P) = \{x \in \mathbb{R}^n : (x, y) \in P \text{ para algum } y \in \mathbb{R}^p\}$.

Lema 7.3.5. Se $P = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : Ax + Gy \leq b\}$, então $\text{proj}_x(P) = \{x \in \mathbb{R}^n : (v^i)^\top (b - Ax) \geq 0, \forall i \in I\}$, em que $\{v^i\}_{i \in I}$ é o conjunto de direções extremas de $Q = \{v \in \mathbb{R}^m : G^\top v = 0, v \geq 0\}$.

Prova: Note que $\bar{x} \in \text{proj}_x(P)$ se, e somente se, $\{y \in \mathbb{R}^p : Gy \leq b - A\bar{x}\} \neq \emptyset$, o que equivale, pelo Lema de Farkas, a $\{v \in Q : (b - A\bar{x})^\top v < 0\} = \emptyset$. Como $Q = \text{rec}(Q) \neq \emptyset$, isso é o mesmo que $(v^i)^\top (b - A\bar{x}) \geq 0$, para todo $i \in I$. ∎

Corolário 7.3.9. $\text{proj}_x(P)$ é um poliedro. Se P é racional, $\text{proj}_x(P)$ também o é.

Exercício 7.3.16. Para poliedros $P, P' \subset \mathbb{R}^n$, mostre que $P + P'$ é poliedro.

O corolário acima é uma ferramenta bastante útil para mostrar que um conjunto é poliedral. Em particular, podemos obter a seguinte recíproca para o Teorema de Minkowski.

Teorema 7.3.4 (Teorema de Weyl). Sejam $Q \subset \mathbb{R}^n$ e $R \subset \mathbb{R}^n$ conjuntos finitos e $P = \text{conv}(Q) + \text{cone}(R)$. Então, P é um poliedro (racional, se Q e R são racionais). Se $Q \neq \emptyset$, então $\text{rec}(P) = \text{cone}(R)$.

Prova: Sejam $A_{n \times |Q|}$ e $B_{n \times |R|}$ as matrizes cujas colunas são os vetores de Q e R, respectivamente. Defina o poliedro $P' = \{(x, y, z) \in \mathbb{R}^n \times \mathbb{R}_+^{|Q|} \times \mathbb{R}_+^{|R|} : x = Ay + Bz, \sum_{k=1}^{|Q|} y_k = 1\}$. Como $P = \text{proj}_x(P')$, o primeiro resultado segue do corolário 7.3.9.

Suponha agora $Q \neq \emptyset$. Seja $d \in \text{cone}(R)$. Tome $x \in Q \subset P$ e $\alpha \geq 0$. Então $x + \alpha d \in \text{conv}(Q) + \text{cone}(R) = P$, mostrando que $d \in \text{rec}(P)$ e, consequentemente, $\text{cone}(R) \subseteq \text{rec}(P)$. Suponha, por absurdo, que exista $d \in \text{rec}(P) \setminus \text{cone}(R)$. Então não existe $\mu \in \mathbb{R}_+^{|Q|}$ tal que $B\mu = d$. Pelo Lema de Farkas, existe π tal que $\pi^\top B \leq 0$ e $\pi^\top d > 0$. Como $d \in \text{rec}(P)$ e $\pi^\top d > 0$, temos que o problema $\max\{\pi^\top x : x \in P\}$ é ilimitado. Por outro lado, podemos mostrar que $\pi^\top x$ é limitado superiormente em P, levando a uma contradição. De fato, se $x \in P$, então $x = A\lambda + B\mu$ para algum $\mu \geq 0$ e $\lambda \geq 0$, satisfazendo $\sum_{i=1}^{|Q|} \lambda_i = 1$. Desse modo, $\pi^\top x = \pi^\top A\lambda + \pi^\top B\mu \leq \pi^\top A\lambda \leq \max_{y \in Q}\{\pi^\top y\} \sum_{i=1}^{|Q|} \lambda_i = \max_{y \in Q}\{\pi^\top y\}$, em que as desigualdades decorrem de $\pi^\top B \leq 0$, $\mu \geq 0$ e $\lambda \geq 0$. Pela contradição, obtemos $\text{rec}(P) \subseteq \text{cone}(R)$. ∎

Quando P é limitado, ou seja, um politopo, temos que $\text{lin}(P) = \text{rec}(P) = \{0\}$. Os teoremas de Minkowski e Weyl nos levam à seguinte caracterização.

Corolário 7.3.10. Politopo é o fecho convexo de um número finito de pontos.

Também a partir do Teorema de Minkowski obtemos facilmente os seguintes resultados sobre programação linear, que serão úteis adiante.

Teorema 7.3.5. Sejam $P \neq \emptyset$ com $\text{lin}(P) = \{0\}$, $c \in \mathbb{R}^n$ e $(\mathcal{P}_L)$ o problema $\max\{c^\top x : x \in P\}$. Então: (i) $(\mathcal{P}_L)$ tem solução ótima se, e somente se, P possui vértice x^* tal que $c^\top x^* \geq c^\top x$, para todo $x \in P$, e (ii) $(\mathcal{P}_L)$ é ilimitado se, e somente se, P possui direção extrema d^* tal que $c^\top d^* > 0$.

Prova: Se P possui vértice (resp. direção extrema) com a propriedade desejada, diretamente concluímos que $(\mathcal{P}_L)$ tem solução ótima (resp. é ilimitado). Para obter as implicações diretas, considere $Q = \{x^k : k \in K\} \neq \emptyset$ e $R = \{d^j : j \in J\}$ os conjuntos de vértices e direções extremas de P. Seja $x \in P$. Pelo teorema 7.3.3, existem $\alpha_k \geq 0$, $k \in K$, e $\beta_j \geq 0$, $j \in J$, com $\sum_{k \in K} \alpha_k = 1$, tais que $x = \sum_{k \in K} \alpha_k x^k + \sum_{j \in J} \beta_j d^j$. Tome $x^* \in \arg\max\{c^\top x^k : k \in K\}$. Assim,

$$c^\top x \leq c^\top x^* + \sum_{j \in J} \beta_j c^\top d^j.$$

Se $c^\top d^j \leq 0$, para todo $j \in J$, então $c^\top x \leq c^\top x^*$ e, portanto, $(\mathcal{P}_L)$ não é ilimitado. Isso mostra a implicação direta em (ii), pela contrapositiva. Além disso, por (ii), se $(\mathcal{P}_L)$ tem solução ótima, deve ocorrer $c^\top d^j \leq 0$, para todo $j \in J$, levando a $c^\top x \leq c^\top x^*$. ■

Podemos especializar a primeira parte do teorema acima e mostrar que todo vértice é a única solução ótima para pelo menos um vetor c.

Teorema 7.3.6. Para todo vértice $\bar{x}$ de P, existe $c \in \mathbb{R}^n$ tal que $\bar{x}$ é a única solução ótima de $\max\{c^\top x : x \in P\}$. Caso P seja racional, podemos tomar $c \in \mathbb{Z}^n$.

Prova: Sejam $P = P(A,b)$ e $c = \sum_{i \in M_{\bar{x}}^=} a^i$. Tome $x \in P \setminus \{\bar{x}\}$. Como $\bar{x}$ é face de dimensão zero de P, existe $i \in M_{\bar{x}}^=$ tal que $(a^i)^\top x < b_i$. Posto que $(a^i)^\top x \leq b_i$, para todo $i \in M$, segue que

$$c^\top x = \sum_{i \in M_{\bar{x}}^=} (a^i)^\top x < \sum_{i \in M_{\bar{x}}^=} b_i = \sum_{i \in M_{\bar{x}}^=} (a^i)^\top \bar{x} = c^\top \bar{x}.$$

Portanto, $\bar{x}$ é a única solução ótima. Caso P seja racional, podemos considerar que $a^i \in \mathbb{Q}^n$ e, por conseguinte, $c \in \mathbb{Q}^n$. Então existe $\lambda > 0$ tal que $\hat{c} = \lambda c \in \mathbb{Z}^n$ leva à mesma solução ótima. ■

7.4 Fecho inteiro

Nesta seção, queremos estudar a envoltória convexa de pontos inteiros de um poliedro racional. Mostramos que tal conjunto também é um poliedro racional e que sua caracterização é importante na resolução de problemas de otimização combinatória e de programação inteira.

7.4.1 Caracterização

Definição 7.4.1. O fecho inteiro (ou envoltória inteira) de um poliedro $P \subseteq \mathbb{R}^n$, denotado por P_I, é o fecho convexo dos pontos inteiros em P, ou seja, $P_I = \text{conv}(P \cap \mathbb{Z}^n)$.

Para a caracterização de P_I, usamos os seguintes resultados auxiliares.

Lema 7.4.1. Sejam $A, B \subseteq \mathbb{R}^n$. Então:
- i) $\text{conv}(A+B) = \text{conv}(A) + \text{conv}(B)$,
- ii) $\text{conv}((A+B) \cap \mathbb{Z}^n) \supseteq \text{conv}(A \cap \mathbb{Z}^n) + \text{conv}(B \cap \mathbb{Z}^n)$,
- iii) se $A = \text{cone}(a^1, \ldots, a^k)$, $\{a^1, \ldots, a^k\} \subset \mathbb{Z}^n$, então $A = \text{conv}(A \cap \mathbb{Z}^n)$.

Prova:

i) Seja $z \in \text{conv}(A+B)$. Então $z = \sum_i \alpha_i z^i$, onde $\sum_i \alpha_i = 1$ e, para todo i, $\alpha_i \geq 0$, $z^i = x^i + y^i$, $x^i \in A$, $y^i \in B$. Logo, $z = x + y$, onde $x = \sum_i \alpha_i x^i \in \text{conv}(A)$ e $y = \sum_i \alpha_i y^i \in \text{conv}(A)$, ou seja, $z \in \text{conv}(A) + \text{conv}(B)$. Tome agora $z \in \text{conv}(A) + \text{conv}(B)$. Então $z = x + y$, onde $x = \sum_i \alpha_i x^i$, $y = \sum_j \beta_j y^j$, com $\sum_i \alpha_i = \sum_j \beta_j = 1$ e, para todos i, j, $\alpha_i, \beta_j \geq 0$, $x^i \in A$, $y^j \in B$. Definindo $\theta_{ij} = \alpha_i \beta_j$ e $z^{ij} = x^i + y^j$, temos que $z = \sum_i \sum_j \beta_j \alpha_i x^i + \sum_j \sum_i \alpha_i \beta_j y^j = \sum_i \sum_j \theta_{ij} z^{ij}$, com $\sum_i \sum_j \theta_{ij} = \sum_i \alpha_i \sum_j \beta_j = 1$ e, para todos i, j, $\theta_{ij} \geq 0$ e $z^{ij} \in A + B$. Portanto, $z \in \text{conv}(A+B)$.

ii) $\text{conv}(A \cap \mathbb{Z}^n) + \text{conv}(B \cap \mathbb{Z}^n) = \text{conv}((A \cap \mathbb{Z}^n) + (B \cap \mathbb{Z}^n)) \subseteq \text{conv}((A+B) \cap \mathbb{Z}^n))$, em que a igualdade segue do primeiro item e a inclusão decorre de $(A \cap \mathbb{Z}^n) + (B \cap \mathbb{Z}^n) \subseteq ((A+B) \cap \mathbb{Z}^n)$.

iii) Como $A \cap \mathbb{Z}^n \subseteq A$ e A é convexo, temos que $\text{conv}(A \cap \mathbb{Z}^n) \subseteq A$. Para a inclusão inversa, tome $x \in A$. Se $x = 0$, trivialmente $x \in \text{conv}(A \cap \mathbb{Z}^n)$. Do contrário, $x = \sum_{i \in I} \alpha_i a^i$ para $I \neq \emptyset$, $I \subseteq \{1, 2, \ldots, k\}$, $\alpha_i > 0, \forall i \in I$. Defina $\alpha = \sum_{i \in I} \lceil \alpha_i \rceil \in \mathbb{Z}$, $\beta_i = \alpha_i / \alpha > 0$, para todo $i \in I$, $\beta_0 = 1 - \sum_{i \in I} \beta_i \geq 0$ e $a^0 = 0 \in A$. Então, $x = \sum_{i \in I \cup \{0\}} \beta_i (\alpha a^i)$, com $\sum_{i \in I \cup \{0\}} \beta_i = 1$, $\beta_i \geq 0$ e $\alpha a^i \in A \cap \mathbb{Z}^n$, para todo $i \in I \cup \{0\}$. Logo, $x \in \text{conv}(A \cap \mathbb{Z}^n)$.

■

Teorema 7.4.1 ([166]). Seja P um poliedro não vazio, racional, com $\text{lin}(P) = \{0\}$. Então P_I é um poliedro racional. Mais ainda, se $P_I \neq \emptyset$, então $\text{rec}(P_I) = \text{rec}(P)$.

Prova: Pelo teorema 7.3.3, $P = C + D$, onde $C = \text{conv}(Q)$, $D = \text{cone}(R) = \text{rec}(P)$, $Q \subset \mathbb{Q}^n$ é o conjunto (não vazio) de vértices de P e $R = \{d^j : j \in J\} \subset \mathbb{Z}^n$ é o conjunto de direções extremas de P. Defina o poliedro $B = \{\sum_{j \in J} \mu_j d^j : 0 \leq \mu_j \leq 1 \, \forall j \in J\}$. Note que $C + B$ é um poliedro limitado contido em P. Além disso, $D = D_I$ pelo lema 7.4.1(iii). Queremos mostrar que $P_I = (C+B)_I + D$. Com isso, observando que $(C+B) \cap \mathbb{Z}^n$ é um conjunto finito, os resultados seguem pelo teorema 7.3.4.

Primeiro, devido a $C + B \subseteq P$, $D = D_I$ e o lema 7.4.1(ii), temos que $(C+B)_I + D \subseteq P_I + D_I \subseteq (P+D)_I = P_I$. Para a inclusão inversa, posto que $(C+B)_I + D$ é um poliedro e, portanto, convexo, é suficiente mostrar que tal conjunto contém $P \cap \mathbb{Z}^n$. Seja então $y \in P \cap \mathbb{Z}^n$, isto é, $y = x + d \in \mathbb{Z}^n$, onde $x \in C$ e $d = \sum_{j \in J} \mu_j d^j \in D$, com $\mu_j \geq 0$ para todo $j \in J$. Note que $d = \bar{d} + \hat{d}$, onde $\bar{d} = \sum_{j \in J} (\mu_j - \lfloor \mu_j \rfloor) d^j \in B$ e $\hat{d} = \sum_{j \in J} \lfloor \mu_j \rfloor d^j \in D$. Considerando ainda que $y = (x + \bar{d}) + \hat{d} \in \mathbb{Z}^n$ e $\hat{d} \in \mathbb{Z}^n$, concluímos que $x + \bar{d} \in (C+B) \cap \mathbb{Z}^n$ e, por conseguinte, $y \in (C+B)_I + D$. ■

Como antecipado na seção 9.1, o fecho inteiro permite a definição de um problema linear equivalente a (7.1). Considere novamente nosso problema alvo

$$(\mathcal{P}) \quad \max\{c^\top x : x \in P \cap \mathbb{Z}^n\},$$

em que $c \in \mathbb{R}^n$ e P é um poliedro racional. Estudamos a seguir suas relações com os problemas de programação linear:

$$(\mathcal{P}_I) \quad \max\{c^\top x : x \in P_I\} \quad \text{e} \quad (\mathcal{P}_L) \quad \max\{c^\top x : x \in P\}.$$

Proposição 7.4.1. Os problemas $(\mathcal{P})$ e $(\mathcal{P}_I)$ são equivalentes no seguinte sentido:

i) $(\mathcal{P})$ é inviável se, e somente se, $(\mathcal{P}_I)$ é inviável;

ii) $(\mathcal{P})$ é ilimitado se, e somente se, $(\mathcal{P}_I)$ é ilimitado;

iii) $(\mathcal{P})$ tem solução ótima se, e somente se, $(\mathcal{P}_I)$ tem solução ótima; mais ainda, toda solução ótima de $(\mathcal{P})$ é solução ótima de $(\mathcal{P}_I)$ e toda solução ótima de $(\mathcal{P}_I)$ que é vértice de P_I é solução ótima de $(\mathcal{P})$.

Prova: O item i) segue do fato de que $P_I = \emptyset$ se, e somente se, $P \cap \mathbb{Z}^n = \emptyset$.

Podemos então admitir $P \cap \mathbb{Z}^n \neq \emptyset$ para mostrar ii). Pelo teorema 7.4.1, $\text{rec}(P_I) = \text{rec}(P)$, o que significa que $(\mathcal{P}_I)$ é ilimitado se, e somente se, $(\mathcal{P}_L)$ é ilimitado. Se $(\mathcal{P}_L)$ é

ilimitado, então, pelo teorema 7.3.5 e proposição 7.3.9, existe $d \in \mathbb{Z}^n$, direção extrema de P, tal que $c^\top d > 0$. Seja $x \in P \cap \mathbb{Z}^n$. Temos que $x + \alpha d \in P \cap \mathbb{Z}^n$, para todo $\alpha \in \mathbb{Z}_+$. Logo, $(\mathcal{P})$ é ilimitado. Por outro lado, se $(\mathcal{P})$ é ilimitado, então $(\mathcal{P}_L)$ é ilimitado, pois $P \cap \mathbb{Z}^n \subseteq P$.

Agora mostramos a segunda parte de (iii), que leva à primeira. Seja $x^* \in P \cap \mathbb{Z}^n$ uma solução ótima de $(\mathcal{P})$. Tome $x \in P_I$, ou seja, $x = \sum_i \alpha_i x^i$, com $\sum_i \alpha_i = 1$, $\alpha_i \geq 0$ e $x^i \in P \cap \mathbb{Z}^n$ para todo i. Como $c^\top x^i \leq c^\top x^*$, obtemos $c^\top x = \sum_i \alpha_i c^\top x^i \leq c^\top x^*$, mostrando que x^* é ótimo para $(\mathcal{P}_I)$. Suponha agora que $(\mathcal{P}_I)$ tenha solução ótima. Pelo teorema 7.3.5, uma das soluções ótimas é um vértice x^* de P_I, que é um ponto inteiro e, portanto, viável para $(\mathcal{P})$. Logo, o valor ótimo de $(\mathcal{P})$ é no mínimo $c^\top x^*$. Por outro lado, como $P \cap \mathbb{Z}^n \subseteq P_I$, o valor ótimo de $(\mathcal{P})$ é no máximo o de $(\mathcal{P}_I)$. Concluímos então que x^* é ótimo para $(\mathcal{P})$. ■

Proposição 7.4.2. Se $P \cap \mathbb{Z}^n \neq \emptyset$, então:

i) $(\mathcal{P})$ é ilimitado se, e somente se, $(\mathcal{P}_L)$ é ilimitado;

ii) $(\mathcal{P})$ tem solução ótima se, e somente se, $(\mathcal{P}_L)$ tem solução ótima; o valor ótimo de $(\mathcal{P}_L)$ é um limite superior para o valor ótimo de $(\mathcal{P})$.

Prova: Suponha $P \cap \mathbb{Z}^n \neq \emptyset$, implicando que $(\mathcal{P})$ e $(\mathcal{P}_L)$ sejam ambos viáveis. Conforme prova da proposição 7.4.1(ii), temos que $(\mathcal{P})$ é ilimitado se, e somente se, $(\mathcal{P}_L)$ é ilimitado. Consequentemente, um dos problemas tem solução ótima se, e somente se, o outro também possui. E o valor ótimo de $(\mathcal{P}_L)$ é no mínimo o de $(\mathcal{P})$, posto que $P \supseteq P \cap \mathbb{Z}^n$. ■

7.4.2 Determinação da dimensão

Como P_I é um poliedro, valem os resultados obtidos na seção 7.3 para conjuntos poliedrais. Em particular, usamos aqui as propriedades derivadas na subseção 7.3.2 para detalhar formas de determinar a dimensão de P_I. Primeiro, lembre que $\dim(S) = \dim(\text{conv}(S))$ para qualquer conjunto S. Além disso, se (π, π_0) é válida para S e $H = \{x : \pi^\top x = \pi_0\}$, então $\text{conv}(S) \cap H = \text{conv}(S \cap H)$ (exercício 7.3.6). Logo, podemos usar apenas vetores afim-independentes de $P \cap \mathbb{Z}^n$ para determinar a dimensão de P_I e suas faces.

Consideramos, ao longo desta subseção, que $P_I \neq \emptyset$.

Para mostrarmos que $\dim(P_I) = k$, podemos demonstrar alternativamente que:

d: existe matriz (D, d) tal que $\text{posto}(D) = n - k$, $Dx = d$ $\forall x \in P_I$ e ocorre um dos seguintes casos:

- d1: P contém $k+1$ pontos inteiros afim-independentes, ou
- d2: $P_I \subseteq \{x \in \mathbb{R}^n : a^\top x = a_0\} \Rightarrow (a^\top, a_0) = u^\top (D, d)$ para algum u, ou
- d3: $\text{afim}(P_I) = \{x : Dx = d\}$, ou melhor, todo ponto que satisfaz $Dx = d$ é combinação afim de pontos inteiros de P;

d': P contém $k+1$ pontos inteiros afim-independentes e todo ponto inteiro de P é combinação afim destes.

Em particular, para concluirmos que $P_I \subseteq \mathbb{R}^n$ tem dimensão plena, podemos usar uma das seguintes alternativas:

D1: P contém $n+1$ pontos inteiros afim-independentes, ou

D2: $P_I \subseteq \{x \in \mathbb{R}^n : a^\top x = a_0\} \Rightarrow a = 0, a_0 = 0$, ou

D3: $\text{afim}(P_I) = \mathbb{R}^n$, ou melhor, todo ponto em $\mathbb{R}^n$ é combinação afim de pontos inteiros de P.

Exercício 7.4.1 (cont. exercício 7.3.3). O conjunto dos vetores de incidência de conjuntos independentes em um grafo $G=(V,E)$, com $V=\{1,2,\ldots,n\}$, é $P\cap\mathbb{Z}^n$, onde

$$P=\{x\in\mathbb{R}^n : x_u+x_v\le 1\,\forall uv\in E,\ 0\le x_u\le 1\,\forall u\in V\}.$$

Logo, P_I é o politopo de conjuntos independentes. Para $u\in\{1,2,\ldots,n\}$, seja $e^u\in\mathbb{B}^n$ tal que $e^u_j=1$, se $u=j$, e $e^u_j=0$, caso contrário. Defina $C=\{e^u : u\in V\}\cup\{0\}$ e note que $C\subset P_I$. Podemos mostrar que P_I tem dimensão plena através das seguintes alternativas:

D1: O conjunto C contém $n+1$ pontos afim-independentes.

D2: Suponha $P_I\subseteq\{x\in\mathbb{R}^n : a^\top x=a_0\}$. Como $0\in P_I$, temos que $a_0=0$. Para todo $u\in V$, como $e^u\in P_I$, temos que $a_u=a^\top e^u=a_0=0$.

D3: Seja $x\in\mathbb{R}^n$. Como $x=\sum_{u\in V}x_u e^u=\sum_{u\in V}x_u e^u+(1-\sum_{u\in V}x_u)0$, temos que $x\in\operatorname{afim}(C)\subseteq\operatorname{afim}(P_I)$.

Exercício 7.4.2 (cont. exercício 7.3.4). Sejam $D=(V,A)$ um digrafo acíclico com $|V|=n$ e $|A|=m$, $N^-(v)=\{u : uv\in A\}$ para $v\in V$ e $S=\{v : N^-(v)=\emptyset\}$. Considere o politopo P_I, onde

$$\begin{aligned} P=\{x\in\mathbb{R}^{n+m} : x_{vv}+\sum_{u\in N^-(v)}x_{uv}\ge 1 \quad \forall v\in V, \\ 0\le x_{uv}\le 1 \quad \forall v\in V \quad \forall u\in N^-(v)\cup\{v\}\}. \end{aligned} \tag{7.6}$$

Note que (7.6) torna-se $x_{uv}\ge 1$ quando $v\in S$. Portanto, $x_{vv}=1$, para todo $v\in S$, constituem $|S|$ igualdades implícitas, linearmente independentes, de P_I. Denote por $Dx=d$ esse sistema de igualdades. Assim, $\operatorname{posto}(D)=|S|$ e, por conseguinte, $\dim(P_I)\le n+m-|S|$. Podemos mostrar que esse limite superior é atingido através de uma das seguintes alternativas:

d1: Para $v\in V$, $u\in N^-(v)$, defina $e^{uv}\in\mathbb{B}^{n+m}$ tal que $e^{uv}_{wz}=1$, se $w=u$ e $z=v$, e $e^{uv}_{wz}=0$, caso contrário. Considere o conjunto de $m+n-|S|+1$ pontos dado por

$$\begin{aligned} &X=\textstyle\sum_{u\in V}e^{uu}, \qquad X^{uv}=X+e^{uv}\ \forall(u,v)\in A,\\ &X^v=X-e^{vv}+e^{r(v)v}\ \forall v\in V\setminus S \text{ com } r(v)\in N^-(v). \end{aligned} \tag{7.7}$$

Esses pontos pertencem a P_I e são AI (ver exercício 7.3.4).

d2: Suponha $P_I\subseteq\{x\in\mathbb{R}^n : a^\top x=a_0\}$. Seja $(u,v)\in A$. Como $X,X^{uv}\in P_I$, temos que $a^\top(X^{uv}-X)=0$, ou seja, $a_{uv}=0$. Seja $v\in V\setminus S$. Dado que $X,X^v\in P_I$, obtemos $a^\top(X^v-X)=a_{r(v)v}-a_{vv}=0$, ou ainda, $a_{vv}=a_{r(v)v}=0$. Adicionalmente, por $X\in P_I$ calculamos $a_0=a^\top X=\sum_{v\in S}a_{vv}$. Tomando $\mu_v=a_{vv}$, para todo $v\in S$, obtemos então que $(a^\top,a_0)=\mu^\top(D,d)$.

d3: Seja $x\in\mathbb{R}^n$ tal que $x_{vv}=1$ para todo $v\in S$. Desejamos mostrar que x é combinação afim dos pontos em (7.7), ou seja, que existem α_0, α_{uv}, para todo $(u,v)\in A$, e α_v, para todo $v\in V\setminus S$, tais que $x=\alpha_0X+\sum_{(u,v)\in A}\alpha_{uv}X^{uv}+\sum_{v\in V\setminus S}\alpha_vX^v$ e $1=\alpha_0+\sum_{(u,v)\in A}\alpha_{uv}+\sum_{v\in V\setminus S}\alpha_v$. Mais precisamente, queremos que tenha solução o sistema

$$\begin{aligned} \alpha_0+\sum_{(u,v)\in A}\alpha_{uv}+\sum_{v\in V\setminus S}\alpha_v &= 1\\ \alpha_0+\sum_{(u,v)\in A}\alpha_{uv}+\sum_{v\in V\setminus(S\cup\{u\})}\alpha_v &= x_{uu}\ \forall u\in V\setminus S\\ \alpha_{uv} &= x_{uv}\ \forall(u,v)\in A, u\ne r(v)\\ \alpha_{uv}+\alpha_v &= x_{uv}\ \forall(u,v)\in A, u=r(v). \end{aligned}$$

Verifique que uma solução é dada por $\alpha_0=1-\sum_{(u,v)\in A}x_{uv}$, $\alpha_u=1-x_{uu}$, para $u\in V\setminus S$, e α_{uv} igual a x_{uv} (se $u\ne r(v)$) ou $x_{uv}+x_{vv}-1$ (se $u=r(v)$).

Observe que a estratégia d' é obtida pela conjunção de d1, d3 e o fato de que $P \subseteq \{x : Dx = d\}$.

7.4.3 Obtenção de facetas

Consideramos, ao longo desta subseção, que $P_I \neq \emptyset$. Similarmente à anterior, usando os resultados da subseção 7.3.4, podemos mostrar quando uma desigualdade (π, π_0), válida para P_I, induz faceta $F = \{x \in P_I : \pi^\top x = \pi_0\}$. Primeiro, devemos verificar que $F \neq P_I$, ou melhor, $P_I \setminus F \neq \emptyset$. Então, tendo $P_I \subseteq \mathbb{R}^n$ dimensão k e sendo (D, d) uma matriz que satisfaça posto$(D) = n - k$ e $Dx = d$ $\forall x \in P_I$, podemos mostrar (entre outras possibilidades) que:

f1: F contém k pontos inteiros afim-independentes, ou

f2: $F \subseteq \{x \in P_I : a^\top x = a_0\} \Rightarrow (a^\top, a_0) = \alpha(\pi^\top, \pi_0) + u^\top(D, d)$ para algum α e u (ou simplesmente $a^\top = \alpha\pi^\top + u^\top D$ para algum α e u, caso $F \neq \emptyset$), ou

f3: todo ponto em $\{x : Dx = d, \pi^\top x = \pi_0\}$ é combinação afim de pontos inteiros de F.

Caso P_I tenha dimensão plena, podemos reescrever tais alternativas como:

F1: F contém n pontos inteiros afim-independentes, ou

F2: $F \subseteq \{x \in P_I : a^\top x = a_0\} \Rightarrow (a^\top, a_0) = \alpha(\pi^\top, \pi_0)$ para algum α (ou simplesmente $a^\top = \alpha\pi^\top$ para algum α, caso $F \neq \emptyset$), ou

F3: todo ponto em $\{x : \pi^\top x = \pi_0\}$ é combinação afim de pontos inteiros de F.

Note ainda que $F \neq P_I$ equivale a (π, π_0) não ser combinação linear das linhas de (D, d) ou, simplesmente, $(\pi, \pi_0) \neq 0$ no caso de dimensão plena.

Por outro lado, para mostrarmos que a face $F = \{x \in P : \pi^\top x = \pi_0\}$, induzida por uma desigualdade válida (π, π_0), não é faceta, podemos, alternativamente, mostrar que:

n1: F contém $k < \dim(P_I)$ pontos inteiros afim-independentes e todo outro ponto inteiro de F é combinação afim daqueles;

n2: existe desigualdade válida (τ, τ_0) que domina (π, π_0), ou seja, tal que $F \subsetneq \{x \in P_I : \tau^\top x = \tau_0\}$, ou ainda, que não é equivalente a (π, π_0) e satisfaz $F \subseteq \{x \in P_I : \tau^\top x = \tau_0\}$;

n3: existem $p \geq 1$ desigualdades válidas (τ^i, τ_0^i), $i \in \{1, \ldots, p\}$, pelo menos uma delas não equivalente a (π, π_0), tais que $\pi = \sum_{i=1}^p \alpha_i \tau^i$ e $\pi_0 \geq \sum_{i=1}^p \alpha_i \tau_0^i$, com $\alpha_i > 0$, para todo $i \in \{1, \ldots, p\}$.

Observe que em (n3) é suficiente obter a desigualdade $\pi \leq \sum_{i=1}^p \alpha_i \tau^i$, quando $P \subseteq \mathbb{R}^n_+$, pois a igualdade pode ser obtida com a adição de múltiplos de algumas das desigualdades $-x \leq 0$.

Exercício 7.4.3 (cont. exercício 7.4.1). Vamos mostrar que a desigualdade (π, π_0), dada por $\sum_{v \in K} x_v \leq 1$, induz uma faceta de P_I se, e somente se, K é clique maximal em G. Primeiro, suponha que $K \subseteq V$ seja uma clique maximal em G. Como no máximo um vértice de qualquer clique pode estar em um conjunto independente, temos que a desigualdade (π, π_0) é válida para P_I. Seja $F = \{x \in P_I : \pi^\top x = \pi_0\}$. Trivialmente temos $F \neq P_I$, posto que P_I tem dimensão plena. Sendo K maximal, para todo $v \in V \setminus K$, podemos eleger um vértice em K, a ser denotado $\bar{v} \in K$, tal que $v\bar{v} \notin E$. Podemos mostrar que F é faceta através das seguintes alternativas:

F1: Considere os pontos e^v, para todo $v \in K$, e $e^v + e^{\bar{v}}$, para todo $v \in V \setminus K$. Esses n pontos estão em F. Mais ainda, são LI e, portanto, AI.

F2: Suponha $F \subseteq \{x \in P_I : a^\top x = a_0\}$. Para $v \in V \setminus K$, temos que $e^{\bar{v}}$ e $e^v + e^{\bar{v}}$ estão em F. Logo, $a^\top(e^v + e^{\bar{v}}) = a^\top e^{\bar{v}}$, implicando $a_v = 0$. Além disso, para $v \in K$, $e^v \in F$ leva a $a_v = a_0$. Assim, tomando $\alpha = a_0$, temos que $(a^\top, a_0) = \alpha(\pi^\top, \pi_0)$.

F3: Seja $x \in \mathbb{R}^n$ tal que $\sum_{v \in K} x_v = 1$. Defina $\bar{K} = \{\bar{v} \in K : v \in V \setminus K\}$. Queremos mostrar que existem α_v, para todo $v \in V$, tais que $x = \sum_{v \in K} \alpha_v e^v + \sum_{v \in V \setminus K} \alpha_v (e^v + e^{\bar{v}})$ e $\sum_{v \in V} \alpha_v = 1$. Verifique que obtemos tais propriedades, tomando $\alpha_v = x_v$, se $v \in V \setminus \bar{K}$, e $\alpha_{\bar{v}} = x_{\bar{v}} - x_v$, se $\bar{v} \in \bar{K}$.

Suponha agora que K não seja clique maximal. Não sendo clique, a desigualdade (π, π_0) não é válida. Não sendo maximal, existe $u \in V \setminus K$ tal que $K \cup \{u\}$ também é clique. Então, $\sum_{v \in K \cup \{u\}} x_v \leq 1$ e $-x_u \leq 0$ são válidas para P_I. Usando a alternativa n3, concluímos que (π, π_0) não define faceta. Mostre o mesmo resultado, usando n1 e n2.

Exercício 7.4.4 (cont. exercício 7.4.2). Seja $w \in V$. Queremos mostrar que a desigualdade (π, π_0), dada por $-x_{ww} - \sum_{u \in N^-(w)} x_{uw} \leq -1$, induz faceta de P_I se, e somente se, $w \in V \setminus S$. Por ser uma desigualdade da descrição de P, ela é válida para P_I. Seja $F = \{x \in P_I : \pi^\top x = \pi_0\}$. Lembre que $F = P_I$ se $w \in S$ e, logo, não é faceta. Para $w \in V \setminus S$, podemos mostrar que F é faceta, usando uma das seguintes alternativas:

f1: Considere o conjunto de $m + n - |S|$ pontos dado por

$$\begin{array}{c} X = \sum_{u \in V} e^{uu}, \quad X^v = X - e^{vv} + e^{r(v)v} \ \forall v \in V \setminus (S \cup \{w\}) \text{ com } r(v) \in N^-(v), \\ X^{uv} = X + e^{uv} \ \forall (u,v) \in A, v \neq w, \quad X^{uw} = X - e^{ww} + e^{uw} \ \forall (u,w) \in A. \end{array} \tag{7.8}$$

Em relação ao exercício 7.4.2, note que redefinimos X^{uw}, para $u \in N^-(w)$, de modo a ter todos esses $m + n - |S|$ pontos em F. Verifique que eles são AI.

f2: Suponha $F \subseteq \{x \in P_I : a^\top x = a_0\}$. Sejam $v \in V \setminus (S \cup \{w\})$ e $(u,v) \in A$. Os pontos $X, X^{uv}, X^v \in F$ levam a $a^\top(X^{uv} - X) = a^\top(X^v - X) = 0$, ou ainda, $a_{uv} = 0$ e $a_{vv} = a_{r(v)v} = 0$. Para $u \in N^-(w)$, obtemos $a_{uw} = a_{ww}$ a partir $X, X^{uw} \in F$. Mais ainda, $a_0 = a^\top X = a_{ww}$. Tomando $\alpha = -a_{ww}$ e $\mu = 0$, chegamos a $(a^\top, a_0) = \alpha(\pi^\top, \pi_0) + \mu^\top(D, d)$.

f3: Seja $x \in \mathbb{R}^n$ tal que $x_{vv} = 1$ para todo $v \in S$. Desejamos mostrar que x é combinação afim dos pontos em (7.8), ou seja, que existem α_0, α_{uv}, para todo $(u,v) \in A$, e α_v, para todo $v \in V \setminus (S \cup \{w\})$, tais que $x = \alpha_0 X + \sum_{(u,v) \in A} \alpha_{uv} X^{uv} + \sum_{v \in V \setminus (S \cup \{w\})} \alpha_v X^v$ e $1 = \alpha_0 + \sum_{(u,v) \in A} \alpha_{uv} + \sum_{v \in V \setminus (S \cup \{w\})} \alpha_v$. Em outros termos, queremos provar que tem solução o sistema

$$\begin{aligned} \alpha_0 + \sum_{(u,v) \in A} \alpha_{uv} + \sum_{v \in V \setminus (S \cup \{w\})} \alpha_v &= 1 \\ \alpha_0 + \sum_{(u,v) \in A} \alpha_{uv} + \sum_{v \in V \setminus (S \cup \{w,u\})} \alpha_v &= x_{uu} \ \forall u \in V \setminus (S \cup \{w\}) \\ \alpha_0 + \sum_{(u,v) \in A, v \neq w} \alpha_{uv} + \sum_{v \in V \setminus (S \cup \{w\})} \alpha_v &= x_{ww} \\ \alpha_{uv} &= x_{uv} \ \forall (u,v) \in A, u \neq r(v) \text{ ou } v = w \\ \alpha_{uv} + \alpha_v &= x_{uv} \ \forall (u,v) \in A, u = r(v) \text{ e } v \neq w. \end{aligned}$$

Mostre que uma solução é dada por $\alpha_0 = x_{ww} - \sum_{(u,v) \in A, v \neq w} x_{uv}$, $\alpha_u = 1 - x_{uu}$, para $u \in V \setminus (S \cup \{w\})$, e α_{uv} igual a x_{uv} (se $v = w$ ou $u \neq r(v)$) ou $x_{uv} + x_{vv} - 1$ (se $v \neq w$ e $u = r(v)$).

7.4.4 Lifting

Quando uma desigualdade não induz faceta, podemos procurar mudar alguns de seus coeficientes, de modo a torná-la válida (caso já não seja) e fortalecê-la (incluindo

mais pontos afim-independentes na face que ela passa a definir). Uma estratégia usual para esse fim, introduzida em [105], chama-se lifting (de variáveis). No lifting sequencial, modifica-se o coeficiente de uma variável por vez [215]; no simultâneo, um subconjunto de variáveis é avaliado ao mesmo tempo [222]. A seguir apresentamos uma técnica similar, introduzida em [218], que denominamos lifting de desigualdades, da qual podemos derivar o lifting de variáveis. Por simplicidade, vamos nos restringir a politopos, como é usual na apresentação de liftings.

Proposição 7.4.3. Seja P um politopo racional. Sejam (λ, λ_0) uma desigualdade válida para P_I e (π, π_0) uma desigualdade válida para a face $F = \{x \in P_I : \lambda^\top x = \lambda_0\}$. Se $F \neq P_I$, então

$$(\pi + \alpha\lambda)^\top x \leq \pi_0 + \alpha\lambda_0 \tag{7.9}$$

é desigualdade válida para P_I, para todo

$$\alpha \geq \max_{x \in P \cap \mathbb{Z}^n} \left\{ \frac{\pi^\top x - \pi_0}{\lambda_0 - \lambda^\top x} : \lambda^\top x < \lambda_0 \right\}. \tag{7.10}$$

Se α satisfaz (7.10) na igualdade, então a dimensão da face de P_I induzida por (7.9) é maior que a dimensão da face de F definida por (π, π_0). Para tal α, se F é faceta de P_I e (π, π_0) induz faceta de F, então (7.9) induz faceta de P_I.

Prova: Primeiro observe que o máximo em (7.10) existe, pois é calculado sobre um conjunto não vazio finito, posto que P é limitado e $F \neq P_I$. Seja $x \in P \cap \mathbb{Z}^n$. Se $x \in F$, então $\lambda^\top x = \lambda_0$ e $\pi^\top x \leq \pi_0$, de modo que x satisfaz (7.9) para qualquer α. Caso contrário, $\lambda^\top x < \lambda_0$. Por (7.10), temos que $\alpha(\lambda_0 - \lambda^\top x) \geq \pi^\top x - \pi_0$. Novamente, x verifica (7.9). Logo, tal desigualdade é válida para $P \cap \mathbb{Z}^n$ e, por conseguinte, para P_I.

Suponha agora que $\alpha = \frac{\pi^\top \bar{x} - \pi_0}{\lambda_0 - \lambda^\top \bar{x}}$, para $\bar{x} \in P \cap \mathbb{Z}^n$ com $\lambda^\top \bar{x} < \lambda_0$. Sejam $F' = \{x \in F : \pi^\top x = \pi_0\}$ e $\bar{F}$ a face de P_I definida por (7.9). Note que $(\pi + \alpha\lambda)^\top \bar{x} = \pi_0 + \alpha\lambda_0$ e $\lambda^\top \bar{x} < \lambda_0 = \lambda^\top x$, para todo $x \in F'$. Por conseguinte, $\bar{x} \in \bar{F}$, $F' \subset \bar{F}$ e $\bar{x}$ é AI com os pontos de F'. Logo, $\dim(\bar{F}) \geq \dim(F') + 1$. ∎

A figura 7.16 ilustra a proposição 7.4.3, apresentando o lifting da desigualdade (π, π_0) válida para a faceta de P_I definida por (λ, λ_0). A figura 7.16(a) considera o caso em que (π, π_0) não é válida para P_I, enquando a figura 7.16(b) exemplifica o caso complementar.

Exercício 7.4.5 (cont. exercício 7.4.1). Considere o poliedro P definido no exercício 7.4.1, em que o grafo $G = (V, E)$ é uma roda ímpar, ou seja, um ciclo ímpar junto com um vértice adicional, adjacente a todos os vértices do ciclo. Precisamente, $V = \{0, 1, 2, \ldots, n\}$, com $n \geq 5$ ímpar, e $E = \{(0, i), (i, i+1) : i \in \{1, 2, \ldots, n-1\}\} \cup \{(0, n), (n, 1)\}$. A figura 7.17 exibe G quando $n = 7$. Pelo exercício 7.4.3, $x_0 + x_1 + x_2 \leq 1$ induz faceta de P_I. Defina $F = \{x \in P_I : x_0 + x_1 + x_2 = 1\}$. Mostre que $(n-3)/2x_0 + \sum_{i=3}^n x_i \leq (n-3)/2$ induz faceta de F. Usando a proposição 7.4.3, mostre que $\lfloor n/2 \rfloor x_0 + \sum_{i=1}^n x_i \leq \lfloor n/2 \rfloor$ define faceta de P_I.

Quando $x_i \geq 0$ ou $x_i \leq 1$ são desigualdades válidas para P_I, o lifting de desigualdades válidas para as faces $\{x \in P_I : x_i = 0\}$ ou $\{x \in P_I : x_i = 1\}$ modifica apenas um coeficiente e pode ser obtido com a solução de um problema linear 0-1, gerando o lifting de variáveis. Esse é o caso da figura 7.16(a). Note que uma desigualdade $x_i \geq 0$ ou $x_i \leq 1$ pode ser genericamente descrita como $(2\delta - 1)x_i \leq \delta$, para $\delta \in \mathbb{B}$.

Corolário 7.4.1. Suponha $P \subseteq [0, 1]^n$, racional. Sejam $\delta \in \{0, 1\}$ e $S^\delta = \{x \in P \cap \mathbb{B}^n : x_1 = \delta\}$. Se $x_1 = \delta$ não é igualdade de P_I e $\sum_{j=2}^n \pi_j x_j \leq \pi_0$ é desigualdade válida para S^δ, então $\alpha(2\delta - 1)x_1 + \sum_{j=2}^n \pi_j x_j \leq \pi_0 + \delta$ é válida para P_I, para todo $\alpha \geq \alpha_\delta$, onde

$$\alpha_\delta = \max \left\{ \sum_{j=2}^n \pi_j x_j - \pi_0 : x \in S^{1-\delta} \right\}.$$

Mais ainda, a desigualdade resultante induz faceta de P_I se a desigualdade original induz faceta de $\text{conv}(S^\delta)$, $(2\delta-1)x_i \leq \delta$ induz faceta de P_I e $\alpha = \alpha_\delta$.

Exercício 7.4.6 (cont. exercício 7.4.5). Seja P o poliedro usado no exercício 7.4.5, em que o grafo $G=(V,E)$ é uma roda ímpar. Defina $P^0 = \{x \in P : x_0 = 0\}$. Mostre que $x_0 \geq 0$ define faceta de P_I e $\sum_{i=1}^n x_i \leq \lfloor n/2 \rfloor$ define faceta de P_I^0. Usando o corolário 7.4.1, mostre que $\lfloor n/2 \rfloor x_0 + \sum_{i=1}^n x_i \leq \lfloor n/2 \rfloor$ induz faceta de P_I.

7.5 Poliedros inteiros

Nesta seção, estudamos condições necessárias ou suficientes para que um poliedro coincida com seu fecho inteiro. Sob outra perspectiva, derivamos condições para garantir que um conjunto de desigualdades (possivelmente indutoras de facetas) descreva o fecho inteiro de um poliedro.

Por simplicidade, concentramo-nos em poliedros racionais com vértices, embora os resultados aqui apresentados possam ser adaptados/generalizados para poliedros racionais quaisquer.

7.5.1 Caracterização

Definição 7.5.1. Poliedro inteiro é um poliedro em que todas as faces (minimais) não vazias contêm ponto inteiro.

Mostramos a seguir que poliedros inteiros são exatamente aqueles que coincidem com seu fecho inteiro.

Proposição 7.5.1. Para um poliedro racional $P \subseteq \mathbb{R}^n$ com $\text{lin}(P) = \{0\}$, é equivalente dizer que:

i) P é inteiro;
ii) os vértices de P são inteiros;
iii) os vértices de P estão em $\text{conv}(P \cap \mathbb{Z}^n)$;
iv) $P = \text{conv}(P \cap \mathbb{Z}^n)$.

Prova: O resultado é trivial se $P = \emptyset$. Suponha $P \neq \emptyset$ e $\text{lin}(P) = \{0\}$. Pelo corolário 7.3.4, as faces minimais não vazias de P são seus vértices. Daí segue a equivalência entre (i) e (ii). Além disso, todo ponto inteiro de P está trivialmente em $P_I := \text{conv}(P \cap \mathbb{Z}^n)$, levando a $(ii) \Rightarrow (iii)$. Suponha agora que $P_v \subset P_I$, em que $P_v \neq \emptyset$ é conjunto de vértices de P. Pelos teoremas 7.3.3, 7.3.4 e 7.4.1, $P = \text{conv}(P_v) + \text{rec}(P) \subseteq \text{conv}(P_I) + \text{rec}(P_I) = P_I \subseteq P$, o que mostra a implicação $(iii) \Rightarrow (iv)$. Finalmente, note que obtemos $(iv) \Rightarrow (ii)$ pelo fato de todo vértice de P_I ser inteiro (um ponto não inteiro de P_I é combinação convexa de pontos inteiros de P). ■

Outra caracterização importante pode ser estabelecida usando programação linear.

Proposição 7.5.2. Seja $P \subseteq \mathbb{R}^n$ um poliedro racional com $\text{lin}(P) = \{0\}$. Para cada $c \in \mathbb{R}^n$, seja $\mathcal{P}(c)$ o problema $\max\{c^\top x : x \in P\}$. É equivalente dizer que:

i) P é inteiro;
ii) $\mathcal{P}(c)$ tem solução ótima inteira para todo $c \in \mathbb{R}^n$ tal que $\mathcal{P}(c)$ tem solução ótima;
iii) $\mathcal{P}(c)$ tem solução ótima inteira para todo $c \in \mathbb{Z}^n$ tal que $\mathcal{P}(c)$ tem solução ótima;
iv) $\mathcal{P}(c)$ tem valor ótimo inteiro para todo $c \in \mathbb{Z}^n$ tal que $\mathcal{P}(c)$ tem solução ótima.

Prova: A proposição 7.5.1 e o teorema 7.3.5 levam a (i)⇒(ii). Já as implicações (ii)⇒(iii)⇒(iv) são diretas. Resta então mostrar que (iv)⇒(i) ou, equivalentemente, sua contrapositiva. Suponha que P não seja inteiro, isto é, que exista vértice $\bar{x} \in P$, onde $\bar{x}_j \notin \mathbb{Z}$ para algum $j \in \{1,2,\ldots,n\}$. Pelo teorema 7.3.6, existe $\bar{c} \in \mathbb{Z}^n$ tal que $\bar{x}$ é a única solução de $\max\{\bar{c}^\top x : x \in P\}$. Logo, existe $q \in \mathbb{Z}$, suficientemente grande, tal que $\bar{x}$ também é solução de $\max\{\hat{c}^\top x : x \in P\}$, onde $\hat{c} = \bar{c} + \frac{1}{q} e_j$. Com isso, $\bar{x}$ é solução dos problemas $\max\{q\hat{c}^\top x : x \in P\}$ e $\max\{q\bar{c}^\top x : x \in P\}$, cuja diferença de valor ótimo é $q\hat{c}^\top \bar{x} - q\bar{c}^\top \bar{x} = \bar{x}_j \notin \mathbb{Z}$. Assim, o valor ótimo de pelo menos um dos dois problemas não é inteiro. E, como $q\hat{c} \in \mathbb{Z}^n$ e $q\bar{c} \in \mathbb{Z}^n$, obtemos a negação de (iv). ∎

7.5.2 Sistemas TDI

A caracterização de poliedros inteiros através da proposição 7.5.2(iv) motiva a seguinte definição, que se relaciona ao dual do problema $\mathcal{P}(c)$.

Definição 7.5.2. Um sistema de desigualdades $Ax \leq b$, com A e b racionais, é dito TDI (totalmente dual-inteiro) se $\min\{b^\top y : A^\top y = c, y \geq 0\}$ tem solução ótima inteira, para todo $c \in \mathbb{Z}^n$, tal que $\mathcal{P}(c)$ tem solução ótima.

Vale observar que a definição de TDI não está relacionada ao poliedro, mas a uma descrição sua. A seguir usamos essa definição para caracterizar poliedros inteiros.

Proposição 7.5.3 ([83, 118]). Suponha $P(A,b) \subseteq \mathbb{R}^n$ racional com $\mathrm{lin}(P) = \{0\}$. Se $Ax \leq b$ é TDI e $b \in \mathbb{Z}^m$, então P é inteiro.

Prova: Seja $c \in \mathbb{R}^n$ tal que $\mathcal{P}(c)$ tem solução ótima. Como o dual de $\mathcal{P}(c)$ tem solução ótima inteira e $b \in \mathbb{Z}^m$, o valor ótimo desse dual (o mesmo de $\mathcal{P}(c)$) é inteiro. Pela proposição 7.5.2(iv), temos que P é inteiro. ∎

Para obtermos a recíproca da proposição acima, usamos o seguinte resultado auxiliar.

Lema 7.5.1. Seja $R = \{x^1,\ldots,x^r\} \subset \mathbb{Z}^n$ um conjunto finito. Então existe conjunto finito $S = \{x^{r+1},\ldots,x^{r+s}\} \subseteq \mathrm{cone}(R) \cap \mathbb{Z}^n$ tal que $\mathrm{cone}(R) \cap \mathbb{Z}^n = \{\sum_{i=1}^{r+s} \alpha_i x^i : \alpha_i \in \mathbb{Z}_+, i \in \{1,\ldots,r+s\}\}$.

Prova: Defina $S = \{\sum_{i=1}^{r} \alpha_i x^i : 0 \leq \alpha_i < 1, i \in \{1,\ldots r\}\} \cap \mathbb{Z}^n$. Note que S é um subconjunto finito de $\mathrm{cone}(R) \cap \mathbb{Z}^n$; digamos $S = \{x^{r+1},\ldots,x^{r+s}\}$. Logo, $\{\sum_{i=1}^{r+s} \alpha_i x^i : \alpha_i \in \mathbb{Z}_+, i \in \{1,\ldots,r+s\}\} \subseteq \mathrm{cone}(R) \cap \mathbb{Z}^n$. Para a inclusão inversa, tome $x \in \mathrm{cone}(R) \cap \mathbb{Z}^n$. Portanto, $x = y + z$, onde $y = \sum_{i=1}^{r} \lfloor \alpha_i \rfloor x^i$ e $z = \sum_{i=1}^{r} (\alpha_i - \lfloor \alpha_i \rfloor) x^i$, com $\alpha_i \geq 0$, para todo $i \in \{1,\ldots,r\}$. Temos que $x \in \mathbb{Z}^n$, $y \in \mathbb{Z}^n$, $z = y - x \in \mathbb{Z}^n$ e, assim, $z \in S$. Logo, $z = x^{r+k}$ para algum $k \in \{1,\ldots,s\}$ e, por conseguinte, $x = \sum_{i=1}^{r} \lfloor \alpha_i \rfloor x^i + x^{r+k}$, o que mostra a inclusão desejada. ∎

Proposição 7.5.4 ([96]). Suponha $P(A,b) \subseteq \mathbb{R}^n$ racional com $\mathrm{lin}(P) = \{0\}$. Se P é inteiro, então existe sistema TDI $A'x \leq b'$, com (A',b') inteiro, tal que $P = \{x \in \mathbb{R}^n : A'x \leq b'\}$.

Prova: Podemos admitir que (A,b) é uma matriz inteira. Suponha $P \neq \emptyset$, do contrário o resultado é trivial. Dada uma face $F \neq \emptyset$ de $P(A,b)$, defina o cone $C(F) = \mathrm{cone}\{a^i : i \in M_F^=\}$. Note que F é o conjunto de soluções ótimas de $\max\{d^\top x : x \in P\}$, para todo $d \in C(F)$. De fato, sendo $d = \sum_{i \in M_F^=} \alpha_i a^i$, com $\alpha_i \geq 0$ para todo $i \in M_F^=$, temos que $d^\top x = \sum_{i \in M_F^=} \alpha_i (a^i)^\top x \leq \sum_{i \in M_F^=} \alpha_i b_i$, para todo $x \in P$, sendo que a igualdade ocorre sempre que $x \in F$.

Pelo lema 7.5.1, existe conjunto finito $K(F) \subset \mathbb{Z}^n$ tal que

$$C(F) \cap \mathbb{Z}^n = \{ \sum_{\pi \in K(F)} y_\pi \pi : y_\pi \in \mathbb{Z}_+ \ \forall \pi \in K(F) \}.$$

Seja $\pi \in K(F)$. Como $\pi \in C(F) \cap \mathbb{Z}^n$ e P é inteiro, o $\max\{\pi^\top x : x \in P\}$ é finito (atingido em F) e inteiro; digamos $\pi_0 \in \mathbb{Z}$. Logo, $\pi^\top x \leq \pi_0$ é válida para P, de modo que P pode ser definido pelo sistema (com inequações redundantes)

$$A'x \leq b' \equiv Ax \leq b,\ \pi^\top x \leq \pi_0,\ \forall \pi \in K(F)\ \forall F \text{ face minimal de } P.$$

Note que A' e b' são inteiros. Para mostrar que $A'x \leq b'$ é TDI, considere $c \in \mathbb{Z}^n$ tal que $\max\{c^\top x : A'x \leq b'\}$ tenha solução ótima, atingida em uma face minimal F_c de P. O dual é dado por

$$\begin{aligned} (\mathcal{D}_c) \quad \min \quad & b^\top u + \sum_F \sum_{\pi \in K(F)} y_\pi \pi_0 \\ \text{s.a:} \quad & A^\top u + \sum_F \sum_{\pi \in K(F)} y_\pi \pi = c, \\ & u \geq 0,\ y_\pi \geq 0\ \forall \pi \in K(F)\ \forall F. \end{aligned}$$

Vamos mostrar que $(\mathcal{D}_c)$ tem solução inteira. De fato, como $c \in C(F_c) \cap \mathbb{Z}^n$, existe vetor $y^* \in \mathbb{Z}_+^{|K(F_c)|}$ tal que $c = \sum_{\pi \in K(F_c)} y^*_\pi \pi$. Então y^* junto com $y_\pi = 0$, para todo $\pi \in K(F)$ e $F \neq F_c$, e $u = 0$ definem uma solução viável para $(\mathcal{D}_c)$. Na verdade, essa é uma solução ótima, já que, para todo $x^* \in F_c$, $c^\top x^* = \sum_{\pi \in K(F_c)} y^*_\pi \pi^\top x^* = \sum_{\pi \in K(F_c)} y^*_\pi \pi_0$, que é o valor da solução dual construída. ∎

A proposição acima garante a existência de uma descrição inteira TDI para um poliedro inteiro. Observamos, porém, que nem toda descrição inteira de um poliedro inteiro é TDI. Na verdade, para um poliedro inteiro de dimensão plena, pode-se mostrar que existe uma única descrição TDI minimal (com respeito à remoção de restrições) com coeficientes inteiros [189].

Vale mencionar que, mesmo quando $\text{lin}(P) \neq \{0\}$, vale a proposição 7.5.2 e consequentemente a proposição 7.5.3 [66]. Também continua valendo a proposição 7.5.4 [96] e a unicidade da descrição TDI minimal inteira [189].

Exercício 7.5.1. Verifique que $P = \{x \in \mathbb{R}^3 : x_1 + 2x_2 + 4x_3 \leq 4, x \geq 0\}$ é inteiro, embora o sistema que o descreve não seja TDI. Mostre, por outro lado, que $P = \{x \in \mathbb{R}^3 : x_1 + 2x_2 + 4x_3 \leq 4, x_2 + 2x_3 \leq 2, x_2 + 3x_3 \leq 3, x_3 \leq 1, x \geq 0\}$ e que essa descrição é TDI. Obtenha as novas inequações da segunda representação de P, usando os conjuntos $K(F)$, para todo F vértice de P.

7.5.3 Total unimodularidade

Embora a caracterização de poliedros inteiros por sistemas TDI seja importante, seu uso para reconhecer tais poliedros pode ser limitado. Nesta subseção, estudamos outras possibilidades para essa identificação.

Definição 7.5.3. Uma matriz inteira com posto k é dita unimodular se toda submatriz de ordem k invertível possui determinante ± 1. É dita totalmente unimodular (TU) se o determinante de toda submatriz invertível é ± 1.

Exercício 7.5.2. Uma matriz A é TU se, e somente se, $[A\ I]$ é unimodular.

Proposição 7.5.5. Seja $A \in \mathbb{Z}^{m \times n}$, com $\text{posto}(A) = m$. Então A é unimodular se, e somente se, $P_A^=(b) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ é inteiro para todo $b \in \mathbb{Z}^m$.

Prova: Primeiro admita que A é unimodular. Seja $b \in \mathbb{Z}^m$ tal que $P_A^=(b) \neq \emptyset$. Pelo corolário 7.3.4, $P_A^=(b)$ possui vértice. Tome então $\bar{x} \in \mathbb{R}^n$ vértice de $P_A^=(b)$. Pelo corolário 7.3.8, deduzimos que $\bar{x}$ é solução de um sistema $A'x = b'$, em que $A' = \begin{bmatrix} A \\ J \end{bmatrix}$ é invertível e $b' = \begin{bmatrix} b \\ 0 \end{bmatrix}$, sendo $J_{(n-m)\times n}$ submatriz de $-I$. Note que $\det(A') = \pm\det(\bar{A})$, em que $\bar{A}$ é submatriz de A. Logo, $\det(A') = \pm 1$. Como A' e b' são inteiros, pela Regra de Cramer, concluímos que $\bar{x} \in \mathbb{Z}^n$. Portanto, $P_A^=(b)$ é inteiro.

Suponha agora que $P_A^=(b)$ seja inteiro, para todo $b \in \mathbb{Z}^m$. Seja $\bar{A}$ submatriz $m \times m$ invertível de A. Sem perda de generalidade, podemos particionar $A = [\bar{A}_{m\times m}\ \hat{A}_{m\times(n-m)}]$. Vamos mostrar que $\bar{A}^{-1}$ é inteira. Considere sua i-ésima coluna $\bar{A}_i^{-1}$. Tome $z \in \mathbb{Z}^m$ tal que $z + \bar{A}_i^{-1} \geq 0$ e $b = \bar{A}z + e_i \in \mathbb{Z}^m$. Note que $\bar{x} = \begin{bmatrix} z+\bar{A}_i^{-1} \\ 0 \end{bmatrix} \in \mathbb{R}^n$ satisfaz $\begin{bmatrix} \bar{A} & \hat{A} \\ 0 & -I_{n-m} \end{bmatrix} \bar{x} = \begin{bmatrix} b \\ 0 \end{bmatrix}$. Pelo corolário 7.3.8, deduzimos que $\bar{x}$ é vértice de $P_A^=(b)$ e, por conseguinte, $\bar{x} \in \mathbb{Z}^n$. Em particular, $z + \bar{A}_i^{-1} \in \mathbb{Z}^m$, o que implica que $\bar{A}_i^{-1} \in \mathbb{Z}^m$. Mostramos assim que $\bar{A}^{-1}$ é inteira, assim como $\bar{A}$. Então os determinantes de $\bar{A}$ e $\bar{A}^{-1}$ são inteiros. Como $|\det(\bar{A})||\det(\bar{A}^{-1})| = 1$, temos que $|\det(\bar{A})| = 1$. Logo, A é unimodular. ■

Corolário 7.5.1 ([119]). Seja $A \in \mathbb{Z}^{m\times n}$. Então A é TU se, e somente se, $P_A^{\leq}(b) = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$ é inteiro para todo $b \in \mathbb{Z}^m$.

Prova: Note que x é vértice de $P_A^{\leq}(b)$ se, e somente se, $(x, b - Ax)$ é vértice de $P_{[A\ I]}^=(b) = \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^m : Ax + y = b, x \geq 0, y \geq 0\}$. Logo, $P_A^{\leq}(b)$ é inteiro se, e somente se, $P_{[A\ I]}^=(b)$ é inteiro. Como A é TU se, e somente se, $[A\ I]$ é unimodular, o resultado segue diretamente da proposição 7.5.5. ■

Certas operações sobre matrizes TU mantêm tal propriedade.

Exercício 7.5.3. As seguintes afirmações são equivalentes:

i) A é TU;
ii) $A^\top$ é TU;
iii) $\begin{bmatrix} A \\ I \end{bmatrix}$ é TU;
iv) $\begin{bmatrix} A & I \end{bmatrix}$ é TU;
v) A matriz obtida $\left\{ \begin{array}{l} \text{multiplicando-se uma linha (ou coluna) por -1} \\ \text{trocando-se duas linhas (ou colunas)} \\ \text{duplicando-se linhas (ou colunas)} \end{array} \right\}$ é TU;
vi) $\begin{bmatrix} A \\ -A \end{bmatrix}$ é TU;
vii) $\begin{bmatrix} A & -A \end{bmatrix}$ é TU.

As equivalências acima permitem outras caracterizações similares àquela do corolário 7.5.1 ou dela decorrentes.

Exercício 7.5.4. Para uma matriz $A \in \mathbb{Z}^{m\times n}$, mostre que as seguintes assertivas são equivalentes:

i) A é TU;
ii) $\{x \in \mathbb{R}^n : b' \leq Ax \leq b, d' \leq x \leq d\}$ é inteiro $\forall b', b \in \mathbb{Z}^m$, $\forall d, d' \in \mathbb{Z}^n$;
iii) $P_A^{\geq}(b) = \{x \in \mathbb{R}^n : Ax \geq b, x \geq 0\}$ é inteiro $\forall b \in \mathbb{Z}^m$;
iv) $D_A^{\geq}(c) = \{u \in \mathbb{R}^m : A^\top u \geq c, u \geq 0\}$ é inteiro $\forall c \in \mathbb{Z}^n$;
v) para todo $c \in \mathbb{Z}^n$ e $b \in \mathbb{Z}^m$, tais que os problemas $\max\{c^\top x : x \in P_A^{\leq}(b)\}$ e $\min\{b^\top u : u \in D_A^{\geq}(c)\}$ têm soluções ótimas, elas são obtidas em pontos inteiros.

Temos ainda que TU é uma condição suficiente, porém não necessária, para integralidade de $P_A^=(b)$.

Corolário 7.5.2. Se $A \in \mathbb{Z}^{m\times n}$ é uma matriz TU, então $P_A^=(b) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ é inteiro para todo $b \in \mathbb{Z}^m$.

Existem várias outras caracterizações de TU (ver, por exemplo, [192]). Aqui, destacamos a seguinte:

Teorema 7.5.1 ([94]). Para uma matriz $A \in Z^{m\times n}$, com linhas e colunas indexadas pelos conjuntos $M = \{1,\ldots,m\}$ e $N = \{1,\ldots,n\}$, respectivamente, as seguintes afirmações são equivalentes:

i) A é TU;
ii) $\forall J \subseteq N$, $\exists$ partição J_1,J_2 de J tal que $\sum_{j\in J_1} a_{ij} - \sum_{j\in J_2} a_{ij} \in \{0,\pm 1\} \quad \forall i \in M$;
iii) $\forall I \subseteq M$, $\exists$ partição I_1,I_2 de I tal que $\sum_{i\in I_1} a_{ij} - \sum_{i\in I_2} a_{ij} \in \{0,\pm 1\} \quad \forall j \in N$.

Prova: Mostramos a equivalência entre (i) e (ii). Então, como A é TU se, e somente se, $A^\top$ é TU, segue a equivalência entre (i) e (iii).
($\Rightarrow$) Seja J um subconjunto arbitrário de N. Defina $z \in \mathbb{B}^n$ tal que $z_j = 1$, se $j \in J$, e $z_j = 0$, caso contrário. Além disso, sejam $d' = 0$, $d = z$, $g = Az$, e $b,b' \in \mathbb{R}^m$ tais que $b'_i = b_i = \frac{1}{2}g_i$, se g_i é par, e $b'_i = \frac{1}{2}(g_i - 1)$, $b_i = \frac{1}{2}(g_i + 1) = b'_i + 1$, se g_i é ímpar. Agora considere o poliedro

$$P(b,b',d,d') = \{x \in \mathbb{R}^n_+ : b' \leq Ax \leq b, d' \leq x \leq d\} \subseteq \mathbb{B}^n.$$

Como A é TU, $b',b \in \mathbb{Z}^m$ e $d',d \in \mathbb{Z}^n$, temos pelo exercício 7.5.4 que P é inteiro. Mais ainda, $P \neq \emptyset$, pois $z/2 \in P$. Então existe $\bar{x} \in P \cap \mathbb{B}^n$. Logo, $\bar{x}_j = 0$, para $j \notin J$, e $\bar{x}_j \in \{0,1\}$, para $j \in J$. Além disso, para todo $i \in I$, $\sum_{j\in J} a_{ij}\bar{x}_j \in \mathbb{Z}$ é igual a $\frac{1}{2}g_i$ (se g_i é par) ou $\frac{1}{2}(g_i \pm 1)$ (se g_i é ímpar). Note que $z_j - 2\bar{x}_j = \pm 1$ para $j \in J$.

Seja $J_1 = \{j \in J : z_j - 2\bar{x}_j = 1\}$ e $J_2 = \{j \in J : z_j - 2\bar{x}_j = -1\}$. Para $i \in M$, temos

$$\sum_{j\in J_1} a_{ij} - \sum_{j\in J_2} a_{ij} = \sum_{j\in J} a_{ij}(z_j - 2\bar{x}_j) = \begin{cases} g_i - g_i = 0, & g_i \text{ é par,} \\ g_i - (g_i \pm 1) = \pm 1, & g_i \text{ é ímpar.} \end{cases}$$

($\Leftarrow$) Por (ii) para $|J| = 1$, temos que $A \in \{0,\pm 1\}^{m\times n}$. Seja B uma submatriz $k \times k$ não singular de A. Devemos mostrar que $|\det B| = 1$. A prova é por indução em k. Para $k = 1$, o resultado decorre trivialmente de $a_{ij} \in \{0,\pm 1\}$ para todo $i \in I$ e $j \in J$. Suponha que o determinante de cada submatriz invertível de A, de ordem $(k-1)$, seja ± 1.

Sejam $r = |\det B|$ e $B^* = rB^{-1}$. Pela hipótese de indução e pela Regra de Cramer, temos que $b^*_{ij} = \{0,\pm 1\}$, para todo i e j. Além disso, $BB^*_1 = re_1$, em que B^*_1 é a primeira coluna de B^*.

Seja $J = \{i : b^*_{i1} \neq 0\}$. Note que $J \neq \emptyset$, pois B^* é invertível. Além disso, para todo $i \in \{2,\ldots,k\}$, temos

$$(BB^*_1)_i = \sum_{j\in J: b^*_{i1}=1} b_{ij} - \sum_{j\in J: b^*_{i1}=-1} b_{ij} = 0,$$

implicando que $|j \in J : b_{ij} \neq 0|$ seja par (lembre que $b_{ij} \in \{0,\pm 1\}$). Logo, para qualquer partição (J_1,J_2) de J, segue que $\sum_{j\in J_1} b_{ij} - \sum_{j\in J_2} b_{ij}$ é par para todo $i \in \{2,\ldots,k\}$.

Por hipótese, existe uma partição $(\bar{J}_1,\bar{J}_2)$ de J tal que $|\sum_{j\in\bar{J}_1} b_{ij} - \sum_{j\in\bar{J}_2} b_{ij}| \leq 1$, para todo $i \in \{1,\ldots,k\}$. Assim,

$$\sum_{j\in\bar{J}_1} b_{ij} - \sum_{j\in\bar{J}_2} b_{ij} = 0, \quad \text{para todo } i \in \{2,\ldots,k\}.$$

Agora considere o valor de $\alpha_1 = |\sum_{j\in\bar{J}_1} b_{1j} - \sum_{j\in\bar{J}_2} b_{1j}| \leq 1$. Defina $y \in \mathbb{R}^k$ tal que $y_i = 1$, se $i \in \bar{J}_1$; $y_i = -1$, se $i \in \bar{J}_2$; e $y_i = 0$, caso contrário. Então $By = \pm\alpha_1 e_1$. Como B é não

singular e $y \neq 0$ (pois $J \neq \emptyset$), temos que $\alpha_1 = 1$ e, consequentemente, $By = \pm e_1$. Logo, $BB_1^* = re_1 = \pm rBy$, ou seja, $B(B_1^* \pm ry) = 0$. Sendo B invertível, temos $B_1^* \pm ry$. Como y e B_1^* são vetores com entradas $\{0, \pm 1\}$, segue que $r = 1$. ∎

Corolário 7.5.3. Seja A uma matriz com elementos $\{0, \pm 1\}$ e não mais que dois elementos não nulos por coluna. Então A é TU se, e somente se, as linhas de A podem ser particionadas em dois subconjuntos I_1 e I_2 tais que, para toda coluna com dois elementos não nulos, acontece o seguinte:

1. Se ambos os elementos possuem o mesmo sinal, um é indexado por I_1 e o outro por I_2;
2. Se possuem sinais diferentes, ambos são indexados por I_1 ou ambos por I_2.

Prova: Suponha que A seja TU. Seja $I = M$ e suponha que (I_1, I_2) satisfaça o teorema 7.5.1(iii). Então claramente (I_1, I_2) satisfaz (1) e (2). Agora, suponha que I_1 e I_2 satisfaçam (1) e (2). Tome $I \subseteq M$. Defina $I_1' = I \cap I_1$ e $I_2' = I \cap I_2$. Então I_1' e I_2' satisfazem o teorema 7.5.1(iii). ∎

Vale destacar que o reconhecimento de matrizes TU pode ser feito em tempo polinomial, usando-se decomposição de matroides [194, 52]. O algoritmo de menor complexidade atualmente conhecido foi proposto em [203], e uma versão simplificada dele foi apresentada mais recentemente [210].

Exercício 7.5.5. Seja A uma matriz binária em que, para cada linha, os elementos não nulos aparecem consecutivamente em um único bloco de colunas, isto é, para cada linha i, existe intervalo J_i tal que $a_{ij} = 1$ se, e somente se, $j \in J_i$. Mostre que A é TU.

Exercício 7.5.6. Considere o problema de fluxo de custo mínimo. Sejam $G = (V, A)$ um grafo direcionado, $c : A \to \mathbb{R}$ uma função que define o custo unitário de fluxo em cada arco, $w : A \to \mathbb{Z}$ uma função que define a capacidade de cada arco e $b : V \to \mathbb{Z}$ uma função que marca a oferta ou demanda de cada nó ($b(v) > 0$ é a oferta de v; $-b(v) > 0$ é a demanda de v), tal que $\sum_{v \in V} b(v) = 0$. O problema, que consiste em escoar pela rede G um fluxo de custo mínimo, dos nós de oferta aos nós de demanda, pode ser modelado como:

$$\begin{array}{ll} \min & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{s.a:} & \sum_{j:(i,j) \in A} x_{ij} - \sum_{k:(k,i) \in A} x_{ki} = b_i \quad \forall i \in V, \\ & 0 \leq x_{ij} \leq w_{ij} \quad \forall (i,j) \in A. \end{array}$$

Mostre que o problema tem solução inteira.

7.5.4 Reconhecimento de poliedros inteiros

Pelos resultados deste capítulo, podemos mostrar que um poliedro não vazio $P(A, b) \subseteq \mathbb{R}_+^n$, com $A \in \mathbb{Z}^{m \times n}$ e $b \in \mathbb{Z}^m$, é inteiro por uma das seguintes alternativas:

I1 Todo vértice de P é inteiro ou, equivalentemente, todo ponto não inteiro de P é combinação convexa estrita de outros pontos de P;

I2 Para todo $c \in \mathbb{Z}^n$, $\max\{c^\top x : x \in P\}$ é ilimitado ou tem solução de valor inteiro;

I3 P tem representação TDI com lado direito inteiro, ou seja, $P(A, b) = P(\hat{A}, \hat{b})$, em que $\hat{A} \in \mathbb{Z}^{\hat{m} \times n}$, $\hat{b} \in \mathbb{Z}^{\hat{m}}$ e, para todo $c \in \mathbb{Z}^n$, $\min\{\hat{b}^\top u : \hat{A}^\top u \geq c, u \geq 0\}$ tem solução inteira ou é inviável;

I4 A é TU;

I5 $P = \text{conv}(P \cap \mathbb{Z}^n)$ - Se (π, π_0) induz faceta de $\text{conv}(P \cap \mathbb{Z}^n)$, então (π, π_0) é equivalente a uma das desigualdades (a^i, b_i);

I6 $P = \text{conv}(P \cap \mathbb{Z}^n)$ - Para todo $c \in \mathbb{R}^n$, $c \neq 0$, $\max\{c^\top x : x \in \text{conv}(P \cap \mathbb{Z}^n)\}$ é ilimitado ou uma das desigualdades (a^i, b_i) é satisfeita na igualdade por todas as suas soluções ótimas.

Exercício 7.5.7. Mostre que de fato (I6) equivale a $P = \text{conv}(P \cap \mathbb{Z}^n)$.

Ilustramos as abordagens acima com o seguinte exemplo. Seja $G = (V, E)$ um ciclo par com $V = \{1, 2, \ldots, n\}$ e $E = \{e_i = (i, i+1) : i \in \{1, 2, \ldots, n-1\}\} \cup \{e_n = (n, 1)\}$. Defina

$$P(G) = \{x \in \mathbb{R}^n_+ : x_i + x_j \leq 1 \, \forall ij \in E\}. \tag{7.11}$$

Note que $P(G) \cap \mathbb{Z}^n = P(G) \cap \mathbb{B}^n$ é o conjunto de vetores de incidência dos conjuntos independentes de G.

I1 Seja $x \in P(G) \setminus \mathbb{B}^n$. Sejam $U = \{i \in V : x_i \in (0,1)\} \neq \emptyset$ e $\varepsilon = \min_{i \in U} \min\{x_i, 1 - x_i\} > 0$. Defina x^0 e x^1 tais que: $x^0_i = x^1_i = x_i$, para todo $i \in V \setminus U$; $x^0_i = x_i - (-1)^i \varepsilon$, $x^1_i = x_i + (-1)^i \varepsilon$, se $i \in U$. Pela definição de ε, temos que $0 \leq x^0, x^1 \leq 1$. Seja $(i, j) \in E$. Como n é par, i e j têm paridades diferentes. Logo, se ambos i e j pertencem a U ou a $V \setminus U$, temos $x^0_i + x^0_j = x^1_i + x^1_j = x_i + x_j \leq 1$. Suponha agora $i \in U$ e $j \in V \setminus U$. Como $x_i > 0$, $x_j \in \{0, 1\}$ e $x_i + x_j \leq 1$, temos $x_j = 0$. Portanto, $x^k_i + x^k_j = x_i \pm \varepsilon \leq 1$, para $k = 0, 1$. Consequentemente, temos que $x^0, x^1 \in P(G)$. Finalmente, observe que $x = (1/2)x^0 + (1/2)x^1$, mostrando que x não é vértice.

I2 Sejam $c \in \mathbb{Z}^n$ e x solução ótima de $\max\{c^\top x : x \in P\}$. Se $x \in \mathbb{B}^n$, trivialmente temos o resultado desejado. Admita então que $x \notin \mathbb{B}^n$. Como em (I1), construa x^0 e x^1, ambos pontos de $P(G)$. Observe que $0 \geq c^\top(x^0 - x) = -c^\top(x^1 - x) \geq 0$. Logo, x^0 e x^1 também são soluções ótimas. Adicionalmente, se $\varepsilon = \min\{x_\ell, 1 - x_\ell\}$, então a componente ℓ de x^0 torna-se 0 ou 1 ou a componente ℓ de x^1 torna-se 0 ou 1. Em qualquer caso, obtemos uma outra solução ótima com pelo menos uma componente inteira a mais que x. Após um número finito de passos, obtemos uma solução ótima inteira, como desejado.

I3 Associando a variável w_i à restrição relativa a e_i, obtemos o problema dual

$$\min \textstyle\sum_{i=1}^n w_i \quad \text{s.a} \quad w_i + w_j \geq c_j \, \forall (i, j) \in L, \, w \geq 0,$$

onde $L = \{(i, i+1) : i \in \{1, 2, \ldots, n-1\}\} \cup \{(n, 1)\}$. Sejam w^* uma solução ótima e $U = \{i \in \{1, \ldots, n\} : w^*_i \notin \mathbb{Z}\}$. Se $U = \emptyset$, o resultado segue. Do contrário, seja $\varepsilon_1 = \min\{w^*_i : i \in U\} > 0$ e $\varepsilon_2 = \min\{w^*_i + w^*_j - c_j : |\{i, j\} \cap U| = 1\}$. Note que $\varepsilon_2 > 0$, pois w^* é viável, $c_j \in \mathbb{Z}$ e exatamente um entre w^*_i e w^*_j não é inteiro. Tome $\varepsilon = \min\{\varepsilon_1, \varepsilon_2\} > 0$. Considere um intervalo maximal I contido U. Pela simetria do ciclo, podemos admitir que $I = \{1, 2, \ldots, p\}$, $p \geq 1$. Defina $\hat{w}$ tal que: $\hat{w}_i = w^*_i$, se $i \notin I$; $\hat{w}_i = w^*_{e_i} + (-1)^i \varepsilon$, se $i \in I$. Temos que $\hat{w}$ é dual viável (verifique!). Além disso, $\sum_{i=1}^n (\hat{w}_{e_i} - w^*_{e_i})$ é igual a 0 (se p é par) ou $-\varepsilon$ (se p é ímpar). Logo, p é par e $\hat{w}$ é igualmente solução ótima. Porém, $\hat{w}$ tem pelo menos uma componente inteira a mais que w^*. Repetindo o mesmo processo a partir de $\hat{w}$ um número finito de vezes, obtemos uma solução ótima inteira para o dual.

I4 Note que, na matriz $A \in \mathbb{B}^{n \times n}$ que define $P(G) = \{x \in \mathbb{R}^n_+ : Ax \leq 1\}$, a linha i associa-se à aresta e_i de G, para todo $i \in \{1, 2 \ldots, n\}$, enquanto a coluna j refere-se ao vértice j, para todo $j \in \{1, 2, \ldots, n\}$. Há exatamente dois elementos não nulos, iguais a 1, em cada linha i, nas colunas associadas aos vértices que definem e_i. Defina a partição J_0, J_1 dos índices das colunas, compreendendo os índices pares e ímpares, respectivamente. Como cada aresta e_i é definida por vértices com paridades diferentes (pois o ciclo é par), temos que, em cada linha, um elemento 1 fica em uma coluna de J_0 e o outro posiciona-se em uma coluna de J_1. Pelo corolário 7.5.3, A é TU.

I5 Seja (π,π_0) uma desigualdade indutora de uma faceta F de $\text{conv}(P(G)\cap\mathbb{Z}^n)$. Consideramos três casos. Primeiro, suponha que exista $i\in V$ tal que $\pi_i<0$. Queremos mostrar que (π,π_0) é equivalente a $-x_i\le 0$. Suponha que não. Então, existe $x\in P(G)$ tal que $\pi^\top x=\pi_0$ e $x_i=1$. O ponto x' obtido de x anulando essa componente também está em $P(G)$. Logo, $\pi^\top(x'-x)\le 0$, ou seja, $-\pi_i\le 0$: um absurdo, pois $\pi_i<0$. Logo, nesse caso F é induzida por $x_i\ge 0$. Agora, suponha que $\pi_i=\varepsilon\ge 0$, para todo $i\in\{1,2\ldots,n\}$. Os pontos x^1 e x^0, que são os vetores de incidência dos conjuntos independentes $\{1,3,5,\ldots,n-1\}$ e $\{2,4,6,\ldots,n\}$, estão em $P(G)$. Logo, $n\varepsilon=\sum_{i=1}^n\pi_i=\pi^\top x^0+\pi^\top x^1\le 2\pi_0$, ou melhor, $n\varepsilon/2\le\pi_0$. Sendo assim, a desigualdade $\sum_{i=1}^n\varepsilon x_i\le(n\varepsilon)/2$, que resulta da combinação cônica das desigualdades $x_i+x_j\le 1$, cada uma delas ponderada por $\varepsilon/2$, domina (π,π_0). Finalmente, sem perda de generalidade, podemos supor que π_1 seja uma entrada de maior valor em π e $\pi_1>\pi_2\ge 0$. Queremos mostrar que todo ponto inteiro x da faceta satisfaz $x_n+x_1=1$. Seja $\bar{x}\in P(G)\cap\mathbb{Z}^n$ tal que $\bar{x}_n+\bar{x}_1<1$, ou seja, $\bar{x}_n=\bar{x}_1=0$. Defina $\hat{x}=\bar{x}+e_1-\bar{x}_2e_2$, isto é, $\hat{x}$ descreve o conjunto independente obtido de $\bar{x}$ incluindo o vértice 1 e retirando o 2, caso este vértice esteja no conjunto original. Temos que $\pi^\top(\hat{x}-\bar{x})=\pi^\top(e_1-\bar{x}_2e_2)=\pi_1-\bar{x}_2\pi_2\ge\pi_1-\pi_2>0$. Logo, $\pi^\top\bar{x}<\pi^\top\hat{x}\le\pi_0$, mostrando que $\bar{x}$ não está na faceta. Portanto, todo ponto inteiro x da faceta satisfaz $x_n+x_1=1$. Isso mostra que (π,π_0) define a mesma faceta que $x_1+x_n\le 1$.

I6 Dado $\pi\in\mathbb{R}^n$, $\pi\ne 0$, suponha que o problema $\max\{\pi^\top x : x\in\text{conv}(P(G)\cap\mathbb{Z}^n)\}$ tenha solução ótima de valor π_0. Logo, $\pi^\top x\le\pi_0$ é desigualdade válida para $\text{conv}(P(G)\cap\mathbb{Z}^n)$, satisfeita na igualdade em toda solução ótima. Assim como em (I5), concluímos que $\sum_{i=1}^n\pi_i\le 2\pi_0$.

Consideramos três casos. Se existe i tal que $\pi_i<0$, toda solução ótima $\bar{x}$ satisfaz $\bar{x}_i=0$. Se $\pi_i=\varepsilon>0$, para todo $i\in\{1,2,\ldots,n\}$, então $\pi_0=n\varepsilon/2$. Consequentemente, em toda solução ótima $\bar{x}$ ocorre $\bar{x}_i+\bar{x}_j=1$, para todo $ij\in E$. Do contrário, teríamos $2\pi_0=\sum_{ij\in E}\varepsilon(\bar{x}_i+\bar{x}_j)<n\varepsilon$. No último caso, sem perda de generalidade, podemos considerar que π_1 é uma entrada de maior valor em π e $\pi_1>\pi_2$. Seja $\bar{x}\in\text{conv}(P(G)\cap\mathbb{Z}^n)$ tal que $\bar{x}_n+\bar{x}_1<1$. Para concluir, resta mostrar que $\bar{x}$ não é ótima. Temos que $\bar{x}=\sum_{i=1}^p\lambda_i x_i$, onde $\sum_{i=1}^p\lambda_i=1$, $\lambda_i>0$ e $x^i\in P(G)\cap\mathbb{Z}^n$, para todo $i\in\{1,2,\ldots,p\}$. Como $\bar{x}_n+\bar{x}_1<1$ e $x^i_n+x^i_1\le 1$, para todo $i\in\{1,2,\ldots,p\}$, deve existir $k\in\{1,\ldots,p\}$ tal que $x^k_n+x^k_1<1$, ou seja, $x^k_n=x^k_1=0$. Seja $y=x^k+e_1-x^k_2e_2$, isto é, y descreve o conjunto independente obtido de x^k incluindo o vértice 1 e retirando o 2, caso este vértice esteja no conjunto original. Logo, $\hat{x}=\sum_{i\ne k}\lambda_i x^i+\lambda_k y\in\text{conv}(P(G)\cap\mathbb{Z}^n)$. Mais ainda, $\pi^\top(\hat{x}-\bar{x})=\lambda_k\pi^\top(e_1-x^k_2e_2)=\lambda_k(\pi_1-x^k_2\pi_2)\ge\lambda_k(\pi_1-\pi_2)>0$. Logo, $\bar{x}$ não é ótima.

Exercício 7.5.8. Considere agora que G é um ciclo ímpar. Mostre que $P(G)$, como definido em (7.11), não é inteiro, porém $\{x\in P(G):\sum_{i=1}^n x_i\le\lfloor n/2\rfloor\}$ é inteiro.

7.6 Método de Planos de Corte

Muitas vezes as desigualdades que descrevem o fecho inteiro de um poliedro racional não são todas conhecidas ou são em grande número. Dessa forma, embora o problema (7.1) possa ser, em teoria, reformulado com um PPL, essa abordagem direta pode ser inviável na prática. Uma alternativa é a aplicação do método de planos de corte, introduzido em [104].

Dados um poliedro racional $P(A,b)\subseteq\mathbb{R}^n$ e um vetor $c\in\mathbb{Q}^n$, uma estratégia para encontrar uma solução ótima de $(\mathcal{P})\ \max\{c^\top x : x\in P\cap\mathbb{Z}^n\}$ ou pelo menos um bom limitante superior consiste em resolver uma sequência de problemas lineares

$$(\mathcal{P}_k)\ \max\{c^\top x : x\in P_k\},\quad k\in\{0,1,2,\ldots\},$$

onde

$$P = P_0 \supset P_1 \supset P_2 \supset \cdots \supseteq P_l$$

e P_{k+1} não contém uma solução ótima x^k (vértice de P_k) obtida para o problema $(\mathcal{P}_k)$, caso fracionária. Por outro lado, se $x^k \in \mathbb{Z}^n$, então x^k é uma solução do problema $(\mathcal{P})$, e o processo se encerra. Em toda iteração, $c^\top x^k$ é um limite superior para o valor ótimo de $(\mathcal{P})$.

Para a definição de P_{k+1}, devemos resolver o chamado problema de separação (entre x^k e P_l), que consiste em encontrar uma ou mais desigualdades (π, π_0), válidas para P_l, tal que $\pi^\top x^k > \pi_0$. Tais desigualdades são chamadas planos de corte. Sua inclusão em P_k "corta"o ponto x^k, definindo assim um novo poliedro mais restrito (ver figura 7.18). O ideal seria que esse plano de corte definisse uma faceta de P_l, o que levaria a uma aceleração na convergência do método.

Exercício 7.6.1. Seja novamente P o poliedro usado no exercício 7.4.5, em que G é uma roda ímpar. Considere o problema

$$(\mathcal{P}) \max\{\sum_{i=0}^{n} x_i : x \in P \cap \mathbb{B}^{n+1}\}.$$

Usando a abordagem de planos de corte, começamos resolvendo $(\mathcal{P}_0)$, cuja solução x^0 está representada na figura 7.19(a) (o número ao lado de um vértice i é o valor da variável x_i). Pelo exercício 7.4.3, temos que $x_0 + x_i + x_j \leq 1$ define faceta de P_l, para todo $i = 1, \ldots, n$ e $j = i+1$, ou $i = n$ e $j = 1$. Essas são as desigualdades induzidas pelas cliques maximais de G, todas elas violadas por x^0. Definindo então $P_1 = \{x \in P_0 : x_0 + x_1 + x_2 \leq 1\}$, obtemos a solução x^1 exibida na figura 7.19(b). Novamente, algumas desigualdades de clique estão violadas, em particular $x_0 + x_2 + x_3 \leq 1$. Acrescentando-a como plano de corte, obtemos o politopo P_2 e a solução associada x^2, descrita pela figura 7.19(c). Agora todas as desigualdades de clique estão satisfeitas. Porém, a desigualdade $\lfloor n/2 \rfloor x_0 + \sum_{i=1}^{n} x_i \leq \lfloor n/2 \rfloor$, que também define faceta, conforme exercício 7.4.5, ainda está violada. Podemos definir P_3 a partir desse plano de corte, levando à solução x^3, apresentada na Figura 7.19(d), que é inteira e, portanto, ótima para o problema $(\mathcal{P})$.

Caso essa última desigualdade tivesse sido a primeira a ser acrescida, a solução obtida já seria inteira, fazendo com que a geração das restrições de clique não fosse necessária. Isso, porém, poderia não acontecer com outra função-objetivo.

Sendo x^k vértice de P_k, várias fórmulas podem ser empregadas para determinar planos de corte, entre as quais podemos citar cortes de Gomory-Chvátal [104], cortes disjuntivos [9, 8], cortes lift-and-project [10] (ver [51] para um survey). Porém, todos esses cortes podem não ser suficientemente fortes ou, em outras palavras, podem determinar faces de baixa dimensão de P_l, o que pode acarretar um número excessivo de iterações do método.

Quando se conhecem desigualdades indutoras de faces de alta dimensão P_l, especialmente facetas, podemos considerar o problema de separação restrito a elas: dado um ponto $\bar{x} \notin P_l$ e uma família $\mathcal{F}$ de desigualdades válidas para P_l (notadamente indutoras de facetas), desejamos encontrar $(\pi, \pi_0) \in \mathcal{F}$ tal que $\pi^\top \bar{x} > \pi_0$, ou mostrar que $\pi^\top \bar{x} \leq \pi_0$, para todo $(\pi, \pi_0) \in \mathcal{F}$.

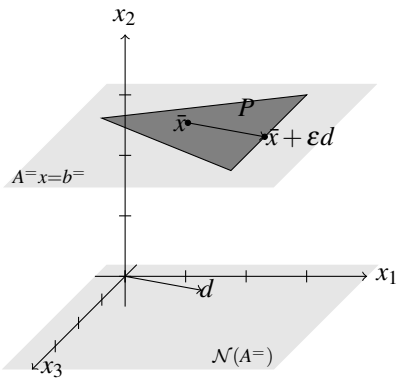
O ponto crucial e, portanto, desafiador do método de planos de corte é o problema de separação. Muitas vezes, tal problema, mesmo restrito a uma classe de facetas, é NP-difícil, o que justifica o desenvolvimento de heurísticas de separação. Os algoritmos de separação são, em geral, específicos para cada problema.

Exercício 7.6.2 (cont. exercício 7.4.3). Considere o problema de separação para a classe de desigualdades dada por $\sum_{v \in K} x_v \leq 1$, para toda clique maximal K de $G = (V, E)$. Dado um ponto $\bar{x} \in \mathbb{R}^{|V|}$, desejamos saber se alguma dessas desigualdades é violada

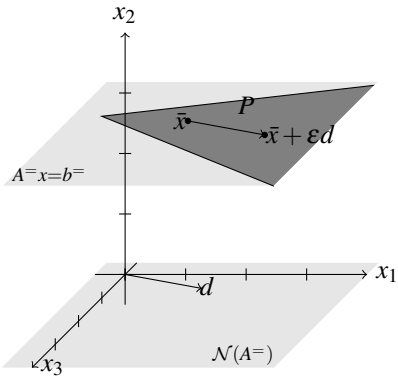
por $\bar{x}$. Mostre que esse problema corresponde a encontrar uma clique de peso máximo em G, em que o peso de cada vértice v é $\bar{x}_v$. Portanto, esse problema de separação é NP-difícil [126].

Exercício 7.6.3. Para o mesmo politopo do exercício 7.6.2, considere agora a classe de desigualdades dada por $\sum_{v \in V_C} x_v \leq (|V_C|-1)/2$, para todo ciclo ímpar $C = (V_C, E_C)$ de $G = (V, E)$. Tais desigualdades induzem facetas sob certas condições [172]. Verifique que elas podem ser reescritas como $\sum_{uv \in E_C}(1 - x_u - x_v) \geq 1$. Assim, o problema de separação equivale a encontrar um ciclo ímpar de peso mínimo em G, em que o peso de cada aresta $uv \in E$ é $w_{uv} = 1 - x_u - x_v$. Considere o grafo bipartido $G' = (V \cup V', E')$, onde $V' = \{v' : v \in V\}$ e $E' = \{u'v, uv' : uv \in E\}$, com peso nas arestas $w_{u'v} = w_{uv'} = w_{uv}$. Mostre que encontrar um ciclo ímpar de peso mínimo em G, contendo um vértice $u \in V$, equivale a encontrar um caminho mínimo em G' entre u e u'. Usando esse fato, argumente que o problema de separação para essa classe de facetas é polinomial.

Agradecimento O autor gostaria de agradecer a Phablo Moura, Jefferson Gurguri, Sérgio de Freitas Filho e alunos da disciplina Combinatória Poliédrica, 2017, do MDCC/UFC, pela cuidadosa leitura desse material e valiosas sugestões.



(a) $Ad \not\leq 0$.



(b) $Ad \leq 0$.

Figura 7.11: Ilustração do lema 7.3.1.

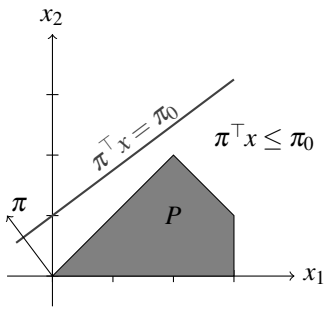
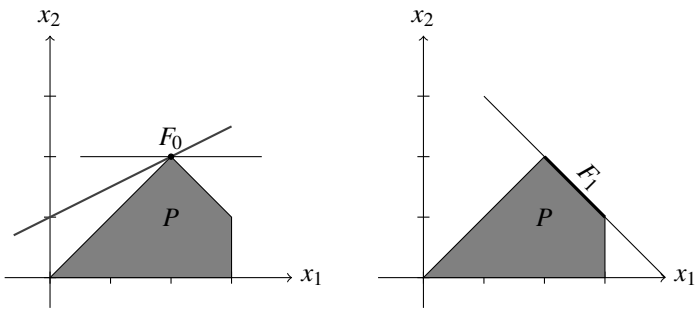


Figura 7.12: Desigualdade válida.



(a) Face de dimensão zero, F_0. (b) Face de dimensão um, F_1.

Figura 7.13: Faces de um poliedro.

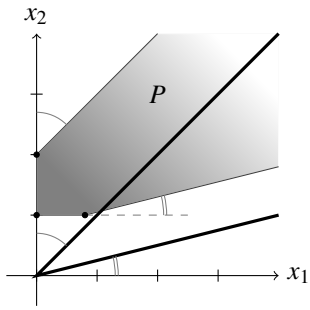
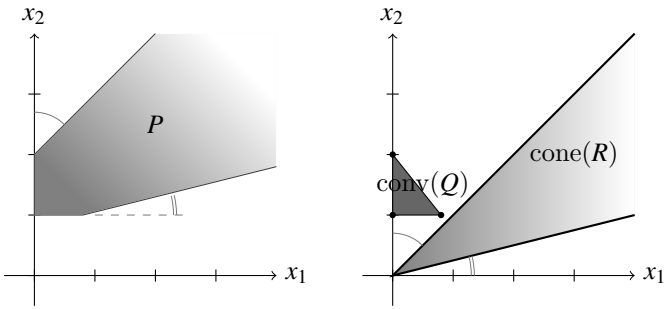


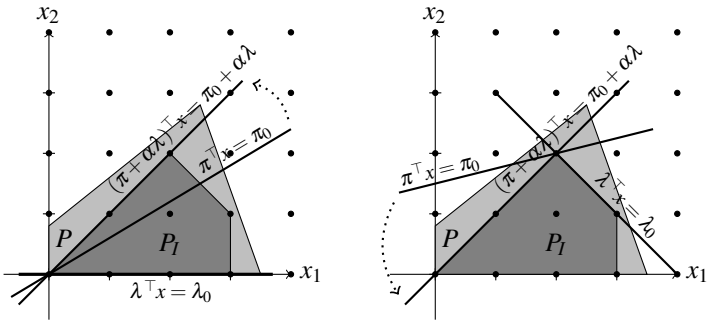
Figura 7.14: Vértices e direções extremas.



(a) Poliedro P, com vértices Q e direções extremas R.

(b) $P = \text{conv}(Q) + \text{cone}(R)$.

Figura 7.15: Ilustração do Teorema de Minkowski.



(a) (π, π_0) não é válida para P_I.

(b) (π, π_0) é válida para P_I.

Figura 7.16: Lifting.

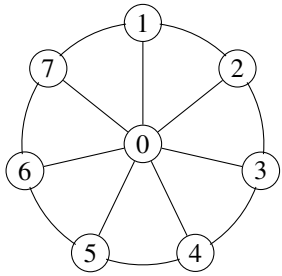
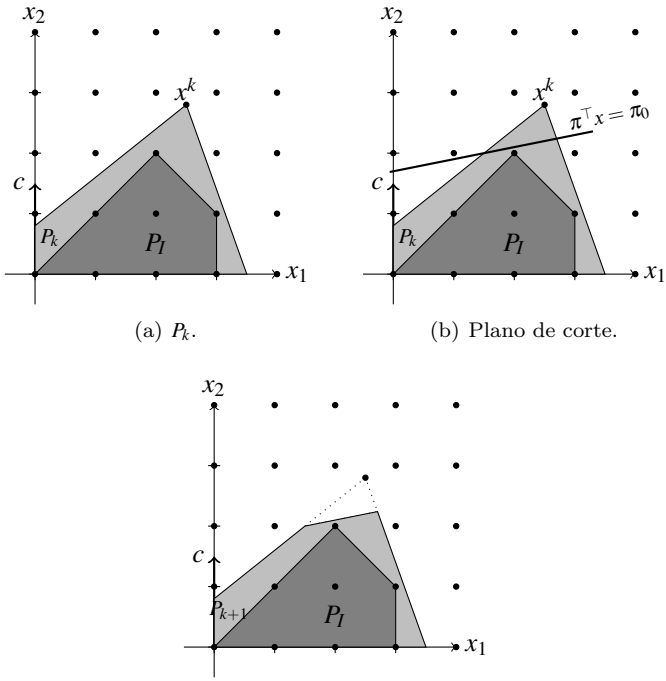


Figura 7.17: Roda ímpar com sete vértices periféricos e um central.

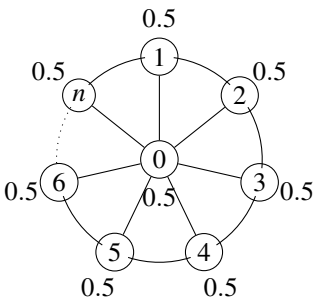


(a) P_k.

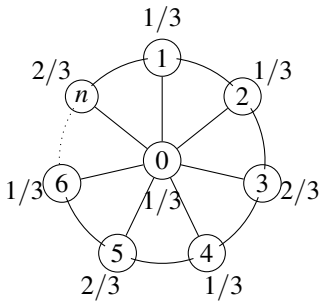
(b) Plano de corte.

(c) P_{k+1}.

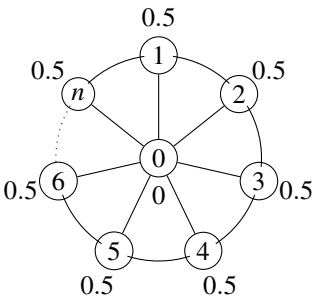
Figura 7.18: Separação e plano de corte.



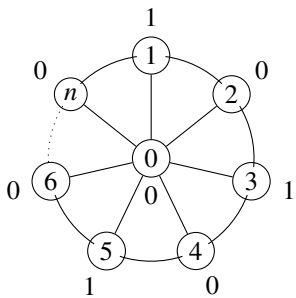
(a) Solução x^0 para $P_0 = P$, com valor $(n+1)/2$.



(b) Solução x^1 para $P_1 = \{x \in P_0 : x_0 + x_1 + x_2 \leq 1\}$, com valor $(3n+1)/6$.



(c) Solução x^2 para $P_2 = \{x \in P_1 : x_0 + x_2 + x_3 \leq 1\}$, com valor $n/2$.



(d) Solução x^3 para $P_3 = \{x \in P_2 : \lfloor n/2 \rfloor x_0 + \sum_{i=1}^{n} x_i \leq \lfloor n/2 \rfloor\}$, com valor $(n-1)/2$.

Figura 7.19: Resolução por planos de corte.

8

Otimização Não Linear Inteira Mista

Wendel Melo
Marcia Fampa
Fernanda Maria Pereira Raupp

8.1 Introdução

Este capítulo aborda a resolução de problemas convexos de programação não linear inteira mista (PNLIM), que são problemas de otimização que englobam simultaneamente termos não lineares e variáveis em domínio discreto. Algebricamente, esses problemas podem ser descritos da seguinte forma:

$$\begin{array}{lll} (P) & \min\limits_{x,y} & f(x,y) \\ & \text{s. a} & g(x,y) \leq 0, \\ & & x \in X,\ y \in Y \cap \mathbb{Z}^{n_y}, \end{array} \tag{8.1}$$

em que X e Y são subconjuntos poliédricos de $\mathbb{R}^{n_x}$ e $\mathbb{R}^{n_y}$, respectivamente, e Y é limitado. Assumimos que $f : \mathbb{R}^{n_x+n_y} \to \mathbb{R}$ e $g : \mathbb{R}^{n_x+n_y} \to \mathbb{R}^m$ são funções convexas e duplamente diferenciáveis. Por consequência da convexidade de suas funções, (P) é classificado como problema de PNLIM convexo. Assumimos aqui também que o problema (P) possui solução ótima.

Diversas aplicações envolvendo modelos baseados em PNLIM têm aparecido na literatura [141]. Em tempos recentes, um número significativo de algoritmos tem sido proposto para resolver o problema (P), valendo-se de diferentes estratégias de resolução. Nós classificamos aqui esses algoritmos nas seguintes categorias:

- Algoritmos de branch-and-bound não linear: De modo geral, esta classe reúne, no contexto não linear, algoritmos com base na metodologia de particionamento de espaço conhecida como branch-and-bound (BB), já amplamente difundida no universo de programação linear inteira mista (PLIM). Trabalhos

sobre BB não linear podem ser encontrados em [112, 33, 140, 198, 199, 30].

- Algoritmos de aproximação linear: Este tipo de algoritmo resolve um problema de PNLIM convexo aproximando-o por uma sequência de problemas de PLIM, cujas resoluções fornecem limitantes inferiores (no caso de minimização) para o problema abordado. Algoritmos dessa classe podem ser encontrados em [65, 79, 92, 181, 212, 135].

- Algoritmos de subproblemas contínuos: Algoritmos desta categoria são caracterizados pela resolução de uma sequência de problemas em domínio contínuo. Diferentemente dos algoritmos de BB, esta classe não faz particionamento explícito do espaço em seu processo de convergência. Nesta categoria, estão alguns algoritmos propostos para casos especiais de (P) que envolvem, dentre outras características, a presença de variáveis inteiras restritas aos valores $\{0, 1\}$ [142, 168, 164, 162].

- Abordagens híbridas: Os algoritmos desta categoria procuram combinar características presentes nos algoritmos das categorias anteriores. O objetivo principal é aproveitar as principais vantagens das abordagens mescladas e, de forma simultânea, remediar seus pontos fracos. Como exemplos, temos [29, 165].

Além das categorias acima, também se observa a existência de algoritmos baseados em aproximação quadrática, como, por exemplo, abordagens baseadas em programação quadrática sequencial [72, 71]. Todavia, segundo nosso melhor conhecimento, não existe ainda demonstração de convergência para esses algoritmos, o que os classificaria então como heurísticas de viabilidade (feasibility heuristics), que são algoritmos voltados para obtenção de (boas) soluções viáveis para o problema abordado. É importante mencionar ainda que em [79] é sugerida também a adoção de uma aproximação quadrática de (P) no algoritmo Aproximação Externa (a ser apresentado na seção 8.3.3), sem o provimento de qualquer resultado computacional sobre essa metodologia. A notada carência de mais trabalhos na literatura nessa direção indica ser este caminho não muito promissor até o momento. Uma possível razão para o fato de a aproximação linear vir sendo preferida em relação à quadrática é que a aproximação linear por meio da série de Taylor fornece, de modo natural, um esquema funcional para obtenção de limitantes inferiores para (P), uma vez que ela se constitui em uma relaxação de (P). Esses limitantes inferiores são fundamentais para fechar o chamado gap de otimalidade e atestar que uma determinada solução viável do problema abordado é ótima. A aproximação quadrática de (P) por meio da série de Taylor, por sua vez, não se configura em uma relaxação deste, o que não permite a obtenção de limitantes inferiores e, por consequência, dificulta a certificação de que uma determinada solução viável é ótima.

8.2 Branch-and-bound baseado em PNL

Os algoritmos de branch-and-bound (BB), juntamente com seus derivados, como branch-and-cut e branch-and-price, se constituem numa técnica extremamente fundamental na área de programação inteira, seja no âmbito linear ou no não linear. Esses algoritmos se baseiam no particionamento do espaço das variáveis inteiras do problema abordado. No nosso contexto, definimos como $(P_{\bar{Y}})$ o subproblema de (P) referente a uma partição $\bar{Y} \subseteq Y$:

$$
\begin{array}{llll}
(P_{\bar{Y}}) & \min\limits_{x,y} & f(x,y) & \\
 & \text{s. a} & g(x,y) \leq 0, & \\
 & & x \in X,\ y \in \bar{Y} \cap \mathbb{Z}^{n_y}. &
\end{array}
\tag{8.2}
$$

Definimos então a relaxação contínua de $(P_{\bar{Y}})$, isto é, o problema derivado deste em que as restrições de integralidade são relaxadas, como $(\tilde{P}_{\bar{Y}})$:

$$
\begin{array}{llll}
(\tilde{P}_{\bar{Y}}) & \min\limits_{x,y} & f(x,y) & \\
 & \text{s. a} & g(x,y) \leq 0, & \\
 & & x \in X,\ y \in \bar{Y}. &
\end{array}
\tag{8.3}
$$

Algoritmo 8.1: Branch-and-bound básico.

Entrada: (P): Problema de PNLIM abordado, ε_c: tolerância de convergência.
Saída: (x^*, y^*): solução ótima de (P).

```
z^u = +∞;
Y^0 = Y;
Seja N = {0} a lista inicial de nós em aberto;
i = 0;
Seja l_b^i o limitante inferior do nó i;
l_b^0 = −∞;
enquanto N ≠ ∅ faça
    Selecione um nó k de N;
    se (P̃_{Y^k}) é inviável então
        N = N\{k};                                   // Poda por inviabilidade
    senão
        Seja (x̃^k, ỹ^k) uma solução ótima de (P̃_{Y^k});
        se f(x̃^k, ỹ^k) < z^u − ε_c então
            se ỹ^k for inteira então
                z^u = f(x̃^k, ỹ^k);
                (x*, y*) = (x̃^k, ỹ^k);
                N = N\{k};                           // Poda por otimalidade
                N = N\{j : l_b^j ≥ z^u − ε_c};       // Podas por limite
            senão
                // Ramificação
                Selecione uma variável y_j com valor ỹ_j^k não inteiro;
                Y^{i+1} = Y^k ∩ {y ∈ ℝ^{n_y} : y_j ≤ ⌊ỹ_j^k⌋};
                Y^{i+2} = Y^k ∩ {y ∈ ℝ^{n_y} : y_j ≥ ⌈ỹ_j^k⌉};
                l_b^{i+1} = l_b^{i+2} = f(x̃^k, ỹ^k);
                N = N ∪ {i+1, i+2}\{k};
                i = i+2 ;
        senão
            N = N\{k} ;                              // Poda por limite
```

O algoritmo básico de BB é apresentado aqui como algoritmo 8.1. Note que, no algoritmo 8.1, uma árvore de busca é percorrida, na qual, a cada nó k explorado, a

relaxação contínua $(\tilde{P}_{Y^k})$ do subproblema na partição Y^k é resolvida. Assim, sejam z^u o melhor limitante superior corrente e $(\bar{x}^k, \bar{y}^k)$ uma solução ótima de $(\tilde{P}_{Y^k})$, caso este seja viável. Temos então quatro possibilidades distintas a considerar:

1. O problema $(\tilde{P}_{Y^k})$ é inviável: neste caso, podemos descartar todas as subpartições de Y^k (subárvore abaixo do nó atual), pois também serão inviáveis (poda por inviabilidade, linha 10 do algoritmo 8.1).
2. $f(\bar{x}^k, \bar{y}^k) \geq z^u$: neste caso, uma vez que o valor $f(\bar{x}^k, \bar{y}^k)$ é um limitante inferior para (P) em toda subpartição de Y^k, temos a certeza de que nenhuma delas apresenta solução que melhore o limitante superior conhecido, e, portanto, podemos descartá-las (poda por limite, linha 27 do algoritmo 8.1).
3. $f(\bar{x}^k, \bar{y}^k) < z^u$ e $\bar{y}^k$ é inteira: neste caso, temos uma solução viável melhor que a melhor solução conhecida. Assim, podemos atualizar z^u e também descartar as subpartições de Y^k (poda por otimalidade, linha 17 do algoritmo 8.1), pois $f(\bar{x}^k, \bar{y}^k)$ ainda é um limitante inferior para as mesmas, e, portanto, nenhuma delas pode fornecer solução que melhore o novo limitante superior. Podemos ainda descartar todas as demais partições em aberto da árvore que apresentam limitante inferior maior que o novo valor de z^u (podas por limite, linha 18 do algoritmo 8.1).
4. $f(\bar{x}^k, \bar{y}^k) < z^u$ e $\bar{y}^k$ não é inteira: neste caso, precisamos avaliar subpartições de Y^k, pois existe a possibilidade de alguma delas abrigar solução viável que melhore o limitante superior. Escolhemos uma variável y_j cujo valor y_j^k seja fracionário para particionar o espaço e ramificamos a árvore, criando assim duas novas subpartições (ramificação, linhas 20-25 do algoritmo 8.1).

O sucesso na aplicação de BB em PLIM está fortemente relacionado à adoção de estratégias auxiliares para aceleração do seu processo de convergência. Repetir esse êxito no universo não linear tem se mostrado um desafio de pesquisa. A dificuldade na resolução de relaxações PNL é um exemplo. Em alguns casos, mesmo as melhores rotinas computacionais de PNL podem não ser capazes de convergir para uma solução ótima de um problema convexo de PNL, ou sequer encontrar uma solução viável de um problema com região viável não vazia. Esse último caso pode comprometer a confiabilidade de um algoritmo de BB, fazendo com que um ramo seja indevidamente tratado como inviável. Ademais, o alto esforço computacional na resolução de problemas PNL também desencoraja a adoção de esquemas de strong branching, fazendo então com que estratégias de pseudocusto (pseudo-cost) sejam bem vistas. Um estudo em que essas duas práticas são avaliadas em BB para PNLIM é realizado em [30]. Nesse estudo, outros mecanismos adotados em BB também são considerados.

No contexto de PNLIM, torna-se também mais difícil a aplicação de técnicas de pré-processamento na resolução de cada subproblema $(\tilde{P}_{Y^k})$. Entretanto, estratégias de pré-processamento de PLIM com base em possíveis restrições lineares presentes em (P) ainda podem ser utilizadas. O processo de geração de cortes na aplicação de BB para problemas de PNLIM também se mostra uma tarefa não trivial e muitas vezes não adotada na prática. Em [199], é avaliado um esquema de geração de cortes para a resolução de um problema PNLIM com função-objetivo linear por meio de um algoritmo de branch-and-cut. No entanto, os cortes são obtidos através de um modelo de PNL cujo esforço necessário para a resolução pode não compensar o possível ganho com o corte. A falta de resultados computacionais efetivos em [199], bem como a escassez de mais trabalhos nessa linha de atuação, parece indicar ser este um caminho não muito promissor.

Em outra direção, Borchers and Mitchel propuseram a ramificação prematura (early branching), que consiste em interromper precocemente a resolução da relaxação

contínua $(\tilde{P}_{Y^k})$ se for detectado que o algoritmo resolvedor está convergindo para uma solução não inteira. Nesse caso, procede-se imediatamente à ramificação. Ao considerar os nós descendentes, retorna-se à resolução de PNL a partir do ponto em que o algoritmo havia sido interrompido quando a ramificação prematura foi realizada, apenas com a mudança de limitante para uma das variáveis imposta pela nova ramificação e com o novo valor inicial para a mesma. Na prática, pode ser difícil implementar esse esquema de modo eficiente. Primeiramente, porque muitas rotinas computacionais não permitem a interrupção da resolução de um problema de PNL e seu posterior reinício de modo prático, especialmente modificando o limitante de uma das variáveis. Mesmo quando essa tarefa é possível, é necessário conhecimento avançado sobre o algoritmo de PNL empregado bem como sobre sua implementação. Adicionalmente, ao se interromper precocemente a resolução de $(\tilde{P}_{Y^k})$, o algoritmo deixa de obter um limitante inferior para o sub-ramo em questão, inviabilizando podas por limite. Todavia, essa desvantagem é amenizada em [140], em que é adicionada, a cada problema $(\tilde{P}_{Y^k})$, a restrição de corte de nível objetivo:

$$f(x,y) \leq z^u - \varepsilon_c, \tag{8.4}$$

para que então as podas que seriam realizadas por limite passem a ser efetuadas por inviabilidade com a adoção da ramificação prematura. Ainda assim, o esforço computacional tende a ser maior, uma vez que é preciso atestar inviabilidade para que podas por limite sejam realizadas. Remediando essa desvantagem, temos em [198] um esquema de ramificação prematura baseado em um algoritmo de plano de corte sequencial que é capaz de prover limitantes inferiores para $(\tilde{P}_{Y^k})$ a cada iteração. Deste modo, a restrição (8.4) pode ser dispensada. Contudo, o algoritmo de plano de corte sequencial possui convergência lenta, o que pode comprometer a eficiência do procedimento de BB.

Observamos na ramificação prematura uma filosofia de certo modo oposta a da geração de cortes. Com os cortes, gasta-se mais esforço computacional em cada nó de BB, na esperança de reduzir o tamanho da árvore explorada. Adotando-se ramificação prematura, procura-se gastar menos esforço em cada nó se não for considerado de interesse. Uma consequência desse fato é que a ramificação prematura pode gerar árvores de BB com maior número de nós, o que na prática pode tornar essa estratégia desvantajosa.

8.3 Algoritmos de aproximação linear

A ideia central dos algoritmos de aproximação linear consiste em aproximar o problema (P) por meio de um problema de programação linear inteira mista (PLIM), comumente denominado problema mestre. Este é construído a partir de linearizações das funções de (P) sobre um conjunto de pontos $T = \{(x^0,y^0),\ldots,(x^k,y^k)\}$. A convexidade das funções de (P) desempenha um papel fundamental neste tipo de metodologia: graças a ela, o problema mestre se constitui em uma relaxação de (P), o que gera limitantes inferiores válidos para ele (no caso de minimização). Desse modo, à medida que esses algoritmos progridem, novos pontos de linearização são adicionados ao conjunto T com o objetivo de gerar aproximações para (P) cada vez melhores. Esse processo se repete até que o limitante inferior obtido esteja suficientemente próximo ao limitante superior dado pela melhor solução viável de (P) encontrada, permitindo assim declarar essa última solução ótima.

O principal atrativo desses métodos consiste no fato de que, ao resolverem o problema mestre por meio de avançados pacotes computacionais de PLIM, como Cplex, Gurobi ou Xpress, eles se valem diretamente de toda a maturidade obtida nessa área

tão extensamente pesquisada. Ademais, esses métodos apresentam uma relativa facilidade em sua compreensão e implementação. De maneira geral, em problemas em que o grau de não linearidade é baixo, esses algoritmos trazem a expectativa de bom desempenho, uma vez que os problemas mestres gerados rapidamente forneceriam uma aproximação adequada do problema de PNLIM original. Como principal desvantagem, há a constatação de que resolver repetidamente problemas de PLIM pode exigir demasiado esforço computacional em alguns casos, mesmo para os softwares mais avançados. Essa característica desfavorável pode se tornar bastante crítica quando for necessário um número elevado de iterações para a convergência do algoritmo, especialmente pelo fato de, a cada iteração, o número de restrições do problema mestre aumentar.

8.3.1 O problema mestre

Para facilitar o entendimento do problema mestre, observamos que (P) pode ser reformulado como um problema com função-objetivo linear através da introdução de uma variável auxiliar α:

$$\begin{array}{lll} (\bar{P}) & \min\limits_{\alpha,x,y} & \alpha \\ & \text{s. a} & f(x,y) \leq \alpha \\ & & g(x,y) \leq 0 \\ & & x \in X,\ y \in Y \cap \mathbb{Z}^{n_y}. \end{array} \tag{8.5}$$

Uma vez que as restrições de $(\bar{P})$ são convexas, ao efetuarmos sua linearização por meio da série de Taylor sobre um dado ponto qualquer $(\bar{x},\bar{y}) \in X \times Y$, obtemos as seguintes desigualdades válidas para $(\bar{P})$:

$$\nabla f(\bar{x},\bar{y})^T \begin{pmatrix} x-\bar{x} \\ y-\bar{y} \end{pmatrix} + f(\bar{x},\bar{y}) \quad \leq \quad \alpha, \tag{8.6}$$

$$\nabla g(\bar{x},\bar{y})^T \begin{pmatrix} x-\bar{x} \\ y-\bar{y} \end{pmatrix} + g(\bar{x},\bar{y}) \quad \leq \quad 0. \tag{8.7}$$

Note que o ponto $(\bar{x},\bar{y})$ não precisa necessariamente satisfazer as restrições de (P). Note também que em $(\bar{x},\bar{y})$ as linearizações aproximam de forma exata as funções originais de (P).

De posse dessas informações, obtemos o problema mestre a partir de um conjunto $L = \{(x^0,y^0),\ldots,(x^k,y^k)\}$ com $k+1$ pontos de linearização que se constitui em uma relaxação de $(\bar{P})$ e de (P):

$$\begin{array}{llllll} (M^L) & \min\limits_{\alpha,x,y} & \alpha & & & \\ & \text{s. a} & \nabla f(x^k,y^k)^T \begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix} + f(x^k,y^k) & \leq & \alpha, & \forall (x^k,y^k) \in L \\ & & \nabla g(x^k,y^k)^T \begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix} + g(x^k,y^k) & \leq & 0, & \forall (x^k,y^k) \in L \\ & & x \in X,\ y \in Y \cap \mathbb{Z}^{n_y}. & & & \end{array} \tag{8.8}$$

Ressaltamos que (M^L) é um problema de PLIM. Seja $(\hat{\alpha},\hat{x},\hat{y})$ uma de suas soluções ótimas. O valor $\hat{\alpha}$ é um limitante inferior válido para $(\bar{P})$ e (P). Se $(\hat{\alpha},\hat{x},\hat{y})$ for viável para $(\bar{P})$, temos que o valor $\hat{\alpha}$ também será um limitante superior para $(\bar{P})$ e (P). Nesse caso, uma vez que $(\hat{\alpha},\hat{x},\hat{y})$ fornece o mesmo valor como limitante inferior e superior para $(\bar{P})$, essa solução será ótima para $(\bar{P})$. Consequentemente, $(\hat{x},\hat{y})$ será uma solução ótima para (P). Por outro lado, se $(\hat{\alpha},\hat{x},\hat{y})$ não for viável para $(\bar{P})$, é necessário acrescentar desigualdades válidas em (M^L) de modo a cortar essa solução de sua região

viável, fortalecendo assim a relaxação fornecida por esse problema. Nessa situação, de um modo geral, o que os algoritmos de aproximação linear fazem é acrescentar um novo ponto ao conjunto L de modo que as novas linearizações geradas sobre ele cortem $(\hat{\alpha},\hat{x},\hat{y})$. Esse procedimento é então repetido iterativamente até que a diferença entre o limitante inferior e superior conhecidos se torne suficientemente pequena, ou que, equivalentemente, a solução ótima do problema mestre $(\hat{\alpha},\hat{x},\hat{y})$ seja uma solução viável para $(\bar{P})$, o que garante que essa solução é ótima para $(\bar{P})$ e que $(\hat{x},\hat{y})$ é ótima para (P). Observe que, ao longo das iterações, com a adição de novos pontos de linearização ao conjunto L, o problema mestre fornece uma sequência não decrescente de limitantes inferiores. Por essa razão, o melhor limitante inferior sempre é obtido na resolução da versão mais recente do problema mestre. Por fim, é oportuno apontar que um elemento-chave para diferenciar um determinado algoritmo dos demais é a estratégia adotada para escolha dos novos pontos de linearização a serem adicionados ao conjunto L.

8.3.2 Algoritmo Plano de Corte Estendido

O algoritmo Plano de Corte Estendido (PCE) foi apresentado por Westerlund e Pettersson [212]. PCE adota a mais simples estratégia para a escolha dos novos pontos adicionados em L: Seja $(\hat{\alpha}^k,\hat{x}^k,\hat{y}^k)$ uma solução ótima de (M^L) na iteração k. Se $(\hat{\alpha}^k,\hat{x}^k,\hat{y}^k)$ não for viável para $(\bar{P})$, então $(\hat{x}^k,\hat{y}^k)$ é adicionada ao conjunto L. Dessa maneira, as novas linearizações sobre $(\hat{x}^k,\hat{y}^k)$ cortarão $(\hat{\alpha}^k,\hat{x}^k,\hat{y}^k)$ da região viável do problema mestre. Originalmente, o algoritmo PCE apenas lineariza no problema mestre a restrição mais violada de $(\bar{P})$ em cada solução $(\hat{x}^k,\hat{y}^k)$ (note que essa restrição pode se remeter à função-objetivo de (P)). Todavia, temos observado que, em muitos casos, linearizar todas as restrições do problema $(\bar{P})$ é uma estratégia mais efetiva para a resolução do problema abordado.

O algoritmo 8.2 traz o algoritmo PCE. Esse método possui uma peculiaridade em relação aos demais métodos de PNLIM: PCE é totalmente baseado na resolução do problema mestre e não faz uso de nenhum procedimento de programação não linear. Por consequência, ele não faz uso de nenhuma informação proveniente de derivadas de segunda ordem, nem de qualquer aproximação destas, além de não necessitar de nenhuma rotina computacional de PNL em sua implementação. Em determinados contextos, essas características podem se mostrar bem vantajosas. Por outro lado, sua convergência não é finita e pode ser demasiadamente lenta em comparação com os demais métodos, já que as novas linearizações construídas a cada iteração geram cortes fracos em comparação com os gerados pelos outros algoritmos. Como consequência, muitas iterações podem ser necessárias para a resolução do problema abordado. Uma vez que, a cada iteração, o número de linearizações no problema mestre aumenta, a exigência de uma larga quantidade de iterações não é uma característica desejável em um método de aproximação linear. Temos observado ainda que, em muitos casos práticos, algumas soluções $(\hat{\alpha}^k,\hat{x}^k,\hat{y}^k)$ subsequentes estão bem próximas umas das outras e o método PCE acaba gastando muitas iterações repetindo a mesma solução inteira $\hat{y}^k$.

8.3.3 Algoritmo Aproximação Externa

O algoritmo Aproximação Externa (AE) foi proposto originalmente por Duran e Grossman em [65] e posteriormente aperfeiçoado por Fletcher e Leyffer em [79]. A ideia principal ainda consiste em adotar o problema mestre (M^L) e, a cada iteração, acrescentar um novo ponto de linearização em L até que o limitante inferior fornecido por (M^L) se torne suficientemente próximo do melhor limitante superior conhecido

Algoritmo 8.2: Plano de Corte Estendido.

Entrada: (P): problema de PNLIM, L_0: conjunto inicial de pontos de linearização do usuário (possivelmente vazio), ε_c: tolerância de convergência.

Saída: (x^*, y^*): solução ótima para (P).

```
z^l = -∞;
z^u = +∞;
Escolha um ponto de linearização inicial (x^0, y^0);
L = L_0 ∪ {(x^0, y^0)};
k = 1;
Seja (M^L) o problema mestre construído a partir de (P̄) sobre os pontos em L;
enquanto z^u − z^l > ε_c faça
    Seja (α̂^k, x̂^k, ŷ^k) uma solução ótima de (M^L);
    z^l = α̂^k;
    se (x̂^k, ŷ^k) é viável para (P) e f(x̂^k, ŷ^k) < z^u então
        (x*, y*) = (x̂^k, ŷ^k);
        z^u = f(x̂^k, ŷ^k);
    L = L ∪ (x̂^k, ŷ^k);
    k = k + 1;
```

para (P). Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de (M^L) na iteração k. Uma tentativa de obtenção de um limitante superior para (P) consiste na resolução do problema $(P_{\hat{y}^k})$, que é o problema de PNL obtido a partir de (P) fixando-se y no valor $\hat{y}^k$:

$$
\begin{array}{lll}
(P_{\hat{y}^k}) & \min\limits_{x} & f(x, \hat{y}^k) \\
 & \text{s. a} & g(x, \hat{y}^k) \leq 0 \\
 & & x \in X.
\end{array}
\tag{8.9}
$$

Se $(P_{\hat{y}^k})$ for viável, seja $(\bar{x}^k, \hat{y}^k)$ uma de suas soluções ótimas. Nesse caso, $f(\bar{x}^k, \hat{y}^k)$ é um limitante superior válido para (P) e $(\bar{P})$, e o ponto $(\bar{x}^k, \hat{y}^k)$ é adicionado ao conjunto L. Observe então que $(\bar{x}^k, \hat{y}^k)$ é a melhor solução para (P), tendo $\hat{y}^k$ como valor para y (ou uma das melhores, se $(P_{\hat{y}^k})$ tiver múltiplas soluções ótimas). Se $(P_{\hat{y}^k})$ for inviável, então o seguinte problema de viabilidade é resolvido:

$$
\begin{array}{lll}
(P^F_{\hat{y}^k}) & \min\limits_{u,x} & \sum_{i=1}^{m} u_i \\
 & \text{s. a} & g(x, \hat{y}^k) \leq u \\
 & & u \geq 0, \\
 & & x \in X,\ u \in \mathbb{R}^m.
\end{array}
\tag{8.10}
$$

Seja $(\check{u}^k, \check{x}^k)$ uma solução ótima de $(P^F_{\hat{y}^k})$ no contexto descrito. Então, o ponto $(\check{x}^k, \hat{y}^k)$ é adicionado ao conjunto L. Após a atualização de L, com a inclusão de $(\bar{x}^k, \hat{y}^k)$ se o problema $(P_{\hat{y}^k})$ for viável, ou com a inclusão de $(\check{x}^k, \hat{y}^k)$ caso contrário, o algoritmo inicia uma nova iteração, tendo como critério de parada uma tolerância máxima para a diferença entre os melhores limitantes inferior e superior obtidos. Conforme demonstrado em [29, 79], assumindo que as condições de KKT são satisfeitas nas soluções de $(P_{\hat{y}^k})$ e $(P^F_{\hat{y}^k})$, essa estratégia de atualização do conjunto L faz com que, a

cada iteração, a solução ótima do problema mestre $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ seja cortada pelas novas linearizações se ela não for solução viável para $(\bar{P})$. Foi demonstrado ainda que essa estratégia garante que uma determinada solução $\hat{y}^k$ para a variável inteira y não seja visitada mais que uma vez pelo algoritmo, exceto se ela fizer parte da solução ótima de (P) (nesse caso a solução poderia ser visitada no máximo duas vezes). Como o número de soluções inteiras é finito por hipótese, já que Y é limitado, o algoritmo encontra a solução ótima de (P) em um número finito de iterações. Assim, em comparação com o algoritmo PCE, AE tende a requerer um número menor de iterações, com o custo de necessitar resolver, a cada iteração, um ou dois problemas de PNL. Todavia, é válido ressaltar que, quando o problema tratado é puramente discreto, isto é, não possui qualquer variável contínua, AE e PCE têm desempenhos fundamentalmente semelhantes, uma vez que, nesta situação, os problemas $(P_{\hat{y}^k})$ e $(P^F_{\hat{y}^k})$ deixam de fazer sentido e, assumindo que ambos os algoritmos partem da mesma solução inicial (x^0, y^0), as soluções acrescentadas ao conjunto L seriam rigorosamente as mesmas.

Algoritmo 8.3: Aproximação Externa.

Entrada: (P): problema de PNLIM, L_0: conjunto inicial de pontos de linearização do usuário (possivelmente vazio), ε_c: tolerância de convergência.

Saída: (x^*, y^*): solução ótima para (P).

1 $z^l = -\infty$;
2 $z^u = +\infty$;
3 Escolha um ponto de linearização inicial (x^0, y^0) (usualmente a solução ótima de $(\bar{P})$ (relaxação contínua de (P)));
4 $L = L_0 \cup \{(x^0, y^0)\}$;
5 $k = 1$;
6 Seja $\left(M^L\right)$ o problema mestre construído a partir de $(\bar{P})$ sobre os pontos em L;
7 Seja $(P_{\hat{y}^k})$ o problema PNL obtido ao se fixar a variável y de (P) em $\hat{y}^k$;
8 Seja $(P^F_{\hat{y}^k})$ o problema PNL de viabilidade obtido a partir de $(P_{\hat{y}^k})$;
9 enquanto $z^u - z^l > \varepsilon_c$ faça
10 Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de $\left(M^L\right)$;
11 $z^l = \hat{\alpha}^k$;
12 se $(P_{\hat{y}^k})$ é viável então
13 Seja x^k uma solução ótima de $(P_{\hat{y}^k})$;
14 se $f(x^k, \hat{y}^k) < z^u$ então
15 $z^u = f(x^k, \hat{y}^k)$;
16 $(x^*, y^*) = (x^k, \hat{y}^k)$;
17 senão
18 Seja (u^k, x^k) uma solução ótima de $(P^F_{\hat{y}^k})$;
19 $L = L \cup \{(x^k, \hat{y}^k)\}$;
20 $k = k + 1$;

O algoritmo 8.3 traz o algoritmo Aproximação Externa. Na linha 3, um ponto de linearização inicial precisa ser obtido. Uma boa escolha é utilizar a solução da relaxação contínua de (P), que é o problema de PNL obtido ao se relaxar a restrição

de integralidade de y, obtendo-se assim o seguinte problema:

$$
\begin{array}{lll}
(\tilde{P}) & \min\limits_{x,y} & f(x,y) \\
 & \text{s. a} & g(x,y) \leq 0, \\
 & & x \in X,\ y \in Y.
\end{array}
\tag{8.11}
$$

Note que essa mesma estratégia poderia ser adotada no algoritmo PCE (algoritmo 8.2). Entretanto, se isso for feito, o algoritmo passará a necessitar de uma rotina de PNL e possivelmente de informação proveniente de derivada de segunda ordem. Podemos ressaltar aqui que a rotina computacional utilizada para otimizar os problemas $(\tilde{P})$, $(P_{\hat{y}^k})$ e $(P^F_{\hat{y}^k})$ pode apresentar algum tipo de falha nesse processo. Nesse caso, pode-se recorrer à estratégia de PCE de adicionar ao conjunto L a solução $(\hat{x}^k, \hat{y}^k)$.

Uma alternativa à resolução do problema de viabilidade $(P^F_{\hat{y}^k})$, no caso em que todas as variáveis inteiras de (P) forem binárias, é adicionar o seguinte corte binário no problema mestre quando $(P_{\hat{y}^k})$ for inviável [65]:

$$
\sum_{j \in B^k} y_j - \sum_{j \in N^k} y_j \leq |B^k| - 1, \tag{8.12}
$$

em que $B^k = \{j \mid y^k_j = 1\}$, $N^k = \{j \mid y^k_j = 0\}$ e y^k é a parte inteira (binária) da solução do problema mestre. Note que cada restrição (8.12) adicionada corta apenas a solução inteira y^k sobre a qual foi construída. De fato, pode-se também adicionar esse corte mesmo quando $(P_{\hat{y}^k})$ for viável. Entretanto, ao se incluir esse corte no problema mestre quando existem soluções viáveis para (P) tendo y^k como valor para y, o problema mestre deixa de ser, a rigor, uma relaxação (e também uma aproximação externa) de (P). Todavia, ainda assim o algoritmo terminará sua execução sem qualquer prejuízo quando $z^l \geq z^u - \varepsilon_c$ ou o problema mestre se tornar inviável. É válido ressaltar que o corte gerado a partir da restrição (8.12) é considerado mais fraco que os gerados através da solução de $(P^F_{\hat{y}^k})$, uma vez que (8.12) elimina apenas a solução y^k da região viável.

8.3.4 Algoritmo Decomposição de Benders Generalizada

O algoritmo AE possui similaridades com outro algoritmo de aproximação linear clássico para PNLIM, o algoritmo de Decomposição Benders Generalizada (DBG), proposto por Geoffrion [92]. A principal diferença entre ambos consiste na forma como as funções são linearizadas no problema mestre. Enquanto AE e os demais algoritmos de aproximação linear linearizam as funções com base na série de Taylor em (8.6) e (8.7), DBG adota cortes lagrangianos projetados no espaço da variável y, quando $(P_{\hat{y}^k})$ é viável:

$$
\begin{array}{lll}
\nabla_y f(x^k, \hat{y}^k)^T (y - \hat{y}^k) + f(x^k, \hat{y}^k) & & \\
+ (\lambda^k)^T (\nabla_y g(x^k, \hat{y}^k)^T (y - \hat{y}^k) + g(x^k, \hat{y}^k)) & \leq & \alpha,
\end{array}
\tag{8.13}
$$

em que x^k e λ^k são a solução ótima e os valores correspondentes dos multiplicadores de Lagrange obtidos na resolução de $(P_{\hat{y}^k})$, respectivamente. Se $(P_{\hat{y}^k})$ for inviável, é adotado o corte:

$$
(\mu^k)^T \left(\nabla_y g(x^k, \hat{y}^k)^T (y - \hat{y}^k) + g(x^k, \hat{y}^k) \right) \leq 0, \tag{8.14}
$$

em que x^k e μ^k são a solução ótima e os correspondentes multiplicadores de Lagrange obtidos na resolução de $(P^F_{\hat{y}^k})$, respectivamente.

Embora o algoritmo DBG também possua convergência finita, pode-se encontrar em [65] uma demonstração de que o limitante inferior fornecido pelo problema mestre

(M^L), adotado em AE, é maior ou igual ao fornecido pelo problema mestre adotado por DBG sobre o mesmo conjunto de pontos de linearização, o que tende a fazer com que este último método precise de mais iterações para convergir. Por essa razão, os algoritmos de aproximação linear têm sido utilizados com o problema mestre (M^L) adotado em AE em vez do problema mestre de DBG.

8.3.5 Algoritmo branch-and-bound baseado em PL/PNL

Quesada e Grossmann apresentaram em [181] um algoritmo de aproximação linear baseado na resolução de problemas de programação linear (PL) e programação não linear (PNL) em um esquema de branch-and-bound (branch-and-bound baseado em PL/PNL, ou BB-PL/PNL). A estratégia de atualização do conjunto L é a mesma de AE, isto é, fundamentada na solução dos problemas ($P_{\hat{y}^k}$) e ($P^F_{\hat{y}^k}$). A principal motivação desse algoritmo vem da observação de que, em geral, o problema mestre (M^L) é resolvido sucessivas vezes nos algoritmos de aproximação linear. Essa resolução é comumente realizada por meio de um procedimento caixa-preta especializado, geralmente uma sub-rotina que implementa alguma metodologia baseada em BB. Essas sub-rotinas podem vir de avançados pacotes de PLIM, como Cplex, Gurobi ou Xpress. Quesada e Grossmann propuseram então abrir esse procedimento caixa-preta, integrando a resolução do problema mestre com a resolução do problema de PNLIM abordado através de um esquema único de BB. Nesse esquema de BB, em cada partição Y^k explorada, o algoritmo resolve a seguinte relaxação contínua do problema mestre:

$$
\begin{array}{llllll}
\left(\tilde{M}^L_{Y^k}\right) & \min\limits_{\alpha,x,y} & \alpha & & & \\
 & \text{s. a} & \nabla f(x^k,y^k)^T \begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix} + f(x^k,y^k) & \leq & \alpha, & \forall (x^k,y^k) \in L \\
 & & \nabla g(x^k,y^k)^T \begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix} + g(x^k,y^k) & \leq & 0, & \forall (x^k,y^k) \in L \\
 & & x \in X,\ y \in Y^k. & & &
\end{array}
\tag{8.15}
$$

Observamos que o problema mestre é dinamicamente atualizado através da inclusão de novos pontos de linearização em L, que se originam da resolução dos problemas ($P_{\hat{y}^k}$) e ($P^F_{\hat{y}^k}$) quando uma solução inteira porventura é encontrada.

O algoritmo branch-and-bound baseado em PL/PNL é mostrado como algoritmo 8.4. Note que apenas podas por inviabilidade e limite são realizadas. Quando uma solução inteira é obtida na resolução de ($\tilde{M}^L_{Y^k}$) (linha 14), o algoritmo entra em um laço de atualização e resolução do problema mestre até que uma solução não inteira seja obtida (linhas 15-25). Apontamos que a atualização do problema mestre não segue a ideia de particionamento do espaço do algoritmo BB, isto é, quando linearizações são acrescentadas na exploração de uma determinada partição do espaço, as novas restrições passarão a ser impostas a todos os subproblemas ($\tilde{M}^L_{Y^k}$) resolvidos daquela iteração em diante, independentemente do ramo em que estejam na árvore de BB.

À primeira vista, pode parecer difícil conseguir implementar o algoritmo 8.4 de forma tão eficiente quanto AE, pois, ao abrir o procedimento de resolução do problema mestre, o algoritmo BB-PL/PNL perde, em princípio, a habilidade de utilizar as engenhosas estratégias que avançados solvers de PLIM empregam para ampliar sua eficiência. No entanto, ressaltamos aqui que, em geral, esses mesmos solvers fornecem um meio prático de implementar algoritmos como esse através do uso de lazy constraints (delayed rows) juntamente com funções callback que permitem modificar o comportamento padrão de seu procedimento de BB. Adicionando-se as novas

Algoritmo 8.4: Branch-and-bound baseado em PL/PNL.

Entrada: (P): Problema de PNLIM, L_0: conjunto inicial de pontos de linearização do usuário (possivelmente vazio), ε_c: tolerância de convergência.

Saída: (x^*, y^*): solução ótima para (P).

$z^u = +\infty$;
Escolha um ponto de linearização inicial (x^0, y^0), (usualmente a solução ótima de $(\tilde{P})$ (relaxação contínua de (P)));
$L = L_0 \cup \{(x^0, y^0)\}$;
$Y^0 = Y$;
Seja $N = \{0\}$ a lista inicial de nós em aberto da enumeração de BB;
Seja l_b^i o limitante inferior do nó i;
$l_b^0 = -\infty$;
$i = 0$;
enquanto $N \neq \emptyset$ faça
  Selecione um nó k de N;
  se $\left(\tilde{M}^L_{Y^k}\right)$ é inviável então
    $N = N \backslash \{k\}$; // Poda por inviabilidade
  senão
    Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de $\left(\tilde{M}^L_{Y^k}\right)$;
    enquanto $\hat{y}^k$ possui valor inteiro e $\hat{\alpha}^k < z^u - \varepsilon_c$ faça
      se $(P_{\hat{y}^k})$ é viável então
        Seja $\tilde{x}^k$ uma solução ótima de $(P_{\hat{y}^k})$;
        se $f(\tilde{x}^k, \hat{y}^k) < z^u$ então
          $z^u = f(\tilde{x}^k, \hat{y}^k)$;
          $(x^*, y^*) = (\tilde{x}^k, \hat{y}^k)$;
          $N = N \backslash \{j : l_b^j \geq z^u - \varepsilon_c\}$; // Poda por limite
      senão
        Seja $(\tilde{u}^k, \tilde{x}^k)$ uma solução ótima de $(P^F_{\hat{y}^k})$;
      $L = L \cup \{(\tilde{x}^k, \hat{y}^k)\}$;
      Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de $\left(\tilde{M}^L_{Y^k}\right)$;
    se $\hat{\alpha}^k < z^u - \varepsilon_c$ então
      // Ramificação
      Selecione uma variável y_j com valor não inteiro $\hat{y}^k_j$;
      $Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \leq \lfloor \hat{y}^k_j \rfloor\}$;
      $Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \geq \lceil \hat{y}^k_j \rceil\}$;
      $l_b^{i+1} = l_b^{i+2} = \hat{\alpha}^k$;
      $N = N \cup \{i+1, i+2\} \backslash \{k\}$;
      $i = i + 2$;
    senão
      $N = N \backslash \{k\}$; // Poda por limite

linearizações do problema mestre como lazy constraints, é possível implementar o algoritmo 8.4 facilmente com o uso de pacotes como Cplex ou Gurobi, continuando assim a se valer de todos os recursos disponíveis nesses softwares. Desse modo, uma vez que considera um único procedimento de BB, o branch-and-bound para PLIM não precisa ser reiniciado a cada iteração, conforme ocorre em AE. Assim, espera-se que o algoritmo 8.4 apresente uma carga de computação reduzida. De fato, resultados apresentados em [163] apontaram melhor desempenho deste algoritmo em comparação com os demais algoritmos de PNLIM em geral nas instâncias de teste consideradas.

8.3.6 Algoritmo Hiperplano de Suporte Estendido

O algoritmo Hiperplano de Suporte Estendido (HSE) foi proposto por Kronqvist et al. [135]. HSE está baseado em um princípio interessante: fortalecer a relaxação fornecida pelo problema mestre antes de efetivamente começar a sequência de resoluções desse problema de PLIM. Observe que, em geral, algoritmos de aproximação linear aproximam o problema (P) através do problema mestre. Até que essa aproximação se torne adequada para efetivamente garantir a resolução do problema, um número de linearizações precisa ser adicionado ao problema mestre, que é então sucessivamente resolvido. Uma vez que o problema mestre é de PLIM, essa sequência de resoluções pode se mostrar demasiadamente custosa. Vale então ressaltar que, se um conjunto ideal de pontos de linearização L fosse conhecido a priori, apenas uma única resolução de problema de PLIM seria necessária. Desse modo, o algoritmo HSE tenta construir um problema mestre que forneça algum nível razoável de aproximação de (P), antes de efetivamente começar a resolver o problema (M^L), com pontos de linearizações iniciais. Esses primeiros pontos de linearizações são obtidos através da resolução da relaxação contínua de (M^L). Uma vez que essas relaxações são problemas de programação linear contínua, esse conjunto inicial de pontos é gerado de forma bem menos custosa em relação aos demais pontos. Assim, ao se iniciar a sequência de resoluções de (M^L), HSE já partirá de uma aproximação inicial possivelmente razoável para o problema (P), o que pode vir a economizar algumas resoluções do problema mestre. Essa etapa inicial de obtenção dos primeiros pontos termina quando a solução da relaxação contínua de (P) é encontrada, ou algum critério de parada adicional é alcançado.

Além de adotar resolução de problemas de PL nas primeiras iterações, HSE também utiliza um esquema mais simples que AE para a obtenção de novos pontos de linearização (x^k, y^k). No lugar da resolução dos problemas (P_{y^k}) e $(P^F_{\hat{y}^k})$, HSE adota um procedimento de busca linear entre a solução $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ obtida com a resolução do problema mestre (ou sua relaxação) e uma solução interior $(\dot{x}, \dot{y})$ previamente obtida. Se $(\hat{x}^k, \hat{y}^k)$ não for viável para (P), (x^k, y^k) é então calculada como a solução na fronteira da região viável entre $(\hat{x}^k, \hat{y}^k)$ e $(\dot{x}, \dot{y})$. Se $(\hat{x}^k, \hat{y}^k)$ for viável para (P), é suficiente linearizar apenas a função-objetivo $f(x, y)$ em $(\hat{x}^k, \hat{y}^k)$. A solução interior $(\dot{x}, \dot{y})$ pode ser obtida, por exemplo, com a resolução do seguinte problema:

$$
\begin{array}{lll}
(P^I) & \min\limits_{u,x,y} & u \\
 & \text{s. a} & g(x,y) \leq u \\
 & & u \in \mathbb{R},\ x \in X,\ y \in Y.
\end{array}
\tag{8.16}
$$

Conforme [135] aponta, o problema (P^I) não precisa ser resolvido até a otimalidade, mas sim até a obtenção de uma solução viável $(\dot{u}, \dot{x}, \dot{y})$ com $\dot{u} < 0$. Também é válido mencionar que apenas as restrições não lineares de (P) precisam estar inativas em $(\dot{u}, \dot{x}, \dot{y})$. Possíveis restrições lineares presentes em (P) podem estar ativas em $(\dot{x}, \dot{y})$.

O algoritmo HSE é exibido como algoritmo 8.5. Originalmente, seus autores propõem linearizar apenas restrições ativas de $(\bar{P})$ nas soluções de L. Entretanto,

Algoritmo 8.5: Hiperplano de Suporte Estendido.

Entrada: (P): problema de PNLIM, L_0: conjunto inicial de pontos de linearização do usuário (possivelmente vazio), $\tilde{\varepsilon}_c$: tolerância de convergência no laço de resolução PL, ε_c: tolerância de convergência, K^{PL}: número máximo de iterações no laço de resolução PL.

Saída: (x^*, y^*): solução ótima para (P).

$\tilde{z}^l = -\infty$;
$\tilde{z}^u = +\infty$;
Seja $(\dot{x}, \dot{y})$ uma solução interior para (P), possivelmente obtida com a resolução de (P^I) ;
$L = L_0 \cup \{(\dot{x}, \dot{y})\}$;
$k = 1$;
Seja $(\tilde{P})$ a relaxação contínua de (P);
Seja $\left(M^L\right)$ o problema mestre construído a partir de $(\tilde{P})$ sobre os pontos em L e $\left(\tilde{M}^L\right)$ sua relaxação contínua;
enquanto $\tilde{z}^u - \tilde{z}^l > \tilde{\varepsilon}_c$ e $k < K^{PL}$ faça
  Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de $\left(\tilde{M}^L\right)$;
  $\tilde{z}^l = \hat{\alpha}^k$;
  se $(\hat{x}^k, \hat{y}^k)$ é viável para $(\tilde{P})$ então
    se $f(\hat{x}^k, \hat{y}^k) < \tilde{z}^u$ então
      $\tilde{z}^u = f(\hat{x}^k, \hat{y}^k)$;
    Adicione uma restrição de linearização da função-objetivo $f(x, y)$ sobre $(\hat{x}^k, \hat{y}^k)$ em $\left(M^L\right)$ e $\left(\tilde{M}^L\right)$ segundo expressão (8.6);
  senão
    Seja (x^k, y^k) o resultado da busca linear entre $(\hat{x}^k, \hat{y}^k)$ e $(\dot{x}, \dot{y})$;
    $L = L \cup (x^k, y^k)$;
  $k = k + 1$;
$z^l = \tilde{z}^l$;
$z^u = +\infty$;
enquanto $z^u - z^l > \varepsilon_c$ faça
  Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de $\left(M^L\right)$;
  $z^l = \hat{\alpha}^k$;
  se $(\hat{x}^k, \hat{y}^k)$ é viável para (P) então
    se $f(\hat{x}^k, \hat{y}^k) < z^u$ então
      $z^u = f(\hat{x}^k, \hat{y}^k)$;
      $(x^*, y^*) = (\hat{x}^k, \hat{y}^k)$;
    Adicione uma restrição de linearização da função-objetivo $f(x, y)$ sobre $(\hat{x}^k, \hat{y}^k)$ em $\left(M^L\right)$ segundo expressão (8.6);
  senão
    Seja (x^k, y^k) o resultado da busca linear entre $(\hat{x}^k, \hat{y}^k)$ e $(\dot{x}, \dot{y})$;
    $L = L \cup (x^k, y^k)$;
  $k = k + 1$;

temos observado melhores resultados em nossos experimentos linearizando todas as restrições de $(\bar{P})$ nessas soluções. Quando o problema mestre (ou sua linearização) fornece uma solução viável para (P), pode-se optar por linearizar apenas a função-objetivo nessa solução (linhas 14 e 28). Apontamos que, entre as linhas 8 e 18, o algoritmo resolve relaxações contínuas do problema mestre. Deste modo, este trecho do algoritmo está solucionando a relaxação contínua de (P) através do método de hiperplano de suporte [208]. Note que AE e BB-PL/PNL podem também realizar tal resolução (linha 3 do algoritmo 8.3 e linha 2 do algoritmo 8.4), entretanto, HSE abre esse procedimento de modo a aproveitar soluções intermediárias para fortalecer a relaxação provida por (M^L) antes de efetivamente começar a considerar a integralidade do problema. No laço entre as linhas 21 e 32, HSE se remete à solução de (P), resolvendo, a cada iteração, a versão corrente do problema mestre.

8.4 Algoritmos de subproblemas contínuos

Os algoritmos de subproblemas contínuos buscam resolver o problema (P) através da resolução de subproblemas em domínio contínuo, sem a realização de particionamento explícito do espaço de soluções, como ocorreria em um algoritmo de branch-and-bound. Esse tipo de abordagem tem sido proposta na literatura para casos particulares do problema (P), em que todas as variáveis inteiras são binárias, isto é, restritas aos valores $\{0,1\}$. Desse modo, os algoritmos se baseiam na construção de problemas contínuos, substituindo a restrição

$$y \in \{0,1\}^{n_y}$$

por alguma expressão que force as variáveis inteiras a só assumirem os valores binários, por exemplo:

$$\sum_{i=1}^{n_y} y_i(1-y_i) = 0 \tag{8.17}$$

Note que o lado esquerdo da equação (8.17) pode ser usado para medir o quão distante uma solução para y está de satisfazer a integralidade. Em outras palavras, temos aqui uma função para medir o gap de integralidade de uma solução.

Em geral, as abordagens adotam algum esquema para tratar os entraves gerados pela não convexidade da equação (8.17). Por exemplo, Murray e Ng apresentam em [168] um algoritmo especializado para problemas de PNLIM com restrições lineares em que a equação (8.17) é penalizada na função-objetivo. Adicionalmente, um esquema de suavização é adotado. Essa estratégia de penalizar a equação (8.17) na função-objetivo já havia sido apresentada por Raghavachari em [182], ainda no âmbito de PLIM. Por sua vez, Melo (2016) e Melo, Fampa e Raupp (2018) propõem em [164, 162] o Algoritmo de Minimização de Gap de Integralidade 1, que considera o lado esquerdo da equação (8.17) de forma isolada na função-objetivo. Esse algoritmo é discutido a seguir.

8.4.1 Algoritmo de Minimização de Gap de Integralidade 1 (para problemas binários)

O Algoritmo de Minimização de Gap de Integralidade 1 (AMGI1) foi proposto por Melo et al. em [164] para resolução de problemas de PNLIM em que todas as variáveis são binárias. Fundamentado na substituição das restrições de integralidade pela equação (8.17), sendo seu lado esquerdo levado à função-objetivo, o problema

(contínuo) de minimização de gap de integralidade (P^G) é construído como:

$$\begin{array}{lll} (P^G) & \min\limits_{x,y} & \sum_{i=1}^{n_y} y_i(1-y_i) \\ & \text{s. a:} & g(x,y) \leq 0, \\ & & 0 \leq y \leq 1, \\ & & x \in X,\ y \in Y. \end{array} \tag{8.18}$$

Embora a função-objetivo de (P^G) seja não convexa (côncava), ela traz menos dificuldades para a convergência em geral de pacotes de PNL em comparação com a adoção direta da restrição (8.17). Uma vez que a função-objetivo $f(x,y)$ foi completamente substituída pelo lado esquerdo de (8.17), a resolução do problema (P^G) pode conduzir, na iteração k, a qualquer solução inteira $(\hat{x}^k, \hat{y}^k)$ que seja viável para (P). Para remediar esse fato, os autores adotam duas estratégias: a primeira consiste na realização de uma busca local sobre $\hat{y}^k$ através da resolução de $(P_{\hat{y}^k})$, que é o subproblema de (P) obtido ao se fixar y em $\hat{y}^k$, conforme apresentado em (8.9); a segunda estratégia consiste em adotar, a partir da segunda iteração, um corte de nível objetivo no problema de minimização de gap, conduzindo assim ao problema $(\hat{P}^G(z^u, \varepsilon_c))$:

$$(\hat{P}^G(z^u, \varepsilon_c)) \quad \min_{x,y} \quad \sum_{i=1}^{n_y} y_i(1-y_i) \tag{8.19}$$

$$\text{s. t.:} \quad g(x,y) \leq 0, \tag{8.20}$$

$$f(x,y) \leq z^u - \varepsilon_c, \tag{8.21}$$

$$0 \leq y \leq 1, \tag{8.22}$$

$$x \in X,\ y \in Y, \tag{8.23}$$

em que z^u é o melhor limitante superior obtido para (P), e $\varepsilon_c > 0$ é uma tolerância de convergência. Note que a finalidade da restrição (8.21) é não permitir que uma solução com maior valor objetivo que a melhor solução conhecida seja obtida com a resolução de $(\hat{P}^G(z^u, \varepsilon_c))$.

A partir dessas definições, a base dos algoritmos de minimização de gap de integralidade é apresentada como algoritmo 8.6. O fato de necessitar resolver os problemas de minimização de gap (P^G) e $(\hat{P}^G(z^u, \varepsilon_c))$ globalmente faz com que o algoritmo 8.6 não possua utilidade prática. Para remediar isso, Melo et al. propuseram resolver os problemas de minimização de gap por meio de pacotes PNL locais em um esquema de BB. Para isso, considera-se o problema de minimização de gap $(\hat{P}^G(Y^k, z^u, \varepsilon_c))$ no contexto de cada partição Y^k:

$$\begin{array}{lll} (\hat{P}^G(Y^k, z^u, \varepsilon_c)) & \min\limits_{x,y} & \sum_{i=1}^{n_y} y_i(1-y_i) \\ & \text{s. t:} & g(x,y) \leq 0, \\ & & f(x,y) \leq z^u - \varepsilon_c, \\ & & 0 \leq y \leq 1, \\ & & x \in X,\ y \in Y^k. \end{array} \tag{8.24}$$

AMGI1 é apresentada como algoritmo 8.7. Ressaltamos que esse algoritmo foi construído de modo a utilizar rotinas de PNL local. Uma vez que os problemas de minimização não fornecem limitantes inferiores para o problema abordado, o esquema de BB de AMGI1 apenas realiza podas por inviabilidade. Todavia, conforme apontado em [164], devido à incorporação do corte de nível objetivo, podas que seriam realizadas por otimalidade e limite em um BB padrão acabam sendo realizadas por inviabilidade em AMGI1. Embora os resultados em [164] apontem que AMGI1 não é uma abordagem competitiva com os demais algoritmos exatos de PNLIM, foi constatado que o algoritmo tende a encontrar soluções viáveis rapidamente, o que

Algoritmo 8.6: Base para os Algoritmos de Minimização de Gap de Integralidade.

Entrada: (P): problema de PNLIM, ε_c: tolerância de convergência.
Saída: (x^*, y^*): Solução ótima para (P).

1 Seja $(\hat{x}^0, \hat{y}^0)$ uma solução ótima global de (P^G);
2 Seja $\tilde{x}^0$ uma solução ótima de $(P_{\hat{y}^0})$;
3 $(x^*, y^*) = (\tilde{x}^0, \hat{y}^0)$;
4 $z^u = f(\tilde{x}^0, \hat{y}^0)$;
5 $k = 1$;
6 enquanto $(\hat{P}^G(z^u, \varepsilon_c))$ for viável faça
7 Seja $(\hat{x}^k, \hat{y}^k)$ uma solução ótima global de $(\hat{P}^G(z^u, \varepsilon_c))$;
8 se $\hat{y}^k$ não for inteira então
9 pare o algoritmo;
10 Seja $\tilde{x}^k$ uma solução ótima de $(P_{\hat{y}^k})$;
11 $(x^*, y^*) = (\tilde{x}^k, \hat{y}^k)$;
12 $z^u = f(\tilde{x}^k, \hat{y}^k)$;
13 $k = k + 1$;

o torna uma boa opção como heurística de viabilidade. Os autores chegam ainda a apresentar uma segunda versão do algoritmo, denominada AMGI2, a ser utilizada exclusivamente como heurística de viabilidade.

Há uma relação evidente entre o BB padrão e AMGI1. A principal diferença diz respeito ao fato de que um BB padrão otimiza em cada partição observando somente a função-objetivo original do problema tratado, o que pode levar a soluções nas partições que estejam longe da integralidade. Por sua vez, AMGI1 otimiza nas partições buscando encontrar alguma solução inteira que melhore o limitante superior conhecido. Por essa razão, AMGI1 tende a encontrar soluções viáveis mais rapidamente que o BB padrão.

8.5 Algoritmos híbridos

8.5.1 Algoritmo branch-and-bound Híbrido Bonmin

Bonami et al. apresentaram em [29] um algoritmo que mistura a metodologia de aproximação linear com um esquema de resolução das relaxações não lineares adotadas no particionamento de BB (seção 8.2). Esse algoritmo é um dos algoritmos de PNLIM implementados no solver Bonmin, razão pela qual o denominamos aqui branch-and-bound Híbrido Bonmin (BBHB). A base dessa abordagem é o algoritmo branch-and-bound baseado em PL/PNL (subseção 8.3.5), no qual, em alguns nós da enumeração de BB, relaxações não lineares são resolvidas na partição corrente para a obtenção de novos pontos de linearização a serem adicionados ao conjunto L. Adicionalmente, o algoritmo Aproximação Externa (subseção 8.3.3) pode ser aplicado por um período de tempo antes de o procedimento de BB se iniciar. Os autores chegaram a sugerir o uso de AE como procedimento de busca local em determinados ramos da árvore de BB, mas não forneceram resultados em que efetivamente adotem

Algoritmo 8.7: Algoritmo de Mimização do Gap de Integralidade

1. Entrada: (P): problema de PNLIM, ε_c: tolerância de convergência.
   Saída: (x^*, y^*): Solução ótima para (P).

1 $z^u = \infty$;
2 $Y^0 = Y$;
3 Seja $N := \{0\}$ a lista inicial de nós em aberto da enumeração de BB;
4 $i = 0$;
5 enquanto $N \neq \emptyset$ faça
6 Selecione um nó k de N;
7 se $(\hat{P}^G(Y^k, z^u, \varepsilon_c))$ é viável então
8 Seja $(\bar{x}^k, \bar{y}^k)$ uma solução ótima (possivelmente local) de $(\hat{P}^G(Y^k, z^u, \varepsilon_c))$;
9 enquanto $\bar{y}^k$ é inteira faça
10 Seja $\bar{x}^k$ uma solução ótima de $(P_{\bar{y}^k})$;
11 $(x^*, y^*) = (\bar{x}^k, \bar{y}^k)$;
12 $z^u = f(\bar{x}^k, \bar{y}^k)$;
13 se $(\hat{P}^G(Y^k, z^u, \varepsilon_c))$ é inviável então
14 vá para linha 21;
15 Seja $(\bar{x}^k, \bar{y}^k)$ uma solução ótima (possivelmente local) de $(\hat{P}^G(Y^k, z^u, \varepsilon_c))$;
 // Ramificação
16 Selecione uma variável y_j com valor não inteiro $\bar{y}^k_j$;
17 $Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j = 0\}$;
18 $Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j = 1\}$;
19 $N = N \cup \{i+1, i+2\}$;
20 $i = i+2$;
21 $N = N \setminus \{k\}$;

tal estratégia.

O algoritmo BBHB é mostrado como algoritmo 8.8. Assim como no caso de branch-and-bound baseado em PL/PNL, esse algoritmo pode ser implementado em pacotes avançados de PLIM através do uso de lazy constraints (delayed rows) juntamente com funções callback que permitem a modificação do comportamento padrão do procedimento de BB. Na linha 3 do algoritmo 8.8, o algoritmo Aproximação Externa é executado por um tempo limite de até α segundos. Para tornar a apresentação desse algoritmo mais simples, omitimos alguns detalhes relacionados à aplicação de AE. Primeiramente, se AE efetivamente resolver o problema em até α segundos, obviamente o algoritmo 8.8 pode ser interrompido após a linha 3. Ainda, a execução de AE compartilhará o mesmo conjunto de pontos de linearização L com o algoritmo 8.8. Em outras palavras, os pontos de linearização calculados por AE serão aproveitados no restante do procedimento. Por fim, se AE encontrar alguma solução viável para (P) no tempo estipulado, então o limitante superior corrente z^u e a melhor solução conhecida (x^*, y^*) serão atualizados.

Observe ainda que o parâmetro β controla a frequência com que $(\tilde{P}_{Y^k})$ será resol-

Algoritmo 8.8: Branch-and-bound Híbrido Bonmin.

Entrada: (P): Problema de PNLIM abordado, L_0: conjunto inicial de pontos de linearização do usuário (possivelmente vazio), ε_c: tolerância de convergência, α: tempo para busca local, β: parâmetro para a resolução da relaxação contínua de PNL.

Saída: (x^*, y^*): solução ótima de (P).

1 $z^u = +\infty$;
2 Seja (x^0, y^0) uma solução ótima de $(\tilde{P})$; $L = L_0 \cup (x^0; y^0)$; $Y^0 = Y$;
3 Executar AproximaçãoExterna((P), L, ε_c) por, no máximo, α segundos;
4 Seja $N = \{0\}$ a lista inicial de nós em aberto;
5 Seja l_b^i o limitante inferior do nó i;
6 $l_b^0 = -\infty$; $i = 0$; $d = 0$;
7 enquanto $N \neq \emptyset$ faça
8 Selecione um nó k de N;
9 $N = N \backslash \{k\}$;
10 se $d \equiv 1 \pmod{\beta}$ então
11 se $(\tilde{P}_{Y^k})$ é inviável então
12 vá para linha 7;
13 Seja $(\tilde{x}^k, \tilde{y}^k)$ uma solução ótima de $(\tilde{P}_{Y^k})$;
14 $L = L \cup \{(\tilde{x}^k, \tilde{y}^k)\}$;
15 se $\tilde{y}^k$ é inteira e $f(\tilde{x}^k, \tilde{y}^k) < z^u$ então
16 $z^u = f(\tilde{x}^k, \tilde{y}^k)$; $(x^*, y^*) = (\tilde{x}^k, \tilde{y}^k)$;
17 $N = N \backslash \{j : l_b^j \geq z^u - \varepsilon_c\}$; // Podas por limite
18 se $f(\tilde{x}^k, \tilde{y}^k) \geq z^u - \varepsilon_c$ então
19 vá para linha 7;
20 se $\left(\tilde{M}_{Y^k}^L\right)$ é viável então
21 Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de $\left(\tilde{M}_{Y^k}^L\right)$;
22 enquanto $\hat{y}^k$ é inteira e $\hat{\alpha}^k < z^u - \varepsilon_c$ faça
23 se $(P_{\hat{y}^k})$ é viável então
24 Seja $\bar{x}^k$ uma solução ótima de $(P_{\hat{y}^k})$;
25 se $f(\bar{x}^k, \hat{y}^k) < z^u$ então
26 $z^u = f(\bar{x}^k, \hat{y}^k)$; $(x^*, y^*) = (\bar{x}^k, \hat{y}^k)$;
27 $N = N \backslash \{j : l_b^j \geq z^u - \varepsilon_c\}$; // Podas por limite
28 senão
29 Seja $(\bar{u}^k, \bar{x}^k)$ uma solução ótima de $(P_{\hat{y}^k}^F)$;
30 $L = L \cup \{(\bar{x}^k, \hat{y}^k)\}$;
31 Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima $\left(\tilde{M}_{Y^k}^L\right)$;
32 se $\hat{\alpha}^k < z^u - \varepsilon_c$ então
// Ramificação
33 Selecione uma variável y_j com valor fracionário $\hat{y}_j^k$;
34 $Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \leq \lfloor \hat{y}_j^k \rfloor\}$;
35 $Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \geq \lceil \hat{y}_j^k \rceil\}$;
36 $l_b^{i+1} = l_b^{i+2} = \hat{\alpha}^k$;
37 $N = N \cup \{i+1, i+2\}$;
38 $i = i + 2$; $d = d + 1$;

vido ao longo da exploração das partições (linhas 10-19). O valor sugerido em [29] é $\beta = 10$. As soluções obtidas com a resolução desse problema são adicionadas a L com o objetivo de fortalecer a relaxação dada pelo problema mestre. No entanto, apontamos que o número elevado de problemas PNL sendo resolvidos por esse algoritmo, juntamente com o possível excesso de pontos de linearização em L, pode fazer com que BBHB se torne desvantajoso em relação a branch-and-bound baseado em PL/PNL, que é o algoritmo que lhe forneceu arcabouço.

8.5.2 Branch-and-bound Híbrido com Aproximação Externa

O algoritmo branch-and-bound Híbrido com Aproximação Externa (BBHAE) proposto por Melo et al. [165]. Sua proposta é a integração de um algoritmo de aproximação linear (Aproximação Externa) em uma metodologia de branch-and-bound, com a expectativa de que a mesclagem de algoritmos de diferentes classes poderia gerar uma nova abordagem capaz de aproveitar as vantagens de ambas. Desse modo, BBHAE toma a exploração de partições de um branch-and-bound não linear, utilizando o algoritmo de aproximação externa como sub-rotina, sempre com uma janela de tempo limitada de execução, em duas situações distintas:

1. Em algumas partições do espaço, com o objetivo de buscar por melhores soluções ou efetivamente resolvê-las. A cada $freq_AE_subprob$ partições exploradas, AE será executado;
2. No problema (P) original, a cada $freq_AE_prob$ partições exploradas. Nessa situação, o conjunto de pontos de linearização acumula os pontos obtidos entre as diferentes chamadas ao procedimento de AE.

O algoritmo BBHAE é mostrado como algoritmo 8.9. Além dos parâmetros de entrada usuais, BBHAE também define os seguintes parâmetros de entrada adicionais:

- $freq_AE_subprob$: frequência de aplicação de AE em subproblemas;
- $freq_AE_prob$: frequência de aplicação de AE em (P);
- $tempo_AE_subprob$: tempo máximo (em segundos) para cada aplicação de AE em subproblemas;
- $tempo_AE_prob$: tempo máximo (em segundos) para cada aplicação de AE em (P);
- $AE((\bar{P}), L_I, z^u, tempo)$: procedimento que implementa o algoritmo Aproximação Externa que aborda $(\bar{P})$, com um conjunto inicial de pontos de linearização L_I, limitante superior z^u, por no máximo $tempo$ segundos. AE retorna: $status$: status de execução do algoritmo, $(\bar{x}, \bar{y})$: melhor solução obtida para $(\bar{P})$, L_F: conjunto final de pontos de linearização, $\bar{z}^l$: limitante inferior para o valor objetivo ótimo de $\bar{P}$.

A principal motivação de BBHAE é a constatação de que o conjunto L pode ser utilizado para salvar o estado corrente de execução do algoritmo Aproximação Externa. Em outras palavras, AE pode ser interrompido ao final de uma determinada iteração k, para ser posteriormente reiniciado do ponto em que parou na iteração $k+1$, bastando, para isso, armazenar o conjunto de pontos de linearização L. Através do armazenamento de L, o problema mestre pode ser corretamente reconstituído. Assim, de certo modo, o algoritmo de BBHAE pode ser visto como sendo a execução do algoritmo Aproximação Externa com interrupções (linha 34). Ao longo dessas interrupções, iterações de branch-and-bound não linear são executadas para explorar partições do espaço. Essas iterações de branch-and-bound podem vir a encontrar soluções viáveis para (P) que serão então adicionadas a L, para melhorar a relaxação

Algoritmo 8.9: Algoritmo híbrido de branch-and-bound e Aproximação Externa.

Entrada: (P): problema de PNLIM, ε_c: tolerância de convergência.
Saída: (x^*, y^*): solução ótima para (P).

1 Seja l_b^i o limitante inferior do nó i;
2 $L_P = \emptyset$; // Conjunto inicial de pontos de linearização para (P)
3 Seja $N = \{0\}$ o conjunto inicial de nós em aberto da árvore de BB;
4 $z^u = +\infty$; $Y^0 = Y$; $l_b^0 = -\infty$; $i = 0$; $iter = 0$;
5 enquanto $N \neq \emptyset$ faça
6 $\quad iter = iter + 1$;
7 $\quad$ Selecione um nó k de N; $N = N \setminus \{k\}$;
8 $\quad$ se $(\tilde{P}_{Y^k})$ é viável então
9 $\quad\quad$ Seja (x^k, y^k) uma solução ótima de $(\tilde{P}_{Y^k})$;
10 $\quad\quad$ se $f(x^k, y^k) < z^u - \varepsilon_c$ então
11 $\quad\quad\quad$ se y^k é inteira então
12 $\quad\quad\quad\quad z^u = f(x^k, y^k)$; $(x^*, y^*) = (x^k, y^k)$; $L_P = L_P \cup \{(x^k, y^k)\}$;
13 $\quad\quad\quad\quad N = N \setminus \{j : l_b^j \geq z^u - \varepsilon_c\}$; // Podas por limite
14 $\quad\quad\quad$ senão
15 $\quad\quad\quad\quad \bar{l}_b = f(x^k, y^k)$;
16 $\quad\quad\quad\quad$ se $iter \equiv 0 \pmod{freq_AE_subprob}$ and $k > 0$ então
$\quad\quad\quad\quad\quad$ // Aplicando AE ao subproblema
17 $\quad\quad\quad\quad\quad L_S = \{(x^k, y^k)\}$;
18 $\quad\quad\quad\quad\quad [status, (\bar{x}, \bar{y}), L_F, \bar{z}^l] = AE((P_{Y^k}), L_S, z^u, tempo_AE_subprob)$;
19 $\quad\quad\quad\quad\quad$ se $(\bar{x}, \bar{y})$ é viável para (P) então
20 $\quad\quad\quad\quad\quad\quad$ se $f(\bar{x}, \bar{y}) < z^u$ então
21 $\quad\quad\quad\quad\quad\quad\quad z^u = f(\bar{x}, \bar{y})$; $(x^*, y^*) = (\bar{x}, \bar{y})$;
22 $\quad\quad\quad\quad\quad\quad\quad N = N \setminus \{j : l_b^j \geq z^u - \varepsilon_c\}$; // Podas por limite
23 $\quad\quad\quad\quad\quad\quad L_P = L_P \cup \{(\bar{x}, \bar{y})\}$;
24 $\quad\quad\quad\quad\quad$ se $\bar{z}^l \geq z^u - \varepsilon_c$ OU ($status$ é "solução ótima" ou "problema inviável") então
25 $\quad\quad\quad\quad\quad\quad$ vá para linha 5;
26 $\quad\quad\quad\quad\quad \bar{l}_b = \max\{\bar{l}_b, \bar{z}^l\}$;
27 $\quad\quad\quad\quad$ Selecione uma variável y_j com valor fracionário y_j^k;
$\quad\quad\quad\quad$ // Ramificação
28 $\quad\quad\quad\quad Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \leq \lfloor y_j^k \rfloor\}$;
29 $\quad\quad\quad\quad Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \geq \lceil y_j^k \rceil\}$;
30 $\quad\quad\quad\quad N = N \cup \{i+1, i+2\}$;
31 $\quad\quad\quad\quad l_b^{i+1} = l_b^{i+2} = \bar{l}_b$;
32 $\quad\quad\quad\quad i = i + 2$;
33 $\quad$ se $iter \equiv 1 \pmod{freq_AE_prob}$ então
34 $\quad\quad [status, (\bar{x}, \bar{y}), L_F, \bar{z}^l] = AE((P), L_P, z^u, tempo_AE_prob)$; // Aplicando AE
35 $\quad\quad$ se $(\bar{x}, \bar{y})$ é viável e $f(\bar{x}, \bar{y}) < z^u$ então
36 $\quad\quad\quad z^u = f(\bar{x}, \bar{y})$; $(x^*, y^*) = (\bar{x}, \bar{y})$;
37 $\quad\quad\quad N = N \setminus \{j : l_b^j \geq z^u - \varepsilon_c\}$; // Podas por limite
38 $\quad\quad$ se $status$ é "solução ótima" então Pare o algoritmo ;
39 $\quad\quad L_P = L_P \cup L_F$; // Acumulando pontos de linearização

dada pelo problema mestre quando AE for reiniciado. Adicionalmente, o procedimento AE pode ser executado de modo independente como heurística em algumas partições do espaço durante as iterações de branch-and-bound (linha 18), com o objetivo de encontrar soluções naquela partição, ou ainda efetivamente solucioná-la no tempo limite imposto.

8.6 Considerações finais

8.6.1 Heurísticas

Algumas heurísticas propostas para PLIM foram adaptadas para PNLIM convexa. Por exemplo, [27, 28] apresentam adaptações de feasibility pump (FP) para o caso não linear. FP foi originalmente proposto para PLIM em [18] e é caracterizado por manter duas sequências de soluções. À primeira sequência é imposta a satisfação das restrições presentes na relaxação contínua de (P), sem necessariamente satisfazer a integralidade. À segunda sequência é imposta a satisfação da integralidade de (P), sem necessariamente satisfazer suas demais restrições. O algoritmo procura então fazer com que as duas sequências convirjam para a mesma solução, que, assim, atenderá a todas as restrições de (P). Em [28], é utilizada uma estratégia baseada na resolução de problemas de PLIM para obtenção da sequência de soluções inteiras de FP. Por sua vez, [27] utiliza para essa mesma sequência um esquema baseado em arredondamento. Também foram apresentadas versões de FP para problemas de PNLIM não convexos [54, 15].

Em [27], a heurística de mergulho (diving) é estendida para PNLIM. Essa heurística busca encontrar soluções viáveis simulando uma espécie de "mergulho" em um dos caminhos da árvore de enumeração de branch-and-bound. Em [164], são apresentadas as heurísticas baseadas na ideia de minimização de gap de integralidade (apresentada na seção 8.4.1), AMGI1 e AMGI2 (IGMA1 e IGMA2, em inglês). Uma versão da heurística de melhoramento RINS [56] (proposta originalmente para PLIM) também é apresentada por [27], onde uma busca por soluções de melhoramento é realizada através da fixação de um subconjunto de variáveis inteiras e da resolução do subproblema de PNLIM resultante. Também se valendo de subproblemas nos quais um subconjunto de variáveis estão fixas, a heurística RENS [20] procura pelo melhor arredondamento viável em torno de uma solução para a relaxação contínua de (P). Ainda nessa temática de fixação de um subconjunto de variáveis, [19] resolve subproblemas de PLIM para procurar por soluções viáveis para o problema (P). Uma outra heurística de melhoramento para PLIM baseada na resolução de subproblemas inteiros é a local branching [77], estendida para PNLIM em [169]. Por fim, [144] propõe uma heurística combinando elementos oriundos de metodologias distintas como local branching, branch-and-bound e meta-heurísticas.

8.6.2 Pacotes computacionais

Alguns pacotes computacionais (softwares) voltados para PNLIM convexa estão disponíveis nos dias de hoje. Um exemplo de pacote não comercial é o Muriqui [163, 161], que implementa a maioria dos algoritmos exatos discutidos neste capítulo, juntamente com algumas das heurísticas citadas. Uma série de parâmetros está disponível para customizar as execuções dos algoritmos. Outro resolvedor não comercial que lança mão de um conjunto de algoritmos é o Bonmin [29]. Dentre as opções comerciais, pode-se destacar o pacote Knitro [37]. Outras opções de pacotes comerciais e não comerciais estão listadas em [136].

9

Algoritmos de Aproximação

Flavio Miyazawa
Lehilton Pedrosa

9.1 Introdução

A área de otimização combinatória está repleta de problemas NP-difíceis cujas aplicações demandam soluções de instâncias de grande porte. Para muitos desses problemas, mesmo os melhores algoritmos exatos conhecidos levam tempo proibitivamente elevado. Assim, é natural buscar algoritmos que executam eficientemente, mas que ao mesmo tempo devolvam soluções com valor garantidamente próximo ao valor de uma solução ótima. Na abordagem de algoritmos de aproximação, entende-se por eficiente um algoritmo que executa em tempo polinomial; a garantia de proximidade é feita pela demonstração teórica de que o valor de uma solução devolvida é limitado por um fator do valor de uma solução ótima.

Um algoritmo de aproximação deve produzir soluções com valores próximos ao de uma solução ótima e, muitas vezes, a análise dessa proximidade deve ser feita sem o conhecimento desse valor. Nesse caso, o valor da solução devolvida deve ser comparado com um ou mais limitantes do valor ótimo. Normalmente, um limitante é obtido a partir de uma estrutura parecida com a solução ótima, mas que sabemos construir rapidamente. Por exemplo, em vez de obter um caminho hamiltoniano de um grafo de custo mínimo, podemos encontrar uma árvore geradora mínima. A pressão por soluções melhores leva a limitantes cada vez mais próximos do valor ótimo e, daí, a um conhecimento cada vez mais refinado da estrutura e da dificuldade do problema.

Tais limitantes podem ser obtidos de várias maneiras. Por exemplo, pode-se explorar a natureza combinatória do problema, seja analisando configurações particulares, seja realizando simplificações da instância. Outros limitantes resultam da relaxação de formulações do problema. Entre as incontáveis técnicas e ferramentas para obter limitantes e desenvolver algoritmos de aproximação, certamente aquelas baseadas em programação linear inteira ocupam um lugar de destaque. Neste capítulo, estudamos uma série de técnicas e algoritmos baseados, principalmente, na formulação por programa linear inteiro do problema e de sua relaxação.

O conteúdo deste capítulo está dividido em três seções principais. Na seção 9.2, apresentamos a notação utilizada bem como as principais definições. Para exemplificar esses conceitos, apresentamos alguns algoritmos de aproximação mais básicos, utilizando técnicas variadas e com diferentes garantias de aproximação. Na seção 9.3, introduzimos o uso da relaxação de programa linear inteiro como estrutura para se obter um limitante para o valor ótimo. Nessa abordagem, resolvemos a relaxação de uma formulação linear inteira e arredondamos a solução racional obtida, utilizando diferentes métodos, que podem ser determinísticos ou não determinísticos. Na seção 9.4, descrevemos diversos algoritmos primais-duais. Nesses algoritmos, em vez de resolvermos a relaxação de um programa linear inteiro, utilizamos sua formulação dual como guia na busca de uma solução integral aproximada. As referências para os algoritmos apresentados e outras notas bibliográficas estão concentradas na seção 9.5.

9.2 Definições e algoritmos básicos

Dado um problema de otimização Π, denotamos por $\mathcal{I}_\Pi$ o conjunto das instâncias de Π. Ainda, dado um algoritmo $\mathcal{A}$ para o problema Π, para cada $I \in \mathcal{I}_\Pi$, denotamos por $\mathcal{A}(I)$ o valor da solução produzida pelo algoritmo $\mathcal{A}$ quando esse recebe I como entrada e por $\mathrm{OPT}(I)$ o valor de uma solução ótima para I.

Se Π é um problema de minimização, dizemos que um algoritmo $\mathcal{A}$ para Π é um algoritmo de aproximação com fator de aproximação α, ou que é uma α-aproximação, se $\mathcal{A}$ executa em tempo polinomial e

$$\mathcal{A}(I) \leq \alpha\,\mathrm{OPT}(I), \quad \text{para todo } I \in \mathcal{I}_\Pi.$$

Para problemas de maximização, esses termos são definidos de maneira similar. Se Π é um problema de maximização, dizemos que um algoritmo $\mathcal{A}$ para Π é um algoritmo de aproximação com fator de aproximação α, ou que é uma α-aproximação, se $\mathcal{A}$ executa em tempo polinomial e

$$\mathcal{A}(I) \geq \alpha\,\mathrm{OPT}(I), \quad \text{para todo } I \in \mathcal{I}_\Pi.$$

Dizemos que uma família de algoritmos $\{A_\varepsilon\}$, para $\varepsilon > 0$, é um esquema de aproximação de tempo polinomial (Polynomial Time Approximation Scheme, PTAS) se A_ε tem complexidade de tempo polinomial no tamanho da instância, para cada ε constante, e fator de aproximação $1+\varepsilon$, se o problema for de minimização, e $1-\varepsilon$, se o problema for de maximização. Se a complexidade de tempo do algoritmo for da forma $O(f(1/\varepsilon)n^c)$, em que f é uma função arbitrária e c é uma constante que não depende de ε, um tal algoritmo é chamado de esquema de aproximação de tempo polinomial eficiente (Efficient PTAS, EPTAS). Se o algoritmo tiver complexidade de tempo polinomial tanto no tamanho da entrada quanto em $1/\varepsilon$, o algoritmo é chamado de esquema de aproximação de tempo completamente polinomial (Fully PTAS, FPTAS).

9.2.1 Um primeiro algoritmo de aproximação

Dado um conjunto de elementos E e uma família $\mathcal{S}$ de subconjuntos de E, dizemos que $\mathcal{T} \subseteq \mathcal{S}$ é uma cobertura de E se, para todo elemento $e \in E$, existir algum conjunto de $\mathcal{T}$ que contenha e. Tipicamente, cada conjunto em $\mathcal{S}$ corresponde a algum recurso que satisfaz a demanda de seus elementos, e devemos satisfazer todas as demandas em E. Como cada recurso tem um custo de utilização inerente, não necessariamente queremos usar todos os conjuntos de $\mathcal{S}$ para cobrir E. Com isso, um problema natural

é encontrar um subconjunto de $\mathcal{S}$ que cubra todos os elementos de E e cuja soma dos custos seja mínima.

O Problema da Cobertura por Conjuntos (Set Cover Problem, SC) é definido a seguir. Denote por $\mathcal{I}_{\mathrm{SC}}$ o conjunto de todas as tuplas $(E,\mathcal{S},w)$ em que E é um conjunto finito, $\mathcal{S}$ uma família de subconjuntos de E e w um vetor indexado em $\mathcal{S}$ com $w_S \geq 0$ para cada $S \in \mathcal{S}$. Além disso, suponha que, em toda tupla $(E,\mathcal{S},w) \in \mathcal{I}_{\mathrm{SC}}$, a família $\mathcal{S}$ seja uma cobertura de E, já que, caso contrário, o problema não teria solução.

Problema 9.2.1 (SC). Dada uma instância $(E,\mathcal{S},w) \in \mathcal{I}_{\mathrm{SC}}$, o objetivo do problema é encontrar uma cobertura $\mathcal{T} \subseteq \mathcal{S}$ de E tal que a soma $\sum_{S\in\mathcal{T}} w_S$ seja mínima.

SC é um problema NP-difícil [125], o que implica que não existe algoritmo exato e de tempo polinomial para ele sob a hipótese de que $\mathrm{P} \neq \mathrm{NP}$. Na verdade, sob essa hipótese pode-se dizer algo ainda mais forte: SC não tem fator de aproximação assintoticamente melhor do que ordem logarítmica no número de elementos [64]. Mais precisamente, mostra-se o teorema seguinte.

Teorema 9.2.1. Seja n o número de elementos em uma instância de SC e $\varepsilon > 0$ uma constante. Não há algoritmo de aproximação com fator $(1-\varepsilon)\ln n$ para SC, a menos que $\mathrm{P} = \mathrm{NP}$.

Em seguida, apresentamos um algoritmo guloso que adiciona iterativamente um conjunto por vez na solução em construção, até que todos os elementos de E estejam cobertos por algum dos conjuntos escolhidos. A cada iteração, o conjunto escolhido é aquele que possui a menor razão entre seu custo e o número de novos elementos cobertos pela sua escolha.

Algoritmo 9.1: SC-Guloso$(E,\mathcal{S},w)$

1 Faça $\mathcal{T} \leftarrow \emptyset$ e $N \leftarrow E$.
2 enquanto $N \neq \emptyset$ faça
3 Escolha $Z \in \mathcal{S}$ tal que $\frac{w_Z}{|Z\cap N|}$ seja mínimo.
4 Faça $\mathcal{T} \leftarrow \mathcal{T} \cup \{Z\}$ e $N \leftarrow N \setminus Z$.
5 Devolva $\mathcal{T}$.

No passo 1 de SC-Guloso, a variável $\mathcal{T}$ guarda a solução em construção, o que começa vazia, e a variável N guarda todos os elementos ainda não cobertos, o que inicialmente corresponde a todo conjunto E. A condição do laço do passo 2 garante que iremos escolher algum novo conjunto enquanto existir elementos a serem cobertos. Em cada iteração, no passo 3, é escolhido um conjunto Z de forma que o custo por elemento não coberto seja mínimo, e, no passo 4, esse conjunto Z é incorporado à solução em construção, atualizando o conjunto de elementos não cobertos.

No teorema a seguir, H_K denota o K-ésimo número harmônico, isto é, $H_K = \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{K}$.

Teorema 9.2.2. Seja $I = (E,\mathcal{S},w)$ uma instância de SC e defina $K = \max\{|S| : S \in \mathcal{S}\}$. A solução devolvida por SC-Guloso ao receber I como entrada tem valor de no máximo $H_K\,\mathrm{OPT}(I)$.

Demonstração. Considere a i-ésima iteração do laço do passo 2 do algoritmo e seja Z_i o conjunto escolhido nessa iteração. Além disso, seja N_i o conjunto de elementos não cobertos no início dessa iteração.

Em cada iteração, o conjunto Z_i é escolhido por corresponder à menor razão $\frac{w_{Z_i}}{|Z_i \cap N_i|}$, que é o custo de cobertura calculado por elemento. Com isso, podemos repassar o custo do conjunto Z_i para os elementos não cobertos em $Z_i \cap N_i$. Dividindo-se o valor de Z_i igualmente, cada elemento $e \in Z_i \cap N_i$ recebe um custo dado por $c_e = \frac{w_{Z_i}}{|Z_i \cap N_i|}$. Como o custo c_e é atribuído apenas uma vez a cada elemento, a soma dos custos dos conjuntos escolhidos também pode ser vista como a soma dos custos atribuídos por elemento, isto é,

$$\sum_{Z \in \mathcal{T}} w_Z = \sum_{e \in E} c_e.$$

Agora, considere uma solução ótima $\mathcal{T}^* \subseteq \mathcal{S}$. Considere também um conjunto S em $\mathcal{T}^*$ e denote por $\{e_{|S|}, e_{|S|-1}, \ldots, e_1\}$ os seus elementos. Sem perda de generalidade, suponha que o algoritmo SC-Guloso cubra os elementos de S na ordem $e_{|S|}, e_{|S|-1}, \ldots, e_1$. O algoritmo irá repassar um custo para cada um desses elementos. No início da iteração i em que um elemento e_j foi coberto, os elementos $e_j, e_{j-1}, \ldots, e_1$ não estavam cobertos. Assim, o próprio conjunto S era um candidato a ser escolhido no início dessa iteração e portanto o custo repassado ao elemento e_j pode ser estimado por

$$c_{e_j} = \frac{w_{Z_i}}{|Z_i \cap N_i|} \leq \frac{w_S}{|S \cap N_i|} \leq \frac{w_S}{j}.$$

Somando para todos os elementos, temos que o custo total dos elementos no conjunto S é dado por

$$\sum_{e \in S} c_e = \sum_{j=1}^{|S|} c_{e_j} \leq \sum_{j=1}^{|S|} \frac{w_S}{j} = H_{|S|}\, w_S. \tag{9.1}$$

Por fim, podemos computar o custo total por elemento como

$$\begin{aligned}
\sum_{e \in E} c_e &\leq \sum_{S \in \mathcal{T}^*} \sum_{e \in S} c_e \\
&\leq \sum_{S \in \mathcal{T}^*} H_{|S|}\, w_S \\
&\leq \sum_{S \in \mathcal{T}^*} H_K\, w_S = H_K\, \mathrm{OPT}(I).
\end{aligned}$$

A primeira desigualdade é válida porque $\mathcal{T}^*$ é uma cobertura, e a segunda desigualdade segue da desigualdade (9.1). □

Como $H_K \leq \ln K + 1$, esse teorema implica que SC-Guloso é uma $O(\log n)$-aproximação, em que n é o número de elementos em uma instância de SC. Embora ele seja um algoritmo bastante simples e possa parecer que técnicas mais sofisticadas possam obter fatores de aproximação muito melhores, o teorema 9.2.1 implica que não existe algoritmo com fator assintoticamente melhor, a não ser que $\mathrm{P} = \mathrm{NP}$.

9.2.2 Um segundo algoritmo de aproximação

O algoritmo visto anteriormente para SC é um algoritmo guloso simples que tem um fator de aproximação que depende da instância do problema. Vamos apresentar agora um outro algoritmo guloso, mas para o Problema de Escalonamento de Tarefas em Máquinas Paralelas Idênticas (Identical Machines Scheduling Problem, IMS). Para esse algoritmo, veremos que o fator de aproximação não depende da entrada.

Uma instância para IMS é composta de um conjunto de tarefas $J = \{1,\ldots,n\}$, um conjunto de máquinas $M = \{1,\ldots,m\}$ e um vetor p tal que $p_j \geq 0$ representa o tempo de processamento de uma tarefa $j \in J$ quando ela é escalonada em alguma máquina de M. Denote por $\mathcal{I}_{\text{IMS}}$ o conjunto de todas as tuplas (J,M,p). Um escalonamento de J em M é uma partição $\mathcal{M} = (M_1,\ldots,M_m)$ de J, em que M_i é o conjunto das tarefas que foram atribuídas à máquina i. O valor do escalonamento $\mathcal{M}$ é definido como $\text{val}(\mathcal{M}) = \max\left\{\sum_{j\in M_i} p_j : i \in M\right\}$, que é o menor tempo de processamento necessário para que todas as máquinas tenham terminado de executar suas tarefas correspondentes.

Problema 9.2.2 (IMS). Dada uma instância $(J,M,p) \in \mathcal{I}_{\text{IMS}}$, o objetivo do problema é encontrar um escalonamento $\mathcal{M}$ de J em M tal que $\text{val}(\mathcal{M})$ seja mínimo.

Vamos supor que $m \leq n$, uma vez que o caso $m > n$ é trivialmente resolvido com um algoritmo ótimo que escalona uma tarefa por máquina. Para uma instância $I = (J,M,p)$, defina $\mathcal{L}(I) = \max\left\{\max\left\{p_j : j \in M\right\}, \frac{1}{m}\sum_{j\in J} p_j\right\}$. O seguinte lema segue da definição de escalonamento e sua demonstração é deixada como exercício.

Lema 9.2.1. Se $I \in \mathcal{I}_{\text{IMS}}$, então $\mathcal{L}(I)$ é um limitante inferior para o valor de uma solução ótima.

O algoritmo IMS-Guloso atribui cada tarefa iterativamente à máquina que atualmente tiver o menor tempo de execução.

Algoritmo 9.2: IMS-Guloso (J,M,p)

1 para $i \in M$ faça
2 Faça $M_i \leftarrow \emptyset$.
3 para $j \in J$ faça
4 Seja $i \in M$ uma máquina tal que $\sum_{j\in M_i} p_j$ seja mínimo.
5 Faça $M_i \leftarrow M_i \cup \{j\}$.
6 Devolva o escalonamento $(M_1,\ldots,M_m)$.

Lema 9.2.2. Seja $I = (J,M,p) \in \mathcal{I}_{\text{IMS}}$ e defina $m = |M|$. Ao receber I como entrada, IMS-Guloso constrói em tempo polinomial um escalonamento de J em M com valor de no máximo $(2-1/m)\,\mathcal{L}(I)$.

Demonstração. Seja $\mathcal{M} = (M_1,\ldots,M_m)$ o escalonamento obtido pelo algoritmo para I. Além disso, seja t o índice da máquina com maior tempo de processamento em $\mathcal{M}$ e ℓ a última tarefa atribuída a M_t. Considere a iteração do algoritmo em que a tarefa ℓ foi atribuída e a máquina t foi escolhida. No início dessa iteração, a variável M_t é tal que $\sum_{j\in M_t} p_j$ é mínimo, ou seja, t corresponde à máquina com a menor carga já atribuída. Desse fato segue que, ao final da execução do algoritmo, cada máquina i tem tempo de processamento $\sum_{j\in M_i} p_j \geq \sum_{j\in M_t\setminus\{\ell\}} p_j$. Somando-se essas desigualdades para todo i, obtemos

$$\sum_{j\in J\setminus\{\ell\}} p_j = \sum_{i\in M}\sum_{j\in M_i\setminus\{\ell\}} p_j \geq \sum_{i\in M}\sum_{j\in M_t\setminus\{\ell\}} p_j = m\sum_{j\in M_t\setminus\{\ell\}} p_j.$$

Relembre que o valor do escalonamento é o maior tempo de processamento de uma máquina. Assim,

$$\begin{aligned}
\mathrm{val}(\mathcal{M}) &= \sum_{j \in M_\ell \setminus \{\ell\}} p_j + p_\ell \\
&\leq \frac{1}{m} \sum_{j \in J \setminus \{\ell\}} p_j + p_\ell \\
&= \frac{1}{m} \sum_{j \in J} p_j + \left(1 - \frac{1}{m}\right) p_\ell \\
&\leq \mathcal{L}(I) + \left(1 - \frac{1}{m}\right) \mathcal{L}(I). \qquad \square
\end{aligned}$$

Teorema 9.2.3. IMS-Guloso é uma 2-aproximação para IMS.

Demonstração. Segue dos lemas 9.2.1 e 9.2.2. □

9.2.3 Problemas em grafos e em espaços métricos

Iremos explorar alguns problemas que têm como parte da entrada um grafo com pesos nas arestas. Problemas com esse domínio são recorrentes em aplicações de projeto de rede, para os quais os pesos nas arestas representam custos de construir uma conexão entre dois nós. Um exemplo clássico dessa classe é o Problema da Árvore de Steiner (Steiner Tree Problem, ST). Esse problema ocorre em situações em que se deseja construir uma rede cabeada que conecta um certo conjunto de nós. Os nós podem ser terminais, que devem estar obrigatoriamente ligados na rede, ou não terminais, que não são obrigatórios, mas podem servir como pontos de roteamento para baratear o custo da rede.

Formalmente, denote por $\mathcal{I}_{\mathrm{ST}}$ o conjunto de todas as tuplas (G, R, c) em que G é um grafo conexo, R um subconjunto de vértices de G e c um vetor indexado em $E(G)$ tal que $c_e \geq 0$ represente o peso da aresta $e \in E(G)$.

Problema 9.2.3 (ST). Dada uma instância $(G, R, c) \in \mathcal{I}_{\mathrm{ST}}$, o objetivo do problema é encontrar uma árvore $T \subseteq G$ tal que $R \subseteq V(T)$ e a soma $\sum_{e \in E(T)} c_e$ seja mínima.

Um caso particular ocorre quando a função de pesos nas arestas corresponde a uma função de distância no conjunto de vértices, isto é, o grafo G é completo e o peso de uma aresta corresponde à distância entre os seus dois vértices. Para esses tipos de função de peso, vale a desigualdade triangular, o que significa que o peso de uma aresta entre dois vértices não é maior do que o peso de qualquer caminho entre eles. Uma definição mais formal é dada a seguir.

Definição 9.2.1. Considere um grafo completo G associado a uma função de pesos nas arestas c. Dizemos que a função c é métrica se $c_e \geq 0$ para toda aresta $e \in E(G)$ e $c_{uv} \leq c_{uw} + c_{wv}$ para todo trio de vértices $u, v, w \in V(G)$.

Também dizemos que uma instância $I \in \mathcal{I}_{\mathrm{ST}}$ é métrica quando a função de peso de I é métrica. A priori, resolver um problema com instâncias arbitrárias pode ser mais difícil do que resolver o mesmo problema restrito a instâncias métricas. No caso de ST, no entanto, restringir as funções de peso a métricas não torna o problema mais fácil. Mais precisamente, se uma instância I não é métrica, então podemos construir uma instância métrica I' de forma que uma solução de I' corresponda a uma solução de I

com o mesmo custo. Uma consequência disso é que podemos nos concentrar na versão métrica do problema. No lema a seguir, denotamos por $\mathrm{dist}_G(u,v)$ o comprimento com relação a c de um menor caminho de u a v em G.

Lema 9.2.3. Dada uma instância $I = (G,R,c)$ de ST, seja $I' = (G',R,c')$ uma instância métrica de ST, em que G' é um grafo completo em $V(G)$ e $c'_{uv} = \mathrm{dist}_G(u,v)$ para cada $u,v \in V(G)$. Então, (i) dada uma solução T' de I', podemos construir uma solução T de I tal que $\sum_{e\in E(T)} c_e \leq \sum_{e\in E(T')} c'_e$; e (ii) $\mathrm{OPT}(I) = \mathrm{OPT}(I')$.

Demonstração. A partir de uma solução T' de I', construa um grafo H da seguinte maneira: para cada aresta $(u,v) \in E(T')$, encontre um caminho de G, chamado P_{uv}, com custo $\mathrm{dist}_G(u,v) = c'_{uv}$, e adicione suas arestas a H. Observe que H é um subgrafo de G que contém todos os vértices de T'. Construa uma árvore geradora de H e chame-a de T. Como a árvore T' é uma solução de I', ela contém todos os terminais, então T também os contém. Concluímos que T é uma árvore e $R \subseteq V(T)$, ou seja, T é uma solução de I. O custo de T é

$$\sum_{e\in E(T)} c_e \leq \sum_{e\in E(H)} c_e \leq \sum_{e\in E(T')} \sum_{f\in E(P_e)} c_f = \sum_{e\in E(T')} c'_e.$$

Isso implica que $\mathrm{OPT}(I) \leq \mathrm{OPT}(I')$. Para o outro lado, perceba que uma solução para I também é uma solução para I' com custo menor ou igual, então $\mathrm{OPT}(I') \leq \mathrm{OPT}(I)$ e, portanto, $\mathrm{OPT}(I) = \mathrm{OPT}(I')$. □

Relembre que o Problema da Árvore Geradora Mínima (Minimum Spanning Tree, MST) é o problema de, dado um grafo conexo com pesos nas arestas, encontrar uma árvore com todos os vértices do grafo e cujo peso total seja mínimo. Repare que MST é, de fato, um caso particular de ST em que $R = V(G)$, isto é, em que todos os vértices são terminais. Ao contrário do ST, no entanto, conhecemos algoritmos polinomiais para MST, como o algoritmo de Prim ou de Kruskal. Isso sugere um algoritmo simples para resolver uma instância (G,R,c) de ST: primeiro, calcule o grafo induzido por terminais $G[R]$; depois, obtenha uma árvore geradora mínima de $G[R]$ com pesos c.

Para limitar o fator de aproximação desse algoritmo, precisamos comparar o valor da árvore geradora obtida com um valor limitante para o valor ótimo. Para isso, partiremos de uma solução ótima e construiremos uma árvore geradora de terminais. Para mostrar que essa árvore não tem custo muito maior do que o valor ótimo, vamos supor que a instância seja métrica.

Teorema 9.2.4. Existe uma 2-aproximação para ST.

Demonstração. Seja $I = (G,R,c) \in \mathcal{I}_{\mathrm{ST}}$ e T^* uma solução ótima para I. Duplique as arestas de T^* e obtenha um grafo H tal que todo vértice tenha grau par. Assim, H é euleriano e podemos obter uma rota C de H que passe por todas as arestas exatamente uma vez. Depois, construa um caminho P que passe por todos os vértices de R uma única vez, percorrendo C e fazendo atalhos nos vértices já visitados. Formalmente, seja $v_1, v_2, \ldots, v_r, v_1$ a sequência de vértices da rota C, e, sem perda de generalidade, suponha que $v_1 \in R$. Inicialize uma variável $i = 1$ e um caminho P formado apenas pelo vértice v_1. Agora, até que todos os vértices de R estejam em P, repita: seja v_j o primeiro vértice de $R \setminus V(P)$ na sequência de C que está após v_i, adicione (v_i, v_j) a P e faça $i = j$. Repare que cada aresta de P é da forma (v_i, v_j) com $i < j$. Pela desigualdade triangular, temos

$$c_{v_iv_j} \leq c_{v_iv_{i+1}} + c_{v_{i+1}v_{i+2}} + \cdots + c_{v_{j-1}v_j}.$$

Assim, $\sum_{(v_i,v_j)\in E(P)} c_{v_iv_j} \leq c_{v_1v_2} + c_{v_2v_3} + \cdots + c_{v_{r-1}v_r} \leq \sum_{e\in E(H)} c_e = 2\sum_{e\in E(T^*)} c_e$. Em outras palavras, P custa no máximo duas vezes o valor de uma solução ótima.

Finalmente, seja T uma árvore geradora de $G[R]$ de custo mínimo. Como P também é uma árvore geradora de $G[R]$, sabemos que T tem custo menor ou igual ao de P e, portanto, custa no máximo duas vezes o custo de T^*. Como T pode ser obtida em tempo polinomial, o teorema segue. □

Relembre que qualquer algoritmo de tempo polinomial para SC tem um fator de aproximação maior do que $(1-\varepsilon)\ln n$, para $\varepsilon > 0$. Isso implica que o fator de aproximação de qualquer algoritmo para SC deve depender da instância. No caso dos algoritmos para IMS e ST, os fatores de aproximação não dependem da instância. De fato, em ambos os casos, o fator é 2, que é um número constante. Isso nos permite dizer que esses dois problemas são mais fáceis de aproximar do que SC. A seguir, veremos que IMS é ainda mais fácil de aproximar.

9.2.4 Um esquema de aproximação

Apresentamos um esquema de aproximação de tempo polinomial para IMS. Para cada constante $\varepsilon > 0$, iremos projetar um algoritmo IMS-Ptas$_\varepsilon$ que obtém soluções cujo valor é de no máximo $1+\varepsilon$ vezes o valor de uma solução ótima.

O algoritmo particiona o conjunto de tarefas em duas partes, de acordo com o tempo de processamento: uma parte contém as tarefas grandes, que denotamos por G, e outra, as tarefas pequenas, que denotamos por P. A ideia por detrás dessa partição é que, de certa maneira, a estrutura de uma solução ótima é ditada pela atribuição das tarefas grandes, enquanto a escolha da máquina que recebe uma tarefa pequena altera pouco o valor da solução. Para as tarefas grandes, é possível obter um escalonamento quase ótimo em tempo polinomial. Uma vez que temos um escalonamento quase ótimo para as tarefas grandes, as tarefas pequenas serão escalonadas preenchendo espaços do escalonamento das tarefas grandes.

Em vez de encontrar um escalonamento para tarefas de G considerando os tempos de processamento dados pela função p, iremos encontrar um escalonamento para G utilizando os tempos de processamento dados por uma outra função p', na qual o número de tempos de processamento distintos é limitado por uma contante. A razão para isso é que, nessas condições, o número de configurações para as tarefas grandes é limitado e podemos encontrar uma solução ótima em tempo polinomial. Além disso, a diferença dos tempos de processamento de uma tarefa em p e em p' é limitada, então o escalonamento obtido para a instância com tempos de processamento dados por p' é quase ótimo para a instância com os tempos de processamento dados por p.

Mais precisamente, particionamos J da seguinte maneira. Para uma instância $I = (J, M, p)$ de IMS, seja $L = \mathcal{L}(I)$ o limitante inferior para uma solução ótima apresentado no lema 9.2.1. O conjunto das tarefas grandes é definido como $G = \{j \in J : p_j > \varepsilon L\}$ e o conjunto das tarefas pequenas como $P = \{j \in J : p_j \leq \varepsilon L\}$. Para cada $j \in G$, defina $p'_j = \varepsilon^2 L \lfloor p_j/\varepsilon^2 L \rfloor$, de forma que a diferença entre os valores de p_j e p'_j seja de no máximo $\varepsilon^2 L$. Como toda tarefa de G tem tempo de processamento de pelo menos εL e de no máximo L, a quantidade de valores distintos para os tempos de processamento das tarefas de G dados por p' é no máximo $\lceil (1-\varepsilon)L/(\varepsilon^2 L) \rceil \leq 1/\varepsilon^2 + 1$, que é uma constante. Inicialmente, o algoritmo constrói um escalonamento ótimo para a instância (G, M, p'). Para isso, iremos utilizar um programa linear inteiro.

Programa linear inteiro para tarefas grandes Suponha que os tempos de processamento distintos dados por p' sejam $s_1 < s_2 < \ldots < s_R$. O conjunto de tarefas atribuídas a uma máquina em uma solução ótima pode ser representado por um vetor de configuração $u = (u_1, u_2, \ldots, u_R)$, em que u_k é um inteiro não negativo que representa o número de tarefas escalonadas na máquina, que tem tempo de processamento s_k.

Pelo lema 9.2.2, uma solução ótima tem valor de no máximo $2L$. Assim, como o tempo de processamento de uma tarefa grande é pelo menos εL, o número de tarefas grandes em uma máquina é de no máximo $2/\varepsilon$ e, portanto, a soma das componentes de u é de no máximo $2/\varepsilon$.

Denote por $\mathcal{U}$ o conjunto de vetores de configuração que representam até $2/\varepsilon$ tarefas. Observe que $|\mathcal{U}|$ é limitado por uma constante, já que $|\mathcal{U}| \leq (2/\varepsilon)^R$. Além disso, denote por q_k o número de tarefas $j \in G$ tais que $p'_j = s_k$, para $1 \leq k \leq R$. Iremos construir um programa linear inteiro que, para cada $u \in \mathcal{U}$, contenha uma variável inteira não negativa x_u que represente a quantidade de máquinas com a configuração u e uma variável binária y_u que indique se a configuração u é usada na solução.

$$\begin{array}{lrll}
\text{Minimize} & T, & & \\
\text{sujeito a} & \sum_{u \in U} x_u & = m, & \\
& \sum_{u \in U} u_k x_u & = q_k, & \text{para todo } 1 \leq k \leq R, \\
& y_u \leq x_u & \leq m y_u, & \text{para todo } u \in \mathcal{U}, \\
& \sum_{k=1}^{R} s_k u_k y_u & \leq T, & \text{para todo } u \in \mathcal{U}, \\
& x_u \in \mathbb{N}, y_u & \in \{0,1\}, & \text{para todo } u \in \mathcal{U}.
\end{array} \tag{9.2}$$

Observe que o programa linear inteiro (9.2) resolve a instância (G, M, p') de IMS. A primeira restrição garante que temos exatamente uma configuração para cada uma das m máquinas. A segunda restrição garante que o número total de tarefas de tamanho s_k atende a demanda necessária de q_k. A terceira restrição relaciona as variáveis x_u com suas respectivas variáveis indicadoras de uso. A quarta restrição garante que T seja pelo menos o tempo de processamento das tarefas atribuídas a cada máquina, de forma que o valor da função-objetivo se iguale ao tempo de processamento da máquina mais carregada.

Podemos utilizar um algoritmo de força-bruta para resolver esse programa em tempo $m^{O(|\mathcal{U}|)}$, que é polinomial em m, mas cujo expoente depende de ε. A vantagem de utilizarmos um programa linear inteiro nesse caso é que, como o número de variáveis é limitado por uma constante, podemos usar o algoritmo de Lenstra [139], que é exponencial no número de variáveis, mas polinomial no logaritmo dos coeficientes do programa linear inteiro. Assim, o programa linear inteiro (9.2) pode ser resolvido em tempo $O(\log^{O(1)}(m+n))$.

Esquema de aproximação Uma vez que temos um escalonamento para as tarefas grandes, as tarefas pequenas são escalonadas por meio de um algoritmo de preenchimento. O esquema de aproximação é resumido em IMS-Ptas$_\varepsilon$.

O seguinte teorema afirma que esse algoritmo é um EPTAS e, mais do que isso, que executa em tempo linear.

Teorema 9.2.5. Para todo $\varepsilon > 0$, IMS-Ptas$_\varepsilon$ é uma $(1+\varepsilon)$-aproximação para IMS que executa em tempo linear.

Demonstração. Note que é suficiente obter uma aproximação com fator de aproximação $1+2\varepsilon$, uma vez que para qualquer $\varepsilon' > 0$, podemos utilizar esse algoritmo usando $\varepsilon = \varepsilon'/2$ e obter um fator de aproximação $1+\varepsilon'$.

Denote por I a instância (J, M, p) de IMS passada como entrada e sejam G e P os conjuntos definidos pelo algoritmo. Além disso, denote por I' a instância (G, M, p') de IMS construída pelo algoritmo e seja $\mathcal{M}_G$ a solução ótima de I' obtida no passo 5. Observe que, como G é um subconjunto de J e $p'_j \leq p_j$ para todo $j \in G$, uma solução ótima de I induz uma solução de I' de valor igual ou menor e, daí, $\mathrm{OPT}(I') \leq \mathrm{OPT}(I)$.

Algoritmo 9.3: IMS-Ptas$_\varepsilon$ (J, M, p)

1 Seja $L \leftarrow \max\left\{\max\left\{p_j : j \in J\right\}, \frac{1}{m}\sum_{j\in J} p_j\right\}$.
2 Faça $G \leftarrow \{j \in J : p_j > \varepsilon L\}$ e $P \leftarrow \{j \in J : p_j \leq \varepsilon L\}$.
3 para $j \in G$ faça
4 Defina $p'_j \leftarrow \varepsilon^2 L \lfloor p_j/\varepsilon^2 L \rfloor$.
5 Encontre escalonamento $(M_1, \ldots, M_m)$ de (G, M, p') resolvendo o programa linear inteiro (9.2).
6 para $j \in P$ faça
7 Seja i uma máquina tal que $\sum_{j\in M_i} p_j \leq L$.
8 Faça $M_i \leftarrow M_i \cup \{j\}$.
9 Devolva o escalonamento $(M_1, \ldots, M_m)$.

Considere o valor das variáveis M_i, $1 \leq i \leq m$, imediatamente após o passo 5. Defina T_G como o valor de $\max\{\sum_{j\in M_i} p_j : i \in M\}$ após esse passo e seja k uma máquina tal que $T_G = \sum_{j\in M_k} p_j$. Note que T_G é o valor do escalonamento $\mathcal{M}_G$ de acordo com os tempos de processamento dados por p e k é uma máquina com maior carga em $\mathcal{M}_G$. Como $p_j \leq p'_j + \varepsilon^2 L$ e há no máximo $2/\varepsilon$ tarefas na máquina k,

$$\begin{aligned} T_G &= \sum_{j\in M_k} p_j \\ &\leq \sum_{j\in M_k} (p'_j + \varepsilon^2 L) \\ &\leq \sum_{j\in M_k} p'_j + (2/\varepsilon)\varepsilon^2 L \\ &\leq (1+2\varepsilon)\mathrm{OPT}(I), \end{aligned}$$

em que a última desigualdade vale porque $\sum_{j\in M_k} p'_j \leq \mathrm{OPT}(I')$ e $L \leq \mathrm{OPT}(I)$.

Após as tarefas grandes serem escalonadas, cada tarefa pequena é escalonada em uma máquina com carga de no máximo L. Note que, como há m máquinas, a carga média de cada uma em qualquer escalonamento é de no máximo $\frac{1}{m}\sum_{j\in J} p_j \leq L$. Então existe uma máquina com carga de no máximo L, e o passo 7 está bem definido. Seja $\mathcal{M}$ o escalonamento devolvido pelo algoritmo com todas as tarefas. Além disso, seja t uma máquina com maior tempo de processamento em $\mathcal{M}$ e ℓ a última tarefa atribuída a M_t.

Se ℓ for uma tarefa grande, então M_t não contém nenhuma tarefa pequena. Isso implica que o valor de $\mathcal{M}$ é o mesmo de $\mathcal{M}_G$ e, portanto,

$$\mathrm{val}(\mathcal{M}) = T_G \leq (1+2\varepsilon)\mathrm{OPT}(I).$$

Suponha então que ℓ seja uma tarefa pequena. Nesse caso, imediatamente antes de ℓ ser escalonada, a carga de M_t era de no máximo L e, portanto,

$$\mathrm{val}(\mathcal{M}) = \sum_{j\in M_t\setminus\{\ell\}} p_j + p_\ell \leq L + \varepsilon L \leq (1+\varepsilon)\mathrm{OPT}(I).$$

Em ambos os casos, o escalonamento devolvido tem valor de no máximo $(1+2\varepsilon)\mathrm{OPT}(I)$. Para verificar que o algoritmo executa em tempo linear, observe que o programa linear inteiro (9.2) pode ser construído e resolvido em tempo linear e que as tarefas pequenas também podem ser escalonadas em tempo linear. □

9.3 Método do arredondamento

Nesta seção, veremos algumas técnicas para obter soluções inteiras a partir de relaxações de programas lineares inteiros. Apresentaremos primeiro as relações entre um programa linear inteiro e sua relaxação. Em seguida, veremos diferentes formas de se "arredondarem" soluções advindas de um programa linear relaxado.

9.3.1 Limitantes por programação linear

Para se ter um algoritmo de aproximação com bom fator de aproximação, é necessário ter bons limitantes para o valor de uma solução ótima que possam ser obtidos em tempo polinomial. Uma das principais técnicas utilizadas na obtenção desses limitantes é a formulação dos problemas de interesse como programas lineares inteiros, ou mesmo programas lineares inteiros mistos. De fato, muitos problemas de otimização NP-difíceis podem ser descritos na forma de programas lineares inteiros. Nesse caso, para cada instância I do problema original, cria-se um programa linear inteiro, que pode ser escrito na seguinte forma:

$$\begin{array}{rl} \text{Minimize} & c^\top x, \\ \text{sujeito a} & Ax \leq b, \\ & x \text{ é inteiro.} \end{array} \tag{9.3}$$

Por simplicidade, apresentamos definições adequadas a problemas de minimização, mas conceitos correspondentes a problemas de maximização podem ser definidos de maneira análoga. Além disso, embora por simplicidade escrevamos restrições com desigualdade do tipo menor ou igual, restrições de igualdade ou de desigualdade no sentido oposto também podem ser utilizadas.

A relaxação linear de um programa linear inteiro é obtida simplesmente com remoção da restrição de integralidade nas variáveis, mas com a manutenção de seus limitantes inferior e superior. Assim, a relaxação linear de (9.3) é dada por:

$$\begin{array}{rl} \text{Minimize} & c^\top x, \\ \text{sujeito a} & Ax \leq b. \end{array} \tag{9.4}$$

Apesar de as formulações (9.3) e (9.4) serem muito parecidas, a complexidade computacional para resolver cada uma delas pode ser bem distinta. Enquanto resolver a primeira formulação é um problema NP-difícil em geral, a segunda pode ser resolvida de maneira eficiente, isto é, em tempo polinomial.

Como toda solução que satisfaz as restrições do programa linear inteiro também satisfaz as restrições de sua relaxação, o conjunto das soluções viáveis do programa linear inteiro é um subconjunto das soluções de sua relaxação. Caso um vetor x satisfaça todas as restrições de um programa linear (inteiro), mesmo que possivelmente não minimize a função-objetivo, diremos que x é uma solução viável para tal programa. Se um problema Π puder ser formulado com a utilização da programação linear inteira, denotaremos por $\text{PLI}(I)$ o programa linear inteiro correspondente a uma instância $I \in \mathcal{I}_\Pi$ e por $\text{PL}(I)$ a relaxação de $\text{PLI}(I)$. Também denotaremos por $\text{OPT}(I)$ e $\text{OPT-LP}(I)$ os valores das soluções ótimas de $\text{PLI}(I)$ e $\text{PL}(I)$, respectivamente. Note que, para um problema de minimização, temos $\text{OPT-LP}(I) \leq \text{OPT}(I)$.

Por se tratarem de formulações muito parecidas, para muitos problemas, a mera remoção da restrição de integralidade das variáveis ainda mantém esses dois valores razoavelmente próximos. Com isso, temos na resolução dos programas lineares uma importante ferramenta de obtenção eficiente de limitantes para o valor de uma solução ótima. A maior razão entre o valor de um programa linear inteiro e o valor da

relaxação correspondente, considerando todas as instâncias do problema, é chamada de gap de integralidade.

Para problemas de maximização, a relaxação do programa linear inteiro nos dá um limitante superior para o valor de uma solução ótima, isto é, temos OPT-LP$(I) \geq$ OPT(I). Nesse caso, o gap de integralidade é a maior razão entre o valor da relaxação e o valor do programa linear inteiro.

A seguir, apresentaremos um primeiro método para se obterem algoritmos de aproximação baseados em programação linear, com o uso do arredondamento de suas variáveis.

9.3.2 Um algoritmo de arredondamento determinístico

Como um primeiro exemplo, vamos formular SC através de um programa linear inteiro. Para isso, usaremos uma variável binária x_S para indicar se um conjunto S está ou não na solução. Considere uma instância $(E, \mathcal{S}, w) \in \mathcal{I}_{\text{SC}}$ do problema. Diremos que um conjunto $S \in \mathcal{S}$ cobre um elemento $e \in E$ se $e \in S$. Para que um conjunto $\mathcal{T} \subseteq \mathcal{S}$ seja uma cobertura dos elementos de E, para cada elemento $e \in E$, pelo menos um dos conjuntos de $\mathcal{S}$ que cobre e deve estar em $\mathcal{T}$. Desse modo, se denotarmos por $\mathcal{S}_e = \{S \in \mathcal{S} : e \in S\}$ os conjuntos que cobrem e, pelo menos um conjunto de $\mathcal{S}_e$ deve estar na solução. Isso sugere a seguinte formulação:

$$\begin{aligned} \text{Minimize} \quad & \sum_{S \in \mathcal{S}} w_S x_S, \\ \text{sujeito a} \quad & \sum_{S \in \mathcal{S}_e} x_S \geq 1, && \text{para todo } e \in E, \\ & x_S \in \{0,1\}, && \text{para todo } S \in \mathcal{S}. \end{aligned} \tag{9.5}$$

A primeira restrição garante que todo elemento deve pertencer a algum conjunto da solução. A relaxação linear do programa acima é dada por:

$$\begin{aligned} \text{Minimize} \quad & \sum_{S \in \mathcal{S}} w_S x_S, \\ \text{sujeito a} \quad & \sum_{S \in \mathcal{S}_e} x_S \geq 1, && \text{para todo } e \in E, \\ & x_S \geq 0, && \text{para todo } S \in \mathcal{S}. \end{aligned} \tag{9.6}$$

Não é necessário limitar x_S superiormente por 1, uma vez que todos os pesos w_S são não negativos.

A ideia do algoritmo de arredondamento será escolher um valor γ e arredondar para 1 todas as variáveis x_S que atingirem pelo menos o limiar γ. Esse limiar será adequado para garantir um bom fator de aproximação e a viabilidade da solução obtida. Usaremos o limiar $\gamma = 1/\beta_{\max}$, onde $\beta_{\max} = \max\{|\mathcal{S}_e| : e \in E\}$.

Algoritmo 9.4: SC-Arred$(E, \mathcal{S}, w)$
1 Seja x uma solução ótima de (9.6).
2 Faça $\beta_{\max} \leftarrow \max\{|\mathcal{S}_e| : e \in E\}$ e $\mathcal{T} \leftarrow \{S \in \mathcal{S} : x_S \geq 1/\beta_{\max}\}$.
3 Devolva $\mathcal{T}$.

Teorema 9.3.1. Seja $I = (E, \mathcal{S}, w)$ uma instância de SC e defina $\beta_{\max} = \max\{|\mathcal{S}_e| : e \in E\}$. A solução devolvida por SC-Arred ao receber I como entrada tem valor de no máximo $\beta_{\max}$ OPT(I).

Demonstração. Considerando que toda instância admite uma solução, o programa linear (9.6) também é sempre viável, uma vez que o vetor x com todos os valores $x_S = 1$ satisfaz todas as restrições. Considere a solução ótima x do programa linear (9.6) obtida no passo 1 e seja $\mathcal{T}$ a família de conjuntos obtida no passo 2 do algoritmo.

Primeiro, vamos mostrar que $\mathcal{T}$ é uma cobertura de E. Suponha que essa afirmação seja falsa, isto é, que exista algum elemento $e \in E$ tal que nenhum conjunto de $\mathcal{S}_e$ tenha sido escolhido para estar em $\mathcal{T}$. Pela forma como $\mathcal{T}$ foi construído no passo 2, temos que $x_S < 1/\beta_{\max}$ para todo $S \in \mathcal{S}_e$. Daí,

$$\sum_{S \in \mathcal{S}_e} x_S < \sum_{S \in \mathcal{S}_e} \frac{1}{\beta_{\max}} = \frac{|\mathcal{S}_g|}{\beta_{\max}} \leq 1.$$

Assim, $\sum_{S \in \mathcal{S}_e} x_S < 1$, contrariando o fato de que x satisfaz a formulação (9.6).

Agora, vamos mostrar que o valor da cobertura $\mathcal{T}$ é no máximo $\beta_{\max}$ vezes o valor de uma solução ótima. De fato, o valor de $\mathcal{T}$ é

$$\begin{aligned}
\sum_{S \in \mathcal{T}} w_S &\leq \sum_{S \in \mathcal{T}} w_S x_S \beta_{\max} && (9.7)\\
&\leq \beta_{\max} \sum_{S \in \mathcal{S}} w_S x_S && (9.8)\\
&= \beta_{\max} \text{OPT-LP}(I) && (9.9)\\
&\leq \beta_{\max} \text{OPT}(I),
\end{aligned}$$

em que a desigualdade (9.7) vale porque $x_S \geq 1/\beta_{\max}$ e a desigualdade (9.9) vale porque tanto w_S como x_S são não negativos para todo $S \in \mathcal{S}$. □

A seguir, vamos considerar o Problema da Cobertura por Vértices (Vertex Cover Problem, VC), que é um caso particular de SC. Dado um grafo G, dizemos que um subconjunto $C \subseteq V(G)$ é uma cobertura por vértices de $E(G)$ se toda aresta $e \in E(G)$ tiver pelo menos um dos extremos em C. Denote por $\mathcal{I}_{\text{VC}}$ o conjunto de todas as tuplas (G, w) tal que G seja um grafo não direcionado e w seja um vetor indexado em v com $w_v \geq 0$ para todo $v \in V(G)$.

Problema 9.3.1 (VC). Dada uma instância $(G, w) \in \mathcal{I}_{\text{VC}}$, o objetivo do problema é encontrar uma cobertura por vértices $C \subseteq V(G)$ de $E(G)$ tal que a soma $\sum_{v \in C} w_v$ seja mínima.

A seguinte redução de VC para SC leva a um algoritmo com fator de aproximação 2. Note que, em VC, queremos cobrir todas as arestas utilizando os vértices do grafo. Considerando que um vértice v cobre uma aresta e se, e somente se, e é incidente em v, podemos definir para cada vértice $v \in V(G)$ um conjunto $S_v = \{e \in E(G) : e \text{ incide em } v\}$. Com isso, construímos uma instância $(E, \mathcal{S}, w)$ de SC, onde $E = E(G)$, $\mathcal{S} = \{S_v : v \in V(G)\}$ e $w_{S_v} = w_v$ para cada $v \in V(G)$. Observe que uma solução para a instância $(E, \mathcal{S}, w)$ de SC também é uma cobertura por vértices de G e que o valor dessa solução corresponde ao valor da cobertura por vértices.

Agora, considere a $\beta_{\max}$-aproximação para SC, onde $\beta_{\max} = \max\{|\mathcal{S}_e| : e \in E\}$. Dada uma aresta $e = (u, v)$, temos que $|\mathcal{S}_e| = 2$, pois e está contida em exatamente dois conjuntos: S_u e S_v. O seguinte resultado segue do teorema 9.3.1.

Teorema 9.3.2. Existe uma 2-aproximação para VC.

9.3.3 Arredondamento probabilístico

Assim como nas formulações anteriores, muitos dos programas lineares inteiros possuem variáveis binárias que representam a escolha ou não de certos objetos para a solução. Ao relaxar o programa linear inteiro, nem sempre as variáveis terão valores binários, podendo ter valores fracionários entre zero e um. De certa maneira, podemos interpretar que as variáveis com valor fracionário próximo de 1 têm maior chance de se tornarem 1 do que as variáveis com valor fracionário próximo de 0. Com isso, iremos interpretar o valor dessas variáveis como probabilidades de tais elementos pertencerem à solução.

Quando todas as decisões de um algoritmo dependem somente de sua entrada $I \in \mathcal{I}_\Pi$, dizemos que esse algoritmo é determinístico. Alguns dos algoritmos que veremos tomam decisões após a escolha de certos números aleatórios. Dizemos que um algoritmo é probabilístico se ele também puder fazer escolhas aleatórias, em geral, seguindo em cada uma alguma distribuição de probabilidade.

Diferentes execuções de um algoritmo probabilístico $\mathcal{A}$ podem levar a soluções distintas, então nesse caso a saída do algoritmo é em si uma variável aleatória. Com isso, o valor $\mathcal{A}(I)$ é uma variável aleatória que representa o valor da solução obtida pela execução do algoritmo $\mathcal{A}$ sobre a entrada I. Assim, em vez de nos preocuparmos com o valor $\mathcal{A}(I)$ da solução produzida, analisamos a esperança desse valor, $\mathrm{E}[\mathcal{A}(I)]$. Dizemos que um algoritmo probabilístico é uma α-aproximação probabilística se, para toda instância I, $\mathrm{E}[\mathcal{A}(I)] \leq \alpha\,\mathrm{OPT}(I)$ para um problema de minimização ou $\mathrm{E}[\mathcal{A}(I)] \geq \alpha\,\mathrm{OPT}(I)$ para um problema de maximização.

A seguir, apresentamos um algoritmo probabilístico, SC-Prob, para uma instância $I = (E, \mathcal{S}, w)$ de SC que obtém soluções com valor esperado de no máximo $\mathrm{OPT}(I)$. Neste algoritmo, permitimos que a saída seja inviável, mas as chances de ele falhar serão limitadas. Depois, construiremos um algoritmo que devolve uma solução de custo esperado $O(\log|E|)\mathrm{OPT}(I)$, mas que fornece uma saída inviável com probabilidade muito baixa. Embora já tenhamos visto um algoritmo determinístico com fator de aproximação da mesma ordem de grandeza, esse algoritmo ilustra como algoritmos probabilísticos podem devolver soluções viáveis boas com alta probabilidade.

Algoritmo 9.5: SC-Prob$(E, \mathcal{S}, w)$

1 Seja x uma solução do programa linear (9.6).
2 Seja $\mathcal{T} \leftarrow \emptyset$.
3 para $S \in \mathcal{S}$ faça
4 Inclua S em $\mathcal{T}$ com probabilidade x_S.
5 Devolva $\mathcal{T}$.

O próximo lema mostra que o valor esperado dos conjuntos escolhidos é no máximo o valor de uma solução ótima. Depois, mostraremos que a probabilidade de um elemento não ser coberto é limitada.

Lema 9.3.1. Seja $I = (E, \mathcal{S}, w)$ uma instância de SC. Se $\mathcal{T}$ é a solução devolvida por SC-Prob ao receber I como entrada, então $\mathrm{E}[\sum_{S \in \mathcal{T}} w_S] \leq \mathrm{OPT}(I)$.

Demonstração. Seja $\mathcal{T}$ a solução obtida SC-Prob e W o valor da solução $\mathcal{T}$. Para cada conjunto S, considere também uma variável aleatória binária $\bar{x}_S$ que indica se S está ou não em $\mathcal{T}$. Assim, $\bar{x}_S = 1$ se $S \in \mathcal{T}$ e $\bar{x}_S = 0$ caso contrário. Com essa definição,

podemos escrever $W = \sum_{S \in \mathcal{S}} w_S \bar{x}_S$. Observe que $\mathrm{E}[\bar{x}_S] = x_S$. Isso implica que o valor esperado da solução gerada seja

$$\begin{aligned}
\mathrm{E}[W] &= \mathrm{E}\left[\sum_{S \in \mathcal{S}} w_S \bar{x}_S\right] \\
&= \sum_{S \in \mathcal{S}} w_S \mathrm{E}[\bar{x}_S] \\
&= \sum_{S \in \mathcal{S}} w_S x_S \\
&= \text{OPT-LP}(I) \leq \text{OPT}(I).
\end{aligned}$$

□

Lema 9.3.2. Seja $I = (E, \mathcal{S}, w)$ uma instância de SC e $f \in E$. Se $\mathcal{T}$ é a solução devolvida por SC-Prob ao receber I como entrada, então $\Pr[f \notin \bigcup_{S \in \mathcal{T}} S] \leq \frac{1}{e}$.

Demonstração. Fixe $f \in E$ e seja $\mathcal{S}_f = \{S_1, \ldots, S_t\}$ a família dos conjuntos que contêm f.

$$\Pr\left[f \notin \bigcup_{S \in \mathcal{T}} S\right] = \prod_{S \in \mathcal{S}_f} \Pr[f \notin S] = \prod_{S \in \mathcal{S}_f} (1 - x_S) \tag{9.10}$$

$$\leq \prod_{S \in \mathcal{S}_f} e^{-x_S} = e^{-\sum_{S \in \mathcal{S}_f} x_S} \tag{9.11}$$

$$\leq \frac{1}{e}. \tag{9.12}$$

A primeira igualdade em (9.10) decorre do fato de que cada conjunto foi escolhido para pertencer a $\mathcal{T}$ independentemente dos outros conjuntos; a desigualdade em (9.11), do fato de que $1 - p \leq e^{-p}$ para qualquer p real; a desigualdade em (9.12), do fato de que, como x é solução viável do programa linear (9.6), $\sum_{S \in \mathcal{S}_f} x_S \geq 1$. □

Apesar de SC-Prob encontrar uma família de conjuntos com valor esperado no máximo igual ao de uma solução ótima, não necessariamente todos os elementos serão cobertos pelos conjuntos escolhidos. De fato, a probabilidade de algum elemento não ser coberto pode ser razoavelmente alta. Com isso, iremos fortalecer o processo, repetindo a escolha dos conjuntos de maneira a aumentar as chances de que todos os elementos sejam cobertos. Mais especificamente, queremos que os conjuntos escolhidos formem uma cobertura, com alta probabilidade, isto é, com probabilidade pelo menos $1 - O(1/n^c)$, para alguma constante $c > 0$. Equivalentemente, queremos que a probabilidade de que a família de conjuntos devolvida pelo algoritmo não seja uma cobertura seja de no máximo $O(1/n^c)$.

O seguinte algoritmo executa SC-Prob por k iterações e devolve a união dos conjuntos escolhidos.

Note que cada conjunto $\mathcal{T}_i$ é obtido por SC-Prob de maneira independente. Por um lado, a chance de que a união de todos os conjuntos $\mathcal{T}_1, \ldots, \mathcal{T}_k$ cubra todos os elementos de E é maior; por outro lado, o custo da família de conjuntos devolvida aumenta à medida que k aumenta. Assim, usaremos um valor de k de maneira a termos uma solução viável com alta probabilidade, mas cujo valor esperado seja relativamente baixo. O próximo lema mostra que, quando o algoritmo é executado com $k = 2\ln|E|$, teremos uma solução viável com alta probabilidade e um fator de aproximação probabilístico $O(\ln|E|)$.

Lema 9.3.3. Seja $I = (E, \mathcal{S}, w)$ uma instância de SC. Se $\mathcal{T}$ é a solução devolvida por SC-ProbMC$_k$ com parâmetro k quando recebe I como entrada, então $\mathrm{E}[\sum_{S \in \mathcal{T}} w_S] \leq k\,\text{OPT}(I)$.

Algoritmo 9.6: SC-ProbMC$_k$ $(E, \mathcal{S}, w)$
1 Seja x uma solução do programa linear (9.6).
2 para $i \leftarrow 1$ até k faça
3 Execute SC-Prob $(E, \mathcal{S}, w)$ utilizando x e obtenha uma solução $\mathcal{T}_i$.
4 Devolva $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \ldots \cup \mathcal{T}_k$.

Demonstração. Seja $\mathcal{T}_i$ o conjunto obtido ao final da i-ésima iteração do laço do passo 2. Considerando que cada conjunto foi obtido independentemente e que $\mathcal{T} = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_k$, temos

$$\begin{aligned} \mathrm{E}\left[\sum_{S \in \mathcal{T}} w_S\right] &\leq \mathrm{E}\left[\sum_{i=1}^{k} \sum_{S \in \mathcal{T}_i} w_S\right] \\ &\leq \sum_{i=1}^{k} \mathrm{E}\left[\sum_{S \in \mathcal{T}_i} w_S\right] \\ &\leq \sum_{i=1}^{k} \mathrm{OPT}(I) = k\,\mathrm{OPT}(I). \end{aligned}$$

□

O próximo lema mostra que a família de conjuntos obtida por SC-ProbMC$_k$ é uma cobertura com alta probabilidade.

Lema 9.3.4. Seja $I = (E, \mathcal{S}, w)$ uma instância de SC e defina $n = |E|$. Se $\mathcal{T}$ é a solução devolvida por SC-ProbMC$_k$ com parâmetro $k = 2 \ln n$ quando ele recebe I como entrada, então

$$\Pr[\mathcal{T} \text{ ser cobertura de } E] \geq 1 - \frac{1}{n}.$$

Demonstração. Sejam $\mathcal{T}_1, \ldots, \mathcal{T}_k$ os conjuntos obtidos ao final do laço do passo 2 de SC-ProbMC$_k$, com parâmetro $k = 2 \ln |E|$.

Considere um elemento $f \in E$ e denote por $\mathcal{E}_f$ o evento de f não ser coberto por $\mathcal{T}$. Como $\mathcal{T}$ é a união dos subconjuntos $\mathcal{T}_i$, f não é coberto por $\mathcal{T}$ se, e somente se, f não for coberto por nenhum dos conjuntos $\mathcal{T}_i$, para $i = 1, \ldots, k$. Pelo lema 9.3.2, a probabilidade de f não ser coberto por um conjunto $\mathcal{T}_i$ é de no máximo $\frac{1}{e}$. Como os conjuntos $\mathcal{T}_i$ foram criados independentemente, vale

$$\Pr\left[\mathcal{E}_f\right] = \prod_{i=1}^{2 \ln |E|} \Pr[f \text{ não ser coberto por } \mathcal{T}_i] \leq \left(\frac{1}{e}\right)^{2 \ln |E|} = \frac{1}{|E|^2}.$$

Usamos o limitante da união para mostrar que a probabilidade de algum elemento não ser coberto ainda é pequena.

$$\Pr[\mathcal{T} \text{ não ser cobertura}] = \Pr\left[\bigcup_{f \in E} \mathcal{E}_f\right] \leq \sum_{f \in E} \Pr\left[\mathcal{E}_f\right] \leq \sum_{f \in E} \frac{1}{|E|^2} = \frac{1}{|E|}.$$

□

Dos lemas 9.3.3 e 9.3.4, temos o seguinte teorema:

Teorema 9.3.3. Seja $I = (E, \mathcal{S}, w)$ uma instância de SC e defina $n = |E|$. SC-ProbMC$_k$ com parâmetro $k = 2 \ln n$ produz uma solução de I com valor de no máximo $2 \ln n\, \mathrm{OPT}(I)$ com alta probabilidade.

9.3.4 Explorando o politopo do programa linear

No Problema do Escalonamento de Tarefas em Máquinas Não Relacionadas (Unrelated Machines Scheduling Problem, UMS), a entrada é composta de um conjunto de tarefas $J = \{1,\ldots,n\}$, um conjunto de máquinas $M = \{1,\ldots,m\}$ e um vetor p tal que $p_{ij} \geq 0$ representa o tempo de processamento de uma tarefa $j \in J$ quando ela é escalonada na máquina $i \in M$. Denote por $\mathcal{I}_{\text{UMS}}$ o conjunto de todas as tuplas (J,M,p). Um escalonamento de J em M é uma partição $\mathcal{M} = (M_1,\ldots,M_m)$ de J, em que M_i corresponde às tarefas que foram atribuídas à máquina i. O tempo de processamento do escalonamento $\mathcal{M}$ é definido como $\text{val}(\mathcal{M}) = \max\left\{\sum_{j\in M_i} p_{ij} : i \in M\right\}$.

Problema 9.3.2 (UMS). Dada uma instância $(J,M,p) \in \mathcal{I}_{\text{UMS}}$, o objetivo do problema é encontrar um escalonamento $\mathcal{M}$ de J em M tal que $\text{val}(\mathcal{M})$ seja mínimo.

Em seguida, será apresentada uma 2-aproximação para UMS que explora propriedades dos pontos extremais do programa linear obtido relaxando-se a seguinte formulação:

$$\begin{array}{lrll} \text{Minimize} & T, & & \\ \text{sujeito a} & \sum_{i\in M} x_{ij} & = 1, & \text{para todo } j \in J, \\ & \sum_{j\in J} p_{ij}x_{ij} & \leq T, & \text{para todo } i \in M, \\ & x_{ij} & \in \{0,1\}, & \text{para todo } i \in M \text{ e } j \in J. \end{array} \tag{9.13}$$

A relaxação linear dessa formulação é obtida com a troca da última restrição $x_{ij} \in \{0,1\}$ por $x_{ij} \geq 0$. O gap de integralidade dessa formulação para sua relaxação, no entanto, é ilimitado. Para constatar esse fato, considere uma solução ótima fracionária para uma instância composta apenas de uma tarefa, isto é, $J = \{1\}$, tal que $p_{i1} = m$ para toda máquina $i \in M$. Uma solução ótima para essa instância tem valor igual a m. Porém, uma solução ótima para a formulação relaxada é dada pela atribuição $x_{i1} = 1/m$ para toda máquina i, e, com isso, temos uma solução fracionária com $T = 1$. Dessa forma, podemos construir instâncias para as quais o gap de integralidade, igual a m, seja tão grande quanto se queira.

Por outro lado, se conhecermos um número T correspondente a um tempo de processamento em que as tarefas devem ser finalizadas, podemos fixar $x_{ij} = 0$ sempre que $p_{ij} > T$, ou, equivalentemente, remover essa variável da formulação. Assim, vamos denotar por $\mathcal{E}(T) = \{(i,j) \in M \times J : p_{ij} \leq T\}$ as atribuições de tarefas para máquinas que podem existir em uma solução de tempo de processamento de no máximo T. Com isso, temos a formulação parametrizada em T:

$$\mathrm{P}(T) \quad \begin{array}{rll} \sum_{i:(i,j)\in\mathcal{E}(T)} x_{ij} & = 1, & \text{para todo } j \in J, \\ \sum_{j:(i,j)\in\mathcal{E}(T)} p_{ij}x_{ij} & \leq T, & \text{para todo } i \in M, \\ x_{ij} & \geq 0, & \text{para todo } (i,j) \in \mathcal{E}(T). \end{array} \tag{9.14}$$

Note que $\mathrm{P}(T)$ é um programa linear de viabilidade, já que não tem função-objetivo. Uma solução viável pode ser encontrada por algum algoritmo para programação linear convencional, bastando-se, para isso, definir a função-objetivo com coeficientes nulos. Para encontrar o menor valor T^* tal que o programa linear $\mathrm{P}(T^*)$ seja viável, podemos fazer uma busca binária em T. Seja x^* uma solução ótima fracionária de $\mathrm{P}(T^*)$. Utilizaremos um método para arredondar x^* e obter uma solução inteira dentro de um tempo de processamento de no máximo $2T^*$, analisando a estrutura por detrás dos valores não nulos de x^*.

Para as atribuições "inteiras" de tarefas para máquinas, isto é, para as quais $x^*_{ij} = 1$, o algoritmo simplesmente realiza essa atribuição, alocando a tarefa j para a máquina i. No caso das tarefas que foram alocadas de maneira fracionária para mais de uma máquina, o algoritmo obtém um emparelhamento no grafo bipartido que é induzido por essas atribuições fracionárias, atribuindo cada uma dessas tarefas a uma máquina distinta. O algoritmo é chamado de UMS-Arred.

Algoritmo 9.7: UMS-Arred (J, M, p)

1 Seja T^* o menor valor para o qual $\mathrm{P}(T^*)$ é viável.
2 Seja x^* uma solução que é um ponto extremal de $\mathrm{P}(T^*)$.
3 Seja G o grafo com vértices $V(G) = M \cup J$ e arestas $E(G) = \{(i,j) \in M \times J : 0 < x^*_{ij} < 1\}$.
4 Seja $A \subseteq E(G)$ um emparelhamento máximo de G.
5 para $i \in M$ faça
6 Defina $M_i \leftarrow \{j \in J : x_{ij} = 1\} \cup \{j \in J : (i,j) \in A\}$.
7 Devolva o escalonamento $(M_1, \ldots, M_m)$.

Note que UMS-Arred pode ser implementado em tempo polinomial. No passo 1, pode-se fazer uma busca binária de T no intervalo entre 0 e $\sum_{i \in M} \sum_{j \in J} p_{ij}$ [173] resolvendo o programa linear $\mathrm{P}(T)$ a cada passo. Como uma solução que é ponto extremal de um programa linear viável pode ser obtida em tempo polinomial [111], o passo 2 também pode ser executado em tempo polinomial. A construção do grafo G claramente ocorre em tempo polinomial, e a obtenção de um emparelhamento máximo também pode ser feita em tempo polinomial [146]. De fato, veremos que o grafo G é bem particular, então será suficiente usarmos um algoritmo bem mais simples para obtermos um emparelhamento máximo.

Lema 9.3.5. Seja $T \geq 0$. Se x é um ponto extremal do programa linear $\mathrm{P}(T)$, então x tem no máximo $n+m$ variáveis não nulas.

Demonstração. Seja x um ponto extremal de $\mathrm{P}(T)$ e v o número de variáveis em $\mathrm{P}(T)$. Pela Teoria de Politopos, temos que x deve satisfazer v restrições linearmente independentes com igualdade. Como há $n+m+v$ restrições em $\mathrm{P}(T)$, sendo v restrições da forma $x_{ij} \geq 0$, pelo menos $v-(n+m)$ restrições dessa forma são satisfeitas com igualdade. Assim, no máximo $v-[v-(n+m)] = n+m$ variáveis podem ser não nulas. □

Definição 9.3.1. Um grafo conexo é uma pseudoárvore se contém no máximo um circuito, e um grafo é uma pseudofloresta se cada uma de suas componentes conexas for uma pseudoárvore.

Dado o tempo de processamento limite T e a solução x de $\mathrm{P}(T)$, seja G_x o grafo com vértices $V(G_x) = M \cup J$ e arestas $E(G_x) = \{(i,j) \in \mathcal{E}(T) : 0 < x_{ij} < 1\}$.

Lema 9.3.6. Seja $T \geq 0$. Se x é ponto extremal de $\mathrm{P}(T)$, então G_x é uma pseudofloresta.

Demonstração. Seja C uma componente conexa de G_x, e M' e J' os conjuntos de máquinas e tarefas encontradas nas arestas de C. Sejam $\mathrm{P}_C(T)$ e x_C as restrições do programa linear $\mathrm{P}(T)$ e do vetor x para a componente conexa C e os conjuntos de máquinas e tarefas M' e J'.

Seja $x_{\overline{C}}$ o vetor das demais variáveis de x que não estão em x_C. Por simplicidade, reordene as coordenadas de forma que x possa ser estrito como a concatenação de x_C e $x_{\overline{C}}$, isto é, $x = x_C \,|\, x_{\overline{C}}$. Note que x_C é um ponto extremal de $\mathrm{P}_C(T)$. Para verificar esse fato, suponha por contradição que x_C não seja ponto extremal. Segue que existem soluções distintas x'_C e x''_C de $\mathrm{P}_C(T)$ e valores α e β com $0 < \alpha, \beta < 1$, tais que $x_C = \alpha x'_C + \beta x''_C$ e $\alpha + \beta = 1$. Seja $x' = x'_C | x_{\overline{C}}$ o vetor obtido pela concatenação dos vetores x'_C e $x_{\overline{C}}$, e $x'' = x''_C | x_{\overline{C}}$ o vetor obtido pela concatenação dos vetores x''_C e $x_{\overline{C}}$. Note que x' e x'' são soluções distintas de $\mathrm{P}(T)$. Segue que

$$\begin{aligned} x &= x_C \,|\, x_{\overline{C}} \\ &= (\alpha x'_C + \beta x''_C) \,|\, (\alpha x_{\overline{C}} + \beta x_{\overline{C}}) \\ &= \alpha (x'_C \,|\, x_{\overline{C}}) + \beta (x''_C \,|\, x_{\overline{C}}) \\ &= \alpha x' + \beta x''. \end{aligned}$$

Com isso, temos que x é combinação convexa de x' e x'', contrariando o fato de que x é ponto extremal de $\mathrm{P}(T)$. Portanto, x_C é ponto extremal de $\mathrm{P}_C(T)$.

Analogamente à formulação $\mathrm{P}(T)$, o programa linear $\mathrm{P}_C(T)$ possui $n' + m' + v'$ restrições, em que $n' = |J'|$, $m' = |M'|$ e v' é o número de variáveis em $\mathrm{P}_C(T)$. Dessas restrições, v' são da forma $x_{ij} \geq 0$. Como x_C é ponto extremal de $\mathrm{P}_C(T)$, segue do lema 9.3.5 que x_C possui no máximo $n' + m'$ variáveis não nulas. Como C é um grafo conexo com $n' + m'$ vértices, há $n' + m' - 1$ variáveis de x_C que formam uma árvore de C; portanto, C possui no máximo um circuito e é, assim, uma pseudoárvore de X. □

Lema 9.3.7. Seja $T \geq 0$. Se x é ponto extremal de $\mathrm{P}(T)$, então G_x possui um emparelhamento perfeito.

Demonstração. Note que o grafo G_x é bipartido e, pelo lema 9.3.6, é uma pseudoflorestа. Além disso, note que todas as folhas de G_x correspondem a máquinas, já que cada tarefa deve estar ligada de maneira "fracionária" a pelo menos duas máquinas.

Para construir um emparelhamento perfeito, comece com um emparelhamento vazio. Se G_x possui alguma aresta (i, j) em que i é folha, inclua a aresta (i, j) ao emparelhamento e remova i e j de C. Ao fazer isso, o grafo residual sem a máquina i e a tarefa j continuará sendo uma pseudofloresta em que cada tarefa restante ainda estará ligada a pelo menos duas máquinas distintas. Com isso, repita esse processo até que não haja mais folhas em G_x. Após tais remoções, obtemos ou um grafo vazio, ou um conjunto de circuitos. Como G_x é bipartido, sabemos que cada circuito deve ter tamanho par. Para cada um deles, escolhemos suas arestas alternadamente e as incluímos ao emparelhamento. □

Teorema 9.3.4. UMS-Arred é uma 2-aproximação para UMS.

Demonstração. Considere uma instância $I = (J, M, p)$ de UMS. Sejam T^* e x^* as variáveis obtidas nos passos 1 e 2 ao se executar UMS-Arred para I. Sabemos que T^* é um limitante inferior para o valor de uma solução ótima de I, já que uma solução ótima de valor T induz uma solução viável de $P(T)$ e T^* é mínimo tal que $P(T^*)$ seja viável. Portanto, $T^* \leq \mathrm{OPT}(I)$.

As atribuições de tarefas para máquinas são realizadas de duas maneiras. Em uma delas, consideram-se atribuições de tarefas j para máquinas i quando $x^*_{ij} = 1$; na outra, executa-se um emparelhamento perfeito em G_{x^*} construído a partir das variáveis $0 < x^*_{ij} < 1$. Na primeira parte, as atribuições realizadas com $x^*_{ij} = 1$ levam a um tempo de processamento de no máximo T^*, uma vez que x^* deve satisfazer a restrição $\sum_{j \in J} p_{ij} x^*_{ij} \leq T^*$, para todo $i \in M$, da formulação (9.14). As demais tarefas foram alocadas através de um emparelhamento perfeito A de G_{x^*}. Tal emparelhamento

garante que cada uma dessas tarefas restantes tenha sido alocada a uma máquina distinta. Assim, cada máquina i recebe, nesta segunda parte, a carga de no máximo uma tarefa adicional.

Portanto, a carga total de uma máquina i é de no máximo T^*, o que corresponde às tarefas atribuídas de maneira integral na primeira parte, somadas possivelmente com o tempo de processamento de mais uma tarefa p_{ij}, advinda do emparelhamento. Como cada aresta (i,j) do emparelhamento A pertence a $\mathcal{E}(T^*)$, temos que $p_{ij} \leq T^*$. Portanto, podemos delimitar o tempo de processamento máximo de um escalonamento ótimo como

$$T^* + \max\left\{p_{ij} : (i,j) \in A\right\} \leq 2T^* \leq 2\mathrm{OPT}(I). \qquad \square$$

9.4 Método primal-dual

Nesta seção, vamos apresentar o método primal-dual para obtermos algoritmos de aproximação. Enquanto a estratégia dos algoritmos de arredondamento é baseada em resolver a relaxação de um programa linear inteiro, no método primal-dual, utilizamos a relaxação apenas para obtermos um limitante para o valor ótimo. De fato, aqui estamos interessados em construir uma solução para o problema original com valor comparável ao de uma solução dual viável, que é construída simultaneamente com a solução.

9.4.1 Um algoritmo de crescimento dual

Considere uma instância $(E, \mathcal{S}, w)$ de SC. Relembre que E é um conjunto de elementos, $\mathcal{S}$ é uma família de conjuntos que cobre E e, para cada conjunto S, o custo associado é w_S. Além disso, denotamos por $\mathcal{S}_e$ a família de conjuntos que contêm e. Na seção 9.3.2, obtivemos uma $\beta_{\max}$-aproximação para SC utilizando arredondamento. Vamos obter o mesmo fator de aproximação, mas dessa vez usando o método primal-dual. Relembre que a relaxação do programa linear correspondente é

$$\begin{aligned} \text{Minimize} \quad & \sum_{S \in \mathcal{S}} w_S x_S, \\ \text{sujeito a} \quad & \sum_{S \in \mathcal{S}_e} x_S \geq 1, \quad \text{para todo } e \in E, \\ & x_S \geq 0, \quad \text{para todo } S \in \mathcal{S}. \end{aligned} \tag{9.15}$$

Iremos obter uma $\beta_{\max}$-aproximação sem resolver esse programa linear, mas ainda utilizando o valor da relaxação como um limitante inferior para o valor ótimo do problema. Para isso, primeiro calculamos o programa dual da relaxação:

$$\begin{aligned} \text{Maximize} \quad & \sum_{e \in E} y_e, \\ \text{sujeito a} \quad & \sum_{e \in S} y_e \leq w_S, \quad \text{para todo } S \in \mathcal{S}, \\ & y_e \geq 0, \quad \text{para todo } e \in E. \end{aligned} \tag{9.16}$$

O objetivo aqui não é resolver o problema dual, mas obter uma solução viável y cujo custo possa ser comparado com o valor da solução devolvida pelo algoritmo. Pelo Teorema Fraco da Dualidade, dada uma solução viável qualquer y para (9.16), o valor $\sum_{e \in E} y_e$ é um limitante inferior para o valor de (9.15). Assim, se o custo de uma solução para SC é de no máximo $\beta_{\max} \sum_{e \in E} y_e$, então essa solução custa no máximo $\beta_{\max}$ vezes o valor de (9.15) e, portanto, no máximo $\beta_{\max}$ vezes o valor de uma solução ótima.

O programa dual possui uma variável y_e para cada elemento e. Podemos interpretar y_e como a contribuição do elemento e a algum conjunto da solução devolvida que cobre e. Além disso, para cada conjunto S, uma solução viável y deve satisfazer a restrição $\sum_{e \in S} y_e \leq w_S$. Intuitivamente, podemos pensar que a contribuição de todos os elementos cobertos por S não pode ser maior do que o valor de S.

Por um lado, se, para algum elemento f, existir um conjunto S tal que $f \in S$ e $\sum_{e \in S} y_e = w_S$, então S já cobre e e seu custo pode ser limitado pela soma de variáveis duais correspondentes. Por outro lado, se para algum elemento f ocorre $\sum_{y_e} y_e < w_S$ para todo conjunto S que contenha f, então y_f pode ser aumentado em pelo menos $\varepsilon := \min\{w_S - \sum_{e \in S} y_e : S \in \mathcal{S}_f\}$ de forma que y ainda continue viável. Isso sugere o algoritmo a seguir. Comece com uma solução inviável vazia $\mathcal{T}$ para SC e com uma solução dual viável $y = 0$. Enquanto houver elemento f ainda não coberto, encontre um conjunto $S \in \mathcal{S}_f$ que minimize $w_S - \sum_{e \in S} y_e$, adicione S à solução $\mathcal{T}$ e aumente y_f de $w_S - \sum_{e \in S} y_e$.

Algoritmo 9.8: SC-PrimalDual$(E, \mathcal{S}, w)$

$\mathcal{T} \leftarrow \emptyset$.
$y \leftarrow 0$.
enquanto $\bigcup_{S \in \mathcal{T}} S \neq E$ faça
  Seja f tal que $f \notin \bigcup_{S \in \mathcal{T}} S$.
  Encontre S tal que $f \in S$ e $w_S - \sum_{e \in S} y_e$ seja mínimo.
  Aumente y_f de $w_S - \sum_{e \in S} y_e$.
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{S\}$.
Devolva $\mathcal{T}, y$.

Além da solução dual viável y para (9.16), o algoritmo também devolve uma solução $\mathcal{T}$ para SC, que corresponde a uma solução primal viável inteira x de (9.15). Note que o par de soluções x e y satisfazem folgas complementares primais, isto é, para todo conjunto S, se $x_S > 0$, então $\sum_{e \in S} y_e = w_S$, já que nesse caso $x_S = 1$, o que significa que o conjunto S foi escolhido pelo algoritmo. Se para todo elemento e com $y_e > 0$ também tivéssemos $\sum_{S \in \mathcal{S}_e} x_S = 1$, então x e y satisfariam também folgas complementares duais. Daí, x e y teriam o mesmo valor e a solução devolvida pelo algoritmo seria ótima. No entanto, se um elemento e for coberto por vários conjuntos selecionados, então $\sum_{S \in \mathcal{S}_e} x_S > 1$ e a igualdade não será satisfeita. Assim, relaxamos as folgas complementares duais e, em vez de mostrarmos que $\sum_{S \in \mathcal{S}_e} x_S = 1$, mostramos apenas que $\sum_{S \in \mathcal{S}_e} x_S \leq f$, para um certo fator f. Na prova do próximo teorema, $\beta_{\max}$ será tal fator.

Teorema 9.4.1. Seja $I = (E, \mathcal{S}, w)$ uma instância de SC, e defina $\beta_{\max} = \max\{|\mathcal{S}_e| : e \in E\}$. Então SC-PrimalDual, ao receber I como entrada, executa em tempo polinomial e devolve uma solução com valor de no máximo $\beta_{\max}\,\mathrm{OPT}(I)$.

Demonstração. Primeiro, observe que o algoritmo executa em tempo polinomial, porque, a cada iteração, pelo menos um novo elemento f é coberto, e, assim, há no máximo n iterações. Além disso, a cada iteração basta testar cada um dos conjuntos para encontrar o conjunto S que minimiza $w_S - \sum_{e \in S} y_e$ com $f \in S$.

Agora, limitamos o valor da solução $\mathcal{T}$ devolvida pelo algoritmo. Note que, uma vez que y_f é aumentado de exatamente $w_S - \sum_{e \in S} y_e$, no final da iteração que adiciona

S à solução temos $\sum_{e \in S} y_e = w_S$. Como depois dessa iteração todos os elementos de S já estão cobertos, a variável y_e não muda para cada $e \in S$, e sabemos que essa igualdade se mantém no final do algoritmo. Assim, podemos calcular o custo da solução $\mathcal{T}$ devolvida como

$$\sum_{S \in \mathcal{T}} w_S = \sum_{S \in \mathcal{T}} \sum_{e \in S} y_e.$$

Reescrevendo o somatório duplo, obtemos

$$\sum_{S \in \mathcal{T}} w_S = \sum_{e \in E} |\mathcal{S}_e| y_e \leq \beta_{\max} \sum_{e \in E} y_e,$$

em que a desigualdade decorre do fato de que cada elemento e está em no máximo $\beta_{\max}$ conjuntos. A última soma é exatamente o valor da função-objetivo do programa dual (9.16) para y. Como y é uma solução dual viável, essa soma é um limitante inferior para o valor ótimo; concluímos, então, que o valor de $\mathcal{T}$ é de no máximo $\beta_{\max} \mathrm{OPT}(I)$. □

9.4.2 Algoritmos primais-duais com duas fases

Nesta seção vamos utilizar o método primal-dual para projetar uma aproximação constante para o Problema da Localização de Instalações (Facility Location Problem, FL). Uma entrada do problema é composta de um conjunto de instalações F e um conjunto de clientes C. Uma instalação $i \in F$ pode ser aberta a um custo $f_i \geq 0$ e um cliente j pode ser conectado a uma instalação aberta i a um custo $c_{ij} \geq 0$. O objetivo do problema é escolher um subconjunto de instalações $S \subseteq F$ a serem abertas de forma a minimizar os custos. Denote por $\mathcal{I}_{\mathrm{FL}}$ o conjunto de todas as tuplas (C, F, c, f).

Problema 9.4.1 (FL). Dada uma instância $(C, F, c, f) \in \mathcal{I}_{\mathrm{FL}}$, o objetivo do problema é escolher um subconjunto $S \subseteq F$ tal que a soma $\sum_{i \in S} f_i + \sum_{j \in C} \min\{c_{ij} : i \in S\}$ seja mínima.

Esse problema aparece, por exemplo, na distribuição de mercadorias, quando é preferível abrir ou alugar armazéns próximos das unidades varejistas a pagar os custos de transporte de um único depósito. Normalmente, o custo de conexão de um cliente j a uma instalação i é proporcional à distância entre os dois pontos, como descrito na subseção 9.2.3. Assim, é natural presumir que $F \cup C$ esteja em um espaço métrico. De fato, sem essa restrição não podemos encontrar uma aproximação de fator constante para FL. Em seguida, vamos supor que a função de custo de conexão satisfaça a seguinte forma de desigualdade triangular: para todo par de clientes j, j' e par de instalações i, i', vale a seguinte desigualdade

$$c_{ij} \leq c_{ij'} + c_{i'j'} + c_{i'j}.$$

Podemos formular FL como o seguinte programa linear inteiro, que é similar ao programa (9.15) para SC:

$$\begin{array}{lll} \text{Minimize} & \displaystyle\sum_{i \in F} f_i x_i + \sum_{j \in C} \sum_{i \in F} c_{ij} x_{ij} & \\ \text{sujeito a} & \displaystyle\sum_{i \in F} x_{ij}, \geq 1, & \text{para todo } j \in C, \\ & x_{ij} \leq y_i, & \text{para todo } i \in F, j \in C, \\ & x_{ij}, y_i \in \{0,1\}, & \text{para todo } i \in F, j \in C. \end{array} \tag{9.17}$$

A variável x_{ij} indica se um cliente j está conectado à instalação i e a variável y_i indica se a instalação i deve ser aberta. O primeiro conjunto de restrições garante que cada cliente esteja conectado a pelo menos uma instalação e o segundo conjunto de restrições garante que cada cliente j só esteja conectado a uma instalação i se ela

estiver aberta. Substituindo as restrições do tipo $x_{ij}, y_i \in \{0,1\}$, por $x_{ij}, y_i \geq 0$, podemos calcular o programa linear dual:

$$\begin{array}{lll} \text{Maximize} & \sum_{j \in C} v_j, & \\ \text{sujeito a} & v_j \leq c_{ij} + w_{ij}, & \text{para todo } i \in F, j \in C, \\ & \sum_{j \in C} w_{ij} \leq f_i, & \text{para todo } i \in F, \\ & v_j, w_{ij} \geq 0, & \text{para todo } i \in F, j \in C. \end{array} \tag{9.18}$$

Novamente, vamos interpretar a variável dual v_j como a contribuição de um elemento j para o custo da solução inteira. Enquanto no algoritmo para SC só há custos para escolher conjuntos, os custos de FL são divididos em custos de conexão e custos de abertura. Assim, adotamos o seguinte mecanismo de cobrança: uma parte da contribuição v_j é usada para pagar os custos de conexão entre o cliente j e a instalação associada i, que corresponde à constante c_{ij}; e a parte remanescente é usada para pagar os custos de abertura da instalação i associada, que corresponde à variável w_{ij}.

No algoritmo primal-dual, queremos construir uma solução inteira (x,y) para o problema primal cujo custo seja dado em termos de uma solução dual viável maximal (v,w). Aqui, de novo iremos satisfazer as folgas complementares primais e depois mostrar que a solução obtida satisfaz uma relação das folgas complementares duais relaxada. Para satisfazer as folgas primais, queremos que: (i) se uma instalação i foi escolhida, isto é, se $y_i = 1$, então $\sum_{j \in C} w_{ij} = f_i$, e (ii) se um cliente j se conecta a uma instalação i, isto é, se $x_{ij} = 1$, então $v_j = c_{ij} + w_{ij}$.

Na seção anterior, utilizamos o crescimento das variáveis duais para encontrar um limitante para o problema primal. Uma característica daquele algoritmo é que as variáveis duais começam com o valor 0 e, a cada iteração, exatamente uma das variáveis duais é aumentada. Uma alternativa é aumentar o valor de um conjunto de variáveis simultaneamente. As variáveis cujos valores são aumentados durante uma iteração são chamadas de ativas e as variáveis cujos valores não aumentam são chamadas de inativas.

É conveniente descrever o algoritmo com a seguinte noção temporal: inicialmente todas as variáveis duais são inicializadas com 0 e algumas das variáveis são ativadas. À medida que o tempo passa, as variáveis ativas são aumentadas uniformemente até que um ou mais eventos ocorram. Esses eventos são associados a propriedades sobre as variáveis duais (que normalmente correspondem ao momento em que alguma das desigualdades lineares se torna justa). A cada evento, um conjunto de instruções é executado, que pode ativar ou desativar variáveis, abrir novas instalações ou conectar clientes a instalações abertas.

No algoritmo FL-CrescimentoDual, a variável v_j é aumentada até que o valor seja suficiente para conectar j a alguma instalação i. Nesse momento, se i estiver aberta, então v_j é desativada e j é conectado a i, senão w_{ij} começa a crescer na mesma velocidade que v_j para manter a viabilidade da solução dual e, nesse caso, j começa a contribuir para a abertura de i. Quando a contribuição total para uma instalação i se iguala ao custo de abertura, i é aberta e todos os clientes ainda desconectados que contribuem para a abertura de i são conectados a i. O algoritmo termina quando todos os clientes forem conectados e todas as variáveis forem desativadas.

Quando o algoritmo para, cada cliente j pode ser conectado a alguma instalação $\phi(j)$ em S com custo $c_{ij} \leq v_j$. Para definir $\phi(j)$, vamos considerar o momento em que a variável v_j foi desativada. Se v_j foi desativada durante o processamento do evento 1, então nesse momento $v_j = c_{ij} + w_{ij} \geq c_{ij}$ para alguma instalação i que foi incluída em S em uma iteração anterior e definimos $\phi(j) = i$. Se v_j foi desativada durante o

Algoritmo 9.9: FL-CrescimentoDual(C, F, c, f)

Faça $S \leftarrow \emptyset$, $y \leftarrow 0$, $w \leftarrow 0$.
Ative a variável v_j para cada $j \in C$.
enquanto houver variável ativa faça
  Aumente uniformemente as variáveis ativas até que algum evento aconteça.
  Evento 1: $v_j = c_{ij} + w_{ij}$ para algum par i, j.
    Se $i \notin S$, ative w_{ij}.
    Se $i \in S$, desative v_j e w_{lj} para todo $l \in F$.
  Evento 2: $\sum_{j \in C} w_{ij} = f_i$ para algum i.
    Inclua i em S.
    para $j \in C$, tal que $v_j \geq c_{ij}$, faça
      Se v_j está ativa, desative v_j e w_{lj} para todo $l \in F$.
Devolva S, v, w.

processamento do evento 2, então nessa iteração $v_j \geq c_{ij}$ para alguma instalação i que foi incluída em S durante esse evento e definimos $\phi(j) = i$.

A solução dada por S é muito econômica com os custos de conexão, já que cada cliente j é conectado a uma instalação $\phi(j)$ com $v_j \geq c_{\phi(j)j}$ e, portanto, o custo total de conexão é limitado pelo valor da solução dual $\sum_{j \in C} v_j$. O custo de abertura de instalações, no entanto, pode ser muito grande, porque um cliente j pode contribuir para múltiplas instalações. No pior caso, o custo de instalações é $|F|$ vezes o valor da solução dual. Para obtermos uma aproximação constante, queremos garantir que cada cliente só contribua para a abertura de uma única instalação. Para isso, numa segunda fase, iremos abrir apenas um subconjunto S' de S.

Construa um grafo auxiliar H cujos vértices são os elementos de S e em que há aresta entre dois vértices $i, i' \in S$ se, e somente se, existe $j \in C$ tal que $w_{ij} > 0$ e $w_{i'j} > 0$. Agora construa um conjunto independente maximal S' de H. Esse conjunto pode ser encontrado em tempo polinomial, por exemplo, realizando-se o seguinte procedimento: comece com S' vazio e, enquanto houver um elemento $i \in S \setminus S'$ que não tenha vizinho em S', adicione i a S'. Essa fase de limpeza é detalhada no algoritmo FL-LimpezaDual.

Algoritmo 9.10: FL-LimpezaDual(S, v, w)

Construa um grafo H com vértices S e arestas (i, i') se $w_{ij} > 0$ e $w_{i'j} > 0$ para algum $j \in C$.
Encontre um conjunto independente maximal S' de H.
Devolva S'.

O algoritmo primal-dual completo, FL-PrimalDual, corresponde às duas fases, de crescimento dual e de limpeza da solução.

Para uma instalação $i \in S'$, seja C_i o conjunto de clientes que dão uma contribuição

Algoritmo 9.11: FL-PrimalDual(C,F,c,f)
1 Execute FL-CrescimentoDual(C,F,c,f) e obtenha S, v, w. 2 Execute FL-LimpezaDual(S,v,w) e obtenha S'. 3 Abra instalações em S' e conecte cada cliente j à instalação mais próxima de S'.

positiva à instalação i, isto é, $C_i = \{j \in C : w_{ij} > 0\}$. Como S' é independente, os conjuntos C_i são disjuntos dois a dois. De fato, suponha por contradição que $j \in C_i \cap C_{i'}$ para instalações $i, i' \in S'$ distintas. Assim, $w_{ij} > 0$ e $w_{i'j} > 0$ e existe aresta em H que liga i a i', mas isso é uma contradição, porque S' é um conjunto independente. Portanto, de fato C_i e $C_{i'}$ são disjuntos para cada par $i, i' \in S'$. Seja $D = \bigcup_{i \in S'} C_i$ o conjunto de todos os clientes que contribuem para a abertura de alguma instalação.

Vamos limitar o custo da solução devolvida pelo algoritmo. Primeiro, iremos estimar o custo de conexão. Para cada cliente j, queremos encontrar uma instalação $\phi'(j)$ em S' que esteja próxima de j. Diferenciamos dois casos, dependendo da possibilidade de j contribuir para a abertura de alguma instalação de S'. Se $j \in C_i$ para algum $i \in S'$, então definimos $\phi'(j) = i$. Do contrário, seja $i = \phi(j)$ e considere duas possibilidades, a depender de $i \in S'$. Se esse for o caso, então defina $\phi'(j) = i$. Mas se $i \notin S'$, então existe $i' \in S$ que é vizinho de i em H, já que do contrário poderíamos incluir i em S' e S' não seria maximal; nesse caso, defina $\phi'(j) = i'$.

Lema 9.4.1. Seja $j \in C$.

1. Se $j \in D$, então $c_{\phi'(j)j} = v_j - w_{\phi'(j)j}$.
2. Se $j \notin D$, então $c_{\phi'(j)j} \leq 3v_j$.

Demonstração. Primeiro considere o caso em que $j \in D$, ou seja, $j \in C_i$ para algum $i \in S'$. Então $i = \phi'(j)$ e $w_{ij} > 0$ pela definição de C_i. Segue que durante a execução do algoritmo, w_{ij} foi ativada e, portanto, $v_j = c_{ij} + w_{ij}$.

Agora considere o caso em que $j \notin D$, ou seja $j \notin C_i$ para todo $i \in S'$. Seja $i = \phi(j)$ e $i' = \phi'(j)$. Se $i' = i$, então por definição de $\phi(j) = i$, vale $c_{ij} \leq v_j \leq 3v_j$, que é o que queríamos. Suponha então que $i' \neq i$. Pela definição de i', temos $i' \in S'$, mas $i \notin S'$. Então existe uma aresta em H que liga i a i' e, portanto, existe cliente $j' \in C$ tal que as variáveis $w_{ij'}$ e $w_{i'j'}$ sejam positivas. Como $w_{ij'} > 0$ e $w_{i'j'} > 0$, segue que $c_{ij'} \leq v_{j'}$ e $c_{i'j'} \leq v_{j'}$. Pela desigualdade triangular, concluímos que $c_{i'j} \leq c_{ij} + c_{ij'} + c_{i'j'} \leq v_j + v_{j'} + v_{j'}$.

Resta apenas mostrar que $v_{j'} \leq v_j$. Considere o momento em que a variável v_j foi desativada e que o cliente j foi conectado à instalação $\phi(j) = i$. Sabemos que $i \in S$ nesse momento e que, assim, todo cliente que havia contribuído para a instalação i já estava conectado ou foi conectado nesse instante. Como $w_{ij'} > 0$, o cliente j' contribuiu para i e, portanto, a variável $v_{j'}$ estava desativada ou foi desativada no tempo v_j. Segue que $v_{j'} \leq v_j$ e, portanto, $c_{i'j} \leq 3v_j$. □

Teorema 9.4.2. FL-PrimalDual é uma 3-aproximação para FL.

Demonstração. Primeiro verificamos que FL-CrescimentoDual executa em tempo polinomial. Para isso, observe que o número de eventos é limitado pelo número de restrições do programa dual (9.18) e que o momento em que o próximo evento acontece pode ser calculado em tempo polinomial. Como FL-LimpezaDual também executa em tempo polinomial, segue que FL-PrimalDual tem complexidade de tempo polinomial.

Basta então limitar o custo da solução S'. Como uma instalação i deve ter sido adicionada a S pelo evento 2, temos $f_i = \sum_{j \in C} w_{ij} = \sum_{j \in C_i} w_{ij} = \sum_{j \in C_i} w_{\phi'(j)j}$. Combinando com o lema 9.4.1, derivamos

$$\begin{aligned}
\sum_{i \in S'} f_i + \sum_{j \in C} \min\{c_{ij} : i \in S'\} &\leq \sum_{i \in S'} f_i + \sum_{j \in C} c_{\phi'(j)j} \\
&\leq \sum_{i \in S'} \sum_{j \in C_i} w_{\phi'(j)j} + \sum_{j \in D} \left(v_j - w_{\phi'(j)j}\right) + \sum_{j \in C \setminus D} 3v_j \\
&= \sum_{i \in D} w_{\phi'(j)j} + \sum_{j \in D} \left(v_j - w_{\phi'(j)j}\right) + \sum_{j \in C \setminus D} 3v_j \leq 3 \sum_{i \in C} v_j,
\end{aligned}$$

em que a igualdade decorre do fato de que os conjuntos C_i particionam D. □

9.4.3 Programas lineares de tamanho exponencial

Os programas lineares inteiros para SC e para FL vistos anteriormente têm tamanho polinomial. Isso não é uma condição para se utilizar o método primal-dual, já que não queremos resolver sua relaxação (como é o caso de algoritmos de arredondamento). A seguir, mostraremos como isso pode ser útil ao se formular um problema como um programa linear inteiro.

Vamos considerar o Problema da Árvore de Steiner com Coleta de Prêmios (Prize-Collecting Steiner Tree Problem, PCST). Nesse problema, a entrada é composta de um grafo não direcionado G com custo $c_e \geq 0$ para cada aresta $e \in E(G)$ e penalidade $p_v \geq 0$ para cada vértice $v \in V(G)$, além de um vértice especial $r \in V(G)$, chamado de raiz. Queremos encontrar uma árvore em G que contenha a raiz r de forma a minimizar a soma de custos das arestas da árvore e das penalidades de vértices que não estejam na árvore. Denote por $\mathcal{I}_{\text{PCST}}$ o conjunto de todas as tuplas (G, c, p, r).

Problema 9.4.2 (PCST). Dada uma instância $(G, c, p, r) \in \mathcal{I}_{\text{PCST}}$, o objetivo do problema é encontrar uma árvore T tal que $r \in V(T)$ e a soma $\sum_{e \in E(T)} c_e + \sum_{v \in V(G) \setminus V(T)} p_v$ seja mínima.

Esse problema aparece em aplicações em que é preferível deixar de conectar algum vértice v a fim de diminuir custos associados a conexão, caso em que se paga uma penalidade ou multa de valor p_v. Observe que o caso em que existe um conjunto $R \subseteq V(G)$ tal que $p_v = \infty$ para cada $v \in R$ e $p_v = 0$ para $v \in V(G) \setminus R$ corresponde à instância (G, R, c) de ST. Assim como em ST, vamos supor, sem perda de generalidade, que a função de custo nas arestas seja métrica. Se esse não for o caso, então criamos uma nova instância em um grafo completo e cujo peso entre dois vértices é dado pelo comprimento do caminho mais curto entre eles.

Considere uma instância de PCST e denote por T uma solução viável correspondente. Para cada aresta e do grafo, crie uma variável binária x_e que indique se e está na solução, isto é, $x_e = 1$ se $e \in E(T)$, e $x_e = 0$ caso contrário. Além disso, para cada subconjunto X de vértices, crie uma variável binária z_X que indique se X é o conjunto de vértices não incluídos na solução, isto é, $z_X = 1$ se $X = V(G) \setminus V(T)$, e $z_X = 0$ caso contrário. Para um conjunto de vértices $S \subseteq V(G)$, defina $p(S) = \sum_{v \in S} p_v$ e denote por $\delta(S)$ o conjunto das arestas que tenham uma extremidade em S e outra em $V(G) \setminus S$.

$$\begin{aligned}
\text{Minimize} \quad & \sum_{e \in E(G)} c_e x_e + \sum_{X \subseteq V(G) \setminus \{r\}} p(X) z_X, \\
\text{sujeito a} \quad & \sum_{e \in \delta(S)} x_e + \sum_{X : S \subseteq X} z_X \geq 1, && \text{para todo } S \subseteq V(G) \setminus \{r\}, \\
& x_e \in \{0,1\}, && \text{para todo } e \in E(G), \\
& z_X \in \{0,1\}, && \text{para todo } X \subseteq V(G) \setminus \{r\}.
\end{aligned} \tag{9.19}$$

Para verificar que a formulação está correta, consideramos uma solução do programa linear inteiro (x,z) e construímos uma solução T do problema original de valor menor ou igual. A solução (x,z) corresponde a um conjunto de arestas que induz um grafo H em V. Seja T uma árvore geradora mínima da componente conexa de H que contém r. Observe que T induz uma solução viável (x',z') em que $x'_e = 1$ para cada $e \in E(T)$ e $z'_X = 1$ para $X = V(G) \setminus V(T)$. De fato, para cada restrição correspondente a $S \subseteq V(G) \setminus \{r\}$, ou $S \subseteq X$, ou S separa alguma aresta de T. No primeiro caso, $\sum_{X:S\subseteq X} z'_X \geq 1$ e, no segundo, $\sum_{e\in\delta(S)} x'_e \geq 1$. Além disso, o valor de (x',z') é menor ou igual ao valor de (x,z). Para verificar isso, note que T é subgrafo de H e que, como nenhuma aresta de H cruza $V(T)$, $z_Y = 1$ para algum conjunto Y que contém X. Isso implica que o custo das arestas de x' é de no máximo o custo das arestas de x e que o valor das penalidades de z' é de no máximo o valor das penalidades de z.

Substituindo as restrições $x_e \in \{0,1\}$ e $z_X \in \{0,1\}$ por $x_e \geq 0$ e $z_X \geq 0$, obtemos a relaxação linear do problema. Calculando o dual da relaxação, obtém-se:

$$\begin{array}{lll} \text{Maximize} & \displaystyle\sum_{S\subseteq V(G)\setminus\{r\}} y_S, & \\ \text{sujeito a} & \displaystyle\sum_{S:e\in\delta(S)} y_S \leq c_e, & \text{para todo } e \in E(G), \\ & \displaystyle\sum_{S\subseteq X} y_S \leq p(X), & \text{para todo } X \subseteq V(G)\setminus\{r\}, \\ & y_S \geq 0, & \text{para todo } S \subseteq V(G)\setminus\{r\}. \end{array} \tag{9.20}$$

Vamos agora criar um algoritmo, PCST-PrimalDual, baseado no método primal-dual para PCST. Assim como no algoritmo da seção anterior, construímos uma solução dual y, que inicialmente vale 0, e mantemos um conjunto de variáveis duais ativas que são aumentadas uniformemente até que algum evento ocorra. Como o número de variáveis é exponencial, somente os valores das variáveis ativas são mantidos explicitamente.

Durante a execução do algoritmo, construímos e mantemos uma floresta F que contenha os vértices $V(G)$. Diremos que um conjunto de vértices S está conectado se S estiver contido na componente conexa que contém a raiz. Diremos também que um conjunto de vértices C está saturado se a restrição dual correspondente estiver satisfeita com igualdade, isto é, se $\sum_{S\subseteq C} y_S = p(C)$. Cada componente conexa de F correspondente a um conjunto de vértices S poderá estar ativa ou inativa. As variáveis ativas irão representar componentes conexas que ainda não foram conectadas e nem estão saturadas. Uma variável y_S está ativa em um determinado momento da execução se S for uma componente conexa ativa de F.

O algoritmo de crescimento das variáveis duais funciona da seguinte maneira: no início da execução, F é composta apenas de vértices isolados. Assim, cada vértice, com exceção da raiz, é uma componente conexa ativa. Quando as variáveis ativas aumentam, dois eventos podem acontecer. O primeiro acontece quando uma componente conexa C de F se torna saturada; nesse caso, desativamos y_C e continuamos o algoritmo. O segundo evento acontece quando a restrição dual correspondente a uma aresta e fica justa; nesse caso, as variáveis correspondentes às duas componentes conexas ligadas por e são desativadas e a aresta e é adicionada a F, formando uma nova componente conexa. Essa componente conexa ficará ativa se não tiver a raiz e inativa se a tiver.

O algoritmo termina quando todas as variáveis desativadas. Observe que, após a execução do algoritmo de crescimento das variáveis duais, a componente conexa F' de F que contém a raiz induz uma solução para o problema primal cujo custo corresponde à soma dos custos das arestas de F' e das penalidades de $V(G)\setminus V(F')$. Essa solução, no entanto, pode ser muito cara e, como na seção anterior, após o crescimento guloso, é realizada uma fase de limpeza. Suponha que exista uma aresta

e cuja remoção deixa uma componente conexa C que é saturada e se liga à raiz por meio de e. Então removemos e e toda aresta de F' com ambos os extremos em C. Esse procedimento é repetido até que não haja mais arestas a serem retiradas.

Algoritmo 9.12: PCST-PrimalDual(G,c,p,r)

Faça $F \leftarrow (V(G),\emptyset)$, $y \leftarrow 0$.
Ative a variável f_S para cada $S = \{v\}$, com $v \in V(G) \setminus \{r\}$.
enquanto houver variável ativa faça
  Aumente uniformemente as variáveis ativas até que algum evento aconteça.
  Evento 1: $\sum_{S \subseteq C} y_S = p(C)$ para alguma componente conexa C de F.
    Desative y_C.
  Evento 2: $\sum_{S: e \in \delta(S)} y_S = c_e$ para alguma aresta e.
    Sejam S_1, S_2 as componentes conexas de F ligadas por e.
    Adicione e a F e desative y_{S_1} e y_{S_2}.
    Se $r \notin S_1 \cup S_2$, ative $y_{S_1 \cup S_2}$.
Seja F' a subárvore de F que contém r.
para $e \in E(F')$ na ordem inversa de inserção faça
  Seja $C \subseteq V(G) \setminus \{r\}$, tal que C seja componente conexa de $F' - e$.
  Se $\sum_{S \subseteq C} y_S = p(C)$, então remova aresta e e vértices C de F'.
Devolva F'.

Uma maneira de entender o algoritmo é interpretar geometricamente a variável y_S de cada componente conexa S como um fosso de largura y_S ao redor dos elementos de S. Como os fossos não podem se sobrepor, o comprimento de uma aresta que liga duas componentes conexas deve ser maior ou igual à soma dos raios de todos os fossos por que passa essa aresta. Quando o comprimento de uma aresta que liga dois fossos S_1 e S_2 for igual a essa soma, então os dois fossos que circundam S_1 e S_2 encostam-se e um novo fosso começa a aumentar em volta de $S_1 \cup S_2$. Outra possibilidade é de que a soma de y_S entre todas as componentes conexas S que componham uma componente conexa maximal ativa C se torne igual à penalidade associada a C; nesse caso, é preferível pagar a penalidade a continuar aumentando y_C.

Algumas observações são importantes para a análise do algoritmo. Primeiro, note que, quando ativamos uma variável y_S, S é a união de duas componentes conexas S_1 e S_2 inativas. Assim, a coleção dos conjuntos $S \subseteq V(G) \setminus \{r\}$ com $y_S > 0$ forma uma família laminar, isto é, para todo par de conjuntos $S_1, S_2 \subseteq V(G) \setminus \{r\}$ com $y_{S_1} > 0$ e $y_{S_2} > 0$, ou a interseção de S_1 e S_2 é vazia, ou um conjunto inclui o outro. Além disso, como as variáveis ativas y_S correspondem a componentes conexas de F, o evento 2 só pode ocorrer para uma aresta e que cruze duas componentes conexas de F e, portanto, mantém-se a invariante de que F é uma floresta.

PCST-PrimalDual devolve uma solução F' para o problema original e uma solução viável y para o programa dual. A seguir, mostraremos que o custo de F' é no máximo

duas vezes o valor da solução y. Se $X = V(G) \setminus V(F')$, então o custo da solução devolvida é dado pelos custos das arestas da árvore F' mais o custo das penalidades dos vértices não cobertos, X. Assim como nas seções anteriores, queremos criar um esquema de cobrança de forma que o custo de cada aresta de F' e a penalidade dos vértices em X sejam expressos em termos de y_S, para $S \subseteq V(G) \setminus \{r\}$.

O custo da solução devolvida é expresso como a soma de dois termos: a soma total das penalidades de X, dada por $p(X)$, e o custo das arestas da árvore F', dado por $\sum_{e \in E(F')} c_e$. Primeiro, vamos considerar as penalidades. O lema seguinte mostra que as penalidades de X podem ser pagas apenas pelas componentes conexas completamente contidas em X.

Lema 9.4.2. Se $X = V(G) \setminus V(F')$, então $\sum_{S \subseteq X} y_S = p(X)$.

Demonstração. Primeiro, observe que X pode ser particionado em um conjunto de componentes conexas saturadas $X_1, X_2, \ldots, X_s$. Para isso, considere um vértice $v \in X$ imediatamente antes da fase de limpeza. Se, nesse momento, v não estava conectado à raiz, então a componente conexa C que continha v não estava conectada. Como a variável y_C foi desativada em alguma iteração, concluímos que nesse caso C está saturada. Do contrário, v estava conectado à raiz por uma ou mais arestas removidas na fase de limpeza. Seja e a aresta que, ao ser removida, desconecta v da raiz e seja C a componente conexa conectada à raiz através de e. Então ou a componente conexa C estava saturada, ou e não teria sido removida. Argumentando similarmente para cada vértice, concluímos a observação.

Finalmente,

$$p(X) = \sum_{v \in X} p_v = \sum_{i=1}^{s} \sum_{v \in X_i} p_v = \sum_{i=1}^{s} \sum_{S \subseteq X_i} y_S = \sum_{S \subseteq X} y_S. \qquad \square$$

Agora, consideramos o custo das arestas de F'. Para cada aresta $e \in E(F')$, sabemos que $\sum_{S: e \in \delta(S)} y_S = c_e$ pela definição do evento 2. Então, o custo das arestas é dado por

$$\sum_{e \in E(F')} \sum_{S: e \in \delta(S)} y_S.$$

Queremos reescrevê-lo como um fator da função-objetivo de (9.20), de forma que para cada conjunto de vértices $S \subseteq V(G) \setminus \{r\}$, a variável y_S apareça apenas em um termo da soma. Para cada S, o valor de $|\delta(S) \cap E(F')|$ conta o número de arestas de F' para as quais y_S contribui; então, o custo das arestas pode ser escrito como

$$\sum_{S \subseteq V(G) \setminus \{r\}} |\delta(S) \cap E(F')| y_S.$$

Embora, para cada componente conexa S, a variável y_S possa contribuir para diversas arestas de F', o lema a seguir mostra que, na média, cada componente conexa não contribui para mais do que duas arestas.

Lema 9.4.3. Seja $n = |V(G)|$, $X = V(G) \setminus V(F')$ e $\mathcal{C}$ o conjunto de componentes conexas ativas no início de uma dada iteração de PCST-PrimalDual e que não estão contidas em X. Então,

$$\sum_{S \in \mathcal{C}} |\delta(S) \cap E(F')| \leq \left(2 - \frac{1}{n}\right) |\mathcal{C}|.$$

Demonstração. Observe que cada componente conexa ou é desativada quando a restrição de uma aresta se torna justa, caso em que é parte de uma componente conexa maior, ou é desativada quando se torna saturada, caso em que é uma componente

conexa maximal. Seja $\mathcal{I}$ o conjunto de componentes conexas maximais que estão inativas no início da iteração e que não estão contidas em X. Considere o grafo T obtido a partir de F' após contrairmos cada componente conexa $S \in \mathcal{C} \cup \mathcal{I}$ (removendo laços, se houver) e denote por $g(S)$ o grau do vértice resultante da contração de S. Observe que T é uma árvore e que $\mathcal{C}$ e $\mathcal{I}$ particionam $V(T)$. Assim, a soma dos graus de T é dada por

$$\sum_{S \in \mathcal{C}} g(S) + \sum_{S \in \mathcal{I}} g(S) = 2(|\mathcal{C}| + |\mathcal{I}| - 1).$$

Como $|\delta(S) \cap E(F')| = g(S)$, essa igualdade implica que

$$\sum_{S \in \mathcal{C}} |\delta(S) \cap E(F')| = \sum_{S \in \mathcal{C}} |g(S)| = 2|\mathcal{C}| + \sum_{S \in \mathcal{I}} (2 - g(S)) - 2. \tag{9.21}$$

Considere uma componente conexa inativa I no início de uma iteração com $g(I) = 1$. Iremos mostrar que I é a componente conexa que contém a raiz r. Suponha, por absurdo, que I não contenha r. Como $g(I) = 1$, existe uma única aresta e em F' que é incidente em I. Observe que e deve ter sido adicionada na iteração atual ou em uma iteração posterior e, portanto, depois de cada aresta contida em I ter sido adicionada. Considere agora o momento em que e foi analisada na fase de limpeza do algoritmo. Nesse momento, e liga uma componente conexa C, sem a raiz, à componente conexa que contém a raiz. Como a aresta e não foi removida de F' e cada aresta de I só foi analisada depois na fase de limpeza, concluímos que $I \subseteq C$. Sejam $X_1, X_2, \ldots, X_s \subseteq C$ as componentes conexas de C que foram removidas na fase de limpeza. Note que I é a única componente conexa contida em C que faz parte de F'. Assim, I e $X_1, X_2, \ldots, X_s$ formam uma partição de C. Como I está inativa e não contém r, ela está saturada. Além disso, como as componentes conexas X_i foram removidas na fase de limpeza, elas estão saturadas. Daí,

$$p(C) = p(I) + \sum_{i=1}^{s} p(X_i) = \sum_{S \subseteq I} y_S + \sum_{i=1}^{s} \sum_{S \subseteq X_i} y_S \leq \sum_{S \subseteq X} y_S.$$

Porém, C estava saturada, então e teria sido removida ao ser analisada, contrariando o fato de que e faz parte de F'. Portanto, de fato, a componente conexa I contém r.

Assim, existe uma única componente conexa inativa I com $g(I) = 1$. Separando-se I das demais componentes conexas de $\mathcal{I}$, escrevemos

$$\begin{aligned} \sum_{S \in \mathcal{I}} (2 - g(S)) - 2 &= \sum_{S \in \mathcal{I} \setminus \{I\}} (2 - g(S)) + (2 - g(I)) - 2 \\ &\leq \sum_{S \in \mathcal{I} \setminus \{I\}} (2 - 2) + (2 - 1) - 2 = -1, \end{aligned}$$

em que nos valemos do fato de que $g(S) \geq 2$ para $S \in \mathcal{I} \setminus \{I\}$. Substituindo em (9.21), obtemos

$$\sum_{S \in \mathcal{C}} |\delta(S) \cap E(F')| \leq 2|\mathcal{C}| - 1 \leq \left(2 - \frac{1}{n}\right) |\mathcal{C}|. \qquad \square$$

Usando o lema acima, podemos mostrar que o custo das arestas é de no máximo duas vezes a soma das variáveis duais correspondentes a componentes conexas que intersectam F'.

Lema 9.4.4. Seja $n = |V(G)|$ e $\mathcal{D} = \{S \subseteq V(G) \setminus \{r\} : S \cap V(F') \neq \emptyset\}$. Então,

$$\sum_{e \in E(F')} \sum_{S : e \in \delta(S)} y_S \leq \left(2 - \frac{1}{n}\right) \sum_{S \in \mathcal{D}} y_S.$$

Demonstração. Primeiro, agrupamos os termos que multiplicam y_S e reescrevemos a desigualdade como

$$\sum_{S \subseteq V(G) \setminus \{r\}} |\delta(S) \cap E(F')| y_S \leq \left(2 - \frac{1}{n}\right) \sum_{S \in \mathcal{D}} y_S.$$

Iremos demonstrar essa desigualdade por indução no número de iterações. Antes de qualquer iteração, cada variável y_S foi inicializada com 0 e, portanto, a afirmação vale trivialmente. Considere então o valor de y no início de uma iteração e seja $\mathcal{C}$ o conjunto de componentes conexas ativas nessa iteração. Como as variáveis crescem uniformemente, cada uma das variáveis de $\mathcal{C}$ aumentou do mesmo valor, diga-se, ε. Assim, o lado esquerdo da desigualdade aumentou de exatamente $\sum_{S \in \mathcal{C}} |\delta(S) \cap E(F')| \varepsilon$, enquanto o lado direito aumentou de $\left(2 - \frac{1}{n}\right) |\mathcal{C}| \varepsilon$. Pelo lema 9.4.3, sabemos que $\sum_{S \in \mathcal{C}} |\delta(S) \cap E(F')| \leq \left(2 - \frac{1}{n}\right) |\mathcal{C}|$ e, portanto, a desigualdade continua válida no final da iteração. □

Combinando os lemas acima, obtemos o seguinte lema:

Lema 9.4.5. Seja $n = |V(G)|$ e $X = V(G) \setminus V(F')$, então

$$\sum_{e \in E(F')} c_e + \left(2 - \frac{1}{n}\right) \sum_{v \in X} p_v \leq \left(2 - \frac{1}{n}\right) \sum_{S \subseteq V(G) \setminus \{r\}} y_S.$$

Demonstração. Calculando diretamente, temos:

$$\begin{aligned}
\sum_{e \in E(F')} c_e + \left(2 - \frac{1}{n}\right) \sum_{v \in X} p_v &= \sum_{e \in E(F')} \sum_{S : e \in \delta(S)} y_S + \left(2 - \frac{1}{n}\right) p(X) \\
&\leq \left(2 - \frac{1}{n}\right) \sum_{S \in \mathcal{D}} y_S + \left(2 - \frac{1}{n}\right) \sum_{S \subseteq X} y_S \\
&= \left(2 - \frac{1}{n}\right) \sum_{S \subseteq V(G) \setminus \{r\}} y_S,
\end{aligned}$$

em que utilizamos os lemas 9.4.2 e 9.4.4 na desigualdade. □

Finalmente, mostramos que o algoritmo é uma 2-aproximação.

Teorema 9.4.3. PCST-PrimalDual é uma 2-aproximação para PCST.

Demonstração. Observe que o algoritmo executa em tempo polinomial. Seja F' a árvore devolvida e defina $X = V(G) \setminus V(F')$. Utilizando o lema 9.4.5, sabemos que o custo da solução é $\sum_{e \in E(F')} c_e + p(X) \leq 2 \sum_{S \subseteq V(G) \setminus \{r\}} y_S$. □

9.4.4 Método primal-dual e relaxação lagrangiana

Esta seção considera o Problema da k-Árvore Mínima (k-Minimum Spanning Tree Problem, k-MST). A entrada é composta de um grafo não direcionado G com custo $c_e \geq 0$ para cada aresta $e \in E(G)$, um inteiro $k \geq 1$ e um vértice especial r, chamado de raiz. Queremos encontrar uma árvore de menor custo em G com pelo menos k vértices e que contenha r. Denote por $\mathcal{I}_{k\text{-MST}}$ o conjunto de todas as tuplas (G, c, k, r).

Problema 9.4.3 (k-MST). Dada uma instância $(G, c, k, r) \in \mathcal{I}_{k\text{-MST}}$, o objetivo do problema é encontrar uma árvore T tal que $r \in V(T)$, $|V(T)| \geq k$ e a soma $\sum_{e \in E(T)} c_e$ seja mínima.

Enquanto o Problema da Árvore Geradora Mínima pode ser resolvido em tempo polinomial, k-MST é NP-difícil [184]. Note ainda que, como no caso do PCST, podemos supor que a função de custos nas arestas forme uma métrica. A seguir, mostraremos a relação desse problema com PCST, apresentando a técnica da relaxação lagrangiana para algoritmos primais-duais. Antes, precisamos apresentar uma formulação inteira adequada.

$$\begin{array}{lll} \text{Minimize} & \displaystyle\sum_{e\in E(G)} c_e x_e, & \\ \text{sujeito a} & \displaystyle\sum_{e\in\delta(S)} x_e + \sum_{X:S\subseteq X} z_S \geq 1, & \text{para todo } S\subseteq V(G)\setminus\{r\}, \\ & \displaystyle\sum_{S\subseteq V(G)\setminus\{r\}} |S| z_S \leq n-k, & \\ & x_e \in \{0,1\}, & \text{para todo } e\in E(G), \\ & z_X \in \{0,1\}, & \text{para todo } X\subseteq V(G)\setminus\{r\}. \end{array} \tag{9.22}$$

Como em (9.19), x_e indica se e é uma aresta da árvore T e z_X indica se o complemento do conjunto de vértices de T é X. Argumentando-se da mesma forma que anteriormente, pode-se mostrar que o conjunto de arestas e com $z_e = 1$ induz uma árvore T e que sempre há uma solução ótima em que z_X vale 1 se $X = V(G)\setminus V(T)$ e 0, caso contrário. Há duas diferenças importantes entre (9.19) e (9.22). A primeira é que na função-objetivo não há o termo correspondente a penalidades de vértices que estejam fora de T; a segunda é que há uma restrição a mais, $\sum_{S\subseteq V(G)\setminus\{r\}} |S| z_S \leq n-k$. Essa restrição garante que o complemento de T tenha no máximo $n-k$ vértices, ou, equivalentemente, que T tenha pelo menos k vértices. A relaxação linear de (9.22) é feita de maneira usual, substituindo-se $x_e \in \{0,1\}$ e $z_X \in \{0,1\}$ por $x_e \geq 0$ e $z_X \geq 0$, respectivamente.

Embora os dois programas sejam bastante similares, a restrição extra em (9.22) é uma restrição global, que limita o número de itens não cobertos, o que é difícil de se assegurar quando utilizamos a estratégia primal-dual. A relaxação lagrangiana consiste em relaxar essa restrição, de forma a se obter um programa mais simples. Primeiro reescrevemos a restrição de maneira que o lado direito seja nulo,

$$\sum_{S\subseteq V(G)\setminus\{r\}} |S| z_S - (n-k) \leq 0,$$

e multiplicamos a desigualdade por um parâmetro $\lambda \geq 0$, chamado multiplicador de Lagrange. O valor obtido é adicionado à função-objetivo e a restrição é removida. Obtemos o seguinte programa linear:

$$\begin{array}{lll} \text{Minimize} & \displaystyle\sum_{e\in E(G)} c_e x_e + \lambda \left(\sum_{S\subseteq V(G)\setminus\{r\}} |S| z_S - (n-k) \right), & \\ \text{sujeito a} & \displaystyle\sum_{e\in\delta(S)} x_e + \sum_{X:S\subseteq X} z_X \geq 1, & \text{para todo } S\subseteq V(G)\setminus\{r\}, \\ & x_e \geq 0, & \text{para todo } e\in E(G), \\ & z_X \geq 0, & \text{para todo } X\subseteq V(G)\setminus\{r\}. \end{array} \tag{9.23}$$

Seja OPT o valor de uma solução ótima para (9.22). Como uma solução de (9.22) é viável para (9.23) e nessa solução o termo que multiplica λ é não positivo, o valor ótimo de (9.23) é menor ou igual a OPT. Isso significa que o programa linear alterado é uma relaxação do programa linear inteiro original, chamada de relaxação lagrangiana. Observe que (9.23) é praticamente igual a (9.19) quando $p_v = \lambda$ para todo $v \in V(G)$, com a única diferença de que há um termo constante $-(n-k)\lambda$. Isso sugere que podemos utilizar o algoritmo da seção anterior para algum valor de λ. Para que

possamos utilizar essa ideia, precisamos ser capazes de compará-lo com o valor ótimo de k-MST. Para isso, construímos o dual de (9.23):

$$\begin{array}{lll} \text{Maximize} & \displaystyle\sum_{S \subseteq V(G)\setminus\{r\}} y_S - (n-k)\lambda, & \\ \text{sujeito a} & \displaystyle\sum_{S: e \in \delta(S)} y_S \leq c_e, & \text{para todo } e \in E(G), \\ & \displaystyle\sum_{S \subseteq X} y_S \leq |X|\lambda, & \text{para todo } X \subseteq V(G)\setminus\{r\}, \\ & y_S \geq 0, & \text{para todo } S \subseteq V(G)\setminus\{r\}. \end{array} \tag{9.24}$$

Suponha que executemos PCST-PrimalDual para um valor de λ e obtenhamos uma solução primal integral, dada por T, juntamente com uma solução dual viável de (9.20) para PCST, dada por y. Observe que y também é viável para (9.24). Se a solução T para PCST tiver exatamente k vértices, então T também será uma solução viável para k-MST. Nesse caso, podemos escrever o custo de T em termos da função-objetivo de (9.24), cujo custo total corresponde ao custo das arestas para PCST. Utilizando o lema 9.4.5, o custo de T é

$$\begin{aligned} \sum_{e \in E(T)} c_e &\leq \left(2 - \frac{1}{n}\right) \sum_{S \subseteq V(G)\setminus\{r\}} y_S - \left(2 - \frac{1}{n}\right) \sum_{v \in X} p_v \\ &= \left(2 - \frac{1}{n}\right) \left(\sum_{S \subseteq V(G)\setminus\{r\}} y_S - (n-k)\lambda \right) \\ &\leq \left(2 - \frac{1}{n}\right) \mathrm{OPT}. \end{aligned}$$

Na igualdade acima, utilizamos o fato de que X tem exatamente $n-k$ elementos, cada um com penalidade λ. Na última desigualdade, utilizamos o fato de que, pela dualidade fraca, o valor da função-objetivo é de no máximo o valor ótimo. Como a árvore devolvida tem exatamente k vértices, a desigualdade acima significa que a solução devolvida por PCST-PrimalDual é uma 2-aproximação para k-MST.

Duas observações precisam ser feitas. Primeiro, a análise acima depende de que o fator que multiplica o custo das penalidades ($\sum_{v \in X} p_v$) seja o mesmo que multiplica as soma das variáveis duais ($\sum_{S \subseteq V(G)\setminus\{r\}} y_S$). Informalmente, podemos dizer que o custo das arestas na solução obtida é aproximado, enquanto a penalidade dos vértices não cobertos é calculada de maneira justa. Note que isso não foi necessário no teorema 9.4.3, que poderia ser obtido mesmo com uma versão mais fraca do lema 9.4.5 em que $\sum_{e \in E(T)} c_e + \sum_{v \in X} p_v \leq 2 \sum_{S \subseteq V(G)\setminus\{r\}} y_S$. Para que a estratégia da relaxação lagrangiana funcione, por outro lado, a desigualdade mais forte é fundamental.

Além disso, a análise só funciona quando a árvore devolvida pelo algoritmo para o valor de λ escolhido tem exatamente k vértices. Se T tivesse mais do que k vértices, a solução ainda seria viável para k-MST, mas obteríamos um fator maior do que o valor da função-objetivo de (9.24) e, portanto, insuficiente para obtermos a aproximação. Se T tivesse menos do que k vértices, o fator obtido seria menor do que o valor da função-objetivo, mas T não seria viável para k-MST.

Intuitivamente, quanto maior o valor de λ, maior a penalidade de um vértice não coberto por T; portanto, esperamos que T tenha mais vértices. Por outro lado, quanto menor o valor de λ, maior o incentivo para deixar vértices fora de T. Isso, junto com a afirmação acima, sugere que podemos utilizar uma busca binária para encontrar um valor de λ para o qual PCST-PrimalDual devolva uma solução T com exatamente k vértices. Assim, se a árvore obtida tiver mais do que k vértices, diminuímos o valor de λ, ou, se tiver menos do que k vértices, aumentamos o valor de λ. Se essa busca binária encontrar uma árvore com T vértices em um número polinomial de passos,

então obtemos uma 2-aproximação para o problema. Infelizmente, é possível que para uma instância de k-MST não exista valor de λ para o qual a árvore devolvida tenha exatamente k vértices.

Embora a estratégia acima não forneça uma 2-aproximação, podemos utilizar a busca binária para construir uma boa solução para o problema, mas com fator de aproximação maior. A ideia é que, se encontrarmos λ tal que a árvore devolvida tenha k vértices, então essa árvore já é uma solução boa, que é uma 2-aproximação pela discussão acima. Do contrário, iremos encontrar um valor λ_1, para o qual o algoritmo primal-dual devolve uma árvore T_1 com $k_1 < k$ vértices, e um valor λ_2, para o qual o algoritmo primal-dual devolve uma árvore T_2 com $k_2 > k$ vértices. Realizando um número suficiente de iterações da busca binária, teremos $\lambda_1 < \lambda_2$ e $\lambda_2 - \lambda_1 \leq \varepsilon$, para algum valor $\varepsilon > 0$ suficientemente pequeno.

Antes de continuarmos, vamos pré-processar a instância. Iremos obter uma estimativa do custo D da maior distância entre a raiz e um vértice da árvore em uma solução ótima. Suponha que nos seja dado um valor D. Então podemos obter uma instância reduzida pela remoção de cada vértice cuja distância para a raiz seja estritamente maior do que D, já que, nesse caso, cada um desses vértice não está na solução ótima. Note que, quando $D = 0$, todos os vértices da instância pré-processada estão à distância zero da raiz e uma árvore geradora é uma solução ótima. Assim, suponha que $D > 0$ e seja $c_{\min} > 0$ o menor custo de uma aresta não nula. Nesse caso, como a solução ótima deve conter pelo menos uma aresta não nula, o valor ótimo, OPT, é de pelos menos $c_{\min}$. Como há n vértices, podemos testar cada um dos valores possíveis para D, resolver o problema e devolver a melhor solução obtida. De agora em diante, suponha, sem perda de generalidade, que a instância dada seja tal que a maior distância entre um vértice de G e a raiz seja D e que $c_{\min} \leq D \leq \mathrm{OPT}$.

Para que possamos inicializar a busca binária, precisamos encontrar limitantes inferior e superior para o parâmetro λ, para os quais PCST-PrimalDual devolva uma árvore com menos de k vértices e uma árvore com mais de k vértices, respectivamente. Para o limitante inferior, inicializamos $\lambda_1 = 0$, para o qual PCST-PrimalDual devolve uma árvore T_1 composta somente da raiz r. Para o limitante superior, inicializamos λ_2 com um valor suficientemente grande (p.ex., $\lambda_2 = \sum_{e \in E(G)} c_e$), para o qual PCST-PrimalDual devolve uma árvore T_2 com todos os vértices. Com essas variáveis inicializadas, executamos uma busca binária padrão.

Algoritmo 9.13: k-MST-BuscaBin(G, c, k, r)

1 Faça $\lambda_1 \leftarrow 0$ e $\lambda_2 \leftarrow \sum_{e \in E(G)} c_e$.
2 enquanto $\lambda_2 - \lambda_1 > \frac{c_{\min}}{2n^2}$ faça
3 Faça $\lambda \leftarrow \frac{1}{2}(\lambda_1 + \lambda_2)$.
4 Defina $p_v \leftarrow \lambda$ para cada $v \in V(G)$.
5 Execute PCST-Primal-Dual(G, c, p, r) e obtenha árvore T.
6 se $|V(T)| \leq k$ então
7 $\lambda_1 \leftarrow \lambda$.
8 se $|V(T)| > k$ então
9 $\lambda_2 \leftarrow \lambda$.
10 Devolva λ_1, λ_2.

Lema 9.4.6. Após um número polinomial de iterações, k-MST-BuscaBin devolve parâmetros λ_1 e λ_2 tais que

- se $p_v = \lambda_1$ para todo $v \in V(G)$, então PCST-PrimalDual(G,c,p) devolve uma árvore T_1 com $k_1 \leq k$ vértices associada a uma solução $y^{(1)}$ do programa linear (9.20);
- se $p_v = \lambda_2$ para todo $v \in V(G)$, então PCST-PrimalDual(G,c,p) devolve uma árvore T_2 com $k_2 \geq k$ vértices associada a uma solução $y^{(2)}$ do programa linear (9.20);
- $\lambda_2 - \lambda_1 \leq \frac{c_{\min}}{2n^2}$.

Demonstração. Basta verificar que o número de iterações do algoritmo é $O(\log \frac{2n^2 \sum_{e\in E(G)} c_e}{c_{\min}})$, que é polinomial no tamanho da entrada. □

Quando $k_1 = k$ ou $k_2 = k$, ou T_1 ou T_2 já é uma 2-aproximação para k-MST. Assim, suponha que $k_1 < k < k_2$. Seja $X_1 = V(G) \setminus V(T_1)$ e $X_2 = V(G) \setminus V(T_2)$. Pelo lema 9.4.5, sabemos que

$$\sum_{e\in E(T_1)} c_e \leq \left(2 - \frac{1}{n}\right)\left(\sum_{S\subseteq V(G)\setminus\{r\}} y_S^{(1)} - |X_1|\lambda_1\right) \quad \text{e} \tag{9.25}$$

$$\sum_{e\in E(T_2)} c_e \leq \left(2 - \frac{1}{n}\right)\left(\sum_{S\subseteq V(G)\setminus\{r\}} y_S^{(2)} - |X_2|\lambda_2\right). \tag{9.26}$$

Para obtermos uma desigualdade útil para análise, precisamos construir uma solução dual viável y de forma que $\sum_{S\subseteq V(G)\setminus\{r\}} y_S - (n-k)\lambda_2$ seja um limitante inferior para (9.24) e, portanto, para OPT. Isso sugere que escrevamos y como uma combinação convexa de $y^{(1)}$ e $y^{(2)}$, isto é, $y = \alpha_1 y^{(1)} + \alpha_2 y^{(2)}$, de forma que $n - k = \alpha_1|X_1| + \alpha_2|X_2| = \alpha_1(n-k_1) + \alpha_2(n-k_2)$. Assim, definimos

$$\alpha_1 = \frac{(n-k)-(n-k_2)}{(n-k_1)-(n-k_2)} \quad \text{e} \quad \alpha_2 = \frac{(n-k_1)-(n-k)}{(n-k_1)-(n-k_2)}.$$

Em seguida, combinamos as desigualdades (9.25) e (9.26).

Lema 9.4.7.

$$\alpha_1 \sum_{e\in E(T_1)} c_e + \alpha_2 \sum_{e\in E(T_2)} c_e \leq 2\,\mathrm{OPT}.$$

Demonstração. Defina $y = \alpha_1 y^{(1)} + \alpha_2 y^{(2)}$. Observe que y é uma solução viável para (9.24) com parâmetro λ_2. Para verificar isso, observe que, para cada $e \in E(G)$, temos $\sum_{S:e\in\delta(S)} y_S^{(1)} \leq c_e$ e $\sum_{S:e\in\delta(S)} y_S^{(2)} \leq c_e$. Então,

$$\sum_{S:e\in\delta(S)} y_S = \alpha_1 \sum_{S:e\in\delta(S)} y_S^{(1)} + \alpha_2 \sum_{S:e\in\delta(S)} y_S^{(2)} \leq \alpha_1 c_e + \alpha_2 c_e = c_e.$$

Analogamente, como $\lambda_2 > \lambda_1$, obtemos $\sum_{S\subseteq X} y_S \leq |X|\lambda_2$ para todo $X \subseteq V(G)\setminus\{r\}$ e $y_S \geq 0$ para todo $S \subseteq V(G)\setminus\{r\}$. Assim, y é solução viável para (9.24) com parâmetro λ_2 e seu valor é um limitante inferior para o valor ótimo da relaxação lagrangiana, isto é,

$$\sum_{S\subseteq V(G)\setminus\{r\}} y_S - (n-k)\lambda_2 \leq \mathrm{OPT}.$$

Lembrando que $|X_1| = n - k_1$, $|X_2| = n - k_2$ e fazendo a combinação convexa das desigualdades (9.25) e (9.26), obtemos

$$\begin{aligned}
&\alpha_1 \sum_{e \in E(T_1)} c_e + \alpha_2 \sum_{e \in E(T_2)} c_e \\
&\leq \left(2 - \frac{1}{n}\right) \left[\sum_{S \subseteq V(G) \setminus \{r\}} \left(\alpha_1 y_S^{(1)} + \alpha_2 y_S^{(2)} \right) - \left(\alpha_1 (n-k_1)\lambda_1 + \alpha_2 (n-k_2)\lambda_2 \right) \right] \\
&= \left(2 - \frac{1}{n}\right) \left[\sum_{S \subseteq V(G) \setminus \{r\}} y_S - (n-k)\lambda_2 \right] + \left(2 - \frac{1}{n}\right) \alpha_1 (n-k_1)(\lambda_2 - \lambda_1) \\
&\leq \left(2 - \frac{1}{n}\right) \left[\sum_{S \subseteq V(G) \setminus \{r\}} y_S - (n-k)\lambda_2 \right] + \left(2 - \frac{1}{n}\right) n \frac{c_{\min}}{2n^2} \\
&\leq \left(2 - \frac{1}{n}\right) \mathrm{OPT} + \frac{c_{\min}}{n} \\
&= 2\,\mathrm{OPT} - \frac{\mathrm{OPT} - c_{\min}}{n} \\
&\leq 2\,\mathrm{OPT},
\end{aligned}$$

em que nos valemos do fato de que $\alpha_1 \leq 1$ e $\lambda_2 - \lambda_1 \leq \frac{c_{\min}}{2n^2}$ na segunda desigualdade e de que $c_{\min} \leq \mathrm{OPT}$ na última desigualdade. □

Agora podemos obter uma aproximação.

Teorema 9.4.4. Existe uma 5-aproximação para k-MST.

Demonstração. Execute o algoritmo de busca binária acima e obtenha árvores T_1 e T_2, conforme o lema 9.4.6. Se alguma das árvores tiver exatamente k vértices, então já obtemos uma 2-aproximação. Suponha então que $k_1 < k < k_2$ e relembre que T_2 já é uma solução viável para k-MST. Se $\alpha_2 \geq \frac{1}{2}$, então o lema 9.4.7 acima implica que $\sum_{e \in E(T_2)} c_e \leq 2\,\frac{\mathrm{OPT}}{\alpha_2} \leq 4\mathrm{OPT}$, ou seja, que T_2 já seja uma 4-aproximação.

Suponha que $\alpha_2 < \frac{1}{2}$ e que, então, $\alpha_1 > \frac{1}{2}$. Nesse caso, podemos completar T_1 com vértices de T_2. Realizamos o seguinte procedimento. Seja $F = V(T_2) - V(T_1)$. Duplique as arestas de T_2 e forme uma rota passando por todas as arestas. Depois, forme um ciclo C, percorrendo a rota e tomando atalhos sempre que visitar um vértice já visitado, ou que não esteja em F. Observe que C contém todos os vértices de F e tem custo de no máximo duas vezes o custo de T_2. Encontre um subcaminho P de C com $k - k_1$ vértices e de custo mínimo. Isso é possível, já que $k - k_1 \leq k_2 - k_1 = |V(T_2)| - |V(T_1)| \leq |F|$.

Agora estimamos o custo de P. Observe que existem exatamente $|F|$ subcaminhos de C com tamanho $k - k_1$ e que, somando-se todos eles, cada aresta de C é contada $k - k_1$ vezes. Portanto, o menor desses subcaminhos tem custo

$$\begin{aligned}
\sum_{e \in E(P)} c_e &\leq \frac{(k-k_1)}{|F|} \left(\sum_{e \in E(C)} c_e \right) \\
&\leq \frac{k-k_1}{k_2-k_1} \left(\sum_{e \in E(C)} c_e \right) \\
&\leq \frac{(n-k_1)-(n-k)}{(n-k_1)-(n-k_2)} \left(2 \sum_{e \in E(T_2)} c_e \right) \\
&= 2\alpha_2 \sum_{e \in E(T_2)} c_e.
\end{aligned}$$

Finalmente, construímos uma solução T, que é formada pela união de T_1 e P e pela adição de uma aresta da raiz até um vértice de P. Segundo nossa hipótese, o custo dessa aresta é de no máximo D. Portanto, o custo da solução T obtida é a soma dos custos de T_1, de P e da aresta adicionada,

$$\begin{aligned}\sum_{e\in E(T)} c_e &\leq \sum_{e\in E(T_1)} c_e + \sum_{e\in E(P)} c_e + D \\ &\leq 2\left(\alpha_1 \sum_{e\in E(T_1)} c_e + \alpha_2 \sum_{e\in E(T_2)} c_e\right) + D \\ &\leq 2\cdot 2\,\mathrm{OPT} + \mathrm{OPT} = 5\,\mathrm{OPT},\end{aligned}$$

em que usamos $2\alpha_1 \geq 1$ na segunda desigualdade e o lema 9.4.7 na última desigualdade. □

9.5 Comentários finais

Johnson [123] mostrou que um algoritmo guloso tem fator de aproximação $\ln K + 1$ para SC quando os custos dos conjuntos são todos iguais e $K = \max\{|S| : S \in \mathcal{S}\}$. O algoritmo apresentado na seção 9.2.1 é devido a Chvátal [205] e a prova segue a apresentada por Young [220]. A dificuldade de aproximar SC por um fator melhor que o logarítmico já foi estabelecida por Lund e Yannakakis [151] e depois melhorada por Feige [73], que mostrou que não há algoritmo de tempo polinomial com fator de aproximação $(1-o(1))\ln n$ a não ser que $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{O(\log\log n)})$. Essa hipótese foi fortalecida por Dinur e Steurer [64] para $\mathrm{P} \neq \mathrm{NP}$, que mostraram o teorema 9.2.1.

O algoritmo de arredondamento determinístico para SC dado na seção 9.3.2 e sua aplicação para o VC é devido a Hochbaum [53]. VC foi um dos primeiros problemas a se demonstrarem NP-difíceis [125] e há indicativos de que não existem algoritmos de aproximação com fator constante menor do que 2 [129]. Em particular, uma família de relaxações de programas lineares inteiros para VC tem gap de integralidade de pelo menos $2-o(1)$ [5]. A técnica de arredondamento probabilístico de relaxações de programas lineares inteiros 0-1 foi introduzida por Raghavan e Thompson [183]. A análise do algoritmo para SC da seção 9.3.3 é baseada na descrição de Williamson e Shmoys [213].

ST é um problema NP-difícil clássico [87, 125] e, além disso, não pode ser aproximado por um fator menor do que 96/95 sob a hipótese de que $\mathrm{P} \neq \mathrm{NP}$ [46]. Entre os algoritmos mais conhecidos para se encontrar uma árvore geradora de custo mínimo, estão os algoritmos de Kruskal [137] e de Prim [176]. Embora já seja parte do folclore utilizar uma árvore geradora mínima para aproximar ST (veja, e.g., [95]), encontrar uma aproximação com fator constante menor do que 2 foi um problema em aberto que persistiu durante várias décadas. Essa barreira foi quebrada em uma sequência de algoritmos [221, 127, 177, 185], todos eles baseados na escolha de full-components, que são subárvores maximais cujas folhas correspondem a terminais, de acordo com algum critério. O melhor algoritmo conhecido é uma 1,39-aproximação de Byrka et al. [38], que utilizaram uma relaxação de um programa linear e introduziram uma abordagem inovadora de arredondamento probabilístico para escolher full-components iterativamente.

Os problemas de escalonamento descritos como IMS e UMS são normalmente denotados na literatura [109] como $\mathrm{P}||\mathrm{C}_{\max}$ e $\mathrm{R}||\mathrm{C}_{\max}$, respectivamente. O algoritmo guloso que obtém uma 2-aproximação para IMS é de Graham [108]. Hochbaum e Shmoys [117] obtiveram um PTAS para o problema. O EPTAS descrito neste texto que executa em tempo linear baseia-se no apresentado por Schuurman e Woeginger [193].

O primeiro algoritmo para a resolução de programa linear inteiro com número fixo de variáveis é de Lenstra [139]. Depois disso, outros trabalhos melhoraram o tempo de execução [68].

A 2-aproximação para UMS da seção 9.3.4 é de Lenstra et al. [138]. Supondo $P \neq NP$, a dependência do tempo de processamento de uma tarefa de acordo com a máquina alocada em UMS torna o problema estritamente mais difícil de aproximar que IMS, já que, sob essa hipótese, não há uma $(3/2-\varepsilon)$-aproximação para UMS para $\varepsilon > 0$ constante [138]. Diminuir a diferença entre o fator de inaproximabilidade de $3/2$ e a melhor aproximação de 2 é uma das principais perguntas em aberto da área de escalonamento. É improvável, no entanto, que algoritmos baseados em programação linear melhorem o fator de aproximação, pois Verschae e Wiese [209] mostraram que há instâncias do problema para as quais o gap de integralidade das formulações conhecidas é 2.

O algoritmo primal-dual para FL da seção 9.4.2 está em um artigo seminal de Jain e Vazirani [121]. Esse problema serviu, de fato, como caixa de testes para diversas técnicas de programação linear cujo intuito era obter algoritmos de aproximação. Um algoritmo já clássico é o de arredondamento de Chudak e Shmoys [49]; entre as ferramentas utilizadas pelos autores estão, por exemplo, pipage rounding [200] e filtering [145]. Uma estratégia alternativa para a construção de um algoritmo primal-dual é a chamada dual-fiiting, em que o objetivo é encontrar uma solução primal inteira viável juntamente com uma solução dual de mesmo valor, mas possivelmente inviável [122]. Uma característica interessante desses algoritmos é que neles o fator de aproximação é obtido com a limitação de uma família de programas lineares chamados factor-revealing linear programs ou mesmo com a resolução de um único programa linear [75]. O melhor algoritmo de aproximação conhecido tem fator 1,488 e combina tanto técnicas de arredondamento quanto a estratégia primal-dual [143].

O algoritmo primal-dual para PCST da seção 9.4.3 é de Goemans e Williamson [99]. A 5-aproximação para k-MST é de Garg [89] e a descrição desse algoritmo na seção 9.4.4 com o emprego da relaxação lagrangiana é baseada no trabalho de Chudak et al. [48]. A utilização da relaxação lagrangiana para se obter um algoritmo de aproximação já havia sido explorada antes no artigo de Jain e Vazirani [121], que observaram que relaxação lagrangiana da formulação do problema das k-medianas corresponde à formulação de FL. O melhor algoritmo de aproximação conhecido para k-MST é uma 2-aproximação, também de Garg [90], que consiste em executar uma versão modificada do algoritmo para PCST de Goemans e Williamson, assim como na seção 9.4.4, mas com uma escolha mais refinada do parâmetro λ do que na busca binária.

Embora este capítulo tenha priorizado descrever algumas técnicas para a obtenção de algoritmos de aproximação baseadas principalmente em programação linear, existem diversas outras ferramentas. Uma ferramenta ausente que se destaca aqui são os algoritmos de busca local (ver, por exemplo, [6]). O leitor interessado em algoritmos de aproximação de maneira geral pode consultar também os livros de Cavalho et al. [42], Vazirani [207] e Williamson e Shmoys [213].

Bibliografia

[1] "A Review of: "APPLICATIONS OF GRAPH THEORY", R. J. Wilson and L. W. Beineke (Eds.)" Em: Engineering Optimization 5.1 (1980), pp. 68–69. doi: 10.1080/03052158008913100. eprint: https://doi.org/10.1080/03052158008913100. url: https://doi.org/10.1080/03052158008913100.

[2] K. Aardal e C. P. M. van Hoesel. "Polyhedral techniques in combinatorial optimization I: theory". Em: Statistica Neerlandica 50.1 (1996), pp. 3–26. issn: 1467-9574. doi: 10.1111/j.1467-9574.1996.tb01478.x.

[3] K. Aardal e C. P. M. van Hoesel. "Polyhedral techniques in combinatorial optimization II: applications and computations". Em: Statistica Neerlandica 53.2 (1999), pp. 131–177. issn: 1467-9574. doi: 10.1111/1467-9574.00104. url: http://dx.doi.org/10.1111/1467-9574.00104.

[4] K. Aardal e R. Weismantel. "Polyhedral Combinatorics: an annotated bibliography". Em: Annotated bibliographies in combinatorial optimization. Ed. por M. Dell'Amico, F. Maffioli e S. Martello. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1997, pp. 31–44. isbn: 9780471965749. url: https://books.google.com.br/books?id=jz4ZAQAAIAAJ.

[5] Sanjeev Arora et al. "Proving Integrality Gaps without Knowing the Linear Program". Em: Theory of Computing 2.2 (2006), pp. 19–51. doi: 10.4086/toc.2006.v002a002. url: http://www.theoryofcomputing.org/articles/v002a002.

[6] Vijay Arya et al. "Local Search Heuristics for k-Median and Facility Location Problems". Em: SIAM Journal on Computing 33.3 (2004), pp. 544–562. doi: 10.1137/S0097539702416402. url: http://link.aip.org/link/?SMJ/33/544/1.

[7] L. Bahiense, A. Besso e N. Maculan. Using multifow formulations to solve the Steiner tree problem in graphs - Submitted. 2018.

[8] Egon Balas. “Disjunctive Programming”. Em: Discrete Optimization II. Ed. por P.L. Hammer, E.L. Johnson e B.H. Korte. Vol. 5. Annals of Discrete Mathematics. Elsevier, 1979, pp. 3–51.

[9] Egon Balas. “Intersection Cuts - A New Type of Cutting Planes for Integer Programming”. Em: Operations Research 19.1 (1971), pp. 19–39.

[10] Egon Balas, Sebastián Ceria e Gérard Cornuéjols. “A lift-and-project cutting plane algorithm for mixed 0–1 programs”. Em: Mathematical Programming 58.1 (1993), pp. 295–324.

[11] R.M. Barbosa. Combinatória e Grafos. Nobel, 1974.

[12] Cynthia Barnhart et al. “Branch-and-Price: Column Generation for Solving Huge Integer Programs”. Em: Operations Research 46.3 (1998), pp. 316–329. doi: 10.1287/opre.46.3.316. eprint: https://pubsonline.informs.org/doi/pdf/10.1287/opre.46.3.316. url: https://pubsonline.informs.org/doi/abs/10.1287/opre.46.3.316.

[13] G. Barroso e A. R. Ellenrieder. Programação Linear. Almeida Neves Editores, 1971.

[14] J. E. Beasley, ed. Advances in Linear and Integer Programming. New York, NY, USA: Oxford University Press, Inc., 1996. isbn: 0-19-853856-1.

[15] Pietro Belotti e Timo Berthold. “Three ideas for a feasibility pump for nonconvex MINLP”. Em: Optimization Letters 11.1 (jan. de 2017), pp. 3–15. issn: 1862-4480. doi: 10.1007/s11590-016-1046-0. url: https://doi.org/10.1007/s11590-016-1046-0.

[16] C. Berge. The Theory of Graphs and Its Applications. Methuen, 1962. url: https://books.google.com.br/books?id=Jd9QAAAAMAAJ.

[17] C. Berge e E. Minieka. Graphs and Hypergraphs. Graphs and Hypergraphs. North-Holland Publishing Company, 1973. isbn: 9780444103994. url: https://books.google.com.br/books?id=X32GlVfqXjsC.

[18] Livio Bertacco, Matteo Fischetti e Andrea Lodi. “A feasibility pump heuristic for general mixed-integer problems”. Em: Discrete Optimization 4.1 (2007), pp. 63–76. issn: 1572-5286. doi: 10.1016/j.disopt.2006.10.001.

[19] T. Berthold e A. M. Gleixner. Undercover - a primal heuristic for MINLP based on sub-MIPs generated by set covering. Rel. técn. ZIB-REPORT 09-04. 2009.

[20] Timo Berthold. “RENS”. Em: Mathematical Programming Computation 6.1 (mar. de 2014), pp. 33–54. issn: 1867-2957. doi: 10.1007/s12532-013-0060-9. url: https://doi.org/10.1007/s12532-013-0060-9.

[21] Dimitris Bertsimas e John Tsitsiklis. Introduction to Linear Optimization. 1st. Athena Scientific, 1997. isbn: 1886529191.

[22] Alain Billionnet e Karima Djebali. “Solution of a fractional combinatorial optimization problem by mixed integer programming”. Em: RAIRO-Operations Research (out. de 2006). url: https://www.rairo-ro.org/articles/ro/pdf/2006/02/ro0578.pdf.

[23] R.G. Bland. New Finite Pivoting Rules for the Simplex Method. CORE discussion paper: Center for Operations Research and Econometrics. Center for Operations Research & Econometrics, 1976. url: https://books.google.com.br/books?id=iOpxnQEACAAJ.

[24] P. O. Boaventura. Grafos: Teoria, Modelos, Algoritmos. Blucher. isbn: 9788521203919.

[25] Bela. Bollobas. Graph theory : an introductory course / Bela Bollobas. New York: Springer Verlag, 1979.

[26] Béla Bollobás. Modern Graph Theory. 1ª ed. Graduate Texts in Mathematics 184. Springer-Verlag New York, 1998.

[27] Pierre Bonami e João Gonçalves. “Heuristics for convex mixed integer nonlinear programs”. Em: Computational Optimization and Applications (2008). 10.1007/s10589-010-9350-6, pp. 1–19. issn: 0926-6003.

[28] Pierre Bonami et al. “A Feasibility Pump for mixed integer nonlinear programs”. Em: Mathematical Programming 119 (2 2009). 10.1007/s10107-008-0212-2, pp. 331–352. issn: 0025-5610.

[29] Pierre Bonami et al. “An algorithmic framework for convex mixed integer nonlinear programs”. Em: Discrete Optimization 5.2 (mai. de 2008), pp. 186–204. doi: 10.1016/j.disopt.2006.10.011.

[30] Pierre Bonami et al. “On Branching Rules for Convex Mixed-integer Nonlinear Optimization”. Em: J. Exp. Algorithmics 18 (nov. de 2013), 2.6:2.1–2.6:2.31. issn: 1084-6654. doi: 10.1145/2532568. url: http://doi.acm.org/10.1145/2532568.

[31] J. A. Bondy e U. S. R. Murty. Graph Theory with Applications. New York: Elsevier, 1976.

[32] J.A. Bondy e U.S.R Murty. Graph Theory. 1st. Springer Publishing Company, Incorporated, 2008. isbn: 1846289696.

[33] Brian Borchers e John E. Mitchell. “An improved branch and bound algorithm for mixed integer nonlinear programs”. Em: Comput. Oper. Res. 21 (4 abr. de 1994), pp. 359–367. issn: 0305-0548. doi: 10.1016/0305-0548(94)90024-8.

[34] A. Brandstadt, V.B. Le e J.P. Spinrad. Graph Classes: A Survey. Monographs on Discrete Mathematics and Applications. Society for Industrial e Applied Mathematics, 1999. isbn: 9780898714326. url: https://books.google.com.br/books?id=bAGW1L84hRQC.

[35] P. F. Bregalda, A. A. F. Oliveira e C. T. Bornstein. Introdução à Programação Linear. Rio de Janeiro: Editora Campus, 1981.

[36] R.G. Busacker e T.L. Saaty. Finite Graphs and Networks: An Introduction with Applications. Finite Graphs and Networks: An Introduction with Applications v. 10. McGraw-Hill, 1965. isbn: 9780070093058. url: https://books.google.com.br/books?id=Yt9QAAAAMAAJ.

[37] Richard H. Byrd, Jorge Nocedal e Richard A. Waltz. “KNITRO: An integrated package for nonlinear optimization”. Em: Large Scale Nonlinear Optimization, 35-59, 2006. Springer Verlag, 2006, pp. 35–59.

[38] Jaroslaw Byrka et al. “Steiner Tree Approximation via Iterative Randomized Rounding”. Em: J. ACM 60.1 (fev. de 2013), 6:1–6:33. issn: 0004-5411. doi: 10.1145/2432622.2432628. url: http://doi.acm.org/10.1145/2432622.2432628.

[39] Manoel Campêlo, Victor A. Campos e Ricardo C. Corrêa. “On the asymmetric representatives formulation for the vertex coloring problem”. Em: Discrete Applied Mathematics 156.7 (2008), pp. 1097–1111. issn: 0166-218X. doi: http://dx.doi.org/10.1016/j.dam.2007.05.058. url: http://www.sciencedirect.com/science/article/pii/S0166218X07002922.

[40] Robert D. Carr e Goran Konjevod. “Polyhedral Combinatorics”. Em: Tutorials on Emerging Methodologies and Applications in Operations Research. Ed. por Harvey J. Greenberg. New York, NY: Springer, 2005, II.1–II.46. isbn: 978-0-387-22827-3. doi: 10.1007/0-387-22827-6{_}2.

[41] B. Carré. Graphs and Networks. Oxford applied mathematics and computing science series. Clarendon Press, 1979. isbn: 9780198596158. url: https://books.google.com.br/books?id=A-nuAAAAMAAJ.

[42] M. H. Carvalho et al. Uma Introdução Sucinta a Algoritmos de Aproximação. Rio de Janeiro: IMPA - Instituto de Matemática Pura e Aplicada, 2001.

[43] A Cayley. "A Theorem on Trees". Em: The Quarterly Journal of Mathematics 23 (1889), pp. 376–378.

[44] Der-San Chen, Robert Batson e Yu Dang. Applied Integer Programming: Modeling and Solution. Hoboken, NJ, USA: John Wiley & Sons, INC., 2010. isbn: 978-0-470-37306-4.

[45] B.V. Cherkasky. "Algorithm of construction of maximal flow in networks with complexity of". Em: Mathematical Methods of Solution of Economical Problems (1977), pp. 117–125.

[46] Miroslav Chlebík e Janka Chlebíková. "The Steiner tree problem on graphs: Inapproximability results". Em: Theoretical Computer Science 406.3 (2008), pp. 207–214. issn: 0304-3975. doi: http://dx.doi.org/10.1016/j.tcs.2008.06.046. url: http://www.sciencedirect.com/science/article/pii/S0304397508004660.

[47] N. Christofides. Graph Theory: An Algorithmic Approach. Computer science and applied mathematics : a series of monographs and textbooks. Academic Press, 1975. isbn: 9780121743505. url: https://books.google.com.br/books?id=pN9QAAAAMAAJ.

[48] Fabián A. Chudak, Tim Roughgarden e David P. Williamson. "Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation". Em: Mathematical Programming 100.2 (jun. de 2004), pp. 411–421. issn: 1436-4646. doi: 10.1007/s10107-003-0479-2. url: https://doi.org/10.1007/s10107-003-0479-2.

[49] Fabián A. Chudak e David B. Shmoys. "Improved Approximation Algorithms for the Uncapacitated Facility Location Problem". Em: SIAM Journal on Computing 33.1 (2004), pp. 1–25.

[50] V. Chvatal. Linear Programming. Series of books in the mathematical sciences. W. H. Freeman, 1983. isbn: 9780716715870. url: https://books.google.com.br/books?id=DN20%5C_tW%5C_BV0C.

[51] Michele Conforti, Gérard Cornuéjols e Giacomo Zambelli. "50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art". Em: ed. por Michael Jünger et al. Berlin, Heidelberg: Springer, 2010. Cap. Polyhedral Approaches to Mixed Integer Linear Programming, pp. 343–385.

[52] W. H. Cunningham e J.Edmonds. “A Combinatorial Decomposition Theory”. Em: Candian Journal of Mathematics 32 (1980), pp. 734–765.

[53] D.S. Hochbaum. “Approximation algorithms for the set covering and vertex cover problems”. Em: SIAM Journal on Compututing 11.3 (1982), pp. 555–556.

[54] Claudia D’Ambrosio et al. “A storm of feasibility pumps for nonconvex MINLP”. Em: Mathematical Programming 136.2 (dez. de 2012), pp. 375–402. issn: 1436-4646. doi: 10.1007/s10107-012-0608-x. url: https://doi.org/10.1007/s10107-012-0608-x.

[55] R. J. Dakin. “A tree-search algorithm for mixed integer programming problems”. Em: The Computer Journal 8.3 (jan. de 1965), pp. 250–255. issn: 0010-4620. doi: 10.1093/comjnl/8.3.250. eprint: https://academic.oup.com/comjnl/article-pdf/8/3/250/961820/080250.pdf. url: https://doi.org/10.1093/comjnl/8.3.250.

[56] Emilie Danna, Edward Rothberg e Claude Le Pape. “Exploring relaxation induced neighborhoods to improve MIP solutions”. Em: Mathematical Programming 102.1 (jan. de 2005), pp. 71–90. issn: 1436-4646. doi: 10.1007/s10107-004-0518-7. url: https://doi.org/10.1007/s10107-004-0518-7.

[57] G.B. Dantzig. Linear Programming and Extensions. Landmarks in Physics and Mathematics. Princeton University Press, 1998. isbn: 9780691059136. url: https://books.google.com.br/books?id=2j46uCX5ZAYC.

[58] George B. Dantzig e Philip Wolfe. “Decomposition Principle for Linear Programs”. Em: Operations Research 8.1 (fev. de 1960), pp. 101–111. issn: 0030-364X. doi: 10.1287/opre.8.1.101. url: https://doi.org/10.1287/opre.8.1.101.

[59] Tatjana Davidović et al. Mathematical programming-based approach to scheduling of communicating tasks. Rel. técn. 2004.

[60] Narsingh Deo. Graph Theory with Applications to Engineering and Computer Science (Prentice Hall Series in Automatic Computation). USA: Prentice-Hall, Inc., 1974. isbn: 0133634736.

[61] R. Diestel. Graph Decompositions: A Study in Infinite Graph Theory. Oxford science publications. Clarendon Press, 1990. isbn: 9780198532101. url: https://books.google.com.br/books?id=PP6EAAAAIAAJ.

[62] R. Diestel. Graph Theory. Electronic library of mathematics. Springer, 2006. isbn: 9783540261834. url: https://books.google.com.br/books?id=aR2TMYQr2CMC.

[63] Yefim Dinitz. "Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation". Em: Soviet Math. Dokl. 11 (jan. de 1970), pp. 1277–1280.

[64] Irit Dinur e David Steurer. "Analytical Approach to Parallel Repetition". Em: Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing. STOC '14. New York, New York: ACM, 2014, pp. 624–633. isbn: 978-1-4503-2710-7. doi: 10.1145/2591796.2591884.

[65] Marco Duran e Ignacio Grossmann. "An outer-approximation algorithm for a class of mixed-integer nonlinear programs". Em: Mathematical Programming 36 (3 1986). 10.1007/BF02592064, pp. 307–339. issn: 0025-5610. url: http://dx.doi.org/10.1007/BF02592064.

[66] Jack Edmonds e Rick Giles. "A Min-Max Relation for Submodular Functions on Graphs". Em: Studies in Integer Programming. Ed. por P.L. Hammer et al. Vol. 1. Annals of Discrete Mathematics Supplement C. Elsevier, 1977, pp. 185–204. doi: https://doi.org/10.1016/S0167-5060(08)70734_9. url: http://www.sciencedirect.com/science/article/pii/S0167506008707349.

[67] Jack Edmonds e Richard M. Karp. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems". Em: J. ACM 19.2 (1972), pp. 248–264. issn: 0004-5411. doi: 10.1145/321694.321699. url: https://doi.org/10.1145/321694.321699.

[68] Friedrich Eisenbrand. "Fast Integer Programming in Fixed Dimension". English. Em: Algorithms - ESA 2003. Ed. por Giuseppe Di Battista e Uri Zwick. Vol. 2832. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 196–207. isbn: 978-3-540-20064-2. doi: 10.1007/978-3-540-39658-1_20.

[69] A. Cayley Esq. "XXVIII. On the theory of the analytical forms called trees". Em: The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 13.85 (1857), pp. 172–176. doi: 10.1080/14786445708642275. eprint: https://doi.org/10.1080/14786445708642275. url: https://doi.org/10.1080/14786445708642275.

[70] Shimon Even. Graph Algorithms. 2nd. USA: Cambridge University Press, 2011. isbn: 0521736536.

[71] Oliver Exler, Thomas Lehmann e Klaus Schittkowski. “A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization”. English. Em: Mathematical Programming Computation 4 (4 2012), pp. 383–412. issn: 1867-2949. doi: 10.1007/s12532-012-0045-0. url: http://dx.doi.org/10.1007/s12532-012-0045-0.

[72] Oliver Exler e Klaus Schittkowski. “A trust region SQP algorithm for mixed-integer nonlinear programming”. English. Em: Optimization Letters 1 (3 2007), pp. 269–280. issn: 1862-4472. doi: 10.1007/s11590-006-0026-1. url: http://dx.doi.org/10.1007/s11590-006-0026-1.

[73] U. Feige. “A threshold of $\ln n$ for approxiamting set-cover”. Em: In 28th ACM Symposium on Theory of Computing (1996), pp. 314–318.

[74] P. Feofiloff e C.L. Lucchesi. Algoritmos para igualdades minimax em grafos. Escola de computacao. UNICAMP, IMECC, 1988. url: https://books.google.com.br/books?id=iaboGwAACAAJ.

[75] Cristina G. Fernandes et al. “A systematic approach to bound factor-revealing LPs and its application to the metric and squared metric facility location problems”. Em: Mathematical Programming 153.2 (2015), pp. 655–685. issn: 1436-4646. doi: 10.1007/s10107-014-0821-x. url: http://dx.doi.org/10.1007/s10107-014-0821-x.

[76] C.E. Ferreira e Y. Wakabayashi. Combinatória poliédrica e planos-de-corte faciais. X Escola de Computação. UNICAMP-Instituto de Computação, 1996.

[77] Matteo Fischetti e Andrea Lodi. “Local branching”. Em: Mathematical Programming 98 (1 2003). 10.1007/s10107-003-0395-5, pp. 23–47. issn: 0025-5610. url: http://dx.doi.org/10.1007/s10107-003-0395-5.

[78] Marshall L. Fisher. “The Lagrangian Relaxation Method for Solving Integer Programming Problems”. Em: Management Science 50.12_supplement (2004), pp. 1861–1871. doi: 10.1287/mnsc.1040.0263. url: https://doi.org/10.1287/mnsc.1040.0263.

[79] Roger Fletcher e Sven Leyffer. “Solving mixed integer nonlinear programs by outer approximation”. Em: Mathematical Programming 66 (1 1994). 10.1007/BF01581153, pp. 327–349. issn: 0025-5610. url: http://dx.doi.org/10.1007/BF01581153.

[80] L. R. FORD e D. R. FULKERSON. Flows in Networks. Princeton University Press, 1962. isbn: 9780691625393. url: http://www.jstor.org/stable/j.ctt183q0b4 (acesso em 08/10/2022).

[81] L. R. Ford e D. R. Fulkerson. "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem". Em: Canadian Journal of Mathematics 9 (1957), pp. 210–218. doi: 10.4153/CJM-1957-024-0.

[82] L. R. Ford e D. R. Fulkerson. "Maximal Flow Through a Network". Em: Canadian Journal of Mathematics 8 (1956), pp. 399–404. doi: 10.4153/CJM-1956-045-5.

[83] D. R. Fulkerson. "Blocking and anti-blocking pairs of polyhedra". Em: Mathematical Programming 1.1 (1971), pp. 168–194. issn: 1436-4646. doi: 10.1007/BF01584085. url: https://doi.org/10.1007/BF01584085.

[84] A. L. Furtado. Teoria dos Grafos - Algoritmos. Livros Tecnicos Cientificos., 1973.

[85] Zvi Galil. "A New Algorithm for the Maximal Flow Problem". Em: SFCS '78. USA: IEEE Computer Society, 1978, pp. 231–245. doi: 10.1109/SFCS.1978.5. url: https://doi.org/10.1109/SFCS.1978.5.

[86] Zvi Galil e Amnon Naamad. "An O(EVlog2V) algorithm for the maximal flow problem". Em: Journal of Computer and System Sciences 21.2 (1980), pp. 203–217. issn: 0022-0000. doi: https://doi.org/10.1016/0022-0000(80)90035-5. url: https://www.sciencedirect.com/science/article/pii/0022000080900355.

[87] Michael R. Garey e David S. Johnson. Computers and Intractability; A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman Co., 1990. isbn: 0716710455.

[88] R. Garfinkel, R.S. Garfinkel e G.L. Nemhauser. Integer Programming. A Wiley-Interscience publication. Wiley, 1972. isbn: 9780471291954. url: https://books.google.com.br/books?id=CV5RAAAAMAAJ.

[89] N. Garg. "A 3-approximation for the minimum tree spanning k vertices". Em: Proceedings of 37th Conference on Foundations of Computer Science. Out. de 1996, pp. 302–309. doi: 10.1109/SFCS.1996.548489.

[90] Naveen Garg. “Saving an Epsilon: A 2-approximation for the k-MST Problem in Graphs”. Em: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing. STOC ’05. Baltimore, MD, USA: ACM, 2005, pp. 396–402. isbn: 1-58113-960-8. doi: 10.1145/1060590.1060650. url: http://doi.acm.org/10.1145/1060590.1060650.

[91] B. Gavish e S.C. Graves. The Travelling Salesman Problem and Related Problems. Working Paper OR 078-78. Massachusetts Institute of Technology, Operations Research Center, 1978.

[92] A. M. Geoffrion. “Generalized Benders decomposition”. Em: Journal of Optimization Theory and Applications 10 (4 1972). 10.1007/BF00934810, pp. 237–260. issn: 0022-3239.

[93] A.M.H. Gerards e A.W.J. Kolen. “Polyhedral Combinatorics in Combinatorial Optimization”. Em: Statistica Neerlandica 41.1 (1987), pp. 1–25. issn: 1467-9574. doi: 10.1111/j.1467-9574.1987.tb01168.x. url: http://dx.doi.org/10.1111/j.1467-9574.1987.tb01168.x.

[94] Allain Ghouila-Houri. “Caractérisation des matrices totalement unimodulaires”. Em: C. R. Acad. Sci. Paris 254 (1962), pp. 1192–1194.

[95] E. N. Gilbert e H. O. Pollak. “Steiner Minimal Trees”. Em: SIAM Journal on Applied Mathematics 16.1 (1968), pp. 1–29. doi: 10.1137/0116001. eprint: https://doi.org/10.1137/0116001.

[96] F.R. Giles e W.R. Pulleyblank. “Total dual integrality and integer polyhedra”. Em: Linear Algebra and its Applications 25.Supplement C (1979), pp. 191–196. issn: 0024-3795. doi: https://doi.org/10.1016/0024-3795(79)90018-1. url: http://www.sciencedirect.com/science/article/pii/0024379579900181.

[97] P.C. Gilmore e R.C. Gomory. “A linear programming approach to the cutting stock problem”. Em: Operations Research 9 (1961), pp. 849–859.

[98] P.C. Gilmore e R.C. Gomory. “A linear programming approach to the cutting stock problem – Part 2”. Em: Operations Research 11 (1963), pp. 863–888.

[99] Michel X. Goemans e David P. Williamson. “A General Approximation Technique for Constrained Forest Problems”. Em: SIAM Journal on Computing 24.2 (1995), pp. 296–317. doi: 10.1137/S0097539793242618. eprint: https://doi.org/10.1137/S0097539793242618.

[100] E. Goldbarg, M. Goldbarg e H. Luna. Programação Linear e Fluxos em Redes. ELSEVIER EDITORA, 2014. isbn: 9788535278149. url: https://books.google.com.br/books?id=EMcMvgAACAAJ.

[101] Andrew V Goldberg. "An Efficient Implementation of a Scaling Minimum-Cost Flow Algorithm". Em: Journal of Algorithms 22.1 (1997), pp. 1–29. issn: 0196-6774. doi: https://doi.org/10.1006/jagm.1995.0805. url: https://www.sciencedirect.com/science/article/pii/S019667748570805X.

[102] Andrew V. Goldberg, Éva Tardos e Robert E. Tarjan. "Network Flow Algorithms". Em: 1989.

[103] Andrew V. Goldberg e Robert E. Tarjan. "A new approach to the maximum flow problem". Em: Symposium on the Theory of Computing. 1986.

[104] Ralph E. Gomory. "Outline of an algorithm for integer solutions to linear programs". Em: Bull. Amer. Math. Soc. 64.5 (set. de 1958), pp. 275–278.

[105] Ralph E. Gomory. "Some polyhedra related to combinatorial problems". Em: Linear Algebra and its Applications 2.4 (1969), pp. 451–558.

[106] M. Gondran e M. Minoux. Graphes et algorithmes. Collection de la Direction des études et recherches d'Electricité de France. Eyrolles, 1979. url: https://books.google.com.br/books?id=i4hTnQEACAAJ.

[107] M. Gondran e M. Minoux. Graphs and Algorithms. Wiley, 1984.

[108] R. L. Graham. "Bounds for certain multiprocessor anomalies". Em: Bell System Technical Journal 45 (1966), pp. 1563–1581.

[109] R.L. Graham et al. "Optimization and approximation in deterministic sequencing and scheduling: A survey". Em: Annals of Discrete Mathematics 5 (1979), pp. 287–326.

[110] Martin Grötschel. "Polyhedral Combinatorics". Em: Combinatorial Optimization. Ed. por Michael O'Heigeartaigh, Jan Lenstra e Alexander Kan. 1984, pp. 1–10.

[111] Martin Grötschel, László Lovász e Alexander Schrijver. Geometric Algorithms and Combinatorial Optimization. New York: Springer-Verlag, 1988.

[112] Omprakash K. Gupta e A. Ravindran. “Branch and Bound Experiments in Convex Nonlinear Integer Programming”. Em: Management Science 31.12 (1985), pp. 1533–1546. doi: 10.1287/mnsc.31.12.1533.

[113] F. Harary. Graph Theory. Addison Wesley series in mathematics. Addison-Wesley, 1971. url: https://books.google.com.br/books?id=q8OWtwEACAAJ.

[114] Frank Harary, Robert Z. Norman e Dorwin Cartwright. Structural Models: An Introduction to the Theory of Directed Graphs. 4ª ed. New York: John Wiley & Sons, 1965.

[115] Michael Held, Philip Wolfe e Harlan P. Crowder. “Validation of Subgradient Optimization”. Em: Math. Program. 6.1 (dez. de 1974), pp. 62–88. issn: 0025-5610. doi: 10.1007/BF01580223. url: https://doi.org/10.1007/BF01580223.

[116] D. Hilbert, A. Speiser e H. Weyl. Gesammelte Abhandlungen von Hermann Minkowski. v. 2. B. G. Teubner, 1911.

[117] D. S. Hochbaum e D. B. Shmoys. “A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approach”. Em: SIAM Journal on Computing 17 (1988), pp. 539–551.

[118] A. J. Hoffman. “A generalization of max flow—min cut”. Em: Mathematical Programming 6.1 (dez. de 1974), pp. 352–359. issn: 1436-4646. doi: 10.1007/BF01580250. url: https://doi.org/10.1007/BF01580250.

[119] Alan J. Hoffman e Joseph B. Kruskal. “Integral Boundary Points of Convex Polyhedra”. Em: Linear Inequalities and Related Systems. Ed. por H.W. Kuhn e A.J. Tucker. Reimpresso em 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art, p.49-76, 2010. Princeton University Press, 1956, pp. 223–246. isbn: 978-3-540-68279-0. doi: 10.1007/978-3-540-68279-0_3. url: https://doi.org/10.1007/978-3-540-68279-0%5C_3.

[120] Alon Itai e Yossi Shiloach. “Maximum Flow in Planar Networks”. Em: SIAM J. Comput. 8 (1979), pp. 135–150.

[121] K. Jain e V. V. Vazirani. “Approximation algorithms for metric facility location and k-Median problems using the primal-dual schema and Lagrangian relaxation”. Em: Journal of the ACM 48.2 (2001), pp. 274–296.

[122] K. Jain et al. “Greedy Facility Location Algorithms Analyzed using Dual Fitting with Factor-Revealing LP”. Em: Journal of the ACM 50.6 (2003), pp. 795–824.

[123] David S. Johnson. “Approximation algorithms for combinatorial problems”. Em: Journal of Computer and System Sciences 9.3 (1974), pp. 256–278. issn: 0022-0000. doi: 10.1016/S0022-0000(74)80044-9. url: http://www.sciencedirect.com/science/article/pii/S0022000074800449.

[124] Camille Jordan. “Sur les assemblages de lignes.” Em: Journal für die reine und angewandte Mathematik (Crelles Journal) (1869), pp. 185–190.

[125] R.M. Karp. “Reducibility among Combinatorial Problems”. Em: In: Miller R.E., Thatcher J.W., Bohlinger J.D. (eds) Complexity of Computer Computations. The IBM Research Symposia Series (1972), pp. 85–103.

[126] Richard M. Karp. “Reducibility among Combinatorial Problems”. Em: Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations. Ed. por Raymond E. Miller, James W. Thatcher e Jean D. Bohlinger. Boston, MA: Springer, 1972, pp. 85–103.

[127] Marek Karpinski e Alexander Zelikovsky. “New Approximation Algorithms for the Steiner Tree Problems”. Em: Journal of Combinatorial Optimization 1.1 (mar. de 1997), pp. 47–65. issn: 1573-2886. doi: 10.1023/A:1009758919736. url: https://doi.org/10.1023/A:1009758919736.

[128] Alexander Karzanov. “Determining the maximal flow in a network by the method of preflows”. Em: Doklady Mathematics 15 (fev. de 1974), pp. 434–437.

[129] Subhash Khot e Oded Regev. “Vertex cover might be hard to approximate to within 2-ε”. Em: Journal of Computer and System Sciences 74.3 (2008). Computational Complexity 2003, pp. 335–349. issn: 0022-0000. doi: 10.1016/j.jcss.2007.06.019. url: http://www.sciencedirect.com/science/article/pii/S0022000007000864.

[130] G. Kirchhoff. “Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird”. Em: Annalen der Physik und Chemie 148 (1847), pp. 497–508.

[131] Jon Kleinberg e Eva Tardos. Algorithm Design. USA: Addison-Wesley Longman Publishing Co., Inc., 2005. isbn: 0321295358.

[132] Donald E. Knuth. “Aztec Diamonds, Checkerboard Graphs, and Spanning Trees”. Em: J. Algebraic Comb. 6.3 (1997), pp. 253–257. issn: 0925-9899. doi: 10.1023/A:1008605912200. url: https://doi.org/10.1023/A:1008605912200.

[133] Viviane Köhler, Marcia Fampa e Olinto Araújo. “Mixed-Integer Linear Programming Formulations for the Software Clustering Problem”. Em: Computational Optimization and Applications 55.1 (mai. de 2013), pp. 113–135. issn: 1573-2894. doi: 10.1007/s10589-012-9512-9. url: https://doi.org/10.1007/s10589-012-9512-9.

[134] Hugo Harry Kramer et al. “Column generation approaches for the software clustering problem”. Em: Computational Optimization and Applications 64.3 (jul. de 2016), pp. 843–864. issn: 1573-2894. doi: 10.1007/s10589-015-9822-9. url: https://doi.org/10.1007/s10589-015-9822-9.

[135] Jan Kronqvist, Andreas Lundell e Tapio Westerlund. “The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming”. Em: Journal of Global Optimization 64.2 (2016), pp. 249–272. issn: 1573-2916. doi: 10.1007/s10898-015-0322-3. url: http://dx.doi.org/10.1007/s10898-015-0322-3.

[136] Jan Kronqvist et al. “A Review and Comparison of Solvers for Convex MINLP”. Em: Submitted for publication in Optimization and Engineering (2018). url: http://www.optimization-online.org/DB%5C_HTML/2018/06/6650.html.

[137] Joseph B. Kruskal. “On the shortest spanning subtree of a graph and the traveling salesman problem”. Em: Proceedings of the American Mathematical Society 7.1 (1956), pp. 48–48.

[138] J. K. Lenstra, D. B. Shmoys e Eva Tardos. “Approximation algorithms for scheduling unrelated parallel machines”. Em: Mathematical Programming 46 (1990), pp. 259–271.

[139] H. W. Lenstra Jr. “Integer Programming with a Fixed Number of Variables”. Em: Mathematics of Operations Research 8 (1990), pp. 538–548.

[140] Sven Leyffer. “Integrating SQP and Branch-and-Bound for Mixed Integer Nonlinear Programming”. Em: Comput. Optim. Appl. 18 (3 mar. de 2001), pp. 295–309. issn: 0926-6003. doi: 10.1023/A:1011241421041. url: http://dl.acm.org/citation.cfm?id=380781.380829.

[141] Sven Leyffer et al. “Applications and algorithms for mixed integer nonlinear programming”. Em: Journal of Physics: Conference Series 180.1 (2009), p. 012014. url: http://stacks.iop.org/1742-6596/180/i=1/a=012014.

[142] D. Li, J. Wang e X.L. Sun. “Computing exact solution to nonlinear integer programming: Convergent Lagrangian and objective level cut method”. English. Em: Journal of Global Optimization 39 (1 2007), pp. 127–154. issn: 0925-5001. doi: 10.1007/s10898-006-9128-7. url: http://dx.doi.org/10.1007/s10898-006-9128-7.

[143] Shi Li. “A 1.488 approximation algorithm for the uncapacitated facility location problem”. Em: Information and Computation 222.0 (2013). 38th International Colloquium on Automata, Languages and Programming (ICALP 2011), pp. 45–58. issn: 0890-5401. doi: 10.1016/j.ic.2012.01.007. url: http://www.sciencedirect.com/science/article/pii/S0890540112001459.

[144] Leo Liberti, Nenad Mladenović e Giacomo Nannicini. “A recipe for finding good solutions to MINLPs”. Em: Mathematical Programming Computation 3 (4 2011). 10.1007/s12532-011-0031-y, pp. 349–390. issn: 1867-2949. url: http://dx.doi.org/10.1007/s12532-011-0031-y.

[145] J.-H. Lin e J. S. Vitter. “ε-Approximations with Minimum Packing Constraint Violation”. Em: Proc. 24th Annu. ACM Sympos. Theory Comput. 1992, pp. 771–782.

[146] L. Lovasz e M.D. Plummer. Matching Theory. Vol. 29. Annals of Discrete Mathematics. Amsterdam: North-Holland, 1986.

[147] L. Lovász. Combinatorial Problems and Exercises. AMS/Chelsea publication. North-Holland Publishing Company, 1993. isbn: 9780821869475. url: https://books.google.com.br/books?id=e99fXXYx9zcC.

[148] L. Lovász e M.D. Plummer. Matching Theory. AMS Chelsea Publishing Series. AMS Chelsea Pub., 2009. isbn: 9780821847596. url: https://books.google.com.br/books?id=OaoJBAAAQBAJ.

[149] Marco E. Lübbecke e Jacques Desrosiers. “Selected Topics in Column Generation”. Em: Oper. Res. 53.6 (nov. de 2005), pp. 1007–1023. issn: 0030-364X. doi: 10.1287/opre.1050.0234. url: https://doi.org/10.1287/opre.1050.0234.

[150] Simon Lucchesi C. Aspectos Teóricos da Computação. IMPA. isbn: 9788524401114.

[151] Carsten Lund e Mihalis Yannakakis. “On the Hardness of Approximating Minimization Problems”. Em: J. ACM 41.5 (set. de 1994), pp. 960–981. issn: 0004-5411. doi: 10.1145/185675.306789.

[152] A.F.U.S. Macambira et al. Programação Linear. Editora da UFPB, 2016. isbn: 9788523712488.

[153] N. Maculan. “Linear 0-1 Programming”. Em: Handbook of Applied Optimization. Oxford University Press, 2002, pp. 440–445.

[154] N. Maculan e M. H. C. Fampa. Otimização Linear. Brasília: EdUnB, 2006. isbn: 9788523009274. url: https://books.google.com.br/books?id=sO2VPgAACAAJ.

[155] N. Maculan, G. Plateau e A. Lisser. “Integer linear models with a polynomial number of variables and constraints for some classical combinatorial optimization problems”. Em: Pesquisa Operacional 23.1 (2003), pp. 161–168.

[156] N. Maculan et al. “A new formulation for scheduling unrelated processor under precedence constraints”. Em: RAIRO Operations Research 33.1 (1999), pp. 87–92.

[157] N. Maculan et al. “Column-generation in integer linear programming”. Em: RAIRO - Operations Research 37 (2003), pp. 67–83.

[158] Vishv Malhotra, M.Pramodh Kumar e S.N. Maheshwari. “An O(V3) Algorithm for Finding Maximum Flows in Networks”. Em: Information Processing Letters 7 (out. de 1978), pp. 277–278. doi: 10.1016/0020-0190(78)90016-9.

[159] R. E. Marsten, W. W. Hogan e J. W. Blankenship. “The Boxstep Method for Large-Scale Optimization”. Em: Oper. Res. 23.3 (jun. de 1975), pp. 389–405. issn: 0030-364X. doi: 10.1287/opre.23.3.389. url: https://doi.org/10.1287/opre.23.3.389.

[160] Roy E. Marsten. THe Use of the Box Step Method in Discrete Optimization. NBER Working Papers 0086. National Bureau of Economic Research, Inc, mai. de 1975. url: https://ideas.repec.org/p/nbr/nberwo/0086.html.

[161] Wendel Melo. “Agoritmos para Programação Não Linear Inteira Mista”. dissertação de mestrado. Rio de Janeiro, RJ, Brasil: COPPE, Universidade Federal do Rio de Janeiro, 2012.

[162] Wendel Melo. “Novas Abordagens de Solução para Programação Não Linear Inteira Mista Binária e Programação Quadrática Não Convexa.” tese de doutorado. Rio de Janeiro, RJ, Brasil: COPPE, Universidade Federal do Rio de Janeiro, 2016.

[163] Wendel Melo, Marcia Fampa e Fernanda Raupp. “An overview of MINLP algorithms and their implementation in Muriqui Optimizer”. Em: Annals of Operations Research (mai. de 2018). issn: 1572-9338. doi: 10.1007/s10479-018-2872-5. url: https://doi.org/10.1007/s10479-018-2872-5.

[164] Wendel Melo, Marcia Fampa e Fernanda Raupp. “Integrality gap minimization heuristics for binary mixed integer nonlinear programming”. Em: Journal of Global Optimization 71.3 (jul. de 2018), pp. 593–612. issn: 1573-2916. doi: 10.1007/s10898-018-0623-4. url: https://doi.org/10.1007/s10898-018-0623-4.

[165] Wendel Melo, Marcia Fampa e Fernanda Raupp. “Integrating nonlinear branch-and-bound and outer approximation for convex Mixed Integer Nonlinear Programming”. English. Em: Journal of Global Optimization 60.2 (2014), pp. 373–389. issn: 0925-5001. doi: 10.1007/s10898-014-0217-8. url: http://dx.doi.org/10.1007/s10898-014-0217-8.

[166] R. R. Meyer. “On the existence of optimal solutions to integer and mixed-integer programming problems”. Em: Mathematical Programming 7.1 (1974), pp. 223–235. issn: 1436-4646. doi: 10.1007/BF01585518. url: https://doi.org/10.1007/BF01585518.

[167] C.E. Miller, A.W. Tucker e R.A. Zemlin. “Integer Programming Formulation of Travelling Salesman Problems”. Em: J. ACM 7 (1960), pp. 326–329.

[168] Walter Murray e Kien-Ming Ng. “An algorithm for nonlinear optimization problems with binary variables”. Em: Computational Optimization and Applications 47 (2 2010). 10.1007/s10589-008-9218-1, pp. 257–288. issn: 0926-6003. url: http://dx.doi.org/10.1007/s10589-008-9218-1.

[169] G. Nannicini, P. Belotti e L. Liberti. “A local branching heuristic for MINLPs”. Em: ArXiv e-prints (dez. de 2008). arXiv: 0812.2188. url: http://adsabs.harvard.edu/abs/2008arXiv0812.2188N.

[170] G.L. Nemhauser e L.A. Wolsey. Integer and Combinatorial Optimization. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2014. isbn: 9781118626863. url: https://books.google.com.br/books?id=MvBjBAAAQBAJ.

[171] O. Ore. Theory of graphs. American Mathematical Society colloquium publications. American Math. Soc., 1967. url: https://books.google.com.br/books?id=QifJxwEACAAJ.

[172] Manfred W. Padberg. “On the facial structure of set packing polyhedra”. Em: Mathematical Programming 5.1 (1973), pp. 199–215.

[173] Christos H. Papadimitriou. “Efficient search for rationals”. Em: Information Processing Letters 8.1 (1979), pp. 1–4. issn: 0020-0190.

[174] A. Pessoa et al. “Automation and Combination of Linear-Programming Based Stabilization Techniques in Column Generation”. Em: INFORMS Journal on Computing 30.2 (2018), pp. 339–360. doi: 10.1287/ijoc.2017.0784. eprint: https://doi.org/10.1287/ijoc.2017.0784. url: https://doi.org/10.1287/ijoc.2017.0784.

[175] Boris Polyak. “A General Method for Solving Extremum Problems”. Em: Soviet Mathematics. Doklady 8 (jan. de 1967).

[176] R. C. Prim. “Shortest connection networks and some generalizations”. Em: The Bell System Technical Journal 36.6 (nov. de 1957), pp. 1389–1401. issn: 0005-8580. doi: 10.1002/j.1538-7305.1957.tb01515.x.

[177] Hans Jürgen Prömel e Angelika Steger. “A New Approximation Algorithm for the Steiner Tree Problem with Performance Ratio 5/3”. Em: Journal of Algorithms 36.1 (2000), pp. 89–101. issn: 0196-6774. doi: https://doi.org/10.1006/jagm.2000.1086. url: http://www.sciencedirect.com/science/article/pii/S0196677400910861.

[178] A. L. Puccini. Introdução à Programação Linear. Livros Técnicos e Científicos - LTC, 1975.

[179] W. R. Pulleyblank. “Polyhedral Combinatorics”. Em: Mathematical Programming The State of the Art: Bonn 1982. Ed. por Achim Bachem, Bernhard Korte e Martin Grötschel. Berlin, Heidelberg: Springer, 1983, pp. 312–345. isbn: 978-3-642-68874-4. doi: 10.1007/978-3-642-68874-4{_}13. url: $https://doi.org/10.1007/978-3-642-68874-4%5C_13$.

[180] W. R. Pulleyblank. “Polyhedral Combinatorics”. Em: Optimization. Ed. por G. L. Nemhauser, A. H. G. Rinnooy Kan e M. J. Todd. New York, NY: Elsevier, 1989, pp. 371–446. isbn: 0-444-87284-1. url: http://dl.acm.org/citation.cfm?id=94386.94403.

[181] Ignacio Quesada e Ignacio E. Grossmann. “An LP/NLP based branch and bound algorithm for convex MINLP optimization problems”. Em: Computers & Chemical Engineering 16.10-11 (1992). An International Journal of Computer Applications in Chemical

Engineering, pp. 937–947. issn: 0098-1354. doi: 10.1016/0098-1354(92)80028-8.

[182] M. Raghavachari. “On Connections Between Zero-One Integer Programming and Concave Programming Under Linear Constraints”. Em: Operations Research 17.4 (July/August 1969), pp. 680–684. doi: 10.1287/opre.17.4.680.

[183] P. Raghavan e C. D. Thompson. “Randomized rounding”. Em: Combinatorica 7 (1987), pp. 365–374.

[184] R. Ravi et al. “Spanning Trees—Short or Small”. Em: SIAM Journal on Discrete Mathematics 9.2 (1996), pp. 178–200. doi: 10.1137/S0895480194266331. eprint: https://doi.org/10.1137/S0895480194266331. url: https://doi.org/10.1137/S0895480194266331.

[185] Gabriel Robins e Alexander Zelikovsky. “Tighter Bounds for Graph Steiner Tree Approximation”. Em: SIAM Journal on Discrete Mathematics 19.1 (2005), pp. 122–134. doi: 10.1137/S0895480101393155. eprint: http://dx.doi.org/10.1137/S0895480101393155. url: http://dx.doi.org/10.1137/S0895480101393155.

[186] R.T. Rockafellar. Convex Analysis. Princeton University Press, 2015.

[187] T.L. Saaty e P.C. Kainen. The Four-color Problem: Assaults and Conquest. Advanced Book Program. McGraw-Hill International Book Company, 1977. isbn: 9780070543829. url: https://books.google.com.br/books?id=ez%5C_vAAAAMAAJ.

[188] A. Schrijver. Combinatorial Optimization: Polyhedra and Efficiency. Algorithms and Combinatorics. Springer, 2002. isbn: 9783540443896. url: https://books.google.com.br/books?id=mqGeSQ6dJycC.

[189] A. Schrijver. “On total dual integrality”. Em: Linear Algebra and its Applications 38.Supplement C (1981), pp. 27–32. issn: 0024-3795. doi: https://doi.org/10.1016/0024-3795(81)90005-7. url: http://www.sciencedirect.com/science/article/pii/0024379581900057.

[190] A. Schrijver. Theory of Linear and Integer Programming. Wiley Series in Discrete Mathematics & Optimization. John Wiley & Sons, 1998. isbn: 9780471982326. url: https://books.google.com.br/books?id=zEzW5mhppB8C.

[191] Alexander Schrijver. “On the History of the Transportation and Maximum Flow Problems”. Em: 91.3 (fev. de 2002), pp. 437–445. issn: 0025-5610. doi: 10.1007/s101070100259. url: https://doi.org/10.1007/s101070100259.

[192] Alexander Schrijver. “Polyhedral Combinatorics”. Em: Handbook of Combinatorics (Vol. 2). Ed. por R. L. Graham, M. Grötschel e L. Lovász. Cambridge, MA, USA: MIT Press, 1995, pp. 1649–1704. isbn: 0-262-07171-1. url: http://dl.acm.org/citation.cfm?id=233228.233239.

[193] P. Schuurman e G. Woeginger. “Approximation schemes-a tutorial”. Em: Lectures on Scheduling. Ed. por R. Moehring et al. To appear.

[194] P.D. Seymour. “Decomposition of regular matroids”. Em: Journal of Combinatorial Theory, Series B 28.3 (1980), pp. 305–359.

[195] M. H. Simonsen. Introdução à Programação Linear. Notas de Matemática 8. IMPA, 1958.

[196] D. D. Sleator e R. E. Tarjan. “A data structure for dynamic trees”. Em: Journal of Computer and System Sciences 26.3 (1983), pp. 362–391. issn: 00220000. doi: https://doi.org/10.1016/0022-0000(83)90006-5.

[197] J.P. Spinrad. Efficient Graph Representations. Fields Institute Monographs 2472-4173. American Mathematical Society, 2003. isbn: 9781470431464. url: https://books.google.com.br/books?id=BsqOtgEACAAJ.

[198] Claus Still e Tapio Westerlund. “Solving Convex MINLP Optimization Problems Using a Sequential Cutting Plane Algorithm”. Em: Computational Optimization and Applications 34 (1 2006). 10.1007/s10589-005-3076-x, pp. 63–83. issn: 0926-6003.

[199] Robert A. Stubbs e Sanjay Mehrotra. “A branch-and-cut method for 0-1 mixed convex programming”. Em: Mathematical Programming 86 (3 1999). 10.1007/s101070050103, pp. 515–532. issn: 0025-5610.

[200] Maxim Sviridenko. “An Improved Approximation Algorithm for the Metric Uncapacitated Facility Location Problem”. Em: Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization. 2002, pp. 240–257.

[201] J. L. Szwarcfiter. Teoria Computacional de Grafos. ELSEVIER, 2018. isbn: 978-85-352-8884-1.

[202] J.L. Szwarcfiter. Teoria computacional de grafos: Os Algoritmos. ELSEVIER, 2018. isbn: 9788535288858. url: https://books.google.com.br/books?id=OTBbDwAAQBAJ.

[203] Klaus Truemper. “A decomposition theory for matroids. V. Testing of matrix total unimodularity”. Em: Journal of Combinatorial Theory, Series B 49.2 (1990), pp. 241–281.

[204] W.T. Tutte. Connectivity in Graphs. Mathematical expositions. University of Toronto Press, 1966. isbn: 9780802014252. url: https://books.google.com.br/books?id=lppsAAAAMAAJ.

[205] V. Chvátal. “A Greedy Heuristic for the Set-Covering Problem”. Em: Math. of Oper. Res. 4.3 (1979), pp. 233–235.

[206] François Vanderbeck. “Branching in branch-and-price: a generic scheme”. Em: Mathematical Programming 130.2 (dez. de 2011), pp. 249–294. issn: 1436-4646. doi: 10.1007/s10107-009-0334-1. url: https://doi.org/10.1007/s10107-009-0334-1.

[207] V. V. Vazirani. Approximation Algorithms. Springer-Verlag, 2001.

[208] Arthur F. Veinott. “The Supporting Hyperplane Method for Unimodal Programming”. Em: Operations Research 15.1 (1967), pp. 147–152. issn: 0030364X, 15265463. url: http://www.jstor.org/stable/168519.

[209] José Verschae e Andreas Wiese. “On the configuration-LP for scheduling on unrelated machines”. Em: J. Scheduling 17.4 (2014), pp. 371–383. doi: 10.1007/s10951-013-0359-4.

[210] Matthias Walter e Klaus Truemper. “Implementation of a unimodularity test”. Em: Mathematical Programming Computation 5.1 (2013), pp. 57–73.

[211] Douglas B. West. Introduction to Graph Theory. 2ª ed. Prentice Hall, 2000. isbn: 0130144002.

[212] Tapio Westerlund e Frank Pettersson. “An extended cutting plane method for solving convex MINLP problems”. Em: Computers & Chemical Engineering 19, Supplement 1.0 (1995). European Symposium on Computer Aided Process Engineering, pp. 131–136. issn: 0098-1354. doi: 10.1016/0098-1354(95)87027-X.

[213] David P. Williamson e David B. Shmoys. The Design of Approximation Algorithms. Cambridge: Cambridge University Press, 2011. url: http://www.designofapproxalgs.com.

[214] R.J. Wilson. Introduction to Graph Theory. Longman, 1972. isbn: 9780582447622. url: https://books.google.com.br/books?id=3cpTAAAAYAAJ.

[215] L. A. Wolsey. “Valid Inequalities and Superadditivity for 0–1 Integer Programs”. Em: Mathematics of Operations Research 2.1 (1977), pp. 66–77.

[216] L.A. Wolsey. Integer Programming. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998. isbn: 9780471283669. url: https://books.google.com.br/books?id=x7RvQgAACAAJ.

[217] Laurence A. Wolsey. Integer programming. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1998. isbn: 9780471283669. url: http://books.google.com.br/books?id=x7RvQgAACAAJ.

[218] Álinson S. Xavier e Manoel Campêlo. “A New Facet Generating Procedure for the Stable Set Polytope”. Em: Electronic Notes in Discrete Mathematics 37 (2011), pp. 183–188.

[219] XPRESS-MP. Software. 2020. url: http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx.

[220] Neal E. Young. “Greedy Set-Cover Algorithms”. Em: Encyclopedia of Algorithms. Ed. por Ming-Yang Kao. Boston, MA: Springer US, 2008, pp. 379–381. isbn: 978-0-387-30162-4. doi: $10.1007/978-0-387-30162-4_175$.

[221] A. Z. Zelikovsky. “An 11/6-approximation algorithm for the network steiner problem”. Em: Algorithmica 9.5 (mai. de 1993), pp. 463–470. issn: 1432-0541. doi: 10.1007/BF01187035. url: https://doi.org/10.1007/BF01187035.

[222] Eitan Zemel. “Lifting the facets of zero–one polytopes”. Em: Mathematical Programming 15.1 (1978), pp. 268–277.

[223] G.M. Ziegler. Lectures on Polytopes. 2a. Springer, 1998.