Apêndices do livro

"Uma introdução sucinta a algoritmos de aproximação"
(livro do 23o. Colóquio Brasileiro de Matemática - 2001)

APÊNDICE A

Teoria dos Grafos

Este apêndice é um resumo da terminologia e notação básicas da teoria dos grafos. A notação é consistente com a de textos clássicos [BM76, Die00] sobre o assunto.

Um **grafo** é um par (V, E), onde V é um conjunto finito arbitrário e E um conjunto de pares não-ordenados[1] de elementos de V. Os elementos de V são chamados **vértices** e os de E são **arestas** (*edges*). Se vw é uma aresta, os vértices v e w são os **extremos** da aresta. Dois vértices v e w de um grafo são **adjacentes** se existe uma aresta com extremos v e w. Se cada vértice é adjacente a todos os outros, então o grafo é **completo**.

O conjunto de vértices de um grafo G é denotado por V_G e o conjunto de arestas por E_G. Se G é completo, então $|E_G| = \frac{1}{2}n(n-1)$, onde $n := |V_G|$. O **tamanho** (veja apêndice E) de um grafo G é o número $\langle G \rangle := |V_G| + |E_G|$. V_G E_G $\langle G \rangle$

Cortes e graus

Para qualquer conjunto X de vértices de um grafo G, denotamos por $\delta_G(X)$, ou simplesmente por $\delta(X)$, o conjunto de todas as arestas que têm um extremo em X e outro em $V_G \setminus X$. Um **corte** (*cut*) é qualquer conjunto da forma $\delta(X)$, onde X é um subconjunto de V_G. Um conjunto X de vértices **separa** um vértice x de outro y se $x \in X$ enquanto $y \in V_G \setminus X$. Nessas condições, dizemos também que o corte $\delta(X)$ separa x de y. $\delta(X)$

[1] Um par não-ordenado é um conjunto com exatamente dois elementos. Um par não-ordenado como $\{v, w\}$ é denotado, indiferentemente, por vw ou wv.

Se v é um vértice, escrevemos $\delta(v)$ no lugar de $\delta(\{v\})$. O número $|\delta(v)|$ é o **grau** de v. A soma dos graus de todos os vértices de um grafo é igual ao dobro do número de arestas: $\sum_{v \in V_G} |\delta(v)| = 2|E_G|$.

Um vértice é **isolado** se tem grau 0. O **grau máximo** em G é o
Δ número $\Delta(G) := \max_v |\delta(v)|$.

Subgrafos

Um grafo H é **subgrafo** de um grafo G se $V_H \subseteq V_G$ e $E_H \subseteq E_G$. Se uma das inclusões é própria, H é subgrafo **próprio** de G. Se $V_H = V_G$, dizemos que H é um subgrafo **gerador** (*spanning subgraph*) de G.

$G - F$ Para qualquer conjunto F de arestas de G, denotamos por $G - F$ o subgrafo gerador de G cujo conjunto de arestas é $E_G \setminus F$. Se $F = \{f\}$, podemos escrever $G - f$.

Dizemos que H é um subgrafo **induzido** de G se E_H é o conjunto de todas as arestas de G que têm ambos os extremos em V_H. O subgrafo
$G[X]$ induzido de G que tem X por conjunto de vértices é denotado por $G[X]$.
$G - Y$ Para qualquer subconjunto Y de V_G, denotamos por $G - Y$ o grafo $G[V_G \setminus Y]$. Se $Y = \{y\}$, podemos escrever $G - y$.

O subgrafo de G **induzido** por um subconjunto F de E_G é o grafo $(V(F), F)$, onde $V(F)$ é o conjunto de todos os vértices que são extremo
$G[F]$ de algum elemento de F. Podemos denotar esse subgrafo por $G[F]$.

Se F é um conjunto de pares não-ordenados de elementos de V_G,
$G + F$ denotamos por $G + F$ o grafo $(V_G, E_G \cup F)$ de G. Se $F = \{f\}$, podemos escrever $G + f$.

Passeios, ciclos, caminhos e circuitos

Um **passeio** (*walk*) é uma seqüência $(v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ em que $v_0, \ldots, v_k$ são vértices e cada e_i é uma aresta com extremos v_{i-1} e v_i. Dizemos que v_0 é a **origem** e que v_k é o **término** do passeio. Para simplificar, podemos omitir as arestas e representar um passeio por sua seqüência de vértices, $(v_0, v_1, \ldots, v_k)$. O passeio é **fechado** se $v_k = v_0$ e **aberto** em caso contrário.

Uma **trilha** é um passeio cujas arestas são distintas duas a duas, ou seja, um passeio sem arestas repetidas. Um **caminho** (*path*) é um passeio cujos vértices são distintos dois a dois, ou seja, um passeio sem vértices repetidos. A origem e o término do caminho são os seus **extremos.**

Um **ciclo** (*cycle*) é uma trilha fechada com pelo menos três arestas.

Um **circuito** é um ciclo $(v_0, v_1, \ldots, v_k)$ tal que $v_i \neq v_j$ sempre que $i < j$, exceto se $i = 0$ e $j = k$.

Um passeio é **gerador** se contém todos os vértices do grafo. É claro que a definição se aplica, em particular, a trilhas, caminhos, ciclos e circuitos. Um caminho gerador é também conhecido como **caminho hamiltoniano.** Analogamente, um circuito gerador é conhecido como **circuito hamiltoniano.**

O conjunto de arestas de um caminho ou circuito C é denotado por E_C. O **comprimento** de um caminho ou circuito C é $|E_C|$, ou seja, o número de arestas de C. E_C

Conexão e componentes

Um grafo é **conexo** se, para todo par (v, w) de seus vértices, existe um passeio com origem v e término w. É claro que um grafo G é conexo se e somente se $\delta_G(X) \neq \emptyset$ para todo subconjunto não vazio e próprio X de V_G.

Um **componente** de um grafo G é qualquer subgrafo conexo maximal[2] de G. Às vezes convém confundir um componente com o seu conjunto de vértices.

Florestas e árvores

Uma **floresta** é um grafo sem ciclos. Se F é uma floresta, então $|E_F| = |V_F| - c(F)$, onde $c(F)$ é o número de componentes de F.

Uma **árvore** (*tree*) é uma floresta conexa. Portanto, cada componente de uma floresta é uma árvore. Se T é uma árvore, então $|E_T| = |V_T| - 1$. Para qualquer par (x, y) de vértices em uma árvore, existe um e um só caminho de x a y.

Uma **floresta geradora** de um grafo G é qualquer subgrafo gerador de G que seja uma floresta, isto é, qualquer floresta F tal que $V_F = V_G$ e $E_F \subseteq E_G$. Uma **árvore geradora** (*spanning tree*) de um grafo G é qualquer subgrafo gerador de G que seja uma árvore. Todo grafo conexo tem uma árvore geradora.

---

[2] Um subgrafo H de G é **maximal** com relação a uma dada propriedade se H não é subgrafo próprio de um outro subgrafo H' de G que tenha a propriedade em questão. Subgrafos **minimais** são definidos de maneira análoga.

Grafos bipartidos

Uma **bipartição** de um grafo G é qualquer partição $\{X, Y\}$ de V_G tal que $\delta(X) = E_G$. Um grafo admite uma bipartição se e somente se não tem circuito de comprimento ímpar [BM76]. Um **grafo bipartido** (*bipartite graph*) é um grafo munido de uma bipartição.

Grafos planares

Um grafo é **planar** se for isomorfo a um mapa. Mais precisamente, um grafo G é planar se existe um mapa (V, E) e uma bijeção de ψ de V_G em V tal que, para todo par (v, w) de vértices de G, tem-se $vw \in E_G$ se e somente se algum elemento de E tem extremos $\psi(v)$ e $\psi(w)$.

Um **mapa** é um par (V, E) onde (1) V é um conjunto finito de pontos do plano, (2) todo elemento de E é um arco entre dois elementos de V, (3) elementos diferentes de E têm diferentes conjuntos de extremos, (4) o interior de um elemento de E não contém elementos de V nem pontos que pertençam a outro elemento de E.

Um **arco** é a união de um número finito de segmentos de reta no plano que seja homeomorfa ao intervalo fechado $[0, 1]$ da reta; as imagens de 0 e 1 sob esse homeomorfismo são os **extremos** do arco.

Multigrafos

Às vezes, convém admitir a existência de arestas múltiplas em um grafo, ou seja, a existência de várias "cópias" de uma mesma aresta. Dizemos nesse caso que temos um multigrafo. Mais explicitamente, um **multigrafo** consiste em um conjunto finito V e um multiconjunto[3] E de pares não-ordenados de elementos de V. A terminologia e notação definidas para grafos se estendem aos multigrafos com as devidas adaptações. Assim, por exemplo, o grau de um vértice é a somas das multiplicidades das arestas que nele incidem. Outro exemplo: ciclos e circuitos podem ter apenas duas arestas (mas não menos que isso).

Custos e pesos

Suponha que c é uma função que associa um número c_e a cada aresta e de um grafo. Para qualquer conjunto F de arestas do grafo, denotamos

[3] Um multiconjunto pode ter mais de uma cópia de cada um de seus elementos; o número de cópias é a *multiplicidade* do elemento.

por $c(F)$ a soma dos c_f para f em F: $c()$

$$c(F) := \sum_{f \in F} c_f \,.$$

Se $F = \emptyset$, então $c(F) = 0$. Se H é um subgrafo do grafo em questão, definimos $c(H)$ como $c(E_H)$. Se c_e é interpretado como **custo** da aresta e, dizemos que $c(F)$ é o custo de F e $c(H)$ o custo de H. Algo análogo acontece quando c_e é interpretado como **peso** da aresta e.

Vetores e Matrizes

Um **vetor** é uma função que leva um conjunto finito arbitrário — o conjunto de índices — em um conjunto de números reais. Se o conjunto de índices de um vetor x é N, dizemos que x está definido sobre N ou é indexado por N. Se x é um vetor sobre um conjunto N e j é um elemento de N então x_j denota o componente j de x, isto é, o valor da função x em j. Se Q é um subconjunto de N então x_Q denota a restrição de x a Q, ou seja, o vetor cujo componente q é x_q para cada q em Q.

x_Q

Se todos os componentes são números racionais, dizemos que o vetor é **racional**. Se forem todos inteiros, dizemos que o vetor é **inteiro**. Como nossos interesses são computacionais, quase todos os vetores do texto são racionais (a exceção fica no capítulo 7, que trata de programação semidefinida).

Um vetor x é **nulo** se $x_j = 0$ para cada índice j; o vetor nulo será denotado por 0 qualquer que seja o seu conjunto de índices. Se x e y são vetores sobre um mesmo conjunto de índices e $x_j \geq y_j$ para cada j, dizemos que $x \geq y$. A relação $\leq$ é definida de modo análogo.

$x \geq y$

O **vetor característico** de um subconjunto P de N é um vetor x sobre N tal que $x_i = 1$ se $i \in P$ e $x_i = 0$ em caso contrário.

$x \geq 0$

Matrizes

Uma **matriz** é uma função que leva o produto cartesiano de dois conjuntos finitos em um conjunto de números reais. Se uma matriz A tem domínio $M \times N$, dizemos que M é o conjunto de índices de linhas e N o conjunto de índices de colunas de A. Dizemos também que A é uma matriz sobre $M \times N$ ou indexada por $M \times N$. Se i e j são elementos de M e N, respectivamente, então A_{ij} denota o componente (i, j) de A, ou

seja, o valor de A em (i, j).

Uma **linha** de A é um *vetor* sobre N: a linha i de A é o vetor cujo componente j é A_{ij} para cada j em N. A **coluna** j de A é definida analogamente. A coluna j de A será denotada por A_{Mj} ou $A_{\star j}$ e a linha i por A_{iN} ou $A_{i\star}$.

$A_{\star j}$
$A_{i\star}$

Se P e Q são subconjuntos de M e N, respectivamente, então A_{PQ} é a restrição de A a $P \times Q$, ou seja, a matriz sobre $P \times Q$ cujo componente (p, q) vale A_{pq} para cada p em P e cada q em Q.

A_{PQ}

Dizemos que a matriz é **racional** se todos os seus componentes são racionais e **inteira** se todos os seus componentes são inteiros.

A **transposta** de uma matriz A sobre $M \times N$ é a matriz $A^\top$ sobre $N \times M$ definida por $A^\top_{ij} = A_{ji}$. Portanto, a linha i de $A^\top$ é a coluna i de A. É claro que a transposta de $A^\top$ é A. Uma matriz A é **simétrica** se $A^\top = A$.

$A^\top$

Uma matriz é **quadrada** se seus conjuntos de índices de linhas e colunas são iguais. É claro que toda matriz simétrica é quadrada. Uma matriz D é **diagonal** se for quadrada e se $D_{ij} = 0$ sempre que $i \neq j$. É evidente que toda matriz diagonal é simétrica.

Uma matriz **identidade** é uma matriz diagonal I tal que $I_{ii} = 1$ para cada i. Matrizes identidade são denotadas por I, quaisquer que sejam seus conjuntos de índices.

I

Produtos

Para quaisquer vetores x e y sobre um mesmo conjunto N, o produto (interno) de x por y é o número

$$xy \;:=\; \textstyle\sum_{j \in N} x_j y_j \,.$$

É óbvio que $xy = yx$. Por exemplo, se y é o vetor característico de um subconjunto Q de N, então xy é a soma $\sum_{q \in Q} x_q$, que convencionamos denotar por $x(Q)$.

Suponha agora que A é uma matriz sobre $M \times N$. O produto de A por um vetor x sobre N é o vetor

$$Ax$$

sobre M cujo componente i é o produto da linha i de A por x, ou seja, o número $\sum_j A_{ij} x_j$. O produto de um vetor y sobre M por A é o vetor

$$yA$$

cujo componente j é o número $\sum_i y_i A_{ij}$. Em suma, $(Ax)_i = \sum_j A_{ij} x_j$ e $(yA)_j = \sum_i y_i A_{ij}$, ou seja, yA é uma combinação linear das linhas de A enquanto Ax é uma combinação linear das colunas de A.

É evidente que $yA = A^\top y$. É menos evidente, mas não menos verdadeira, a propriedade associativa $y(Ax) = (yA)x$, ou seja, a inversão da ordem das somas:

$$\sum_i y_i (\sum_j A_{ij} x_j) = \sum_j (\sum_i y_i A_{ij}) x_j \,.$$

Para qualquer matriz A sobre $L \times M$ e qualquer matriz B sobre $M \times N$, o produto de A por B é a matriz

$$AB$$

sobre $L \times N$ cujo componente (i,k) é o produto de vetores $A_{i\star} B_{\star k}$. É evidente que $(AB)^\top = B^\top A^\top$. É menos evidente que $(AB)C = A(BC)$, desde que cada produto faça sentido.

Uma matriz quadrada A é **inversível** se existe uma matriz B tal que $AB = BA = I$. Uma tal matriz B é **inversa** de A.

Normas e tamanhos

Denotamos por $\|x\|$ a **norma** do vetor x. Assim, $\|x\| := \sqrt{xx}$. $\|x\|$
Esse número pode ser interpretado como o comprimento geométrico do vetor x.

O **tamanho** de um número inteiro é o número de caracteres (digamos dígitos decimais) na representação do número (veja apêndice E). O tamanho de um número inteiro α é denotado por $\langle\alpha\rangle$. O tamanho de $\langle\alpha\rangle$
um número racional com numerador α e denominador β é a soma dos tamanhos de α e β. O tamanho de α/β é denotado por $\langle\alpha/\beta\rangle$. $\langle\alpha/\beta\rangle$

O **tamanho** de um vetor ou matriz racionais é a soma dos tamanhos de seus componentes. É claro que o tamanho de uma matriz não é inferior ao número de componentes da matriz. O tamanho de uma matriz A é denotado por $\langle A\rangle$. $\langle A\rangle$

O **tamanho** de um conjunto de vetores e matrizes racionais é a soma dos tamanhos dos elementos do conjunto. Assim, o tamanho de um sistema (A, b, c) é $\langle A\rangle + \langle b\rangle + \langle c\rangle$.

Essas definições de tamanho são casos particulares do conceito geral de tamanho a que se refere a seção 1.2 e o apêndice E.

Matrizes positivas semidefinidas

A caracterização de matrizes positivas semidefinidas que descrevemos a seguir é usada no capítulo 7.

Uma matriz quadrada X é **positiva semidefinida** se existe uma matriz quadrada Y tal que

$$X \;=\; YY^{\top}\,.$$

É claro que toda matriz positiva semidefinida é simétrica. O seguinte lema caracteriza tais matrizes.

> **Lema B.1**: *Para toda matriz simétrica X vale uma e apenas uma das seguintes alternativas: existe uma matriz quadrada Y tal que $X = YY^{\top}$ ou existe um vetor y tal que $yXy < 0$.*

A prova do lema depende de alguns conceitos que passamos a definir. Uma matriz quadrada é **elementar** se coincide com a identidade na diagonal e em todas as colunas, exceto talvez uma. Mais precisamente, uma matriz E indexada por $M \times M$ é elementar se existe m em M tal que $E_{mm} = 1$ e $E_{\star j} = I_{\star j}$ para cada j distinto de m. Toda matriz elementar E é inversível e sua inversa, digamos H, também é elementar: $H_{mm} = 1$ e $H_{im} = -E_{im}$ para cada i distinto de m e $H_{\star j} = I_{\star j}$ para cada j distinto de m.

Um **produto de matrizes elementares** é qualquer matriz da forma $E^{(1)} \cdots E^{(p)}$, onde cada $E^{(i)}$ é uma matriz elementar indexada por $M \times M$. Se $H^{(i)}$ é a inversa de $E^{(i)}$ para cada i então o produto de matrizes elementares $H^{(p)} \cdots H^{(1)}$ é a inversa de $E^{(1)} \cdots E^{(p)}$.

Demonstração de B.1: A prova de que apenas uma das alternativas vale é simples: se $X = YY^{\top}$ então

$$yXy = y(YY^{\top})y = (yY)(Y^{\top}y) = (yY)(yY) \geq 0\,.$$

Para provar que uma das alternativas vale, vamos descrever um algoritmo que recebe qualquer matriz simétrica X e devolve uma matriz real Y tal que $X = YY^{\top}$ ou um vetor y tal que $yXy < 0$.

Para qualquer matriz simétrica X indexada por $M \times M$ e qualquer m em M tal que $X_{mm} \neq 0$, é fácil determinar uma matriz elementar E tal que a matriz simétrica $Z := EXE^{\top}$ tem a seguinte estrutura:

$$Z_{mm} = X_{mm} \quad \text{e} \quad Z_{mj} = Z_{jm} = 0$$

	P			$M \setminus P$		
P	-2	0	0	0	0	0
	0	0	0	0	0	0
	0	0	3	0	0	0
$M \setminus P$	0	0	0	0		
	0	0	0		0	
	0	0	0			0

Figura B.1: Estrutura da matriz Z.

para cada j distinto de m. Basta tomar como E a matriz elementar definida por $E_{mm} = 1$, $E_{im} = -X_{im}/X_{mm}$ para cada i distinto de m e $E_{\star j} = I_{\star j}$ para cada j distinto de m.

Segue daí que para qualquer matriz simétrica X indexada por $M \times M$ existe um produto F de matrizes elementares tal que a matriz simétrica

$$Z \;:=\; FXF^\top$$

tem a seguinte estrutura: existe um subconjunto P de M tal que Z_{PP} é uma matriz diagonal, $Z_{P,M\setminus P} = 0$ e, para cada m em $M \setminus P$, $Z_{mm} = 0$ mas $Z_{m\star} \neq 0$. (Veja figura B.1.)

Se $P = M$ e $Z_{pp} \geq 0$ para cada p em P então Z é diagonal e $X = GZG^\top$, onde G é a inversa de F. Portanto,

$$X \;=\; YY^\top\,,$$

onde $Y = G\dot{Z}$ e $\dot{Z}$ é a matriz diagonal definida por $\dot{Z}_{pp} = \sqrt{Z_{pp}}$. Suponha agora que $Z_{mm} < 0$ para algum m em M. Seja u o vetor definido por $u_m = 1$ e $u_i = 0$ para cada i distinto de m. Então $(uF)X(uF) = u(FXF^\top)u = uZu = Z_{mm} < 0$. Em suma,

$$yXy \;<\; 0\,,$$

para $y = uF$. Finalmente, suponha que $P \neq M$ e seja m um elemento de $M \setminus P$. Então $Z_{mm} = 0$ mas $Z_{mj} \neq 0$ para algum j. Seja u o vetor definido da seguinte maneira: $u_j = 1$, $u_i = 0$ para i distinto de m e j e u_m é escolhido de modo que $2u_mZ_{mj} + Z_{jj} < 0$. Então $(uF)X(uF) = uZu < 0$ e portanto

$$yXy \;<\; 0\,,$$

para $y = uF$. □

Convém observar que a matriz Y no lema B.1 poderá não ser racional mesmo que a matriz X seja racional.

```
Algoritmo PosSemidef (X)
 1  Z ← X
 2  F ← G ← I
 3  para m de 1 a n faça
 4     se Z_mm ≠ 0
 5        então E ← H ← I
 6              para i de 1 a n faça E_im ← −Z_im/Z_mm
 7              para i de 1 a n faça H_im ← −E_im
 8              E_mm ← H_mm ← 1
 9              Z ← EZE^T
10              F ← EF
11              G ← GH    ▷   FG = GF = I
12  para m de 1 a n faça
13     se Z_mm < 0
14        então u ← 0
15              u_m ← 1
16              devolva uF
17     se Z_mm = 0
18        então j ← 1
19              enquanto j ≤ n e Z_mj = 0 faça j ← j + 1
20              se j ≤ n
21                 então u ← 0
22                       u_j ← 1
23                       u_m ← −Z_jj/2Z_mj
24                       se Z_mj > 0
25                          então u_m ← u_m − 1
26                          senão u_m ← u_m + 1
27                       devolva uF
28  para i de 1 a n faça Z_ii ← √Z_ii
29  devolva GZ
```

Figura B.2: O algoritmo PosSemidef recebe uma matriz quadrada X indexada por $\{1,\ldots,n\} \times \{1,\ldots,n\}$ e devolve uma matriz quadrada Y tal que $X = YY^\top$ ou um vetor y tal que $yXy < 0$. Se a matriz X é racional, o vetor y é racional e a matriz Y é racional exceto pela raiz quadrada na linha 28. Nessas condições, o consumo de tempo do algoritmo é polinomial, exceto pela raiz quadrada.

Programação Linear

Este apêndice é um resumo da terminologia e de alguns fatos básicos de programação linear [Pad99, Chv83, Sch86, Feo01]. Veja nossas convenções de notação no apêndice B.

Problema de programação linear

Um **problema de programação linear**, ou **programa linear**, ou simplesmente **pl**, consiste no seguinte: Dada uma matriz A indexada por $M \times N$, um vetor b indexado por M, um vetor c indexado por N e partições[1] $\{M_1, M_2, M_3\}$ e $\{N_1, N_2, N_3\}$ de M e N respectivamente, encontrar um vetor x indexado por N que

$$\begin{array}{rrcll} \text{minimize} & cx & & & \\ \text{sob as restrições} & (Ax)_i & \geq & b_i & \text{para cada } i \text{ em } M_1\,, \\ & (Ax)_i & = & b_i & \text{para cada } i \text{ em } M_2\,, \\ & (Ax)_i & \leq & b_i & \text{para cada } i \text{ em } M_3\,, \\ & x_j & \geq & 0 & \text{para cada } j \text{ em } N_1\,, \\ & x_j & \leq & 0 & \text{para cada } j \text{ em } N_3\,. \end{array} \tag{C.1}$$

Usaremos a abreviatura $\mathrm{P}(A, b, c)$ para denotar esse pl; a seqüência de conjuntos $M_1, M_2, M_3, N_1, N_2, N_3$ fica subentendida nessa notação. Embora o pl tenha sido formulado como um problema de minimização, nossa definição inclui, implicitamente, problemas de maximização, uma vez que minimizar cx é o mesmo que maximizar $-cx$. $\mathrm{P}(A,b,c)$

[1] Ao contrário da definição usual, qualquer das partes de nossas partições pode ser vazia.

A tradição impõe uma terminologia que, infelizmente, desrespeita o significado da palavra "solução". Assim, uma **solução viável** (*feasible solution*) do problema é qualquer vetor x que satisfaz as restrições; e uma **solução ótima** é qualquer solução viável x que minimize cx.

$X(A,b)$ O conjunto de todas as soluções viáveis do problema $\mathrm{P}(A,b,c)$ será denotado por $X(A,b)$. Se $X(A,b)$ é vazio, dizemos que o problema é **inviável** (*infeasible*). Caso contrário, o problema é **viável** (*feasible*). Dizemos que o problema é **limitado** (*bounded*) se existe um número ω tal que $cx \geq \omega$ para todo x em $X(A,b)$; caso contrário, o problema é **ilimitado** (*unbounded*). É claro que problemas inviáveis e ilimitados não têm solução ótima.

Dualidade

Há uma importante relação de dualidade entre pls. O **dual** do pl $\mathrm{P}(A,b,c)$ é o pl $\mathrm{P}(-A^\top,-c,-b)$, onde $A^\top$ é a transposta de A e os conjuntos de índices são N_1,N_2,N_3,M_1,M_2,M_3. É claro que esse pl também pode ser escrito assim: encontrar um vetor y indexado por M que

$$\begin{aligned} \text{maximize} \quad & yb \\ \text{sob as restrições} \quad & (yA)_j \leq c_j \quad \text{para cada } j \text{ em } N_1\,, \\ & (yA)_j = c_j \quad \text{para cada } j \text{ em } N_2\,, \\ & (yA)_j \geq c_j \quad \text{para cada } j \text{ em } N_3\,, \\ & y_i \geq 0 \quad \text{para cada } i \text{ em } M_1\,, \\ & y_i \leq 0 \quad \text{para cada } i \text{ em } M_3\,. \end{aligned} \tag{C.2}$$

$\mathrm{D}(A,c,b)$ $Y(A,c)$ Convém usar uma notação específica para o dual: o pl (C.2) será denotado por $\mathrm{D}(A,c,b)$ e seu conjunto de soluções viáveis por $Y(A,c)$.

Em discussões sobre um par dual de pls, é comum dizer que um deles — digamos (C.1) — é "o primal" e o outro — digamos (C.2) — é "o dual". Adotada essa convenção, cada componente de um vetor x indexado por N é uma "variável primal" e cada componente de um vetor y indexado por M é uma "variável dual".

Problemas canônicos

Se $M_1 = M$ e $N_1 = N$, o problema $\mathrm{P}(A,b,c)$ se reduz a encontrar um vetor x que

$$\begin{aligned} \text{minimize} \quad & cx \\ \text{sob as restrições} \quad & Ax \geq b\,, \\ & x \geq 0\,. \end{aligned} \tag{C.3}$$

Diz-se, às vezes, que esse é o pl **canônico primal**, pois qualquer pl é equivalente a um pl que tem essa forma, e portanto qualquer algoritmo que resolva o pl canônico primal poderia ser usado para resolver um pl arbitrário. É claro que poderíamos enunciar (C.3), sem notação matricial, dizendo

$$\begin{array}{rrcl} \text{minimize} & \sum_j c_j x_j & & \\ \text{sob as restrições} & \sum_j A_{ij} x_j & \geq & b_i \quad \text{para cada } i\,, \\ & x_j & \geq & 0 \quad \text{para cada } j\,. \end{array}$$

O dual do pl (C.3) consiste em encontrar um vetor y que

$$\begin{array}{rrcl} \text{maximize} & yb & & \\ \text{sob as restrições} & yA & \leq & c\,, \\ & y & \geq & 0\,. \end{array} \tag{C.4}$$

Pode-se dizer que esse é o pl **canônico dual**.

Lema da dualidade

Há uma relação fundamental entre as soluções viáveis de um pl e as soluções viáveis de seu dual.

Lema C.1 (da dualidade): *Para todo x em $X(A,b)$ e todo y em $Y(A,c)$ tem-se $cx \geq yb$.*

Demonstração: Se $M_1 = M$ e $N_1 = N$ então as restrições de (C.3) e (C.4) garantem as desigualdades $cx \geq (yA)x = y(Ax) \geq yb$. A primeira vale porque $c \geq yA$ e $x \geq 0$; a última, porque $y \geq 0$ e $Ax \geq b$. A demonstração do caso geral é conceitualmente análoga mas tipograficamente indigesta: em virtude das restrições de (C.1) e (C.2),

$$\begin{aligned} cx &= c_{N_1} x_{N_1} + c_{N_2} x_{N_2} + c_{N_3} x_{N_3} \\ &\geq (yA)_{N_1} x_{N_1} + (yA)_{N_2} x_{N_2} + (yA)_{N_3} x_{N_3} \\ &= (y_{M_1} A_{M_1 N_1} + y_{M_2} A_{M_2 N_1} + y_{M_3} A_{M_3 N_1}) x_{N_1} + \\ &\quad (y_{M_1} A_{M_1 N_2} + y_{M_2} A_{M_2 N_2} + y_{M_3} A_{M_3 N_2}) x_{N_2} + \\ &\quad (y_{M_1} A_{M_1 N_3} + y_{M_2} A_{M_2 N_3} + y_{M_3} A_{M_3 N_3}) x_{N_3} \\ &= y_{M_1} (A_{M_1 N_1} x_{N_1} + A_{M_1 N_2} x_{N_2} + A_{M_1 N_3} x_{N_3}) + \\ &\quad y_{M_2} (A_{M_2 N_1} x_{N_1} + A_{M_2 N_2} x_{N_2} + A_{M_2 N_3} x_{N_3}) + \\ &\quad y_{M_3} (A_{M_3 N_1} x_{N_1} + A_{M_3 N_2} x_{N_2} + A_{M_3 N_3} x_{N_3}) \\ &= y_{M_1} (Ax)_{M_1} + y_{M_2} (Ax)_{M_2} + y_{M_3} (Ax)_{M_3} \end{aligned}$$

$$\begin{aligned} &\geq y_{M_1} b_{M_1} + y_{M_2} b_{M_2} + y_{M_3} b_{M_3} \\ &= yb\,, \end{aligned}$$

onde $A_{M_1 N_1}$, por exemplo, denota a restrição de A a $M_1 \times N_1$. □

O lema é às vezes chamado, um tanto pomposamente, de *teorema fraco* da dualidade. Ele leva à seguinte observação, óbvia mas importante: para tornar evidente que um certo vetor x em $X(A,b)$ é solução ótima do problema $\mathrm{P}(A,b,c)$, basta exibir um vetor y em $Y(A,c)$ tal que

$$cx = yb\,.$$

Com isso, estaremos mostrando também que y é solução ótima do problema $\mathrm{D}(A,c,b)$. Eis outra conseqüência do lema C.1: para mostrar que o problema $\mathrm{P}(A,b,c)$ é limitado, basta exibir um elemento de $Y(A,c)$.

Folgas complementares

A demonstração do lema C.1 sugere o seguinte conceito. Dois vetores x e y, indexados por M e N respectivamente, têm **folgas complementares** (*complementary slackness*) se, para cada j em $N_1 \cup N_3$, temos

$$x_j = 0 \quad \text{ou} \quad (yA)_j = c_j \tag{C.5}$$

e, para cada i em $M_1 \cup M_3$, temos

$$y_i = 0 \quad \text{ou} \quad (Ax)_i = b_i\,. \tag{C.6}$$

Há quem diga que (C.5) dá as "condições de folgas complementares primais" e (C.6) as de "folgas complementares duais".

Lema C.2 (das folgas complementares): *Para todo x em $X(A,b)$ e todo y em $Y(A,c)$, tem-se $cx = yb$ se e somente se as folgas de x e y são complementares.*

Demonstração: A demonstração do lema C.1 deixa claro que $cx = yb$ se e somente se $cx = (yA)x$ e $y(Ax) = yb$. Mas $cx = (yA)x$ equivale a (C.5) e $y(Ax) = yb$ equivale a (C.6). □

Teorema da dualidade

O teorema da dualidade dá as condições em que um pl tem solução ótima e garante que as soluções ótimas do pl e de seu dual têm o mesmo valor:

Teorema C.3 (da dualidade): *Para qualquer sistema (A, b, c), se o problema $\mathrm{P}(A, b, c)$ é viável e limitado então tem solução ótima; mais especificamente, existem x em $X(A, b)$ e y em $Y(A, c)$ tais que $cx = yb$. Se A, b e c forem racionais então podemos supor que x e y também são racionais.*

Esse teorema, também conhecido como *teorema forte* da dualidade, pode ser resumido da seguinte maneira: a menos que os dois pls sejam inviáveis,

$$\min_x cx \;=\; \max_y yb$$

para x variando em $X(A, b)$ e y em $Y(A, c)$. Esta identidade vale mesmo quando um dos pls é inviável, desde que estejamos dispostos a dizer que $\min cx = +\infty$ quando o pl primal é inviável, que $\max yb = +\infty$ quando o pl dual é ilimitado, etc.

Algoritmos de programação linear

O célebre **algoritmo Simplex** resolve qualquer par dual de programas lineares. Ao receber um sistema racional (A, b, c) e os conjuntos $M_1, M_2, M_3, N_1, N_2, N_3$ de índices, o Simplex decide se os problemas $\mathrm{P}(A, b, c)$ e $\mathrm{D}(A, c, b)$ são viáveis e em caso afirmativo produz soluções ótimas racionais, x e y, dos dois programas lineares. Cada componente desses vetores tem numerador e denominador limitados superiormente pelo tamanho do sistema (A, b, c).

Embora o Simplex seja eficiente em média, seu consumo de tempo não é limitado por uma função polinomial do número de componentes do sistema (A, b, c). O consumo de tempo não é limitado nem mesmo por uma função polinomial do tamanho de (A, b, c).

Mas existem algoritmos, muito diferentes do Simplex, capazes de resolver qualquer pl em tempo polinomial.[2] O primeiro algoritmo desse tipo foi publicado por Khachiyan [Kha79, PS82, Sch86, GLS93] e ficou conhecido como **algoritmo dos elipsóides** (*ellipsoid method*). (Outra família de algoritmos polinomiais, conhecidos como **métodos de pontos interiores**, apareceu mais tarde [Kar84, Gon89, Gon92].) Para resolver o problema $\mathrm{P}(A, b, c)$, esse algoritmo consome uma quantidade de tempo limitada por um polinômio no tamanho de (A, b, c).

[2] Esses algoritmos são polinomiais mas não fortemente polinomiais. Veja apêndice E.

Fato C.4: *Existe um algoritmo polinomial que, ao receber um sistema racional (A, b, c), decide se o problema $\mathrm{P}(A, b, c)$ tem solução e, em caso afirmativo, devolve uma solução ótima racional x. O tamanho de x é limitado por $2\langle A\rangle + 2\langle b\rangle$. O consumo de tempo do algoritmo é limitados por um polinômio em $\langle A\rangle + \langle b\rangle + \langle c\rangle$.*

Programação linear e algoritmos de separação

Muitos problemas combinatórios podem ser representados, de maneira aproximada, por problemas da forma $\mathrm{P}(A, b, c)$, com A, b e c racionais. Numa tal classe de programas lineares, o número de linhas de A pode
τ ser uma função exponencial do número de colunas. Digamos que τ é o tamanho da maior linha de (A, b), ou seja, $\tau := \max_i\{\langle A_{i\star}\rangle + \langle b_i\rangle\}$. Mesmo que o número de linhas de A não seja limitado por um polinômio em τ, é possível resolver o problema $\mathrm{P}(A, b, c)$ em tempo limitado por um polinômio em τ e $\langle c\rangle$ se dispusermos de um bom algoritmo para o seguinte

Problema da separação: *Decidir se um dado vetor x está em $X(A, b)$ e, em caso negativo, determinar uma restrição violada, ou seja, uma linha i tal que $(Ax)_i < b_i$.*

Suponha que temos um bom algoritmo para esse problema, isto é, um algoritmo cujo consumo de tempo é limitado por um polinômio em τ e $\langle x\rangle$.[3] Grötschel, Lovász e Schrijver mostraram [GLS93, Sch86] que um tal **algoritmo de separação** pode ser combinado com o algoritmo dos elipsóides de modo a resolver o problema $\mathrm{P}(A, b, c)$ em tempo limitado por uma função polinomial de τ e $\langle c\rangle$ apenas.

Suponha agora que o problema de otimização combinatória que deu origem a $\mathrm{P}(A, b, c)$ tem tamanho σ. Se τ e $\langle c\rangle$ são limitados por um polinômio em σ e temos um bom algoritmo de separação (veja o exercício C.3) então o problema $\mathrm{P}(A, b, c)$ pode ser resolvido em tempo limitado por um polinômio em σ.

[3] O consumo de tempo não depende, portanto, do número de linhas do sistema. Veja o exercício C.3.

Problemas de viabilidade

O **problema da viabilidade** consiste em encontrar um vetor viável do problema $\mathrm{P}(A, b, c)$, ou seja, encontrar um elemento de $X(A, b)$. É claro que o problema pode não ter solução. Para certificar a inexistência de solução, basta exibir um **vetor de inviabilidade**, isto é, um vetor y' em $Y(A, 0)$ tal que $y'b > 0$ (veja exercício C.4).

> **Lema C.5** (de Farkas): *Para qualquer sistema (A, b, c), o conjunto $X(A, b)$ é vazio se e somente se existe um vetor y' em $Y(A, 0)$ tal que $y'b > 0$.*

O problema da viabilidade equivale ao caso em que $c = 0$ no problema $\mathrm{P}(A, b, c)$; e esse caso é, às vezes, computacionalmente mais simples que o caso geral. O método primal-dual, descrito no capítulo 5, tira proveito dessa observação para propor um procedimento de resolução de programas lineares.

Exercícios

C.1 Transforme um programa linear arbitrário num programa canônico equivalente.

C.2 Muitos problemas combinatórios envolvem vetores não-negativos b e c e uma matriz A sem componentes negativos e sem linhas nulas. Verifique que, nesse caso, os problemas (C.3) e (C.4) são viáveis. O teorema da dualidade garante então que ambos os problemas têm solução ótima.

C.3 Sejam s e t dois vértices de um grafo G e seja $\mathcal{S}$ a coleção de todos os subconjuntos de V_G que contêm s mas não contêm t. (O número de tais subconjuntos é, tipicamente, uma função exponencial do número $|V_G|$). Seja A a matriz de incidência de $\mathcal{S}$, isto é, a matriz indexada por $\mathcal{S} \times E_G$ cujos componentes são definidos da seguinte maneira: para cada S em $\mathcal{S}$ e cada e em E_G, o componente (S, e) de A vale 1 se $e \in \delta(S)$ e vale 0 em caso contrário. Agora considere o problema de decidir se um dado vetor não-negativo x indexado por E_G satisfaz as restrições

$$Ax \geq \mathit{1}\,, \tag{C.7}$$

onde $\mathit{1}$ denota o vetor cujos componentes são todos iguais a 1. Esboce um algoritmo para o problema. O algoritmo deve receber o

grafo G, vértices s e t e um vetor racional não-negativo x e deve devolver 1 se (C.7) estiver satisfeita e 0 em caso contrário. O consumo de tempo de seu algoritmo deve ser limitado por um polinômio em $\langle G \rangle + \langle x \rangle$. *Sugestão:* Limite-se, numa primeira versão, ao caso em que os componentes de x estão todos em $\{0, 1\}$.

C.4 Seja y' um elemento de $Y(A, 0)$ tal que $y'b > 0$. Mostre que $X(A, b)$ é vazio. Mostre também que se $Y(A, c)$ não é vazio então o problema $\mathrm{D}(A, c, b)$ é ilimitado. Prove afirmações análogas depois de trocar os papéis de $X(A, b)$ e $Y(A, c)$.

Bibliografia

[ABCC98] D. Applegate, R. Bixby, V. Chvátal e W. Cook. On the solution of traveling salesman problems. *Documenta Mathematica*, Vol. Extra III:645–656, 1998. Proceedings of the International Congress of Mathematicians. URL: `http://www.mathematik.uni-bielefeld.de/documenta/xvol-icm/17/17.html`. [p. 21]

[ABI86] N. Alon, L. Babai e A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. [p. 74]

[ACG+99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela e M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999. [p. iii, 2, 93, 97, 99]

[ACP95] G. Ausiello, P. Crescenzi e M. Protasi. Approximate solution of NP optimization problems. *Theoretical Computer Science*, 150(1):1–55, 1995. [p. 98, 99]

[ADP80] G. Ausiello, A. D'Atri e M. Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21:136–153, 1980. [p. 96]

[AG94] M. Aggarwal e N. Garg. A scaling technique for better network design. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 233–240, 1994. [p. 60]

[AGK+98] S. Arora, M. Grigni, D.R. Karger, P.N. Klein e A. Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. In *Proceedings of the Nineth ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 33–41, 1998. [p. 22]

[AHU74] A.V. Aho, J.E. Hopcroft e J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974. [p. 15, 125]

[AK98] N. Alon e N. Kahale. Approximating the independence number via the θ-function. *Mathematical Programming*, 80:231–237, 1998. [p. 85]

[AKR95] A. Agrawal, P. Klein e R. Ravi. When trees collide: an approximation algorithm for the generalizad Steiner problem on networks. *SIAM Journal on Computing*, 24(2):440–456, 1995. [p. 60]

[Ali95] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51, 1995. [p. 85]

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan e M. Szegedy. Proof verification and intractability of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS)*, p. 24–27, 1992. [p. 2, 99]

[AMO93] R.K. Ahuja, T.L. Magnanti e J.B. Orlin. *Network Flows*. Prentice-Hall, 1993. [p. 26, 27, 33]

[AR98] Y. Aumann e Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998. [p. 33]

[Aro94] S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. PhD Thesis, Princeton University, 1994. [p. 2]

[Aro95] S. Arora. Reductions, codes, PCPs, and inapproximability. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, p. 404–413, 1995. [p. 2]

[Aro98] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998. [p. 22]

[AS92] N. Alon e J. Spencer. *The Probabilistic Method*. John Wiley, 1992. [p. 70]

[AW00] T. Asano e D.P. Williamson. Improved approximation algorithms for MAX SAT. In *Proceedings of the Eleventh ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 96–115, 2000. [p. 74]

[BKRV98] A. Blum, G. Konjevod, R. Ravi e S. Vempala. Semidefinite relaxations for minimum bandwidth and other vertex-ordering problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, p. 100–105, 1998. [p. 85]

[Blu91] A. Blum. *Algorithms for Approximate Graph Coloring*. PhD Thesis, Massachusetts Institute of Technology, 1991. [p. 2]

[BM76] J.A. Bondy e U.S.R. Murty. *Graph Theory with Applications*. Macmillan/Elsevier, 1976. [p. 15, 101, 104]

[Bru98] P. Brucker. *Scheduling Algorithms*. Springer, 2. ed., 1998. [p. 21]

[BTV99] D. Bertsimas, C. Teo e R. Vohra. On dependent randomized rounding algorithms. *Operations Research Letters*, 24(3):105–114, 1999. [p. 74]

[BYE81] R. Bar-Yehuda e S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981. [p. 47, 60]

[CFR98] G. Călinescu, C.G. Fernandes e B. Reed. Multicuts in unweighted graphs with bounded degree and bounded treewidth. In R.E. Bixby, E.A. Boyd e R.Z. Ríos-Mercado, editores, *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 1412 of *Lecture Notes in Computer Science*, p. 137–152. Springer, 1998. [p. 33]

[CG89] B. Chor e O. Goldreich. On the power of two-point sampling. *Journal of Complexity*, 5:96–106, 1989. [p. 75]

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Carnegie Mellon University, 1976. [p. 17, 21]

[Chv79] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. [p. 8, 39]

[Chv83] V. Chvátal. *Linear Programming*. Freeman, 1983. [p. 113]

[CJLL95] P. Chrétienne, E.G. Coffman Jr., J.K. Lenstra e Z. Liu, editores. *Scheduling Theory and its Applications*. John Wiley & Sons, 1995. [p. 21]

[CK00] P. Crescenzi e V. Kann. *A Compendium of NP Optimization Problems*, 2000. URL: `http://www.nada.kth.se/~viggo/wwwcompendium/`. [p. 99]

[CKR00] G. Călinescu, H. Karloff e Y. Rabani. An improved approximation algorithm for MULTIWAY CUT. *Journal of Computer and System Sciences*, 60(3):564–574, 2000. [p. 33, 74]

[CLR92] T.H. Cormen, C.E. Leiserson e R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1992. [p. 10, 15, 20, 59, 125, 128, 130]

[Cob65] A. Cobham. The intrinsic computational difficult of functions. In Y. Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science*, p. 24–30. North-Holland, 1965. [p. 128]

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (STOC)*, p. 151–158, 1971. [p. 87, 130]

[CR73] S.A. Cook e R.A. Reckhow. Time-bounded random access machines. *Journal of Computer and System Science*, 7:354–375, 1973. [p. 127]

[CRW01] F. Chudak, T. Roughgarden e D.P. Williamson. Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation. In *Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization Conference (IPCO)*, 2001. Aceito para publicação. [p. 61, 85]

[CS98] B. Chor e M. Sudan. A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics*, 11(4):511–523, 1998. [p. 85]

[Dev86] L. Devroye. *Non-uniform Random Variate Generation*. Springer, 1986. [p. 75, 84]

[Die00] R. Diestel. *Graph Theory*. Springer, 2. ed., 2000. [p. 101]

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, p. 269–271, 1959. [p. 59]

[DJP+94] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour e M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994. [p. 26, 33, 34, 99]

[DL97] M.M. Deza e M. Laurent. *Geometry of Cuts and Metrics*. Springer, 1997. [p. 85]

[Edm65a] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965. [p. 22, 60]

[Edm65b] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. [p. 22, 128]

[EK00] L. Engebretsen e M. Karpinski. Approximation hardness of TSP with bounded metrics. *Eletronic Colloquium on Computational Complexity*, 7(89), 2000. URL: `http://www.eccc.uni-trier.de/eccc/`. [p. 96]

[ENRS95] G. Even, J. Naor, S. Rao e B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, p. 62–71, 1995. [p. 33]

[Erd67] P. Erdős. Gráfok páros körüljárású részgráfjairól (On bipartite subgraphs of graphs, em húngaro). *Matematikai Lapok*, 18:283–288, 1967. [p. 2, 85]

[ES73] P. Erdős e J. Selfridge. On a combinatorial game. *Journal of Combinatorial Theory, Ser. A*, 14:298–301, 1973. [p. 74]

[ES74] P. Erdős e J. Spencer. *The Probabilistic Method in Combinatorics*. Academic Press, 1974. [p. 70]

[Fei99] U. Feige. Randomized rounding for semidefinite programs—variations on the MAX CUT example. In *Randomization, Approximation, and Combinatorial Optimization*, volume 1671 of *Lecture Notes in Computer Science*, p. 189–196. Springer, 1999. [p. 74]

[Feo01] P. Feofiloff. *Algoritmos de Programação Linear*. A ser publicado pela EDUSP, 2001. [p. 113]

[FF56] L.R. Ford e D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. [p. 33, 60]

[FG95] U. Feige e M.X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the Third Israel Symposium on Theory of Computing and Systems (ISTCS)*, p. 182–189, 1995. [p. 74, 85]

[FJ97] A. Frieze e M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, p. 67–81, 1997. [p. 85]

[GGL95] R.L. Graham, M. Grötschel e L. Lovász, editores. *Handbook of combinatorics. Vol. 1, 2*. Elsevier Science B.V., Amsterdam, 1995. [p. 85]

[GGP+94] M.X. Goemans, A.V. Goldberg, S. Plotkin, D.B. Shmoys, É. Tardos e D.P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the Fifth ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 223–232, 1994. [p. 60]

[GGU72] M.R. Garey, R.L. Graham e J.D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the 4th Annual ACM Symposium on the Theory of Computing (STOC)*, p. 143–150, 1972. [p. 2]

[GGW98] H.N. Gabow, M.X. Goemans e D.P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming*, 82(1-2, Ser. B):13–40, 1998. [p. 60, 61]

[GJ75] M.R. Garey e D.S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4:397–411, 1975. [p. 98]

[GJ78] M.R. Garey e D.S. Johnson. Strong NP-completeness results: Motivations, examples and implications. *Journal of the ACM*, 25:499–508, 1978. [p. 91]

[GJ79] M.R. Garey e D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979. [p. 6, 8, 12, 14, 19, 20, 35, 49, 65, 77, 96, 98, 125, 129, 131]

[GLS93] M. Grötschel, L. Lovász e A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 2. ed., 1993. [p. 82, 84, 117, 118]

[Goe97] M.X. Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79:143–161, 1997. [p. 84]

[Gon89] C.C. Gonzaga. An algorithm for solving linear programming problems in $O(n^3L)$ operations. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior Point and Related Methods*, p. 1–28. Springer, 1989. [p. 117]

[Gon92] C.C. Gonzaga. Path following methods for linear programming. *SIAM Review*, 34(2):167–227, 1992. [p. 117]

[Gra66] R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966. [p. 2, 6]

[Gra69] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969. [p. 21]

[Gui98] K.S. Guimarães. Algoritmos de aproximação para problemas de otimização. In H.P. de Moura, editor, *XVII Jornada de Atualização em Informática*, volume II, p. 1–47. SBC, 1998. [p. v, 20]

[GVY96] N. Garg, V.V. Vazirani e M. Yannakakis. Approximate max-flow and min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1996. [p. 26, 33]

[GVY97] N. Garg, V.V. Vazirani e M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997. [p. 33, 59, 61]

[GW94] M.X. Goemans e D.P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994. [p. 67, 69, 74]

[GW95a] M.X. Goemans e D.P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995. [p. 50, 52, 59, 60]

[GW95b] M.X. Goemans e D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995. [p. 79, 85]

[GW98] M.X. Goemans e D.P. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Combinatorica*, 18(1):37–59, 1998. [p. 61]

[GW01] M.X. Goemans e D.P. Williamson. Approximation algorithms for MAX 3-CUT and other problems via complex semidefinite programming. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing (STOC)*, 2001. Aceito para publicação. [p. 85]

[Hal00] E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. In *Proceedings of the Eleventh ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 329–337, 2000. [p. 85]

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, p. 1–10, 1997. [p. 96]

[HH86] N.G. Hall e D.S. Hochbaum. A fast approximation algorithm for the multicovering problem. *Discrete Applied Mathematics*, 15(1):35–40, 1986. [p. 38]

[Hoc82] D.S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982. [p. 24, 36, 38, 60]

[Hoc97] D.S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing, 1997. [p. iii, 2, 20, 33, 38, 59, 60, 74]

[Hoo91] J.A. Hoogeveen. Analysis of Christofides' heuristic: some paths are more difficult than cycles. *Operations Research Letters*, 10:291–295, 1991. [p. 22]

[HPS94] S. Hougardy, H.-J. Prömel e A. Steger. Probabilistically checkable proofs and their consequences for approximation algorithms. *Discrete Mathematics*, 136:175–223, 1994. [p. 99]

[HS87] D.S. Hochbaum e D.B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987. [p. 39]

[HS88] D.S. Hochbaum e D.B. Shmoys. A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approach. *SIAM Journal on Computing*, 17:539–551, 1988. [p. 21, 39, 89, 96]

[Hu63] T.C. Hu. Multicommodity network flows. *Operations Research*, 9:898–900, 1963. [p. 26, 33]

[IK75] O.H. Ibarra e C.E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22:463–468, 1975. [p. 12, 96]

[Ita78] A. Itai. Two-commodity flow. *Journal of the ACM*, 25(4):596–611, 1978. [p. 26, 33]

[JMVW99] K. Jain, I. Măndoiu, V.V. Vazirani e D.P. Williamson. A primal-dual schema based approximation algorithm for the element connectivity problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 484–489, 1999. [p. 61]

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974. [p. 2, 21, 65, 74]

[Joh87] M.E. Johnson. *Multivariate Statistical Simulation.* John Wiley, 1987. [p. 75, 84]

[JRR95] M. Jünger, G. Reinelt e G. Rinaldi. The traveling salesman problem. In M.O. Ball, T.L. Magnanti, C.L. Monma e G.L. Nemhauser, editores, *Network Models*, p. 225–330. North-Holland, 1995. [p. 21]

[JV00] K. Jain e V.V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. *17th International Symposium on Mathematical Programming (ISMP)*, 2000. URL: `http://www.cc.gatech.edu/people/home/kjain/`. [p. 61, 74]

[Kan92] V. Kann. *On the Approximability of NP-complete Optimization Problems*. PhD Thesis, Royal Institute of Technology, Sweden, 1992. [p. 2]

[Kar72] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller e J.M. Thatcher, editores, *Complexity of Computer Computations*, p. 85–103. Plenum, 1972. [p. 87, 90, 91, 130]

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984. [p. 117]

[Kar96] H. Karloff. How good is the Goemans-Williamson MAX CUT algorithm? In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC)*, p. 427–434, 1996. [p. 85]

[KG98] J. Kleinberg e M.X. Goemans. The Lovász theta function and a semidefinite programming relaxation of vertex cover. *SIAM Journal on Discrete Mathematics*, 11(2):196–204, 1998. [p. 85]

[Kha79] L.G. Khachiyan. A polynomial algorithm for linear programming. *Soviet Mathematical Doklady*, 20:191–194, 1979. [p. 117, 130]

[KKS+99] D.R. Karger, P. Klein, C. Stein, M. Thorup e N.E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, p. 668–678, 1999. [p. 34]

[KMS98] D. Karger, R. Motwani e M. Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45(2):246–265, 1998. [p. 85]

[KMSV99] S. Khanna, R. Motwani, M. Sudan e U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1999. [p. 95, 99]

[Knu68] D.E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, 1968. [p. 125]

[Knu98] D.E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 3. ed., 1998. [p. 63, 72, 75, 84]

[KP99] H. Kellerer e U. Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *Journal of Combinatorial Optimization*, 3(1):59–71, 1999. [p. 21]

[KRAR95] P. Klein, S. Rao, A. Agrawal e R. Ravi. An approximate max-flow min-cut relation for undirected multicommodity flow, with applications. *Combinatorica*, 15(2):187–202, 1995. [p. 33]

[KS95] Y. Kohayakawa e J. Soares. *Demonstrações Transparentes e a Impossibilidade de Aproximações*. Livro do XX Colóquio Brasileiro de Matemática. IMPA, 1995. [p. 99, 129]

[Kuh55] H.W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955. [p. 60]

[KZ97] H. Karloff e U. Zwick. A 7/8-approximation for MAX 3-SAT? In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS)*, p. 406–415, 1997. URL: `http://www.math.tau.ac.il/~zwick/`. [p. 74, 85]

[Law76] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, 1976. [p. 38]

[Lev73] L.A. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973. [p. 130]

[LK73] S. Lin e B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973. [p. 21]

[LLR95] N. Linial, E. London e Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. [p. 33]

[LLRS90] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan e D.B. Shmoys, editores. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley, 1990. Reprint of the 1985 original. [p. 21]

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975. [p. 21, 39]

[Lov01] L. Lovász. Semidefinite programs and combinatorial optimization. In *Brazilian Summer School on Combinatorics and Algorithms*, p. 1–58. CIMPA, 2001. A ser publicado pela Springer. [p. 78, 85]

[LP86] L. Lovász e M.D. Plummer. *Matching Theory*. Elsevier, 1986. [p. 17, 22, 129]

[LR99] T. Leighton e S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999. [p. 33]

[Lub86] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. [p. 74]

[LW95] M. Luby e A. Wigderson. Pairwise independence and derandomization. Technical Report TR-035, International Computer Science Institute, 1995. [p. 75]

[Mit99] J.S.B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: a simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. [p. 22]

[MPS98] E. Mayr, H.J. Prömel e A. Steger, editores. *Lectures on Proof Verification and Approximation Algorithms*, volume 1367 of *Lecture Notes in Computer Science*. Springer, 1998. [p. iii, 2, 99]

[MR95a] S. Mahajan e H. Ramesh. Derandomizing semidefinite programming based approximation algorithms. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, p. 162–169, 1995. [p. 85]

[MR95b] R. Motwani e P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. [p. 74]

[MT90] S. Martello e P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley, 1990. [p. 21]

[OM87] P. Orponen e H. Mannila. On approximation preserving reductions: Complete problems and robust measures. Technical Report C-28, University of Helsinki, 1987. [p. 96]

[Pad99] M. Padberg. *Linear Optimization and Extensions*. Springer, 2. rev. exp. ed., 1999. [p. 113]

[Pap94] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. [p. 98, 125]

[PS82] C.H. Papadimitriou e K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982. [p. 59, 117, 125, 131]

[PS85] F.P. Preparata e M.I. Shamos. *Computational Geometry: An Introduction*. Springer, 1985. [p. 127]

[PV00] C.H. Papadimitriou e S. Vempala. On the approximability of the traveling salesman problem. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, p. 126–133, 2000. [p. 99]

[PY91] C.H. Papadimitriou e M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. [p. 95, 99]

[Rag88] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988. [p. 74]

[Rav94] R. Ravi. A primal-dual approximation algorithm for the Steiner forest problem. *Information Processing Letters*, 50(4):185–189, 1994. [p. 60]

[Rei00] G. Reinelt. TSPLIB – *A Library of Sample Instances for the* TSP, 2000. URL: `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`. [p. 21]

[RS97] R. Raz e S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, p. 475–484, 1997. [p. 10, 96]

[RSL77] D.J. Rosenkrantz, R.E. Stearns e P.M. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6:563–581, 1977. [p. 16]

[RT87] P. Raghavan e C.D. Thompson. Randomized rounding. *Combinatorica*, 7:365–374, 1987. [p. 74]

[RV99] S. Rajagopalan e V.V. Vazirani. Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, 28(2):526–541, 1999. [p. 33, 61]

[RW95] R. Ravi e D.P. Williamson. An approximation algorithm for minimum-cost vertex-connectivity problems. In *Proceedings of the Sixth ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 332–341, 1995. [p. 60]

[RZ00] G. Robins e A. Zelikovsky. Improved steiner tree approximation in graphs. In *Proceedings of the Tenth ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 770–779, 2000. [p. 60]

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley, 1986. [p. 59, 82, 113, 117, 118, 130]

[SG76] S. Sahni e T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976. [p. 90, 96, 98]

[Shm98] D.B. Shmoys. Using linear programming in the design and analysis of approximation algorithms: two illustrative problems. In K. Jansen e J. Rolin, editores, *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization*, number 1444 in Lecture Notes in Computer Science, p. 15–32. Springer, 1998. [p. 33]

[Sku98] M. Skutella. Semidefinite relaxations for parallel machine scheduling. In *Proceedings of the 39th Symposium on Foundations of Computer Science (FOCS)*, p. 472–481, 1998. [p. 85]

[Sku99] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling, 1999. [p. 85]

[Spe87] J. Spencer. *Ten Lectures on the Probabilistic Method*. SIAM, 1987. [p. 70, 74]

[Sri99] A. Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM Journal on Computing*, 29(2):648–670, 1999. [p. 33]

[Ste01] A. Steger. Approximability of NP-optimization problems. In *Brazilian Summer School on Combinatorics and Algorithms*, p. 59–115. CIMPA, 2001. A ser publicado pela Springer. [p. 98]

[Tre96] L. Trevisan. *Reductions and (Non-)Approximability*. PhD Thesis, Università di Roma "La Sapienza", 1996. [p. 2]

[Vaz01] V.V. Vazirani. *Approximation Algorithms*. A ser publicado pela Springer, 2001. [p. iii, 2, 21, 33, 74, 85]

[WG94] D.P. Williamson e M.X. Goemans. Computational experience with an approximation algorithm on large-scale Euclidean matching instances. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 355–364, 1994. [p. 60]

[WGMV95] D.P. Williamson, M.X. Goemans, M. Mihail e V.V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15:435–454, 1995. [p. 60]

[Wil93] D.P. Williamson. *On the Design of Approximation Algorithms for a Class of Graph Problems*. PhD Thesis, Massachusetts Institute of Technology, 1993. [p. 2]

[Wil98] D.P. Williamson. Lecture notes on approximation algorithms. Technical Report RC 21409, T.J. Watson Research Center (IBM Research Division), 1998. [p. 20, 85]

[WSV00] H. Wolkowicz, R. Saigal e L. Vandenberghe, editores. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*. Kluwer, 2000. [p. 86]

[Yan94] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17(3):475–502, 1994. [p. 74]

[Zwi99] U. Zwick. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, p. 679–687, 1999. [p. 74, 85]