Algoritmos de aproximação - Método primal

Marina Andretta

ICMC-USP

19 de outubro de 2015

Baseado no livro Uma introdução sucinta a Algoritmos de Aproximação, de M. H. Carvalho, M. R. Cerioli, R. Dahab, P. Feofiloff, C. G. Fernandes, C. E. Ferreira, K. S. Guimarães, F. K. Miyazawa, J. C. Piña Jr., J. A. R. Soares e Y. Wakabayashi.

Introdução

Vamos ver agora como usar programação linear para criar algoritmos de aproximação.

Programação linear é uma ferramenta útil, pois existem algoritmos polinomiais para resolver programas lineares.

Além disso, ela pode fornecer boas delimitações para o valor ótimo do problema em questão, o que é fundamental para a análise da qualidade da solução produzida por um algoritmo de aproximação.

Introdução

Para criar um algoritmo de aproximação baseado em programação linear, necessitamos de um programa linear que seja uma relaxação do problema de otimização: cada solução viável do problema combinatório deve corresponder a uma solução viável do programa linear.

Técnica do arredondamento

Uma técnica simples, mas às vezes eficaz, é a do arredondamento.

Esta técnica consiste em arredondar uma solução ótima de um programa linear que represente o problema original.

Vamos aplicar a técnica do arredondamento ao problema da cobertura mínima por conjuntos.

Problema $\textsc{MinCC}(E, \mathcal{S}, c)$: Dados um conjunto finito E, uma coleção finita $\mathcal{S}$ de conjuntos finitos que cobre E e um custo c_S em $\mathbb{Q}_{\geq}$ para cada S em $\mathcal{S}$, encontrar uma cobertura $\mathcal{T}$ de E que minimize $c(\mathcal{T})$.

Problema MinCC

Para aplicar a técnica do arredondamento, precisamos formular um programa linear que represente o MinCC.

Considere o seguinte programa linear: encontrar um vetor x indexado por $\mathcal{S}$ que

$$\begin{array}{lll} \text{minimize} & cx = \sum_{S \in \mathcal{S}} c_S x_S & \\ \text{sob as restrições} & \sum_{S : e \in \mathcal{S}} x_S \geq 1, & \text{para cada } e \text{ em } E, \\ & x_S \geq 0, & \text{para cada } S \text{ em } \mathcal{S}. \end{array} \tag{1}$$

Problema MinCC

Neste programa, tem-se uma variável x_S para cada S em $\mathcal{S}$.

Estas variáveis devem assumir valor 0 caso um conjunto S **não** esteja na cobertura, e um valor positivo caso S esteja na cobertura.

O primeiro conjunto de restrições exige que, para cada elemento e em E, exista algum conjunto S escolhido para a cobertura que cubra e.

Exemplo

Considere a instância do MinCC com $E = \{a, b, c, d, e, f\}$, $\mathcal{S} = \{\{a, b\}, \{a, c, d\}, \{d, e\}, \{b, c, f\}, \{a, e, f\}\}$, $c_S = 2$ para todo $S \in \{\{a, b\}, \{a, c, d\}\}$ e $c_S = 1$ para todo $S \in \{\{d, e\}, \{b, c, f\}, \{a, e, f\}\}$.

Uma solução ótima para este problema é $\mathcal{T} = \{\{d, e\}, \{b, c, f\}, \{a, e, f\}\}$, com custo $c(\mathcal{T}) = 3$.

Neste caso, o modelo linear associado é dado por

Exemplo

$$
\begin{aligned}
\text{minimizar} \quad & 2x_{\{a,b\}} + 2x_{\{a,c,d\}} + x_{\{d,e\}} + x_{\{b,c,f\}} + x_{\{a,e,f\}} \\
\text{sujeita a} \quad & x_{\{a,b\}} + x_{\{a,c,d\}} + x_{\{a,e,f\}} \geq 1, \\
& x_{\{a,b\}} + x_{\{b,c,f\}} \geq 1, \\
& x_{\{a,c,d\}} + x_{\{b,c,f\}} \geq 1, \\
& x_{\{a,c,d\}} + x_{\{d,e\}} \geq 1, \\
& x_{\{d,e\}} + x_{\{e,f\}} \geq 1, \\
& x_{\{b,c,f\}} + x_{\{e,f\}} \geq 1, \\
& x_{\{a,b\}} \geq 0, \\
& x_{\{a,c,d\}} \geq 0, \\
& x_{\{d,e\}} \geq 0, \\
& x_{\{b,c,f\}} \geq 0, \\
& x_{\{e,f\}} \geq 0.
\end{aligned}
\tag{2}
$$

Problema MinCC

Dada uma cobertura $\mathcal{T}$ de E, chamaremos de vetor característico um vetor x tal que $x_S = 1$ se S está em $\mathcal{T}$ e $x_S = 0$, caso contrário.

O programa linear (1) é viável, já que o vetor característico de qualquer cobertura de E satisfaz todas as suas restrições.

Ele também é limitado, pois $cx \geq 0$ para qualquer solução viável x.

Assim, pelo teorema forte da dualidade, este programa linear tem uma solução ótima racional.

Problema MinCC

Como o vetor característico de qualquer cobertura de E é viável para o programa linear (1), temos a seguinte delimitação inferior para o valor ótimo do problema $\text{MinCC}(E, \mathcal{S}, c)$:

$$opt(E, \mathcal{S}, c) \geq c\hat{x}$$

para toda solução ótima $\hat{x}$ de (1).

Problema MinCC

Se todas as componentes de uma solução ótima $\hat{x}$ de (1) são inteiras, a subcoleção $\{S \in \mathcal{S} : \hat{x}_S > 0\}$ é uma solução ótima do MinCC, ou seja, é uma cobertura de E de custo mínimo.

Em geral, porém, várias componentes de $\hat{x}$ são fracionárias.

Ao arredondarmos os valores dessas componentes de maneira apropriada, podemos obter uma cobertura de E. A seguir, analisamos uma possível maneira de realizar este arredondamento.

Problema MinCC

Para cada e em E, seja f_e a frequência de e em $\mathcal{S}$, isto é, o número de elementos de $\mathcal{S}$ aos quais e pertence.

Seja β a maior das frequências.

Como $\mathcal{S}$ cobre E, temos que $\beta > 0$.

O próximo algoritmo, projetado por Hochbaum, escolhe um conjunto S em $\mathcal{S}$ para fazer parte da cobertura se e somente se o valor da variável $\hat{x}_S$ é pelo menos $\frac{1}{\beta}$.

Algoritmo MinCC-Hochbaum

Algoritmo MinCC-Hochbaum$(E, \mathcal{S}, c)$:

1 seja $\hat{x}$ uma solução ótima racional de (1);

2 para cada e em E, faça $f_e \leftarrow |\{S \in \mathcal{S} : e \in S\}|$;

3 faça $\beta \leftarrow \max_{e \in E} f_e$;

4 faça $\mathcal{T} \leftarrow \{S \in \mathcal{S} : \hat{x}_S \geq 1/\beta\}$;

5 devolva $\mathcal{T}$.

Exemplo

Resolvendo o programa linear (2) do exemplo, temos que a solução ótima $\hat{x}$ é dada por $\hat{x}_{\{a,b\}} = 0$, $\hat{x}_{\{a,c,d\}} = 0.5$, $\hat{x}_{\{d,e\}} = 0.5$, $\hat{x}_{\{b,c,f\}} = 1$ e $\hat{x}_{\{a,e,f\}} = 0.5$.

O elemento de E que aparece em mais subconjuntos de $\mathcal{S}$ é a, que tem frequência 3. Assim, $\beta = 3$.

Com isso, a cobertura que o algoritmo MinCC-Hochbaum calcula é $\mathcal{T} = \{\{a, c, d\}, \{d, e\}, \{b, c, f\}, \{a, e, f\}\}$, com custo $c(\mathcal{T}) = 5$.

Algoritmo MinCC-Hochbaum

Primeiramente, vamos mostrar que a coleção $\mathcal{T}$ é uma cobertura.

Cada e em E pertence a no máximo β conjuntos de $\mathcal{S}$.

Como, para cada $e \in E$, temos uma restrição $\sum_{S:e\in S} x_S \geq 1$, certamente $\hat{x}_S \geq 1/\beta$ para algum S que contém e.

Portanto, $\mathcal{T}$ contém algum S que contém e, ou seja, $\mathcal{T}$ é uma cobertura de E.

Algoritmo MinCC-Hochbaum

Teorema 1: O algoritmo MinCC-Hochbaum é uma β-aproximação polinomial para o MinCC$(E, \mathcal{S}, c)$, com β o número máximo de vezes que um elemento de E aparece em conjuntos de $\mathcal{S}$.

Demonstração: O custo da cobertura $\mathcal{T}$ produzida pelo algoritmo é

$$c(\mathcal{T}) = \sum_{S \in \mathcal{T}} c_S.$$

Como $\beta \hat{x} \geq 1$ para todo S em $\mathcal{T}$,

$$c(\mathcal{T}) = \sum_{S \in \mathcal{T}} c_S \leq \sum_{S \in \mathcal{T}} c_S \beta \hat{x}_S = \beta \sum_{S \in \mathcal{T}} c_S \hat{x}_S \leq \beta c \hat{x}.$$

Como $opt(E, \mathcal{S}, c) \geq c\hat{x}$,

$$c(\mathcal{T}) \leq \beta opt(E, \mathcal{S}, c).$$

Algoritmo MinCC-Hochbaum

O programa linear (1) tem $|\mathcal{S}|$ variáveis e $|E|$ restrições (as restrições $x_S \geq 0$ são implícitas) e, portanto, o tamanho do sistema é $(|S|+1)|E| + \langle c \rangle$.

Como programas lineares podem ser resolvidos em tempo polinomial, a linha 1 do algoritmo MinCC-Hochbaum pode ser executada em tempo polinomial.

As demais linhas claramente também podem ser executadas em tempo polinomial em $|E|$ e $|\mathcal{S}|$. Assim, o algoritmo é polinomial.

Técnica métrica

Como anteriormente, usaremos uma solução ótima $\hat{x}$ de um programa linear associado a um problema combinatório.

Agora, $\hat{x}$ é um vetor indexado pelas arestas de um grafo e cada componente $\hat{x}_e$ será interpretada como o comprimento da aresta e.

Técnica métrica

Vamos descrever a aplicação dessa técnica ao problema do multicorte mínimo.

Dados um grafo G e um conjunto K de pares de vértices, dizemos que um caminho de s a t é um K-caminho se $\{s, t\} \in K$.

Um conjunto M de arestas é um multicorte se não existe K-caminho no grafo $G - M$.

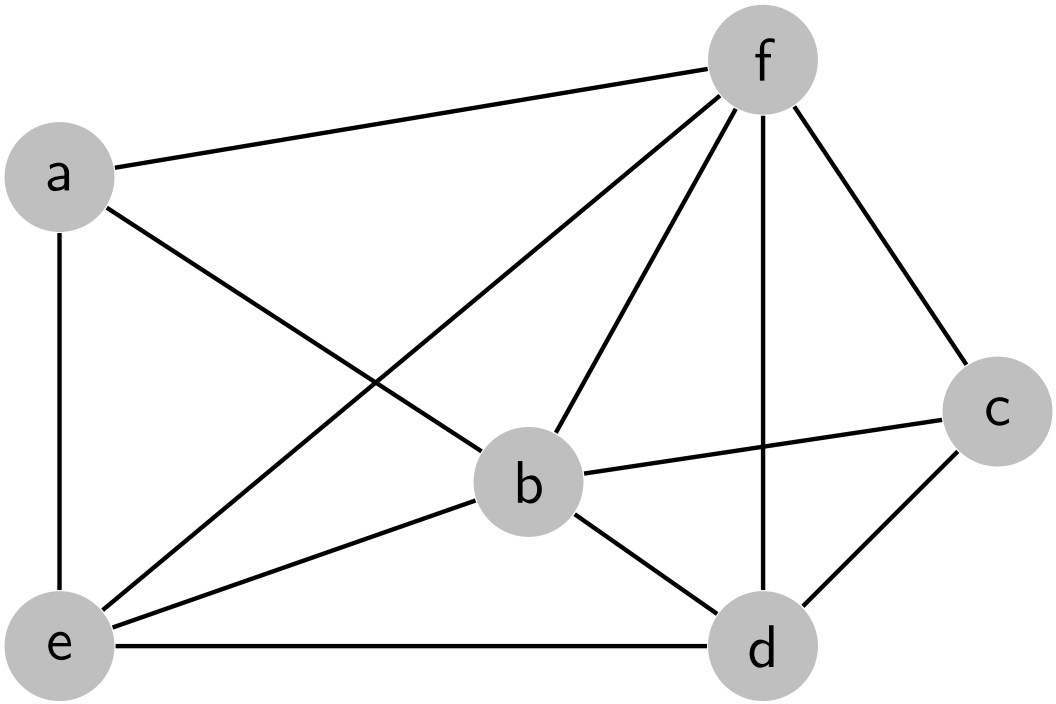
Problema MinMCut

O problema do multicorte mínimo (*minimum multicut problem*) consiste no seguinte:

Problema MinMCut(G, K, c): Dados um grafo G, um conjunto K de pares de vértices e um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e, encontrar um K-multicorte M que minimize $c(M)$.

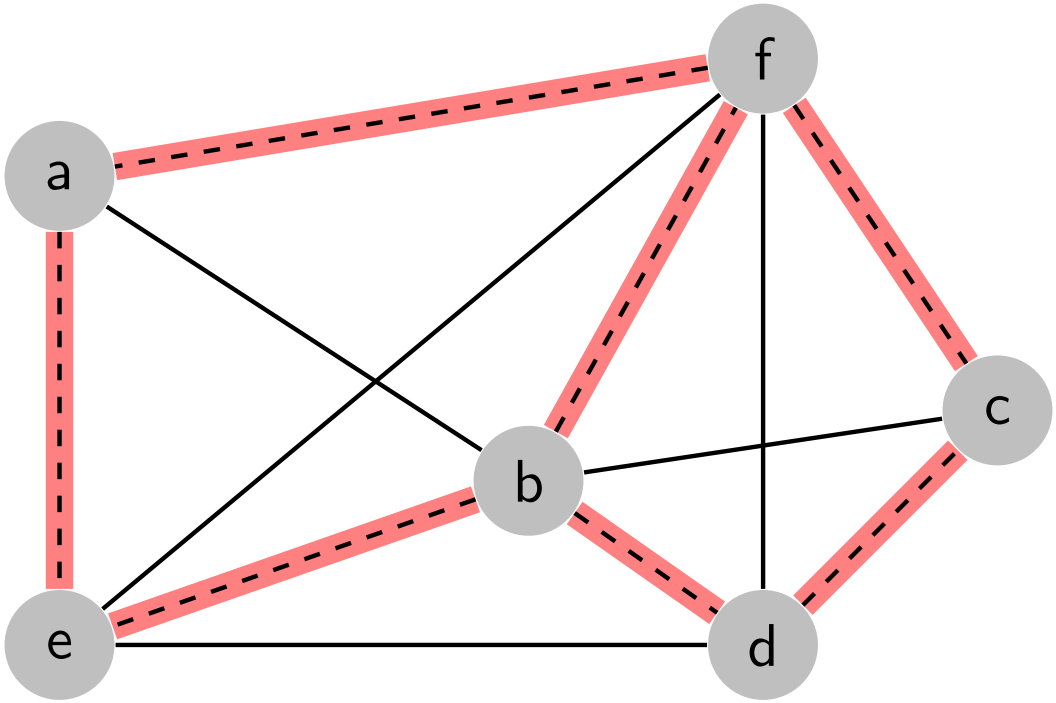
Exemplo

Considere o grafo G dado abaixo, $K = \{\{a, f\}, \{a, e\}, \{c, d\}\}$ e $c_e = 1$ para toda aresta e de G.



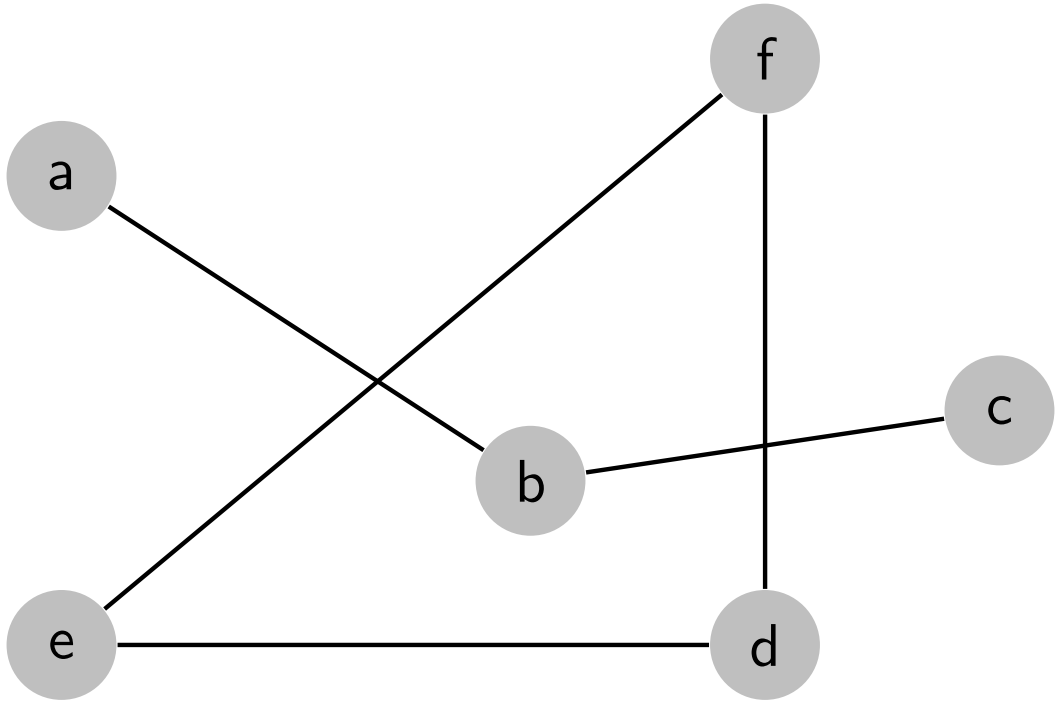
Exemplo

Um K-multicorte é
$M = \{\{a, f\}, \{a, e\}, \{c, d\}, \{b, f\}, \{b, e\}, \{c, f\}, \{b, d\}\}$, com custo $c(M) = 7$.



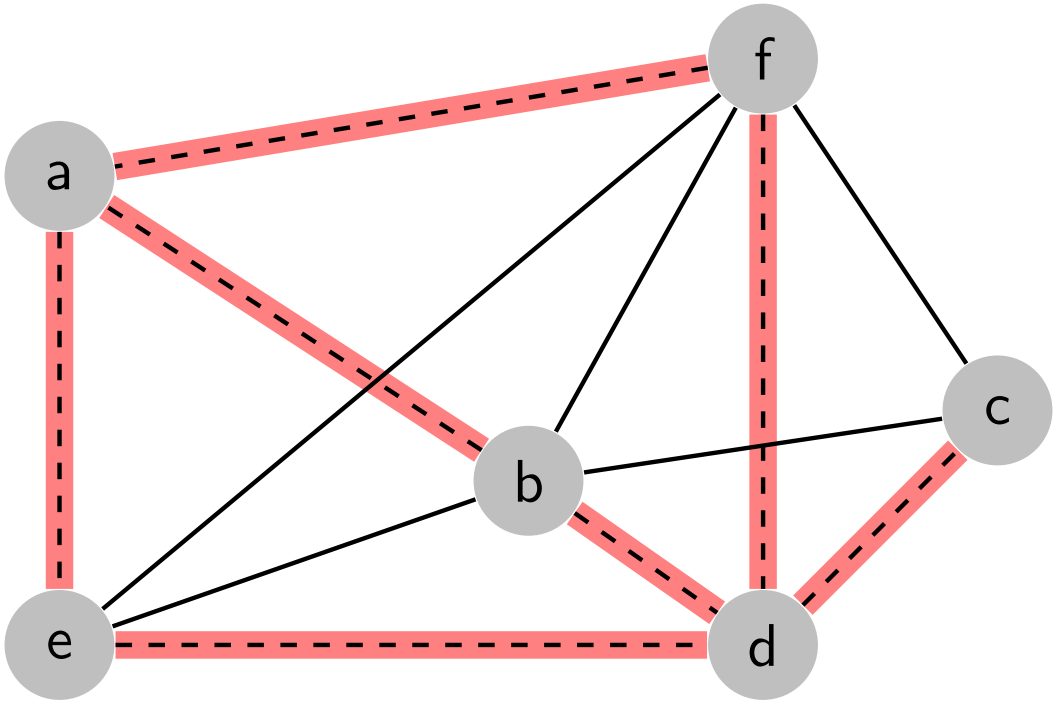
Exemplo

Um K-multicorte é
$M = \{\{a, f\}, \{a, e\}, \{c, d\}, \{b, f\}, \{b, e\}, \{c, f\}, \{b, d\}\}$, com custo $c(M) = 7$.



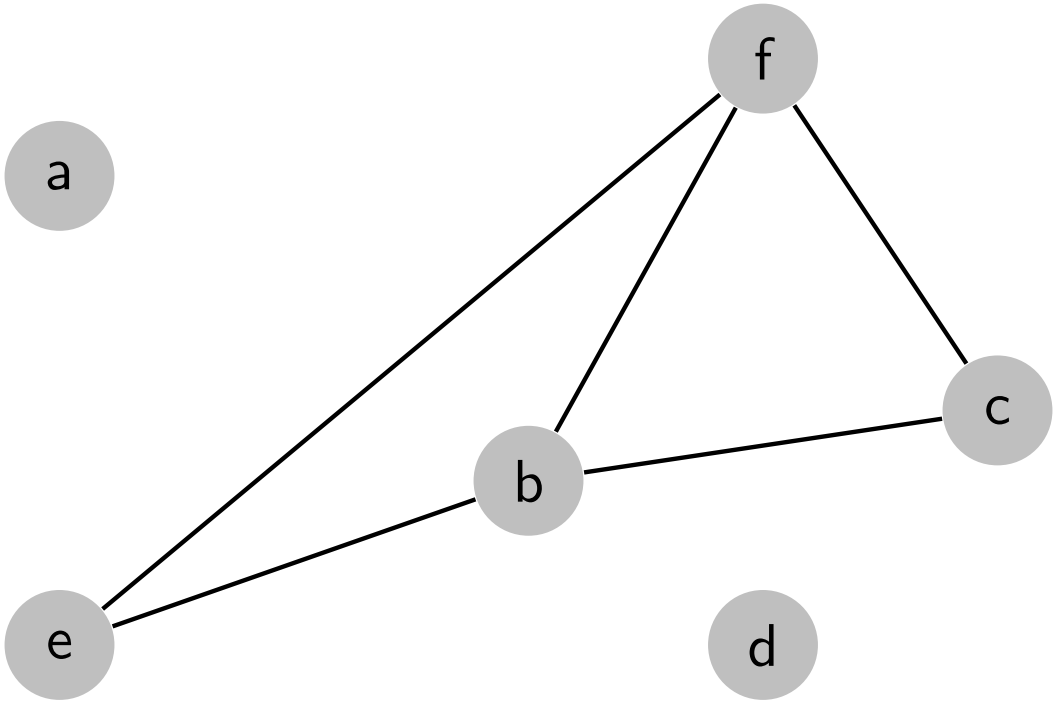
Exemplo

Outro K-multicorte é
$M = \{\{a, f\}, \{a, e\}, \{c, d\}, \{a, b\}, \{b, d\}, \{d, e\}, \{d, f\}\}$, com custo $c(M) = 7$.



Exemplo

Outro K-multicorte é
$M = \{\{a, f\}, \{a, e\}, \{c, d\}, \{a, b\}, \{b, d\}, \{d, e\}, \{d, f\}\}$, com custo $c(M) = 7$.



Problema MinMCut

Há um conhecido algoritmo polinomial, de Ford e Fulkerson, para o caso em que $|K| = 1$, que usa o teorema de fluxo máximo e corte mínimo.

Para o caso em que $|K| = 2$, uma variante deste algoritmo é usada, gerando também um algoritmo polinomial.

Dahlhaus et al. mostraram que o problema é *NP*-difícil mesmo quando restrito às instâncias em que $|K| = 3$.

Problema MinMCut

Denote por $\mathcal{P}$ o conjunto de todos os K-caminhos e considere o seguinte problema de programação linear: encontrar um vetor x indexado por E_G que

$$\begin{array}{lll} \text{minimize} & cx & \\ \text{sob as restrições} & x(E_P) \geq 1, & \text{para cada } P \text{ em } \mathcal{P}, \\ & x_e \geq 0, & \text{para cada } e \text{ em } E_G. \end{array} \tag{3}$$

Problema MinMCut

O vetor característico de E_G é viável em (3) e $cx \geq 0$ para qualquer solução viável de (3).

Portanto, de acordo com o teorema forte da dualidade, o programa linear (3) tem uma solução ótima racional.

Note que o vetor característico de qualquer K-multicorte é viável em (3) e, portanto,

$$opt(G, K, c) \geq c\hat{x}$$

para qualquer solução ótima $\hat{x}$ de (3).

Algoritmo MinMCut-GVY

O algoritmo a seguir foi concebido por Garg, Vazirani e Yannakakis e fornece uma boa razão de aproximação conhecida para o MinMCut.

Ele supõe que $K \neq \emptyset$; em caso contrário, o problema é trivial.

Algoritmo MinMCut-GVY(G, K, c):

1 seja $\hat{x}$ uma solução ótima racional de (3);

2 faça $k \leftarrow |K|$;

3 faça $M \leftarrow$ Central$(G, k, K, c, \hat{x})$;

4 devolva M.

Algoritmo MinMCut-GVY

O algoritmo Central, que detalhamos adiante, produz um K-multicorte M tal que

$$c(M) \leq (4 \ln(2k))c\hat{x}.$$

Teorema 2: O algoritmo MinMCut-GVY é uma $(4 \ln(2k))$-aproximação polinomial para o MinMCut(G, K, c), com $k := |K| > 0$.

Demonstração: Como visto, o custo $c(M)$ do multicorte M que o algoritmo devolve satisfaz

$$c(M) \leq (4 \ln(2k))c\hat{x}$$

.

Além disso, $opt(G, K, c) \geq c\hat{x}$ e, portanto,

$$c(M) \leq (4 \ln(2k))opt(G, K, c).$$

Algoritmo MinMCut-GVY

Note que o número de restrições do programa (3) pode ser exponencial em $|V_G|$.

Chamaremos de algoritmo de separação um algoritmo polinomial que, dado um vetor x e um programa linear, decide se x é viável e, caso não o seja, determina uma restrição violada por x.

Quando tal algoritmo existe, é possível resolver o programa linear em tempo polinomial no tamanho de sua maior restrição e em $\langle x \rangle$.

Algoritmo MinMCut-GVY

A linha 1 do algoritmo MinMCut-GVY pode ser executada em tempo polinomial em $\langle G \rangle + \langle c \rangle$, já que temos um algoritmo de separação para (3):

1. interprete x_e como comprimento da aresta e, calcule um caminho de comprimento mínimo de s a t para cada $\{s, t\}$ em K.
2. Se algum destes caminhos, digamos P, tiver comprimento menor que 1, o vetor x não satisfaz a restrição de (3) correspondente a P.
3. Caso contrário, x satisfaz todas as restrições de (3).

Algoritmo MinMCut-GVY

Como existem algoritmos polinomiais em $\langle G \rangle$ para encontrar um caminho de custo mínimo (algoritmo de Dijkstra, por exemplo), esse algoritmo de separação consome tempo polinomial em $\langle G \rangle$.

Portanto, a linha 1 do algoritmo MinMCut-GVY pode ser executada em tempo polinomial em $\langle G \rangle + \langle c \rangle$.

Algoritmo MinMCut-GVY

A solução $\hat{x}$ é racional e $\langle \hat{x} \rangle$ é limitado por um polinômio em $\langle G \rangle$.

Como mostraremos ao analisar o algoritmo Central, a linha 3 consome tempo polinomial em $\langle G \rangle + \langle c \rangle + \langle \hat{x} \rangle$ e, portanto, polinomial em $\langle G \rangle + \langle c \rangle$.

Assim, podemos concluir que o algoritmo MinMCut-GVY é polinomial.

Algoritmo Central

Vejamos agora como funciona o algoritmo Central.

Ele recebe um grafo G, um inteiro $k \geq 1$, um conjunto K de no máximo k pares de vértices, um vetor racional c e um vetor racional não-negativo x tal que $x(E_P) \geq 1$ para todo K-caminho P.

O algoritmo Central produz um K-multicorte M tal que

$$c(M) \leq (2\ln(2k))(1 + \frac{1}{k}|K|)cx.$$

Algoritmo Central

Para obter um tal multicorte, Central calcula uma partição (S, T) de V_G que separa os dois vértices de algum par em K de modo a manter $c(\delta(S))$ razoavelmente pequeno.

A partição (S, T) induz a partição $(A, \delta(S), B)$ de E_G, com A o conjunto das arestas que têm ambos os extremos em S e B o conjunto das arestas que têm ambos os extremos em T.

O algoritmo é então aplicado, recursivamente, ao grafo (V_G, B).

Algoritmo Central

Para descrever o algoritmo Central, precisamos de alguma notação.

Para quaisquer vértices s e u, seja $x(s,u) := \min_P x(E_P)$, onde o mínimo é tomado sobre todos os caminhos P de s a u.

Se não existe caminho de s a u definimos $x(s,u) := \infty$.

Podemos interpretar $x(s,u)$ como a distância de s a u.

Algoritmo Central

Para qualquer número ρ, denotamos por $V(s, \rho)$ o conjunto de todos os vértices à distância no máximo ρ de s, ou seja,

$$V(s, \rho) := \{v \in V_G : x(s, v) \leq \rho\}.$$

Algoritmo Central

Imagine que as arestas do grafo são tubos, sendo x_e o comprimento e c_e a área da secção transversal do tubo e.

Com essa interpretação em mente, denote por $\mathcal{V}(s, \rho)$ o volume da parte da tubulação que dista no máximo ρ de s. Ou seja,

$$\mathcal{V}(s, \rho) := c_A x_A + \sum_{uv \in \delta(S), u \in S} c_{uv}(\rho - x(s, u)),$$

com $S := V(s, \rho)$ e c_A e x_A as restrições de c e x, respectivamente, ao conjunto de arestas $A := E_{G[S]}$.

Algoritmo Central

É claro que $\mathcal{V}(s, \rho) \leq cx$ qualquer que seja ρ.

Agora estamos prontos para descrever o algoritmo Central.

Algoritmo CENTRAL

Algoritmo CENTRAL(G, k, K, c, x):

1 se $K = \emptyset$

2 então devolva $\emptyset$;

3 senão se $cx = 0$

4 então devolva $\{e \in E_G : x_e > 0\}$;

5 senão sejam s e t os extremos de um K-caminho;

6 seja $v_1, ..., v_n$ uma ordenação dos vértices

7 tal que $x(s, v_1) \leq ... \leq x(s, v_n)$.

8 Para i de 1 a n, faça $p_i \leftarrow x(s, v_i)$;

9 faça $j \leftarrow 1 + \max\{i : p_i = 0\}$;

Algoritmo CENTRAL

10 enquanto $\mathcal{V}(s, p_j) > ((2k)^{2p_j} - 1)\frac{1}{k}cx$;

11 faça $j \leftarrow j + 1$;

12 faça $S \leftarrow V(s, p_{j-1})$;

13 faça $T \leftarrow V_G \setminus S$;

14 faça $B \leftarrow E_{G[T]}$;

15 faça $G_B \leftarrow (V_G, B)$;

16 faça $K_B \leftarrow K \setminus \{\{s', t'\} : S \text{ separa } s' \text{ de } t'\}$;

17 faça $M_B \leftarrow \text{CENTRAL}(G_B, k, K_B, c_B, x_B)$;

18 devolva $\delta(S) \cup M_B$.

Algoritmo Central

Na linha 16, dizemos que S separa s' de t' se S contém somente s' ou somente t'.

Na linha 17, os vetores c_B e x_B são as restrições de c e x a B, respectivamente.

No fim da linha 9, temos $2 \leq j \leq n$, já que $p_1 = x(s, s) = 0$ e $p_n \geq x(s, t) \geq 1$.

Algoritmo Central

Após as linhas 10 e 11, como $p_n \geq 1$ e $k \geq 1$,

$$((2k)^{2p_n} - 1)\frac{1}{k}cx \geq ((2k)^2 - 1)\frac{1}{k}cx > (2k-1)\frac{1}{k}cx \geq cx \geq \mathcal{V}(s, p_n).$$

Na linha 17, podemos aplicar o algoritmo Central, pois $|K_B| \leq |K| \leq k$, x_B é não-negativo e $x_B(E_P) \geq 1$ para todo K_B-caminho P em G_B.

Lema 1: Ao fim da linha 12 do algoritmo CENTRAL, temos que

$$c(\delta(S)) \leq (2\ln(2k))\left(c_A x_A + c_{\delta(S)} x_{\delta(S)} + \frac{1}{k} cx\right),$$

com

- c_A e x_A as restrições de c e x ao conjunto $A := E_{G[S]}$;
- $c_{\delta(S)}$ e $x_{\delta(S)}$ as restrições de c e x a $\delta(S)$.

Demonstração: Primeiramente, vamos verificar que, após as linhas 10 e 11,

$$p_{j-1} < p_j,$$

$$\mathcal{V}(s, p_{j-1}) \geq ((2k)^{2p_{j-1}} - 1)\frac{1}{k}cx$$

e

$$\mathcal{V}(s, p_j) \leq ((2k)^{2p_j} - 1)\frac{1}{k}cx.$$

Algoritmo Central

Como já vimos, $2 \leq j \leq n$ após as linhas 10 e 11.

Suponha que $j = 2$. Então, por causa da linha 9, temos que

$$p_{j-1} = p_1 = 0 < p_j.$$

Além disso, temos que

$$\mathcal{V}(s, p_{j-1}) = \mathcal{V}(s, p_1) = \mathcal{V}(s, 0) = 0 = ((2k)^0 - 1)\frac{1}{k}cx.$$

Temos ainda que $\mathcal{V}(s, p_j) \leq ((2k)^{2p_j} - 1)\frac{1}{k}cx$ em virtude das linhas 10 e 11.

Agora analisemos o caso em que $j > 2$.

Claramente,

$$\mathcal{V}(s, p_{j-1}) \geq ((2k)^{2p_{j-1}} - 1)\frac{1}{k}cx$$

e

$$\mathcal{V}(s, p_j) \leq ((2k)^{2p_j} - 1)\frac{1}{k}cx.$$

devido às linhas 10 e 11.

Disso segue que $p_{j-1} \neq p_j$.

Como $p_{j-1} \leq p_j$, pelas linhas 6 e 7, temos que

$$p_{j-1} < p_j.$$

Agora prosseguimos com a prova do lema.

Como $\mathcal{V}(s, p_{j-1}) \geq ((2k)^{2p_{j-1}} - 1)\frac{1}{k}cx$ e $\mathcal{V}(s, p_j) \leq ((2k)^{2p_j} - 1)\frac{1}{k}cx$, temos que

$$\frac{\mathcal{V}(s, p_j) + \frac{1}{k}cx}{\mathcal{V}(s, p_{j-1}) + \frac{1}{k}cx} \leq (2k)^{2(p_j - p_{j-1})}. \tag{4}$$

Tomando-se o logaritmo natural do lado esquerdo de (4) obtemos

$$\ln(\mathcal{V}(s, p_j) + \frac{1}{k}cx) - \ln(\mathcal{V}(s, p_{j-1}) + \frac{1}{k}cx) = \int_{p_{j-1}}^{p_j} \frac{d}{d\rho} \ln(\mathcal{V}(s, \rho) + \frac{1}{k}cx) d\rho.$$

Como a derivada de $\mathcal{V}(s,\rho) + \frac{1}{k}cx$ em relação a ρ no intervalo (p_{j-1}, p_j) é $c(\delta(S))$, temos

$$\ln(\mathcal{V}(s,p_j) + \frac{1}{k}cx) - \ln(\mathcal{V}(s,p_{j-1}) + \frac{1}{k}cx) = \int_{p_{j-1}}^{p_j} \frac{c(\delta(S))}{\mathcal{V}(s,\rho) + \frac{1}{k}cx} d\rho.$$

Tomando-se o logaritmo natural do lado direito de (4) obtemos

$$2(p_j - p_{j-1})\ln(2k) = \int_{p_{j-1}}^{p_j} (2\ln(2k))d\rho.$$

Como o logaritmo é uma função crescente, concluímos de (4) que

$$\int_{p_{j-1}}^{p_j} \frac{c(\delta(S))}{\mathcal{V}(s,\rho) + \frac{1}{k}cx} d\rho \leq \int_{p_{j-1}}^{p_j} (2\ln(2k))d\rho.$$

Como $p_{j-1} < p_j$, o intervalo (p_{j-1}, p_j), que não é vazio.

Então, para algum ρ no intervalo (p_{j-1}, p_j), temos

$$\frac{c(\delta(S))}{\mathcal{V}(s,\rho) + \frac{1}{k}cx} \leq (2\ln(2k)).$$

Para concluir o lema, resta mostrar que $\mathcal{V}(s,\rho) \leq c_A x_A + c_{\delta(S)} x_{\delta(S)}$.

Para isso, note que, para todo uv em $\delta(S)$ com u em S, temos que $x(s,v) > \rho$ e $x(s,v) \leq x(s,u) + x_{uv}$. Ou seja,

$$\rho - x(s,u) < x_{uv}.$$

Como $S = V(s, p_{j-1}) = V(s,\rho)$, segue que

$$\begin{aligned} \mathcal{V}(s,\rho) &= c_A x_A + \sum_{uv\in\delta(S),u\in S} c_{uv}(\rho - x(s,u)) \\ &\leq c_A x_A + \sum_{uv\in\delta(S)} c_{uv} x_{uv} \\ &= c_A x_A + c_{\delta(S)} x_{\delta(S)}, \end{aligned}$$

como queríamos demonstrar.

Algoritmo Central

Finalmente, estamos preparados para apresentar a análise do algoritmo Central.

Teorema 3: O algoritmo Central(G, k, K, c, x) produz um K-multicorte M tal que

$$c(M) \leq (2\ln(2k))\left(1 + \frac{1}{k}|K|\right)cx$$

e consome tempo polinomial em $\langle G\rangle + \langle c\rangle + \langle x\rangle$.

Demonstração: Vamos começar provando que, após a execução das linhas 10 e 11,

$$p_{j-1} < \frac{1}{2}.$$

Para isso, seja h o menor natural tal que $p_h \geq \frac{1}{2}$.

Tal h existe e é maior do que 1, já que $p_n \geq x(s,t) \geq 1$ e $p_1 = 0$.

Algoritmo Central

Como $k \geq 1$ e $p_h \geq \frac{1}{2}$, temos que $(2k)^{2p_h} - 1 \geq k$.

Além disso, lembre-se que $\mathcal{V}(s, p_h) \leq cx$.

Então

$$\mathcal{V}(s, p_h) \leq ((2k)^{2p_h} - 1)\frac{1}{k}cx.$$

Assim, $j \leq h$ depois da execução da linha 11.

Como $p_{h-1} < \frac{1}{2}$ pela escolha de h e porque $p_1 = 0$, temos que $p_{j-1} < \frac{1}{2}$.

Algoritmo Central

Agora vamos verificar que o conjunto devolvido pelo algoritmo é, de fato, um K-multicorte.

Se $K = \emptyset$, então o conjunto vazio que o algoritmo devolve é um multicorte.

Se $cx = 0$, o conjunto $\{e \in E_G : x_e > 0\}$ é um K-multicorte. Isso porque, como $x(E_P) \geq 1$ para todo K-caminho P, todo K-caminho tem pelo menos uma aresta e com $x_e > 0$.

Algoritmo Central

Suponha agora que $K \neq \emptyset$ e $cx > 0$.

Neste caso, o algoritmo devolve $M := \delta(S) \cup M_B$.

Vamos supor, por hipótese de indução, que M_B é um K_B-multicorte no grafo G_B.

Vamos verificar que M é um K-multicorte em G.

Como distâncias satisfazem a desigualdade triangular e $p_{j-1} < \frac{1}{2}$, temos que para quaisquer dois vértices u e v em S,

$$x(u, v) \leq x(u, s) + x(s, v) < 1. \tag{5}$$

Algoritmo Central

Suponha agora que P é um K-caminho em G.

Como $x(E_P) \geq 1$, a desigualdade (5) garante que P tem pelo menos um extremo fora de S.

Se P tem um vértice em S, então também tem pelo menos uma aresta em $\delta(S)$.

Se P não tem vértices em S, então também é um K_B-caminho e, portanto, tem pelo menos uma aresta em M_B.

Logo, $\delta(S) \cup M_B$ é um K-multicorte.

Algoritmo Central

Vamos ver agora que $c(M) \leq (2\ln(2k))\left(1 + \frac{1}{k}|K|\right) cx$.

Se $K = \emptyset$ ou $cx = 0$, claramente vale a desigualdade.

Suponha então que $K \neq \emptyset$ e $cx > 0$.

Neste caso, o algoritmo devolve $M := \delta(S) \cup M_B$.

Como $\{s, t\}$ está em K mas não em K_B, temos $|K_B| < |K|$.

Isso garante o sucesso da recursão na linha 17 e, portanto, vale que

$$c_B(M_B) \leq (2\ln(2k))\left(1 + \frac{1}{k}|K_B|\right) c_B x_B.$$

Defina $A := E_{G[S]}$, como no Lema 1. Usando a desigualdade anterior e o Lema 1, temos que

$$c(\delta(S) \cup M_B) = c(\delta(S)) + c(M_B)$$

$$\leq (2\ln(2k))(c_A x_A + x_{\delta(S)} x_{\delta(S)} + \frac{1}{k} cx + (1 + \frac{1}{k}|K_B|)c_B x_B)$$

$$= (2\ln(2k))(c_A x_A + x_{\delta(S)} x_{\delta(S)} + c_B x_B + \frac{1}{k} cx + \frac{1}{k}|K_B| c_B x_B).$$

Como $(A, \delta(S), B)$ é uma partição de E_G, temos

$$c_A x_A + c_{\delta(S)} x_{\delta(S)} + c_B x_B = cx.$$

Então,

$$c(\delta(S) \cup M_B) \leq (2\ln(2k))(cx + \frac{1}{k}cx + \frac{1}{k}|K_B|c_B x_B).$$

Como $|K_B| \leq |K| - 1$,

$$c(\delta(S) \cup M_B) \leq (2\ln(2k))(cx + \frac{1}{k}cx + \frac{1}{k}(|K| - 1)c_B x_B).$$

Por fim, como $c_B x_B \leq cx$,

$$c(\delta(S) \cup M_B) \leq (2\ln(2k))(1 + \frac{1}{k}|K|)cx,$$

como queríamos demonstrar.

Algoritmo Central

Quanto ao tempo de execução do algoritmo Central, não é difícil ver que as linhas 1 a 16 consomem tempo polinomial em $\langle G \rangle$, $|K|$ e $\langle x \rangle$.

Em especial, as linhas 10 e 11 do algoritmo consomem tempo $O(|V_G|)$.

Com estas informações, é fácil mostrar, por indução em $|K|$, que o algoritmo consome tempo polinomial em $\langle G \rangle + \langle c \rangle + \langle x \rangle$.