Algoritmos de aproximação - Método primal-dual
Problema da floresta de Steiner

Marina Andretta

ICMC-USP

11 de novembro 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.

Problema da floresta de Steiner

Trataremos agora do problema da floresta de Steiner, um caso particular do problema da transversal mínima.

O método de aproximação primal-dual aplicado a este problema resultará em um algoritmo que, embora semelhante em muitos aspectos ao algoritmo apresentado para encontrar uma solução aproximada para o problema da transversal mínima (algoritmo MinTC-BE), tem uma razão de aproximação substancialmente melhor.

Além disso, um pós-processamento que era inoperante para o algoritmo MinTC-BE, passa a ser crucial.

Problema da floresta de Steiner

Seja G um grafo e $\mathcal{R}$ uma coleção de subconjuntos de V_G.

Uma $\mathcal{R}$-floresta de G é qualquer floresta geradora F de G tal que todo elemento de $\mathcal{R}$ está inteiramente contido em algum componente de F.

Quando $\mathcal{R}$ está subentendida, dizemos simplesmente que F é uma floresta de Steiner de G.

Se $\mathcal{R}$ tem um só elemento, o conceito de floresta de Steiner reduz-se ao de árvore de Steiner: uma árvore (não necessariamente geradora) de G cujo conjunto de vértices contém o único conjunto em $\mathcal{R}$.

Problema da floresta de Steiner

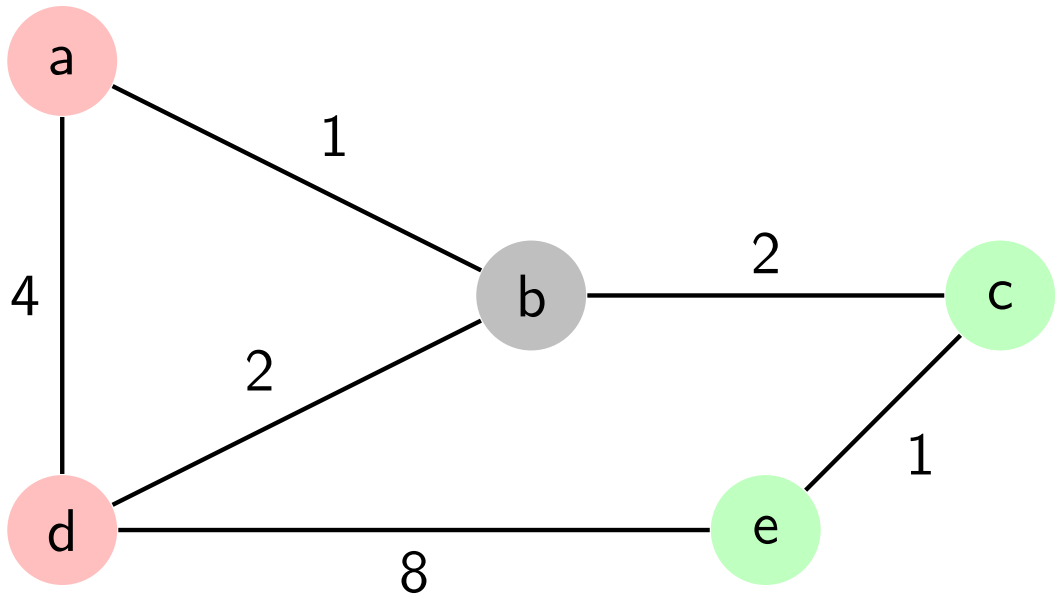
O problema da floresta de Steiner (*Steiner forest problem*) consiste no seguinte:

Problema $\textsc{MinFS}(G, c, \mathcal{R})$: Dados um grafo G, um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e e uma coleção $\mathcal{R}$ de subconjuntos de V_G, encontrar uma $\mathcal{R}$-floresta F que minimize $c(F)$.

Podemos dizer que $c(F)$ é o custo da floresta F. Assim, o problema consiste em encontrar uma floresta de Steiner de custo mínimo.

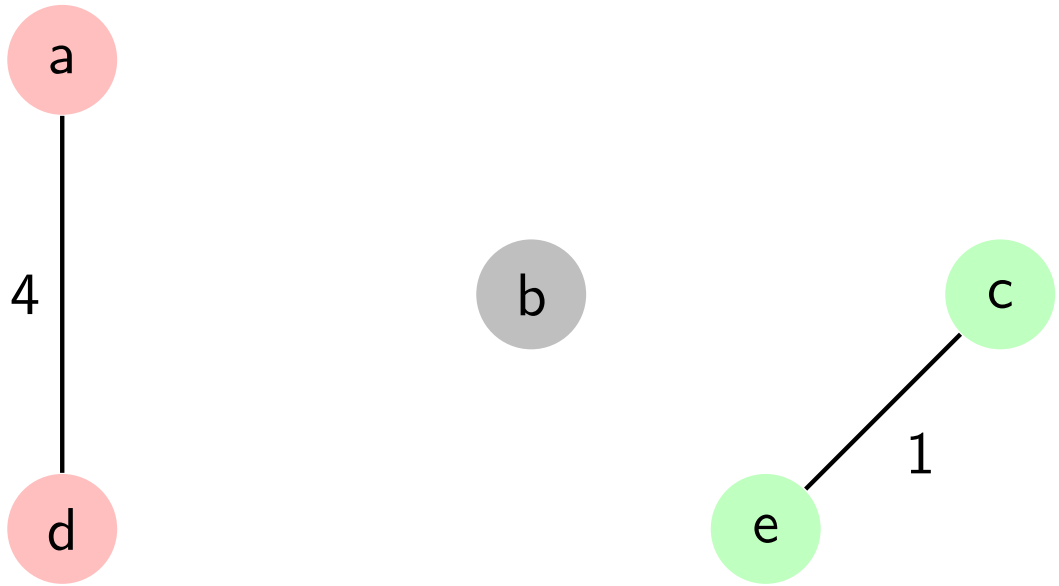
Exemplo

Considere o grafo G abaixo, com os custos nas arestas e $\mathcal{R} = \{\{a, d\}, \{c, e\}\}$.



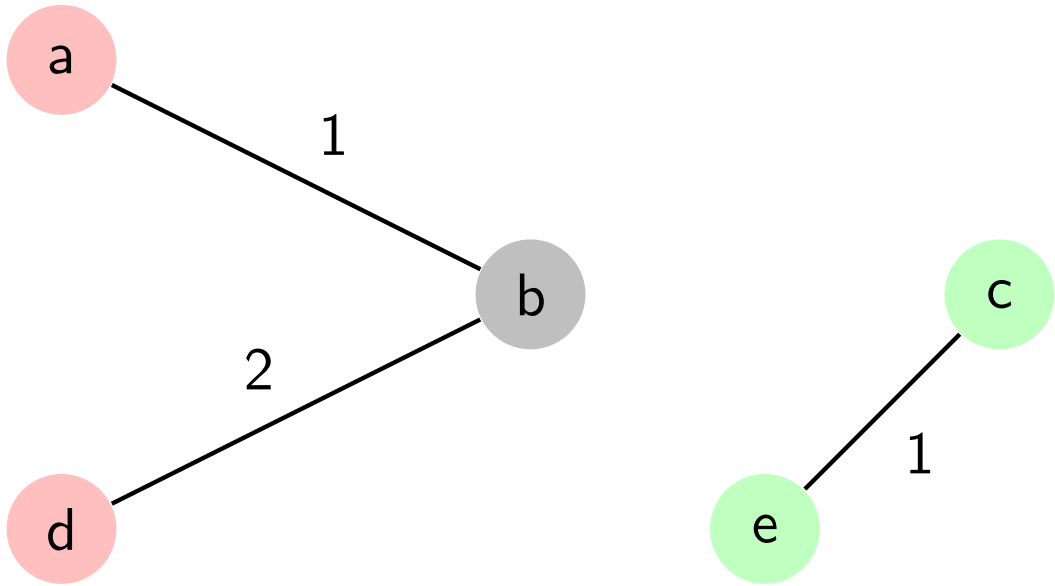
Exemplo

Uma $\mathcal{R}$-floresta F de é dada na figura, com custo $c(F) = 5$.



Exemplo

Uma $\mathcal{R}$-floresta F de custo mínimo é dada na figura, com custo $c(F) = 4$.



Problema da floresta de Steiner

O problema MinFS é NP-difícil, mesmo quando restrito a árvores de Steiner, ou seja, mesmo quando a coleção $\mathcal{R}$ contém um só conjunto.

Dada uma instância $(G, c, \mathcal{R})$ do problema, adotamos as abreviaturas $V := V_G$ e $E := E_G$ e dizemos que um subconjunto S de V é ativo se

$$R \cap S \neq \emptyset \quad \text{e} \quad R \setminus S \neq \emptyset$$

para algum R em $\mathcal{R}$.

A coleção de todos os conjuntos ativos será denotada por $\mathcal{S}$. Um subconjunto de V que não pertence a $\mathcal{S}$ é inativo.

Problema da floresta de Steiner

É fácil verificar que uma floresta geradora F de G é uma $\mathcal{R}$-floresta se e somente se

$$\delta_F(S) \neq \emptyset \text{ para todo } S \text{ em } \mathcal{S}.$$

Portanto, toda floresta de Steiner é uma transversal da coleção de todos os cortes da forma $\delta(S)$ com S em $\mathcal{S}$.

É claro que uma floresta geradora F é uma $\mathcal{R}$-floresta se e somente se V_T é inativo para cada componente T de F.

Programa primal

O seguinte programa linear é uma relaxação de $\textsc{MinFS}(G, c, \mathcal{R})$: encontrar um vetor x indexado por E que

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

Dada uma floresta de Steiner F, é claro que o vetor característico de E_F é uma solução viável de (1).

Portanto, se $\hat{x}$ é uma solução ótima de (1), então $c\hat{x} \leq opt(G, c, \mathcal{R})$.

Programa dual

O dual do programa (1) consiste em encontrar um vetor y indexado pela coleção $\mathcal{S}$ dos subconjuntos ativos de V que

$$\begin{array}{lll} \text{maximize} & y(\mathcal{S}) & \\ \text{sob as restrições} & \sum_{S:e\in\delta(S)} y_S \leq c_e, & \text{para cada } e \text{ em } E, \\ & y_S \geq 0, & \text{para cada } S \text{ em } \mathcal{S}. \end{array} \qquad (2)$$

Exemplo

Para o caso do exemplo, precisamos determinar a coleção $\mathcal{S}$.

Ele é formada por todos os subconjuntos dos vértices $\{a, b, c, d, e\}$ que tem algum elemento, mas não todos, de $\{a, d\}$ e de $\{c, e\}$.

Ou seja,

$$\mathcal{S} = \{\{a\}, \{d\}, \{a, b\}, \{a, c\}, \{a, e\}, \{d, b\}, \{d, c\}, \{d, e\},$$

$$\{a, b, c\}, \{a, b, e\}, \{a, c, e\}, \{d, b, c\}, \{d, b, e\}, \{d, c, e\},$$

$$\{a, b, c, e\}, \{d, b, c, e\}, \{c\}, \{e\}, \{c, b\}, \{e, b\},$$

$$\{c, a, d\}, \{e, a, d\}, \{c, a, b, d\}, \{e, a, b, d\}\}.$$

Exemplo

No entanto, note que $\delta(S) = \delta(V \setminus S)$. Ou seja, podemos eliminar de $\mathcal{S}$ os subconjuntos complementares.

Desta forma, temos

$$\mathcal{S} = \{\{a\}, \{d\}, \{c\}, \{e\},$$

$$\{a, b\}, \{a, c\}, \{a, e\}, \{b, d\}, \{c, d\}, \{d, e\}, \{b, c\}, \{b, e\}\}.$$

Exemplo

Então o programa linear primal é

$$
\begin{aligned}
\text{minimizar} \quad & x_{ab} + 4x_{ad} + 2x_{bd} + 2x_{bc} + x_{ce} + 8x_{de} \\
\text{sob as restrições} \quad & x_{ab} + x_{ad} \geq 1, \\
& x_{ab} + x_{bd} + x_{bc} \geq 1, \\
& x_{bc} + x_{ce} \geq 1, \\
& x_{ce} + x_{de} \geq 1, \\
& x_{ad} + x_{bd} + x_{bc} \geq 1, \\
& x_{ab} + x_{ad} + x_{bc} + x_{ce} \geq 1, \\
& x_{ab} + x_{ad} + x_{ce} + x_{de} \geq 1, \\
& x_{ab} + x_{ad} + x_{bc} + x_{de} \geq 1, \\
& x_{ad} + x_{bd} + x_{bc} + x_{ce} + x_{de} \geq 1, \\
& x_{ad} + x_{bd} + x_{ce} \geq 1, \\
& x_{ab} + x_{bd} + x_{ce} \geq 1, \\
& x_{ab} + x_{bd} + x_{bc} + x_{ce} + x_{de} \geq 1, \\
& x_{ab}, x_{ad}, x_{bd}, x_{bc}, x_{ce}, x_{de} \geq 0.
\end{aligned}
$$

Exemplo

E o programa linear dual é

maximizar

$$y_{\{a\}} + y_{\{d\}} + y_{\{c\}} + y_{\{e\}} + y_{\{a,b\}} + y_{\{a,c\}} + y_{\{a,e\}} + y_{\{b,d\}} + y_{\{c,d\}} + y_{\{d,e\}} + y_{\{b,c\}} + y_{\{b,e\}}$$

sob as restrições

$$\begin{aligned}
y_{\{a\}} + y_{\{a,c\}} + y_{\{a,e\}} + y_{\{b,d\}} + y_{\{b,c\}} + y_{\{b,e\}} &\leq 1,\\
y_{\{a\}} + y_{\{d\}} + y_{\{a,b\}} + y_{\{a,c\}} + y_{\{a,e\}} + y_{\{b,d\}} + y_{\{c,d\}} + y_{\{d,e\}} &\leq 4,\\
y_{\{d\}} + y_{\{a,b\}} + y_{\{c,d\}} + y_{\{d,e\}} + y_{\{b,c\}} + y_{\{b,e\}} &\leq 2,\\
y_{\{c\}} + y_{\{a,b\}} + y_{\{a,c\}} + y_{\{b,d\}} + y_{\{c,d\}} + y_{\{b,e\}} &\leq 2,\\
y_{\{c\}} + y_{\{e\}} + y_{\{a,c\}} + y_{\{a,e\}} + y_{\{c,d\}} + y_{\{d,e\}} + y_{\{b,c\}} + y_{\{b,e\}} &\leq 1,\\
y_{\{e\}} + y_{\{a,e\}} + y_{\{b,d\}} + y_{\{c,d\}} + y_{\{b,e\}} &\leq 8,\\
y_{\{b,d\}}, y_{\{c,d\}}, y_{\{d,e\}}, y_{\{b,c\}}, y_{\{b,e\}} &\geq 0.
\end{aligned}$$

Aproximação primal-dual

Se y é uma solução viável do programa dual (2) e $\hat{x}$ é uma solução ótima do programa primal (1) então, pelo teorema fraco da dualidade, $y(\mathcal{S}) \leq c\hat{x}$.

Portanto,

$$y(\mathcal{S}) \leq opt(G, c, \mathcal{R}).$$

Aproximação primal-dual

Considere agora as condições de folgas aproximadas. A condição de folgas 1-aproximadas no primal exige que

$$\sum_{S:e\in\delta(S)} y_S = c_e \text{ sempre que } x_e > 0$$

e a condição de folgas 2-aproximadas no dual exige que

$$x(\delta(S)) \leq 2 \text{ sempre que } y_S > 0.$$

Aproximação primal-dual

Se soubéssemos encontrar uma solução viável x de (1) e uma solução viável y de (2) que satisfizessem as duas condições de folgas aproximadas, teríamos, pelo lema das folgas aproximadas, uma 2-aproximação para o MinFS.

Sabemos como satisfazer a condição de folgas 1-aproximadas no primal, mas não sabemos como satisfazer a condição de folgas 2-aproximadas no dual.

Ainda assim, é possível obter uma 2-aproximação se respeitarmos a segunda condição em um certo sentido "médio".

Descrição do algoritmo

No início de cada iteração, temos uma floresta geradora F de G e um vetor y indexado por $\mathcal{S}$.

O vetor y é viável em (2) e o par (F, y) satisfaz a condição de folgas 1-aproximadas

$$\sum_{S : e \in \delta(S)} y_S = c_e \text{ para cada } e \text{ em } F.$$

Descrição do algoritmo

Em cada iteração, dizemos que um componente T de F é ativo se $V_T \in \mathcal{S}$, e inativo em caso contrário.

Denotamos por $\mathcal{S}_F$ a coleção dos conjuntos de vértices dos componentes ativos de F.

Abusando um pouco da terminologia, dizemos que $\mathcal{S}_F$ é a coleção dos componentes ativos de F.

Descrição do algoritmo

Cada elemento S de $\mathcal{S}_F$ viola a restrição $x(\delta(S)) \geq 1$ de (1), com x o vetor característico de F.

Isso sugere que devemos acrescentar à F alguma das arestas de $\delta(S)$.

Qualquer aresta desse tipo liga dois componentes distintos de F. Dizemos que esta aresta é externa à F.

Devemos, então, escolher uma aresta externa e acrescentá-la à F. Resta esclarecer como escolher essa aresta.

Descrição do algoritmo

Como o programa dual (2) tem o objetivo de maximizar a soma das variáveis y_S, procuramos aumentar uniformemente os valores das variáveis y_S com S em $\mathcal{S}_F$ de modo a não violar as restrições do programa dual.

Esse aumento gradativo de alguns componentes de y pára quando a restrição $\sum y_S \leq c_e$ correspondente a alguma aresta externa e for satisfeita com igualdade.

A aresta e é então acrescentada à floresta F.

Descrição do algoritmo

O processo iterativo pára quando F não tem componentes ativos.

O algoritmo devolve então uma $\mathcal{R}$-floresta minimal de F, ou seja, uma floresta F_1 tal que, para cada aresta e, a floresta $F_1 - e$ tem algum componente ativo.

A seguir, descrevemos o algoritmo, proposto por Goemans e Williamson, formalmente, mas de maneira pouco detalhada.

O algoritmo supõe que a instância $(G, c, \mathcal{R})$ é viável, ou seja, que cada conjunto em $\mathcal{R}$ está contido em algum componente de G.

Algoritmo MinFS-GW

Algoritmo MinFS-GW$(G, c, \mathcal{R})$:

1 faça $F \leftarrow (V, \emptyset)$;
2 para cada S em $\mathcal{S}$ faça $y_S \leftarrow 0$;
3 enquanto $\mathcal{S}_F \neq \emptyset$, faça:
4 seja y' o vetor característico de $\mathcal{S}_F$;
5 seja θ o maior número tal que
6 $\sum_{S:e\in\delta(S)}(y + \theta y')_S \leq c_e$ para cada aresta externa e;
7 seja f uma aresta externa tal que $\sum_{S:f\in\delta(S)}(y + \theta y')_S = c_f$;
8 faça $y \leftarrow y + \theta y'$;
9 faça $F \leftarrow F + f$;
10 faça $F_0 \leftarrow F$;
11 seja F_1 uma $\mathcal{R}$-floresta minimal de F_0.
12 devolva F_1.

Observações sobre o algoritmo

A cardinalidade da coleção $\mathcal{S}$ não é limitada por uma função polinomial de $|V|$ (por exemplo, $|\mathcal{S}|$ pode ser da ordem de $2^{|V|/2}$). Assim, o número de componentes do vetor y não será polinomial.

É possível contornar essa dificuldade se armazenarmos apenas os componentes não-nulos de y, que é limitado por um polinômio em $|V|$.

O número de iterações não supera $|V|$, o número de componentes que assumem valores não-nulos em cada iteração é limitado por $|\mathcal{S}_F|$ e $|\mathcal{S}_F| < |V|$.

Vale a mesma observação para o vetor y' na linha 4, indexado por $\mathcal{S}$ e definido por $y'_S = 1$ se e somente se $S \in \mathcal{S}_F$.

Observações sobre o algoritmo

O cálculo de θ nas linhas 5 e 6 pode ser organizado da seguinte maneira.

Para cada vértice v, defina $d(v) := \sum_{S:v\in S} y_S$.

Para cada aresta uv externa a F, defina:

- $\theta_{uv} := \infty$, se uv liga dois componentes inativos de F;
- $\theta_{uv} := c_e - d(u) - d(v)$, se uv liga um componente ativo de F a um componente inativo; e
- $\theta_{uv} := \frac{1}{2}(c_e - d(u) - d(v))$, se uv liga dois componentes ativos.

Observações sobre o algoritmo

Finalmente, adote $\theta := \min_e \theta_e$, com o mínimo tomado sobre todas as arestas externas.

Portanto, é possível implementar o algoritmo MinFS-GW de modo que seu consumo de tempo seja $O(|V|^2 \log(|V|))$.

Lema 1: O algoritmo MinFS-GW admite uma implementação polinomial.

Análise do algoritmo

No início de cada iteração, F é uma floresta geradora de G.

Ao final do processo iterativo, a floresta F_0 não tem componentes ativos e, portanto, é uma $\mathcal{R}$-floresta.

A floresta F_1 que o algoritmo devolve também é uma $\mathcal{R}$-floresta.

Antes de delimitar $c(F_1)$, é preciso estabelecer uma relação entre F_1 e a coleção $\mathcal{S}_F$ dos componentes ativos de F em uma iteração arbitrária do algoritmo.

Análise do algoritmo

Lema 2: No início de cada iteração, vale a desigualdade

$$\sum_{S \in \mathcal{S}_F} |\delta_{F_1}(S)| \leq 2|\mathcal{S}_F|,$$

onde F_1 é a floresta de Steiner que o algoritmo devolve.

Análise do algoritmo

Teorema: O algoritmo MinFS-GW é uma 2-aproximação para o MinFS.

Demonstração: Já vimos que o subgrafo F_1 que o algoritmo devolve é uma $\mathcal{R}$-floresta.

Agora vamos mostrar que, no início de cada iteração, vale a desigualdade

$$\sum_{S \in \mathcal{S}} |\delta_{F_1}(S)| y_S \leq 2y(\mathcal{S}). \tag{3}$$

Análise do algoritmo

Como no início da primeira iteração $y_S = 0$ para todo $S \in \mathcal{S}$, a desigualdade (3) é válida.

Suponha agora que a desigualdade (3) seja válida no início de uma iteração qualquer.

Durante a iteração, y_S é acrescido de θ se e somente se $S \in \mathcal{S}_F$.

Análise do algoritmo

Portanto, o lado esquerdo de (3) é acrescido de

$$\sum_{S \in \mathcal{S}_F} |\delta_{F_1}(S)|\theta$$

e o lado direito de (3) é acrescido de

$$2|\mathcal{S}_F|\theta.$$

Como, pelo lema 1, $\sum_{S \in \mathcal{S}_F} |\delta_{F_1}(S)| \leq 2|\mathcal{S}_F|$, temos que o incremento do lado esquerdo não é maior que o do lado direito.

Portanto a desigualdade (3) vale no início da iteração seguinte, como queríamos demonstrar.

Análise do algoritmo

No fim do processo iterativo, o vetor y é viável no programa dual (2).

Além disso, a condição de folgas 1-aproximadas vale com F_0 no papel de F e, portanto, também com F_1 no papel de F:

$$\sum_{S:e\in\delta(S)} y_S = c_e \text{ para cada } e \text{ em } F_1.$$

Análise do algoritmo

Agora podemos analisar o custo $c(F_1)$:

$$\begin{aligned} c(F_1) &= \sum_{e \in F_1} c_e \\ &= \sum_{e \in F_1} \sum_{S : e \in \delta(S)} y_S \\ &= \sum_{S \in \mathcal{S}} |\delta_{F_1}(S)| y_S. \end{aligned}$$

Como provamos que que $\sum_{S \in \mathcal{S}} |\delta_{F_1}(S)| y_S \leq 2y(\mathcal{S})$, temos

$$\begin{aligned} c(F_1) &= \sum_{S \in \mathcal{S}} |\delta_{F_1}(S)| y_S \\ &\leq 2y(\mathcal{S}) \\ &\leq 2opt(G, c, \mathcal{R}), \end{aligned}$$

já que $y(\mathcal{S})$ é uma solução do programa dual (2) e, portanto, $y(\mathcal{S}) \leq opt(G, c, \mathcal{R})$.