Pontifícia Universidade Católica
do Rio de Janeiro

João Paulo de Freitas Araujo

Algoritmos para acelerar a computação de árvores de cortes de Gomory e Hu

Tese de Doutorado

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Produção da PUC-Rio como requisito parcial para obtenção do título de Doutor em Engenharia de Produção.

Orientador: Prof. José Eugenio Leal
Co-orientador: Prof. Fernanda Maria Pereira Raupp

Rio de Janeiro
Agosto de 2016

Pontifícia Universidade Católica
do Rio de Janeiro

João Paulo de Freitas Araujo

Algoritmos para acelerar a computação de árvores de cortes de Gomory e Hu

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-Graduação em Engenharia de Produção da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. José Eugenio Leal
Orientador
Departamento de Engenharia Industrial - PUC-Rio

Prof. Fernanda Maria Pereira Raupp
Co-orientadora
Laboratório Nacional de Computação Científica - LNCC

Prof. Mitre Costa Dourado
Universidade Federal do Rio de Janeiro - UFRJ

Prof. Eduardo Sany Laber
Departamento de Informática – PUC-Rio

Prof. Glaydston Mattos Ribeiro
Universidade Federal do Rio de Janeiro - UFRJ

Rafael Martinelli Pinto
Departamento de Engenharia Industrial - PUC-Rio

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 25 de agosto de 2016

João Paulo de Freitas Araujo

Graduou-se em Física pela Pontifícia Universidade Católica do Rio de Janeiro em 2006. Possui mestrado em Engenharia de Produção pela Pontifícia Universidade Católica do Rio de Janeiro (2011).

Ficha Catalográfica

Araujo, João Paulo de Freitas

Algoritmos para acelerar a computação de árvores de cortes de Gomory e Hu / João Paulo de Freitas Araujo ; orientador: José Eugenio Leal ; co-orientadora: Fernanda Maria Pereira Raupp. – 2016.
73 f. : il. ; 30 cm

Tese (doutorado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, 2016.
Inclui bibliografia

1. Engenharia Industrial – Teses. 2. Fluxos em redes multiterminais. 3. Análise de sensibilidade. 4. Corte mínimo. 5. Árvore de cortes. 6. Clustering. I. Leal, José Eugenio. II. Raupp, Fernanda Maria Pereira. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. IV. Título.

CDD: 658.5

A meu pai.

Agradecimentos

Aos orientadores José Eugenio Leal e Fernanda Maria Pereira Raupp, pela confiança, dedicação e contribuição em minha formação. Pela parceria para a concretização deste trabalho.

À CAPES e à PUC - Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos professores que participaram da Comissão Examinadora.

À Claudia, e a todos os professores e funcionários do DEI.

Aos meus pais, Sergio e Teresa, pela educação, atenção, e carinho em toda minha vida.

Ao meu irmão, Luiz Fernando, e a todos os meus amigos, que sempre me apoiaram e me ajudaram de alguma forma.

Resumo

Araujo, João Paulo de Freitas; Leal, José Eugenio. **Algoritmos para acelerar a computação de árvores de cortes de Gomory e Hu.** Rio de Janeiro, 2016. 73p. Tese de Doutorado - Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

Calcular o valor do fluxo máximo entre um nó origem e um nó destino em uma rede é um problema clássico no contexto de Fluxos em Redes. Sua extensão, chamada de *problema do fluxo máximo multiterminal,* consiste em achar os valores dos fluxos máximos entre todos os pares de nós de uma rede não direcionada. Estes problemas possuem diversas aplicações, especialmente nos campos de transporte, logística, telecomunicações e energia. Neste trabalho, apreciamos a recente teoria da análise de sensibilidade, em que se estuda a influência da variação de capacidade de arestas nos fluxos máximos multiterminais, e estendemos a computação dinâmica dos fluxos multiterminais para o caso de mais de uma aresta com capacidade variável. Através dessa teoria, relacionamos também nós de corte e fluxos multiterminais, o que permitiu desenvolver um método competitivo para solucionar o problema do fluxo máximo multiterminal, quando a rede possui nós de corte. Os resultados dos experimentos computacionais conduzidos com o método proposto são apresentados e comparados com os de um algoritmo clássico, fazendo uso de instâncias geradas e outras conhecidas da literatura. Por último, aplicamos a teoria apresentada em um problema de identificação de complexos de proteínas em redes de interação proteína-proteína. Através da generalização de um algoritmo e de um resultado teórico sobre exclusão de cortes mínimos, foi possível reduzir o número de cálculos de fluxo máximo necessários para identificar tais complexos.

Palavras-chave

Fluxos em redes multiterminais; análise de sensibilidade; corte mínimo; árvore de cortes; clustering.

Abstract

Araujo, João Paulo de Freitas; Leal, José Eugenio (Advisor). **Algorithms for performing the computation of Gomory Hu cut-trees.** Rio de Janeiro, 2016. 73p. Doctoral Thesis - Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

Computing the maximum flow value between a source and a terminal nodes in a given network is a classic problem in the context of network flows. Its extension, namely the multi-terminal maximum flow problem, consists of finding the maximum flow values between the all pairs of nodes in a given undirected network. These problems have several applications, especially in the fields of transports, logistics, telecommunications and energy. In this work, we study the recent theory of sensitivity analysis, which examines the influence of edges capacity variation on the multi-terminals maximum flows, and we extend the dynamic computation of multi-terminals flows to the case of more than one edge with variable capacity. Based on this theory, we also relate cut nodes and multi-terminals flows, allowing us to develop a competitive method to solve the multi-terminal maximum flow problem, when the network has cut nodes. The results of the computational experiments conducted with the proposed method are presented and compared with the results of a classical algorithm, using generated and well-known instances of the literature. Finally, we apply the presented theory on a problem of identifying protein complexes in protein-protein interaction networks. Through the generalization of an algorithm and a theoretical result about exclusion of minimum cuts, it was possible to reduce the number of maximum flow computations necessary to identify such complexes.

Keywords

Multi-terminal network flows; sensitivity analysis; minimum cut; cut-tree; clustering.

Sumário

1
Introdução

Calcular o valor do fluxo máximo entre um nó origem e um nó destino em uma rede é um problema clássico no contexto de Fluxos em Redes. Sua extensão, chamada de *problema do fluxo máximo multiterminal,* consiste em achar os valores dos fluxos máximos entre todos os pares de nós de uma rede não direcionada. Estes problemas possuem diversas aplicações, especialmente nos campos de transporte, logística, telecomunicações e energia. Exemplos podem ser encontrados em [1], [2], [3] e [4].

É importante ressaltar que o problema multiterminal é diferente do problema de multifluxos, ou multiproduto. Nestes, os fluxos entre várias origens e destinos compartilham a rede, enquanto que naquele, apesar de determinar o fluxo máximo entre todos os pares de nós, há somente um fluxo entre uma origem e um destino na rede por vez.

Na década de 50, Ford & Fulkerson [5] popularizaram o problema do fluxo máximo com seu método de resolução. Eles, especialmente, demonstraram a relação entre o valor do fluxo máximo e a capacidade do corte mínimo. Assim, o problema multiterminal numa rede não direcionada, por sua vez, estava resolvido, bastando aplicar o algoritmo $n(n-1)/2$ vezes para determinar o fluxo máximo entre todos os pares de nós, onde *n* é o número de nós da rede.

Alguns anos mais tarde, Gomory & Hu [6] desenvolveram um método para calcular todos os valores de fluxo máximo de uma rede não direcionada resolvendo apenas $(n-1)$ vezes o problema do fluxo máximo. O resultado do algoritmo é exposto através de uma árvore de cortes, que reflete os valores de fluxo máximo e cortes mínimos entre todos os pares dos *n* nós.

Posteriormente, Gusfield [7] apresentou um modo mais simples de obter a mesma árvore de cortes, mas também utilizando $n-1$ vezes o algoritmo de fluxo máximo. Goldberg & Tsioutsiouliklis [8] realizaram um estudo comparando computacionalmente três variações do algoritmo de Gomory e Hu e o algoritmo de Gusfield, e concluíram que o algoritmo de Gomory e Hu, implementado com heurísticas, é mais robusto que o de Gusfield. Para o caso não ponderado (quando cada aresta possui capacidade unitária), Bhalgat et al. [9] mostraram um

algoritmo mais rápido que não utiliza um algoritmo de fluxo máximo como procedimento interno.

No contexto de fluxos multiterminais, a análise de sensibilidade teve seu início na década de 60 com Elmaghraby [10]. Ele estudou os efeitos nos fluxos máximos de uma rede quando fosse permitida a variação de capacidade de uma única aresta (paramétrica) da rede. Recentemente, Berthomé et al. [11] aperfeiçoaram o resultado de Elmaghraby para o caso de uma aresta paramétrica na rede e generalizaram para o caso de *k* arestas paramétricas, observando que um total de 2^k árvores de cortes é suficiente para calcular todos os fluxos máximos para quaisquer valores dos parâmetros.

Com o objetivo de reduzir a complexidade dos algoritmos que necessitam calcular duas ou mais árvores de cortes em sequência, para o caso de variação crescente da capacidade de uma única aresta paramétrica na rede, Barth et al. [12] mostraram como utilizar as informações de uma árvore de cortes já calculada para construir a seguinte. Hartmann & Wagner [13] demonstraram o mesmo para o caso de a capacidade da aresta paramétrica decrescer.

Nesta tese, temos o objetivo de estender as teorias de Barth et al. [12] e Hartmann & Wagner [13] mencionadas acima, englobando não somente o caso de uma única aresta paramétrica na rede, mas, também, quando há mais de uma.

Ainda, fundamentada na teoria da análise de sensibilidade de fluxos multiterminais, mostramos uma propriedade que relaciona nós de corte e árvores de cortes para redes não direcionadas. Com base em tal propriedade, propomos uma nova abordagem para a computação de árvores de cortes em redes que possuem nós de corte.

Experimentos computacionais foram realizados com o intuito de comparar os tempos de execução da nova abordagem proposta com a tradicional. Para estes experimentos, quatro famílias de instâncias são utilizadas: PATH e TREE, de Goldberg & Tsioutsiouliklis [8], CACTUS, da literatura, e PARTED, que foi especialmente desenvolvida para tal. Quando a rede não direcionada possui nós de corte, os resultados numéricos mostram que a computação de árvores de cortes, utilizando a nova abordagem, é eficaz.

De acordo com as extensões feitas na teoria da análise de sensibilidade, no capítulo subsequente são apresentados novos algoritmos que constroem árvores de cortes de Gomory e Hu dinamicamente, para os casos de alteração de capacidade

em mais de uma aresta da rede. Nas variações crescentes, o algoritmo é uma modificação do método tradicional de Gomory e Hu. Para as variações decrescentes, generalizou-se o algoritmo apresentado por Hartmann & Wagner [13].

Por último, aplicamos os resultados teóricos obtidos nesta tese em um problema de clustering no campo da biologia, especificamente, o problema de identificação de complexos de proteínas em redes de interação proteína-proteína (PPI). Mitrofanova et al. [14] desenvolveram um algoritmo que utiliza a computação de seguidas árvores de cortes para identificar os complexos de proteínas. Desse modo, buscamos reduzir o esforço computacional realizado pelo algoritmo de identificação de complexos de proteínas.

O trabalho encontra-se estruturado da seguinte forma. Capítulo 2 apresenta as notações usadas, conceitos básicos relacionados à teoria de grafos e problemas de fluxo máximo, e o método de Gomory & Hu [6] que soluciona eficientemente o problema de fluxo máximo multiterminal. No Capítulo 3, apreciamos a teoria de computação dinâmica de árvores de cortes de Gomory Hu e a estendemos para o caso de mais de uma aresta paramétrica na rede. Capítulo 4 apresenta um método alternativo para construir uma árvore de cortes baseado na identificação de nós de corte, juntamente com um exemplo de sua aplicação e testes computacionais. Capítulo 5 introduz dois novos algoritmos que calculam árvores de cortes dinamicamente frente a mudanças de capacidade em uma ou mais arestas na rede. O problema de identificação de complexos de proteína em redes PPI e o aperfeiçoamento, pelos nossos teoremas e métodos, do algoritmo de Mitrofanova et al. [14], são abordados no Capítulo 6. Os comentários finais e as recomendações para trabalhos futuros encontram-se nos Capítulos 7 e 8, respectivamente, enquanto o Capítulo 9 contém as referências bibliográficas.

2
Definição do problema

2.1
Formulação matemática

Os problemas do Fluxo Máximo e sua extensão multiterminal surgem no contexto de redes, e estas são representadas por grafos. Para entendimento, segue uma introdução à teoria dos grafos e dos fluxos, e uma apresentação do problema de fluxo máximo. Para mais informações sobre grafos e fluxos, ver as referências [5], [15], [16] e [17].

Conceitos básicos da teoria dos grafos:

Definição 1. *Um* ***grafo direcionado*** $G = (V, A)$ *consiste de um conjunto V de vértices v, também chamados de nós, e um conjunto A de arcos a, onde cada arco é um par ordenado* (i, j) *de vértices. O vértice i do par é a fonte do arco e o vértice j o alvo.*

Na Figura 2.1, os arcos são: (v1, v4), (v4, v3), (v3, v2), (v2, v3), (v2, v4) e (v1, v2).

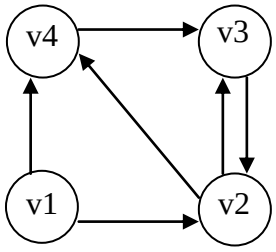


Figura 2.1: Exemplo de grafo direcionado.

Definição 2. *Um* ***grafo não direcionado*** $G = (V, E)$ *consiste de um conjunto V de vértices v, também chamados de nós, e um conjunto E de arestas e, onde cada aresta é um par não ordenado* (i, j) *de vértices.*

Cabe notar que as arestas, representadas pela letra *e*, são diferentes dos arcos, estes representados pela letra *a*. Ao contrário dos arcos, as arestas não possuem sentido, ou igualmente falando, possuem duplo sentido. A Figura 2.2 ilustra uma representação de um grafo não direcionado.

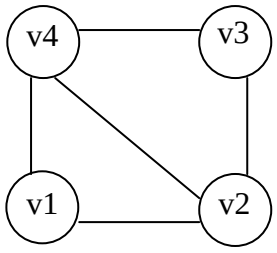


Figura 2.2: Exemplo de grafo não direcionado.

As definições que se seguem são aplicadas a grafos não direcionados, visto que este trabalho lida apenas com este caso.

Definição 3. *Uma aresta e = (i, j) é dita* ***incidente*** *aos nós i e j. Os nós i e j por sua vez são* ***adjacentes*** *entre si e são chamados* ***extremos da aresta*** *e.*

Definição 4. *É chamado de* ***passeio*** *entre dois nós i e j uma sequência de arestas* (i, v_1), (v_1, v_2), ..., (v_{k-1}, j), *ou uma sequência de nós* $i, v_1, v_2, ..., v_{k-1}, j$ *em que dois nós consecutivos são adjacentes, sendo k o número de arestas no passeio.*

Definição 5. *Um passeio é um* ***caminho*** *quando não passa duas vezes por um mesmo nó.*

Denota-se i-j ou $P_{i,j}$ *como um caminho de i até j.*

No grafo da Figura 2.2, um exemplo de caminho entre v1 e v3 é (v1, v2), (v2, v4), (v4, v3).

Definição 6. *Um* ***ciclo*** *é um passeio com no mínimo três arestas, em que o primeiro e o último nó coincidem, mas nenhum outro nó é repetido.*

Um exemplo de ciclo no grafo da Figura 2.2 é (v1, v2), (v2, v4), (v4, v1).

Definição 7. *Um grafo não direcionado G é dito* ***conexo*** *se, para todo par de vértices {s, t} de G, existe um caminho entre s e t.*

Definição 8. *Um grafo é uma* ***árvore*** *se for conexo e não possuir ciclos.*

Uma ***floresta*** *é um grafo que não possui ciclos.*

A Figura 2.3 ilustra exemplos de uma árvore e uma floresta.

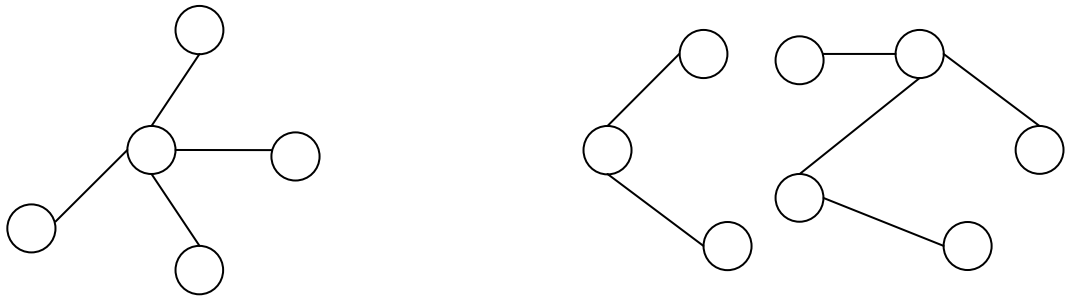


Figura 2.3: Exemplos de uma árvore e de uma floresta.

Definição 9. *O grafo G′ = (V′, E′) é um* ***subgrafo*** *de G = (V, E) se V′ for um subconjunto de V e E′ for um subconjunto de E.*

Uma ***subárvore*** *é um subgrafo de uma árvore.*

Definição 10. *Uma* ***componente conexa*** *é um subgrafo conexo maximal, onde o termo maximal significa que a adição de um novo nó gera um subgrafo desconexo.*

Definição 11. *Uma* ***clique*** *em um grafo não direcionado é um subconjunto de seus nós tais que todo par de nós do subconjunto é conectado por uma aresta.*

Definição 12. *Uma aresta é uma* ***ponte*** *quando sua exclusão desconecta a componente conexa em que se encontra.*

Definição 13. *Um nó é um* ***nó de corte*** *quando sua exclusão desconecta a componente conexa em que se encontra.*

Definição 14. *Uma **componente biconexa** é um subgrafo conexo maximal que não contém nó de corte, onde o termo maximal significa que a adição de um novo nó gera um subgrafo que contém um nó de corte.*

Como é possível ver no exemplo da Figura 2.4, os nós 3 e 6 são nós de corte do grafo em (a), no qual geram três componentes biconexas em (b).

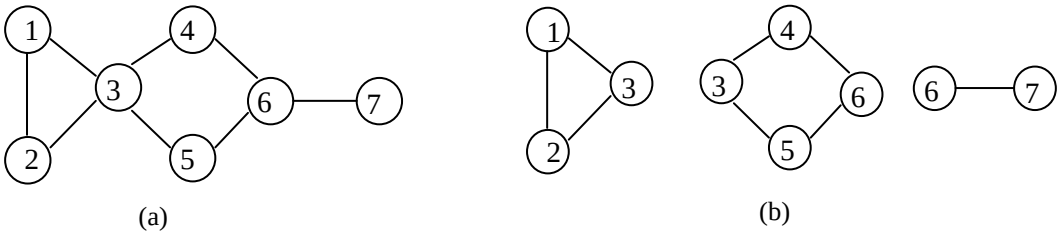


Figura 2.4: Exemplo de um grafo (a) e de suas componentes biconexas (b).

Como nomenclatura, neste trabalho, a letra *n* representará o número de nós em um grafo, e a letra *m* o número de arestas de um grafo.

Conceitos básicos da teoria de fluxos em rede:

Definição 15. *Uma **rede** (não direcionada) é definida como um grafo G = (V, E) associado a uma função c: E →R+ chamada função de **capacidade**.*

*Dois específicos nós s e t são chamados respectivamente a **origem** (source) e o **destino** (terminal) da rede.*

A Figura 2.5 ilustra um exemplo de rede não direcionada.

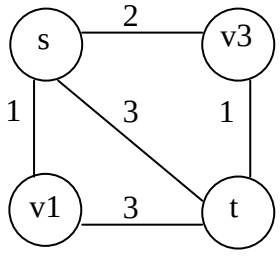


Figura 2.5: Exemplo de rede não direcionada.

Para este trabalho, a menos que seja especificado, toda rede mencionada será uma rede não direcionada.

Definição 16. *Seja uma rede G = (V, E) em que s é o nó origem e t o nó destino. Considere dois subconjuntos complementares S e* $\bar{S}$ *de V, ou seja,* $S \cap \bar{S} = \emptyset$ *e* $S \cup \bar{S} = V$*, onde S contém o nó s e* $\bar{S}$ *contém o nó t. Denote por* (*S,* $\bar{S}$) *o conjunto de arestas com uma das extremidades em S e a outra em* $\bar{S}$*.* (*S,* $\bar{S}$) *é chamado de* ***corte*** *entre, ou que separa, os nós s e t. O corte* (*S,* $\bar{S}$) *é também simbolizado como corte* (*s - t*)*.*

A ***capacidade do corte*** (*S,* $\bar{S}$) *é definida como:*

$$c(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}, (i,j) \in E} c(i, j).$$

A Figura 2.6 ilustra um corte de um grafo, representado pela linha tracejada, entre *s* e *t* de capacidade 8, em que *S* = {s, v1}, $\bar{S}$ = {v3, t}, (*S,* $\bar{S}$) = {(s, v3), (s, t), (v1, t)}, e *c*(s, v3) = 2, *c*(s, t) = 3 e *c*(v1, t) = 3.

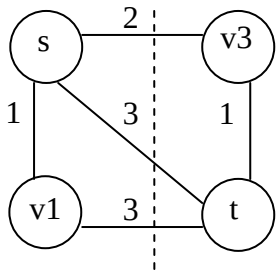


Figura 2.6: Exemplo de corte.

Definição 17. *Dentre todos os possíveis cortes que separam s e t, aquele com a menor capacidade é chamado de* ***corte* (*s - t*) *mínimo****.*

Definição 18. *Seja uma rede não direcionada G = (V, E) com s sendo o nó origem e t o nó destino. Um* ***fluxo*** *em G é uma função f:* $V \times V \mid (v_1, v_2) \in E \rightarrow R+$ *com as seguintes propriedades:*

- ***Sentido único:*** *em uma aresta não pode haver ao mesmo tempo fluxo nos dois sentidos, ou seja,* $\forall i, j \in V,\ f(i,j) * f(j,i) = 0.$
- ***Restrição de capacidade:*** *o fluxo que passa em cada aresta não pode ser superior à capacidade da aresta, ou seja,* $\forall i, j \in V,\ f(i, j) \leq c(i, j).$

- ***Conservação do fluxo:*** *o fluxo total que entra em um nó v tem igual valor ao fluxo total que sai de v, exceto para os casos de s e t. Ou seja,* $\forall v \in V \setminus \{s, t\}, \sum_{i \in V} f(i, v) = \sum_{j \in V} f(v, j).$

O valor do fluxo | f | na rede pode ser obtido por meio das equações seguintes:

1. *fluxo total que sai de s menos o fluxo total que entra em s:*

$$| f | = \sum_{i \in V} f(s, i) - \sum_{j \in V} f(j, s)$$

2. *fluxo total que entra em t menos o fluxo total que sai de t:*

$$| f | = \sum_{i \in V} f(i, t) - \sum_{j \in V} f(t, j)$$

Definição 19. *Determinar o valor máximo do fluxo que se pode enviar do nó origem s para o nó destino t em uma rede, é conhecido como o* ***problema do fluxo máximo****.*

Teorema 1 (Ford & Fulkerson [5]) fluxo máximo - corte mínimo. *Seja f uma função de fluxo de uma rede não direcionada de origem s e destino t, e um corte* $(S, \bar{S})$ *separando s e t. Então:*

1. $| f | \leq c(S, \bar{S})$*;*

2. *Se* $| f | = c(S, \bar{S})$*, então f é máximo e* $(S, \bar{S})$ *é mínimo;*

3. $| f | = c(S, \bar{S})$ *se e somente se,* $\forall i \in S$ *e* $\forall j \in \bar{S}$, $f(i, j) = c(i, j).$

A Figura 2.7 ilustra um fluxo máximo na rede da Figura 2.6. O corte mínimo na rede é $(S, \bar{S}) = \{(s, v1), (s, t), (v3, t)\}$, onde $S = \{s, v3\}$, $\bar{S} = \{v1, t\}$, com capacidade $5 = c(s, v1) + c(s, t) + c(v3, t) = 1 + 3 + 1$. De acordo com a Definição 18, o valor do fluxo pode ser calculado pelo fluxo que sai do nó *s*, logo o valor do fluxo máximo na rede é 5, igual à capacidade do corte mínimo.

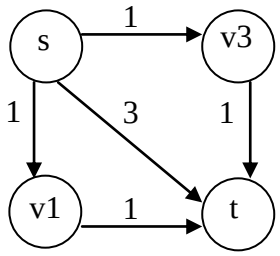


Figura 2.7: Fluxo máximo na rede não direcionada da Figura 2.6. Fluxos com valor zero estão omitidos.

É fácil observar que, em uma rede não direcionada, como as arestas permitem a passagem de fluxo em ambos os sentidos, o fluxo máximo da origem *s* para o destino *t* terá igual valor ao fluxo máximo de *t* para *s*.

Denota-se $f_{s,t}$ como o fluxo máximo entre o par de nós *s* e *t*.

2.2 Procedimento básico

Após a introdução à teoria dos grafos e dos fluxos em rede, foi apresentado, também, o *problema do fluxo máximo* (Definição 19) entre uma origem e um destino em uma rede. Este problema foi primeiramente resolvido por Ford & Fulkerson [5] com base no Teorema 1 do *fluxo máximo e do corte mínimo*.

O problema que trata este trabalho é uma derivação do *problema do fluxo máximo*. Em vez de determinar o fluxo máximo entre somente um par de nós, $\{s, t\}$, por exemplo, busca-se calcular o fluxo máximo entre todos os pares de nós de uma rede; o que é conhecido como o *problema do fluxo máximo multiterminal*.

O *problema do fluxo máximo multiterminal* pode ser obviamente resolvido calculando-se o fluxo máximo $n(n-1)/2$ vezes, uma para cada par de nós não ordenado da rede. Gomory & Hu [6] mostraram que dentre os $n(n-1)/2$ valores

de fluxo máximo existem no máximo $n-1$ valores distintos, e que estes podem ser obtidos resolvendo-se o problema do fluxo máximo exatamente $n-1$ vezes. Estes resultados são apresentados por meio do que eles chamaram de árvore de cortes, pela qual se pode obter os fluxos máximos entre todos os pares de nós da rede.

É importante notar que o algoritmo de Gomory e Hu resolve o problema multiterminal apenas para o caso de uma rede não direcionada. Resoluções eficientes para o caso de uma rede direcionada foram elaboradas primeiramente em Gusfield & Naor [18] e Schnorr [19], mas Benczúr [20] provou estarem erradas. Ao leitor interessado no estudo do caso de redes direcionadas, recomenda-se ver [21].

2.2.1
Método de Gomory e Hu

Após a observação da existência de no máximo $n-1$ valores distintos de fluxo máximo em uma rede, Gomory e Hu desenvolveram um método capaz de obter os $n(n-1)/2$ valores de fluxo máximo por meio de contrações de vértices (Definição 20 a seguir) e $n-1$ execuções do algoritmo de fluxo máximo, cujo resultado final é expresso por uma árvore de cortes (Definição 21 a seguir).

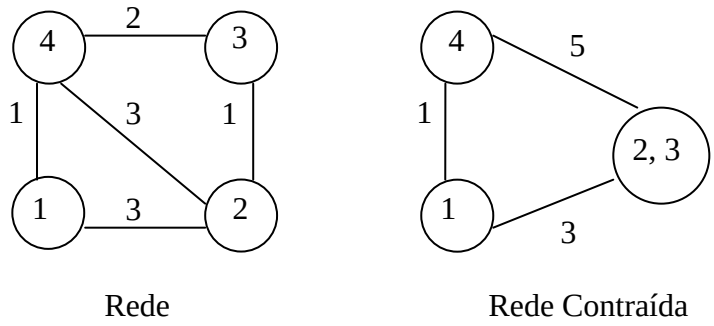
Definição 20. *Seja a rede $G = (V, E)$. Uma **contração** de vértices $v_1, v_2, ..., v_k$ de G em um único vértice $\bar{v}$, chamado super nó, consiste em:*

1. *substituir em G os vértices $v_1, v_2, ..., v_k$ pelo super nó $\bar{v}$;*

2. *substituir, para todo vértice v_i, $i \in \{1,...,k\}$, as arestas (v_i, u), onde $u \in V$ é um vértice não contraído, pela aresta $(\bar{v}, u)$ cuja capacidade $c(\bar{v}, u) = \sum_i c(v_i, u)$.*

*Uma rede que possui um super nó é chamada de **rede contraída**.*

A Figura 2.12 ilustra uma rede e sua forma contraída obtida pela contração dos nós 2 e 3 em um super nó.



Rede Rede Contraída

Figura 2.12: Contração em uma rede.

Para diferenciar das arestas de redes, neste trabalho as arestas das árvores de cortes serão representadas por um par não ordenado [*i*, *j*] de vértices entre colchetes.

Definição 21. *Seja a rede G =* (*V, E*)*. Uma* ***árvore de cortes*** (*cut-tree*) *CT é uma árvore com capacidades nas arestas e com os mesmos vértices de G. Suas propriedades são:*

1. *Para toda aresta* $[i, j] \in CT$*, a capacidade de* [*i, j*] *é igual ao valor do fluxo máximo entre i e j em G:*

$$\forall [i,\ j] \in CT,\ c[i,\ j] = f_{i,j}.$$

2. *Toda aresta de CT corresponde a um corte* (*s - t*) *mínimo em G. A retirada de uma aresta* [*i, j*] *em CT divide-o em dois subconjuntos de vértices X e* $\overline{X}$ *com* $i \in X$ *e* $j \in \overline{X}$*. Assim, o corte* $\left(X, \overline{X}\right)$ *forma um corte mínimo* (*i - j*)*, representado por* $C_{i,j}$*, de G.*

$$\forall [i,\ j] \in CT, [i,\ j] \Rightarrow C_{i,j}.$$

3. *Para todo par de vértices {s, t} de G, o valor $f_{s,t}$ do fluxo máximo entre s e t é a menor capacidade de aresta do caminho único que conecta s e t em CT. Esta aresta de menor capacidade corresponde a um corte mínimo $C_{s,t}$.*

A Figura 2.13 ilustra um exemplo de árvore de cortes construída a partir de uma rede. Pode-se observar, pelas propriedades mencionadas, que o corte mínimo entre os nós 2 e 3 e os nós 1 e 2 são respectivamente as arestas [2, 3] e [1, 2] da árvore de cortes *CT*. Seus fluxos máximos são as capacidades destas arestas, no caso, 3 e 4. O corte mínimo entre os nós 1 e 3 é por sua vez a aresta [2, 3], com fluxo máximo igual a 3.

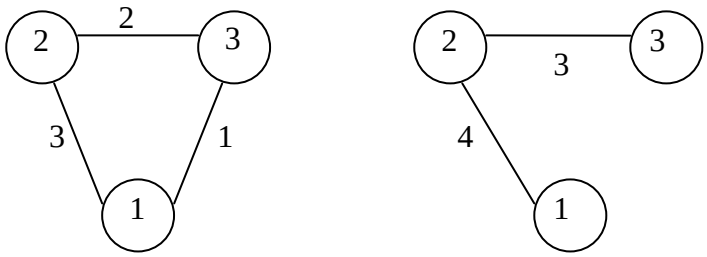


Figura 2.13: Exemplo de uma rede e sua árvore de cortes.

Definição 22. *A **interseção** entre dois cortes mínimos $(X, \overline{X})$ e $(Y, \overline{Y})$ ocorre se todos os quatro conjuntos $X \cap Y$, $X \cap \overline{Y}$, $\overline{X} \cap Y$, $\overline{X} \cap \overline{Y}$ são não vazios.*

Um exemplo de interseção de cortes é ilustrado na Figura 2.14. Sejam os conjuntos $X = \{1, 4\}$, $\overline{X} = \{2, 3\}$, e os conjuntos $Y = \{1, 2\}$ e $\overline{Y} = \{3, 4\}$. De acordo com a Definição 22, os cortes $(X, \overline{X})$ e $(Y, \overline{Y})$ se intersectam, pois os quatro conjuntos $X \cap Y = \{1\}$, $X \cap \overline{Y} = \{4\}$, $\overline{X} \cap Y = \{2\}$, $\overline{X} \cap \overline{Y} = \{3\}$ são não vazios.

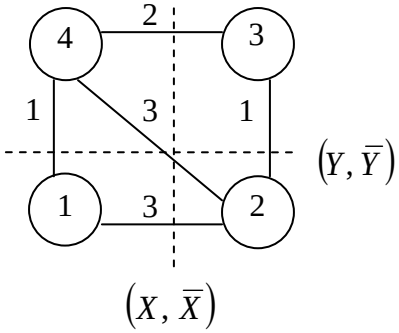


Figura 2.14: Exemplo de interseção de cortes.

O algoritmo criado por Gomory e Hu induz, por meio de processos de contrações de vértices, a formação de cortes mínimos que não se intersectam. Desta forma, eles mostraram que o fluxo máximo calculado entre dois nós não contraídos em uma rede contraída tem igual valor ao calculado na rede original. O corte mínimo encontrado na rede contraída também é corte mínimo na rede original, bastando substituir os super nós pelos nós que os compõem.

A árvore de cortes final é alcançada quando todos os super nós possuem apenas um nó cada, e isto acontece após $n-1$ resoluções do problema de fluxo máximo.

Em geral, é possível encontrar mais de uma árvore de cortes para uma mesma rede. A árvore de cortes apenas será única se todos os cortes (s - t) mínimos da rede forem únicos.

A Figura 2.15 apresenta as linhas gerais do algoritmo de Gomory e Hu (GH). Um exemplo de sua aplicação segue logo após.

Procedimento GH (*G*);

1 Construa *CT* criando um super nó com todos os nós de *G*;

2 **enquanto** existir em *CT* um super nó $\bar{v}$ com mais de um nó **faça**

3 Selecione arbitrariamente dois nós *s* e *t* de $\bar{v}$;

4 Contraia, em *G*, cada componente conexa de $CT \setminus \bar{v}$ em um super nó, gerando uma rede contraída *G'*;

5 Calcule um corte mínimo $(X, \bar{X})$ entre *s* e *t* em *G'*;

6 Divida $\bar{v}$ em dois super nós $\bar{v}_1$ e $\bar{v}_2$, tal que os nós de $\bar{v}_1$ estão em *X* e os nós de $\bar{v}_2$ estão em $\bar{X}$;

7 Conecte $\bar{v}_1$ e $\bar{v}_2$ com uma aresta de capacidade $c(X, \bar{X})$;

8 Conecte os super nós anteriormente adjacentes a $\bar{v}$, a $\bar{v}_1$ ou $\bar{v}_2$, dependendo de suas posições em $(X, \bar{X})$;

9 **finalize enquanto**

10 **retorne** *CT*;

Finalize GH;

Figura 2.15: Pseudocódigo do algoritmo de Gomory e Hu.

O algoritmo de Gomory e Hu será aplicado à rede *G* da Figura 2.16, na qual também está ilustrada a execução da linha 1 do algoritmo.

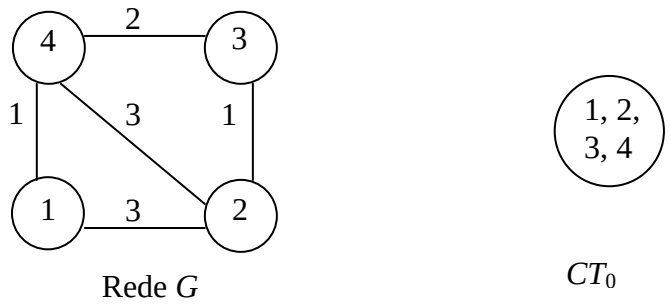


Figura 2.16: Rede *G* e execução da linha 1 do algoritmo GH.

Na linha 2 a condição do laço **enquanto** é analisada e inicia-se a primeira iteração. São escolhidos arbitrariamente os nós 2 e 4 (linha 3). Como é a primeira iteração do algoritmo, $CT_0 \setminus \bar{v}$ é vazio, e portanto, não se gera grafo contraído na linha 4 do algoritmo. O corte mínimo (2-4) em *G* tem capacidade 5 com *X* = {1,2} e $\bar{X}$ = {3, 4} (linha 5). A Figura 2.17 ilustra as linhas 5, 6 e 7 do algoritmo.

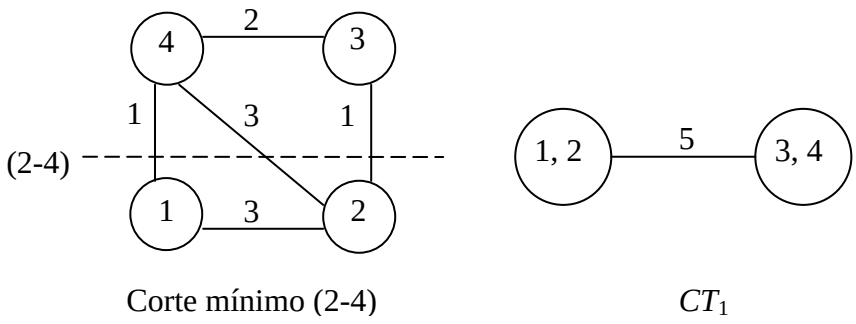


Figura 2.17: Execução das linhas 5, 6 e 7 do algoritmo GH.

A primeira iteração chega ao fim. A condição do laço **enquanto** é analisada novamente e inicia a segunda iteração. Escolhe-se arbitrariamente o super nó $\bar{v} = \{1, 2\}$ e consequentemente como *s* e *t* os nós 1 e 2, respectivamente (linha 3). Na linha 4, determina-se $CT_1 \setminus \bar{v}$, e identifica-se uma componente conexa com os nós 3 e 4 a ser contraída em *G*. Realiza-se o cálculo do corte mínimo entre os nós 1 e 2 na rede contraída (linha 5). O resultado é um corte mínimo (1 - 2) de capacidade 4 com $X = \{1\}$ e $\overline{X} = \{2, \{3, 4\}\}$. A Figura 2.18 ilustra a execução das linhas 4 e 5 do algoritmo.

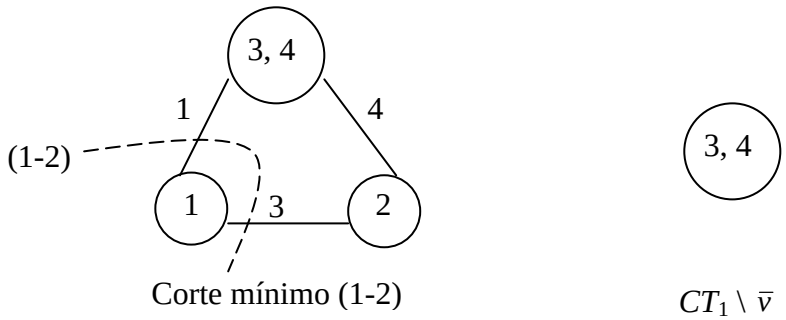
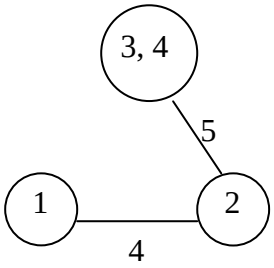


Figura 2.18: Execução das linhas 4 e 5 do algoritmo GH.

As linhas 6, 7 e 8 atualizam a árvore de cortes com o resultado obtido na linha 5. Os vértices 1 e 2, que pertenciam a um super nó, agora estão separados na nova árvore de cortes CT_2. A aresta que os conecta possui a capacidade do corte mínimo (1 - 2), ou seja, 4.

O super nó que contém os vértices 3 e 4 se encontra do mesmo lado do corte mínimo (1 - 2) que o vértice 2. Sendo assim, em CT_2 eles estarão conectados.

A Figura 2.19 ilustra a nova árvore de cortes.

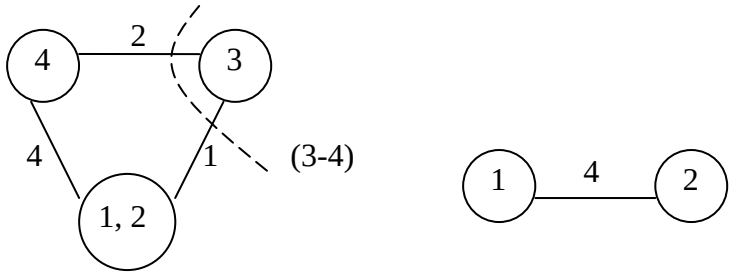


CT_2

Figura 2.19: Nova árvore de cortes CT_2 após a segunda iteração do laço **enquanto**.

A condição do laço **enquanto** é analisada novamente (linha 2) e uma terceira iteração se inicia, visto que na árvore de cortes ainda existe um super nó composto por mais de um nó, no caso, $\bar{v} = \{3, 4\}$.

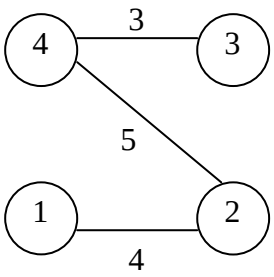
Vértices 3 e 4 são então selecionados como *s* e *t* (linha 3). A Figura 2.20 ilustra $CT_2 \setminus \bar{v}$ e a rede contraída após o processo de contração dos vértices 1 e 2. O corte mínimo (3 - 4), de capacidade 3 e com $X = \{3\}$ e $\overline{X} = \{4, \{1, 2\}\}$, também se encontra ilustrado.



Corte mínimo (3-4) $CT_2 \setminus \bar{v}$

Figura 2.20: Execução das linhas 4 e 5 da terceira iteração.

CT_2 é atualizada dividindo o super nó $\bar{v} = \{3, 4\}$ em dois super nós, estes conectados por uma aresta de capacidade 3. Como o super nó {1, 2} está contido em $\overline{X} = \{4, \{1, 2\}\}$, o super nó de vértice 2 se conectará, em CT_3, ao vértice 4. A Figura 2.21 ilustra CT_3.



CT_3

Figura 2.21: CT_3 e árvore de cortes final de G.

A linha 2 do algoritmo é executada novamente. Desta vez não há, na árvore de cortes, um super nó composto por mais de um nó, e o algoritmo chega ao fim. CT_3 é retornada.

Observe que no exemplo foram computados exatamente $n-1$ (no caso três) algoritmos de fluxo máximo/corte mínimo, nos quais os cortes mínimos não se intersectam. O valor do fluxo máximo entre quaisquer dois nós é a menor capacidade de aresta no caminho que os conecta na árvore. O valor de fluxo máximo, por exemplo, entre 2 e 3 é 3, entre 1 e 4 é 4, e entre 2 e 4 é 5.

3
Análise de sensibilidade de fluxos máximos multiterminal

Este capítulo apresentará a versão paramétrica do *problema de fluxo máximo multiterminal*, também chamada de *análise de sensibilidade de fluxos máximos multiterminal*. Esta consiste em estudar o comportamento do fluxo máximo entre todos os pares de nós de uma rede frente à variação de capacidade de arestas da rede.

O primeiro a lançar um método de resolução para este problema foi Elmaghraby [10]. Ele descreveu um método para analisar a influência nos fluxos máximos entre todos os pares de nós quando a capacidade de uma aresta da rede decresce linearmente.

O problema resolvido por Elmaghraby pode ser sintetizado como:

Problema 1. *Seja uma rede* $G = (V, E)$ *com a aresta* $e = (i, j) \in E$ *possuindo capacidade* $c(e) = \bar{c} - \varepsilon$, $0 \leq \varepsilon \leq \bar{c}$, *sendo* $\bar{c}$ *a capacidade inicial. O objetivo é determinar o fluxo máximo entre todos os pares de nós para todos os valores do parâmetro* ε.

Para cada par de vértices da rede, o valor do fluxo máximo entre eles, frente ao decrescimento de $c(e)$, se comporta dentro das três seguintes possibilidades:

1. Permanece constante para todo valor de $c(e)$. Isto ocorre quando, para qualquer valor de $c(e)$, e não pertence a um corte mínimo entre o par de vértices.

2. Varia linearmente com $c(e)$. Isto é válido quando, para $\varepsilon = 0$, um corte mínimo que separa o par de vértices contém a aresta e.

3. Permanece constante até um dado valor $c^*(e)$ de $c(e)$, quando então começa a variar linearmente com $c(e)$. A partir de $c^*(e)$, a aresta e está contida nos cortes mínimos que separam o par de vértices.

Definição 23. *Um valor c*(e) de c(e) que satisfaça o item 3 acima, em pelo menos um fluxo máximo, é chamado de* ***capacidade crítica****.*

Elmaghraby observa que, no intervalo entre duas capacidades críticas, nenhum fluxo máximo muda de comportamento, ou seja, só ocorrem as possibilidades dos itens 1 e 2 acima. O estudo da *análise de sensibilidade de fluxos máximos multiterminais* passa a ser, então, determinar todas as capacidades críticas, uma vez que, em cada intervalo, a construção de uma árvore de cortes fornece os fluxos máximos desejados. Desta forma, de acordo com Elmaghraby, é necessário calcular uma árvore de cortes para se obter cada capacidade crítica.

Berthomé et al. [11] aperfeiçoaram o resultado de Elmaghraby e demonstraram que, com apenas duas árvores de cortes, é possível determinar todas as capacidades críticas da aresta paramétrica. Para chegar a este resultado, analisaram o problema variando a aresta paramétrica crescentemente, ou seja, de zero a infinito.

Ao final, Berthomé et al. [11] expandiram o resultado acima para o caso em que várias arestas de uma rede possuem capacidade paramétrica. Este problema é conhecido como *fluxos paramétricos em rede multiterminal*. Para a situação em que k arestas de uma rede possuem capacidade paramétrica, os autores estabeleceram que 2^k computações de árvores de cortes são suficientes para se obter o valor do fluxo máximo entre quaisquer pares de vértices da rede.

3.1
Computando árvores de cortes dinamicamente

Com o objetivo de reduzir a complexidade dos algoritmos que necessitam calcular duas ou mais árvores de cortes em sequência, como o apresentado em Berthomé et al. [11], Barth et al. [12] mostraram como utilizar as informações de uma árvore de cortes já computada para construir a seguinte.

Seja $G = (V, E)$ uma rede com dois vértices s e t, respectivamente a origem e o destino. Seja uma aresta única $e = (i, j)$ com capacidade $c(e) = \lambda$ que varia de

acordo com o parâmetro $\lambda \geq 0$. Denota-se por $f_{s,t}^{\lambda}$ o valor do fluxo máximo entre *s* e *t* quando a capacidade da aresta *e* é λ.

Para uma rede que possui apenas uma aresta com capacidade paramétrica, a variação desta capacidade pode não influenciar o valor dos fluxos máximos (e dos cortes mínimos) para vários pares de vértices.

Lema 1 (Barth et al. [12]). *Sejam G uma rede e e = (i, j) uma aresta de G tal que c(e) =* λ*. Sejam s e t um par de vértices de G e* CT^{α} *uma árvore de cortes quando c(e) =* α*. Se os caminhos* $P_{s,t}$ *e* $P_{i,j}$ *em* CT^{α} *não possuírem aresta em comum, então* $f_{s,t}^{\lambda} = f_{s,t}^{\alpha}, \forall \lambda > \alpha \geq 0$.

Demonstração: Utilizando a propriedade de corte da árvore de cortes, existe um corte mínimo $C_{s,t}^{\alpha}$ separando *s* e *t* em que ambos os vértices *i* e *j* (*e* = (*i*, *j*)) se encontram no mesmo lado do corte mínimo. Consequentemente, o corte não contém *e* para $\lambda > \alpha$, e é insensível à variação de λ. □

Para exemplificar o Lema 1, considere a rede *G* da Figura 3.1. Em *G*, a aresta (1, 2) possui capacidade variável, e inicialmente $c(1, 2) = \alpha = 2$. CT^2 encontra-se ilustrada também na Figura 3.1. Sendo 5 e 6 os vértices *s* e *t*, logo $f_{5,6}^{\lambda} = f_{5,6}^{2}, \forall \lambda > 2 \geq 0,$ pois os caminhos $P_{5,6}$ e $P_{1,2}$ em CT^2 não possuem aresta em comum. No caso, $f_{5,6}^{2} = 3$.

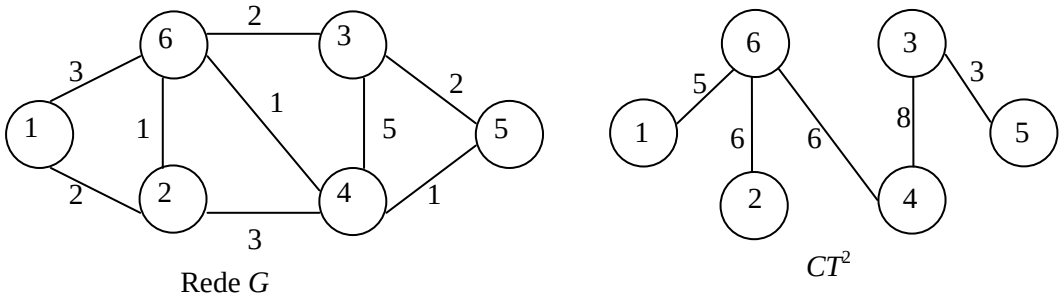
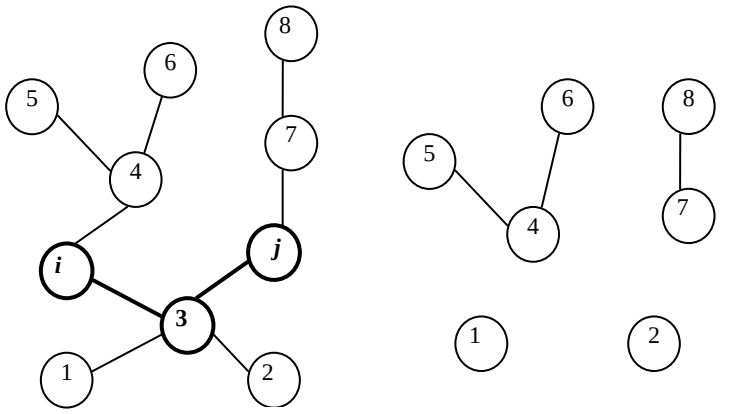


Figura 3.1: Exemplificação do Lema 1.

Para as próximas inferências, a seguinte definição se faz necessária:

Definição 24. *Seja G = (V, E) uma rede conexa e acíclica, ou seja, uma árvore. Sejam i e j um par de vértices de G e* $P_{i,j}$ *o (único) caminho que os conecta. A* ***floresta decomposta - (i, j)*** de *G, denotada como* $F_{i,j}$*, é o conjunto de árvores remanescentes após a exclusão de* $P_{i,j}$.

A Figura 3.2 ilustra a Definição 24. Considere a árvore *G* e o caminho i-3-j em *G* à esquerda da Figura 3.2. Então, a floresta decomposta - (*i, j*) de *G* é formada pelas quatro subárvores à direita.



Árvore *G* Floresta decomposta - (*i, j*) de *G*

Figura 3.2: Ilustração de uma árvore *G* e sua floresta decomposta-(*i, j*).

Lema 2 (Barth et al. [12]). *Seja G uma rede com uma aresta e = (i, j) de capacidade paramétrica. Seja* CT^{α} *a árvore de cortes quando c(e) =* α*. Seja* $F_{i,j}$ *a floresta decomposta - (i, j) de* CT^{α}*. Para cada árvore* $T \in F_{i,j}$*, existe uma árvore de cortes de G com c(e) =* $\lambda > \alpha$ *que contém* T *como subárvore.*

O Lema 2 pode ser demonstrado aplicando o algoritmo de Gomory e Hu da seguinte forma. Primeiramente, sejam $s_1 = x$ um nó do caminho $P_{i,j}$ e $t_1 = y$ um nó adjacente a *x* não contido em $P_{i,j}$, ambos pertencendo ao mesmo super nó. Seja T_y a maior subárvore de CT^{α} enraizada em *y* não contendo *x*. O caminho P_{s_1,t_1} em CT^{α}, reduzido à aresta [*x, y*], não possui aresta em comum com $P_{i,j}$. Assim, do Lema 1, o corte mínimo entre s_1 e t_1 pode ser obtido de CT^{α}. A partir desta escolha, os super nós resultantes são SV_1 com todos os elementos de T_y e

SV_2 com os outros elementos da rede. Continuando o algoritmo de Gomory e Hu para o super nó SV_1 e sempre obtendo os cortes mínimos de CT^{α}, ao final, teremos reconstruído toda a estrutura de T_y.

A Figura 3.3 exemplifica o Lema 2. A rede *G* possui uma aresta paramétrica, a aresta (1, 6). Inicialmente, $c(1, 6) = 1$, e a rede e sua árvore de cortes são nomeadas G^1 e CT^1, respectivamente. Na ilustração de CT^1, encontra-se marcada, em linhas grossas, a floresta decomposta - (1, 6), ou somente $F_{1,6}$, que é composta por uma árvore que contém os nós 3, 5, 7, 8 e 9. Em seguida, tem-se $c(1, 6) = 4$, gerando G^4 e CT^4. Pode-se observar que CT^4 contém a única árvore de $F_{1,6}$, ou seja, $F_{1,6}$ é uma subárvore de CT^4.

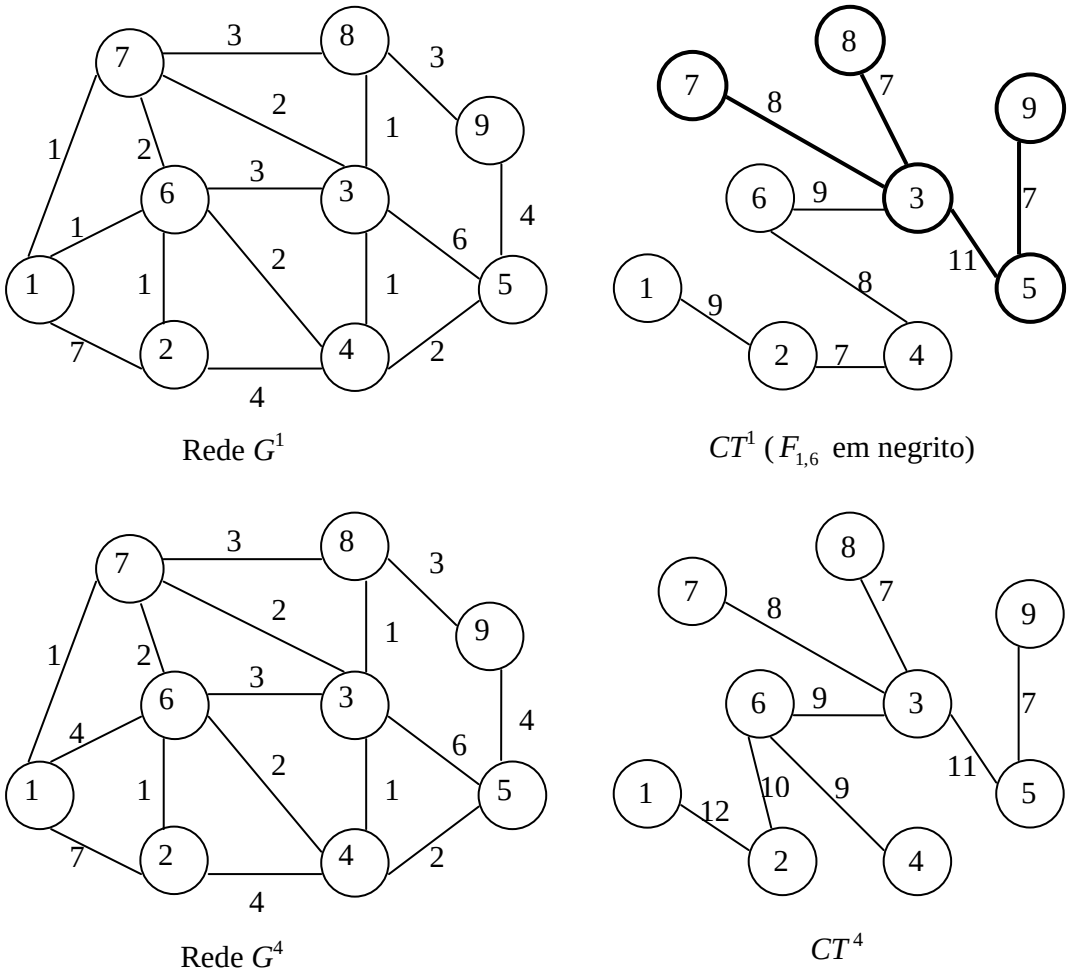


Figura 3.3: Exemplo do Lema 2. Aresta (1, 6) variou de capacidade.

Como nomenclatura, para os próximos teoremas, denota-se $|P_{i,j}|$ como o número de arestas em $P_{i,j}$.

Teorema 2 (Barth et al. [12]). *Seja G uma rede com uma aresta e = (i, j) de capacidade paramétrica c(e) =* λ*. Seja* CT^{α} *uma árvore de cortes obtida quando c(e) =* α*. Seja* $P_{i,j}$ *o caminho entre i e j em* CT^{α}*. Para* $\lambda > \alpha$ *é suficiente calcular* $|P_{i,j}|$ *cortes mínimos em* G^{λ} *a fim de obter a árvore de cortes* CT^{λ}.

A demonstração do Teorema 2 está baseada no algoritmo de Gomory e Hu. O ponto principal é escolher os nós *s* e *t* durante o algoritmo de forma a reconstruir, em primeiro lugar, de acordo com o Lema 2, todas as árvores T de $F_{i,j}$, sem executar, sequer, um algoritmo de corte mínimo. Neste momento, a árvore intermediária encontra-se estruturada com um super nó composto por todos os elementos do caminho $P_{i,j}$ ao qual todas as subárvores T estão conectadas. A partir deste ponto, o algoritmo de Gomory e Hu processará normalmente, executando, então, $|P_{i,j}|$ cortes mínimos.

Relacionando o Teorema 2 com a Figura 3.3, pode-se observar que, para se obter a árvore de cortes CT^4, de posse de CT^1, são necessários apenas três cálculos de fluxo máximo em G^4. Isto ocorre pois $P_{1,6}$ em CT^1 possui apenas três arestas. Além disso, os três cálculos podem ser realizados em uma G^4 contraída, ilustrada na Figura 3.4, conforme a demonstração do Teorema 2.

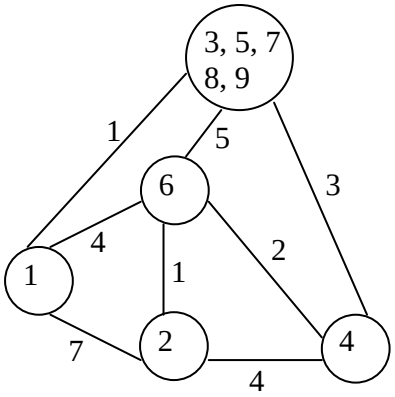


Figura 3.4: Rede G^4 contraída.

Uma extensão do Teorema 2 de Barth et al. [12], para o caso de várias arestas de capacidades paramétricas, é apresentada aqui pelo Teorema 3.

Teorema 3. *Seja G uma rede com k arestas* $e_x = (i_x, j_x)$*, onde* $x = 1, 2, ..., k$*, de capacidades paramétricas* $c(e_x) = \lambda_x$*. Sejam* CT^{α} *uma árvore de cortes obtida quando* $c(e_x) = \alpha_x$ *e* $P_{i,j}^{x}$ *o caminho entre* i_x *e* j_x *em* CT^{α}*. Para* $\lambda_x > \alpha_x$ *é suficiente calcular* $|\bigcup_{x=1}^{k} P_{i,j}^{x}|$ *cortes mínimos em* G^{λ} *a fim de obter a árvore de cortes* CT^{λ}*.*

Demonstração: De acordo com o Lema 1 de Barth et al. [12], as variações crescentes das arestas e_x não influenciarão na árvore de cortes CT^{α}, exceto para a união de seus caminhos $P_{i,j}^{x}$. □

Aplicar o resultado do Teorema 3, ao invés do resultado do Teorema 2, para obter a nova árvore de cortes, quando duas ou mais arestas aumentam de capacidade, pode ser mais eficiente. Considere a rede da Figura 3.5 como sendo uma árvore de cortes de uma determinada rede *G*. Suponha então que as arestas (1, 4) e (1, 6) de *G* aumentam de capacidade. De acordo com o Teorema 3, são necessárias três execuções de fluxo máximo para se obter a nova árvore de cortes, visto que a união dos caminhos $P_{1,4}$ e $P_{1,6}$ é o próprio $P_{1,6}$. Caso apliquemos o Teorema 2, o número de execuções de fluxo máximo pode variar de três a seis.

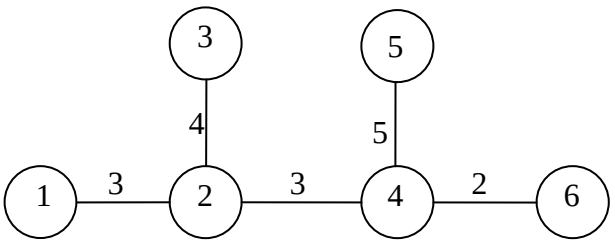


Figura 3.5: Árvore de cortes de uma rede *G*.

A seguir, uma observação sobre adição de arestas:

- Afirmar que uma dada aresta não está contida em uma rede, equivale a afirmar que esta aresta está contida na rede e possui capacidade nula. Portanto, adicionar arestas a uma rede é o caso de variação crescente de capacidade.

Para os algoritmos que geram uma árvore de cortes a partir de informações de uma existente, esse estudo, até aqui, apenas considerou os casos de adição de arestas ou variação crescente da capacidade das mesmas. Para o caso de variação decrescente, e de forma semelhante, exclusão de arestas, Hartmann & Wagner [13] afirmaram:

Teorema 4 (Hartmann & Wagner [13]). *Seja G uma rede com uma aresta e = (i, j) de capacidade paramétrica c(e) =* λ*. Seja* CT^{α} *uma árvore de cortes obtida quando c(e) =* α*. Seja* $P_{i,j}$ *o caminho entre i e j em* CT^{α}*. Para* $0 \leq \lambda < \alpha$ *os cortes mínimos que pertencem a* $P_{i,j}$ *permanecem válidos em* CT^{λ} *com capacidade decrescida de* $(\alpha - \lambda)$*.*

Demonstração: A aresta *e* está contida em todos os cortes mínimos de $P_{i,j}$, e somente neles em CT^{α}. Como *c*(*e*) decresce $(\alpha - \lambda)$, todos os cortes de *G* decrescem no máximo $(\alpha - \lambda)$, logo os cortes mínimos de $P_{i,j}$ permanecem válidos em CT^{λ} com capacidade decrescida de $(\alpha - \lambda)$. □

Cabe notar que, no Teorema 4, não são as arestas de $P_{i,j}$ que permanecem válidas, e sim os cortes mínimos representados por elas. Utilizando novamente a árvore de cortes da Figura 3.5, suponha agora que a aresta (1, 4) de *G* diminui de capacidade em uma unidade. Conforme o Teorema 4, os cortes mínimos representados pelas arestas [1, 2] e [2, 4] da árvore de cortes, nomeados respectivamente C_1 e C_2, podem ser reutilizados, com capacidade decrescida de um, na nova árvore de cortes. A Figura 3.6 ilustra uma possível nova árvore de cortes, em que C_1 e C_2 estão representados pelas arestas [1, 2] e [3, 4], respectivamente.

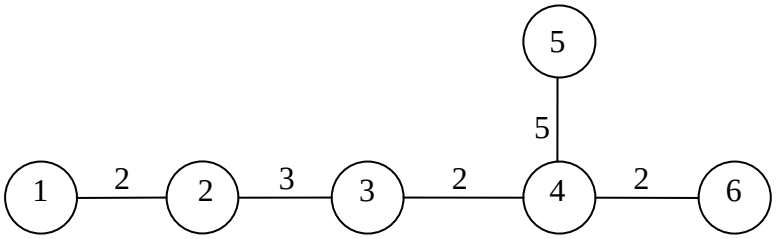


Figura 3.6: Nova árvore de cortes após decréscimo de 1 na capacidade da aresta (1, 4).

Uma extensão ao Teorema 4 de Hartmann & Wagner [13] é definida no Teorema 5.

Teorema 5. *Seja G uma rede com k arestas* $e_x = (i_x, j_x)$ *onde* $x = 1, 2, ..., k$, *de capacidades paramétricas* $c(e_x) = \lambda_x$. *Seja* CT^{α} *uma árvore de cortes obtida quando* $c(e_x) = \alpha_x$. *Seja* $P_{i,j}^{x}$ *o caminho entre* i_x *e* j_x *em* CT^{α}. *Para* $0 \leq \lambda_x < \alpha_x$ *os cortes mínimos que pertencem a* $\bigcap_{x=1}^{k} P_{i,j}^{x}$ *permanecem válidos em* CT^{λ} *com capacidade decrescida de* $\sum_{x=1}^{k}(\alpha_x - \lambda_x)$.

Demonstração: Seguindo a demonstração do Teorema 4, com os decréscimos de $c(e_x)$, todos os cortes de *G* decrescem no máximo $\sum_{x=1}^{k}(\alpha_x - \lambda_x)$, logo os cortes mínimos de $\bigcap_{x=1}^{k} P_{i,j}^{x}$ permanecem válidos em CT^{λ} com capacidade decrescida de $\sum_{x=1}^{k}(\alpha_x - \lambda_x)$. □

Enquanto o Teorema 3 utiliza a operação de conjuntos união, o Teorema 5 faz uso da operação interseção. Seja a rede da Figura 3.7 a árvore de cortes de uma rede *G*. Suponha que as arestas (1, 4) e (2, 6) de *G* têm suas capacidades decrescidas em uma unidade. Para construir a nova árvore de cortes, segundo o Teorema 5, podemos reutilizar o corte mínimo representado pela aresta [2, 4] da árvore de cortes, visto que esta aresta é resultado da interseção entre os caminhos

$P_{1,4}$ e $P_{2,6}$. Observe que, na nova árvore de cortes, o corte reutilizado terá capacidade 1.

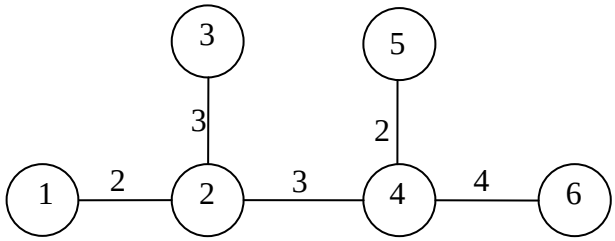


Figura 3.7: Árvore de cortes de uma rede *G*.

Hartmann & Wagner [13] também determinaram outros cortes reutilizáveis para o caso de variação decrescente.

Lema 3 (Hartmann & Wagner [13]). *Seja G uma rede com uma aresta e* = (*i, j*) *de capacidade paramétrica c*(*e*) = λ. *Sejam* CT^{α} *uma árvore de cortes obtida quando c*(*e*) = α *e* $(X, \overline{X})$ *um corte* (*s* - *t*) *mínimo em* G^{λ} *para* $0 \leq \lambda < \alpha$. *Sejam* $i, j \in \overline{X}$ *e a aresta* $[g,h] \in CT^{\alpha}$ *com* $g,h \in X$. *Então* [*g, h*] *é um corte* (*g* - *h*) *mínimo em* G^{λ}.

Demonstração: Suponha a existência de um corte (*g* - *h*) mínimo em G^{λ} com capacidade menor que o corte representado por [*g, h*]. Note que o corte [*g, h*] possui a mesma capacidade em G^{α} e G^{λ}. Tal corte (*g* - *h*) mínimo de capacidade menor separaria, em G^{λ}, *i* e *j* em $\overline{X}$. Ao mesmo tempo, uma vez que o algoritmo de Gomory e Hu induz cortes mínimos que não se intersectam, existe um corte (*g* - *h*) mínimo em G^{λ} que não separa *i* e *j*. Este último, portanto, já possuía capacidade menor em G^{α}. □

Para exemplificar o Lema 3, considere a rede *CT* da Figura 3.8 como sendo uma árvore de cortes de uma rede *G*. Uma alteração decrescente de capacidade ocorre na aresta (1, 4) de *G*, gerando uma nova rede *G´*. Assumindo que o corte mínimo representado pela aresta [4, 5] de *CT* permanece mínimo em *G´*, pode-se concluir, baseado no Lema 3, que os cortes mínimos representados

em *CT* pelas arestas [5, 6] e [5, 7] permanecem válidos para a árvore de cortes de *G´*.

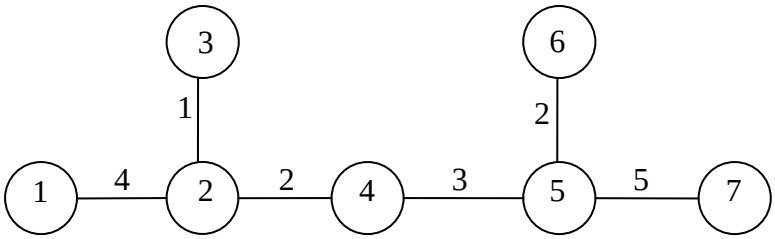


Figura 3.8: Árvore de cortes *CT* de uma rede *G*.

No Lema 4, o Lema 3 é reescrito para considerar variações decrescentes em mais de uma aresta.

Lema 4. *Seja G uma rede com k arestas* $e_x = (i_x, j_x)$*, onde* $x = 1, 2, ..., k$*, de capacidades paramétricas* $c(e_x) = \lambda_x$*. Sejam* CT^{α} *uma árvore de cortes obtida quando* $c(e_x) = \alpha_x$ *e* $P_{i,j}^{x}$ *o caminho entre* i_x *e* j_x *em* CT^{α}*. Sejam* $(X, \overline{X})$ *um corte* (*s* - *t*) *mínimo em* G^{λ} *para* $0 \leq \lambda_x < \alpha_x$ *e y o número de arestas paramétricas que têm* $i_x, j_x \in \overline{X}$*. Sejam a aresta* $[g,h] \in CT^{\alpha}$ *com* $g,h \in X$ *e o conjunto* $L = \{x : [g,h] \in P_{i,j}^{x}\}$*. Se* $(k - y) = |L|$*, então* [*g, h*] *é um corte* (*g* - *h*) *mínimo em* G^{λ} *com capacidade decrescida de* $\sum_{x \in L}(\alpha_x - \lambda_x)$.

Demonstração: Podemos demonstrar o Lema 4 através do algoritmo de Gomory e Hu. Sejam $(X, \overline{X})$ o primeiro corte (*s* - *t*) mínimo encontrado pelo algoritmo e *Y* o conjunto de arestas paramétricas com extremidades $i_x, j_x \in \overline{X}$. A partir da segunda iteração em diante, para $s,t \in X$, todos os cortes (*s* - *t*) mínimos serão calculados em redes contraídas que não contêm arestas de *Y*. Portanto, baseado na prova do Teorema 5, se um corte (*s* - *t*) mínimo *C* em CT^{α} contêm todas arestas paramétricas de *G*, exceto aquelas em *Y*, então *C* permanece válido em CT^{λ} com capacidade decrescida de $\sum_{x \in L}(\alpha_x - \lambda_x)$. □

Seja a rede *CT* da Figura 3.9 uma árvore de cortes de uma rede *G*. Como aplicação do Lema 4, suponha que as arestas (1, 4) e (2, 7) de *G* tenham suas capacidades diminuídas em uma unidade, resultando em uma nova rede *G´*. Assumindo que o corte mínimo C_1 representado pela aresta [4, 5] de *CT* permanece mínimo em *G´*, pode-se concluir, baseado no Lema 4, que o corte mínimo C_2 representado em *CT* pela aresta [5, 7], permanece válido para a árvore de cortes de *G´*. Note que, em *G´*, C_1 e C_2 possuem capacidade 4 e 2, respectivamente.

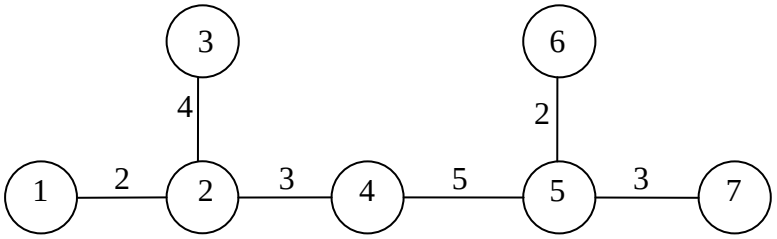


Figura 3.9: Árvore de cortes *CT* de uma rede *G*.

3.2
Propriedades de nós de corte

Nesta seção, apresentamos uma propriedade que relaciona nós de corte e fluxos multiterminais, permitindo-nos computar árvores de cortes de uma forma alternativa, explorando a presença de nós de corte em uma rede. O próximo lema ajudará a provar a propriedade enunciada no Lema 6.

Lema 5. *Se dois nós pertencem a uma componente biconexa A, o fluxo máximo entre eles pode ser calculado em A.*

Prova: Se *s* e *t* são dois nós em *A*, não existe caminho entre *s* e *t* que contenha um nó que não se encontra em *A*. □

Lema 6. *A árvore de cortes de um grafo é a união das árvores de cortes de suas componentes biconexas.*

Prova: Seja *G* um grafo e *A* sua única componente biconexa na qual é o próprio grafo. Sejam *CT* e CT_A as árvores de cortes de *G* e *A*, respectivamente. Adicionando aresta após aresta, e os respectivos nós de suas extremidades, em *G*, pode-se criar uma componente biconexa *B* que compartilha o nó *z* com *A*. Neste processo, seja $e = (i, j)$ a aresta adicionada em cada etapa e $P_{i,j}$ o caminho entre *i* e *j* em *CT*. Uma vez que, em cada etapa, $P_{i,j}$ não possui aresta em comum com CT_A, de acordo com os Lemas 1 e 2, os cortes em CT_A não são influenciados pelo processo e podem estar contidos na *CT* final. Para concluir, baseado no Lema 5, os nós de *A* encontram-se no mesmo lado que *z* em todos os cortes mínimos entre nós de *B*, o que conduz ao resultado de que CT_A pode ser uma subárvore da *CT* final. □

A seguir demonstramos um exemplo da prova do Lema 6. Considere a rede *G* e sua árvore de cortes *CT* ilustradas na Figura 3.10.

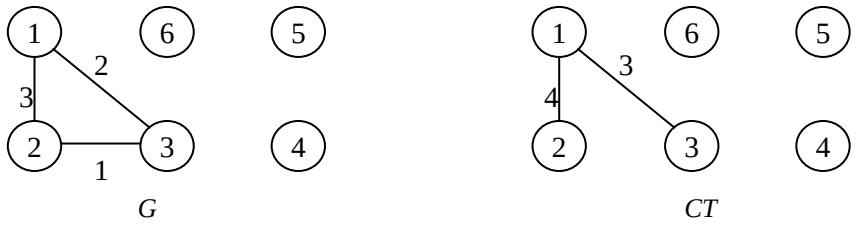


Figura 3.10: Rede *G* (esquerda) e sua árvore de cortes *CT* (direita).

De acordo com o Lema 1, se adicionarmos à rede *G* as arestas (3, 6), (3, 5), (3, 4), (4, 5) e (5, 6), uma por uma, estas adições não influenciariam os cortes em *CT* representados pelas arestas [1, 2] e [1, 3]. Além disso, a nova *CT*, conforme o Lema 2, poderia conter estes cortes. Por último, segundo o Lema 5, o nó 3 será adjacente ao nó 1 na nova *CT*. A Figura 3.11 ilustra a nova rede e sua nova *CT*.

Nova *G* Nova *CT*

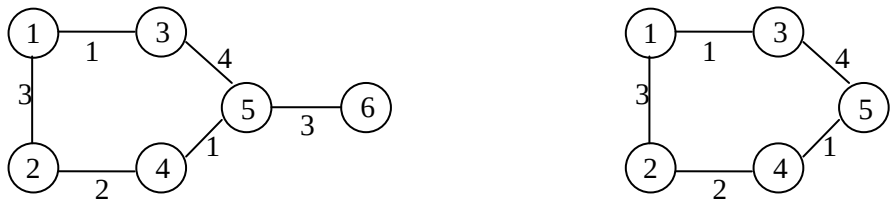
Figure 3.11: Nova rede *G* (esquerda) e sua nova árvore de cortes *CT* (direita).

No próximo capítulo, um algoritmo aqui desenvolvido com base no Lema 6 será apresentado juntamente com experimentos computacionais com o mesmo.

4
Algoritmo para utilizar nós de corte

Baseado no Lema 6, se uma rede possui nós de corte, pode-se resolver o problema do fluxo máximo multiterminal computando a árvore de cortes de cada uma de suas componentes biconexas e reconectando-as no final. A seguir, o método proposto, denotado CN (*cut node*), será descrito.

Para calcular o fluxo máximo entre todos os pares de nós de uma rede $G = (V, E)$ com n nós, não direcionada e com capacidades nas arestas, CN identifica as componentes biconexas e em seguida realiza um teste. Se não existir uma componente biconexa com mais nós do que 80% de n, então o algoritmo aplica o método de Gusfield [7] em cada componente biconexa. Observe que esta condição é diferente da de apenas possuir um nó de corte. Ela evita a situação ilustrada na Figura 4.1, onde uma componente biconexa possui tamanho quase igual ao da rede, o que impossibilita compensar o *overhead* do gerenciamento de componentes biconexas com cálculos de fluxo máximo em redes menores que a original. A explicação para a escolha da porcentagem de 80% será abordada no Subcapítulo 4.1. Por último, todas as árvores de cortes das componentes biconexas são unidas, resultando na árvore de cortes de G. Caso contrário, se existir uma componente biconexa com mais nós que 80% de n, o método de Gusfield é aplicado em G. Observe que, uma rede sem nós de corte possui uma componente biconexa, a própria rede. A Figura 4.2 exibe um pseudocódigo do algoritmo CN.



Rede G Componente biconexa de G

Figura 4.1: Rede G (esquerda) e sua maior componente biconexa (direita).

Procedimento CN (*G*);

1 Identifique todas as componentes biconexas em *G*;

2 **se** não existir componente biconexa com mais do que (0,8**n*) nós **então**

3 Aplique o algoritmo de Gusfield a cada componente biconexa separadamente;

4 Junte as árvores de cortes das componentes biconexas em uma só;

5 **senão**

6 Aplique o algoritmo de Gusfield a *G*;

7 **retorne** *CT*;

Finalize CN;

Figura 4.2: Pseudocódigo do algoritmo CN.

Em nossa implementação de CN, utilizamos o algoritmo de Gusfield para construir as árvores de cortes devido à sua simplicidade, mas pode-se também aplicar o algoritmo de Gomory e Hu. Além disso, o procedimento usado para identificar as componentes biconexas na rede foi baseado no algoritmo de Hopcroft & Tarjan [22]. Ainda, o algoritmo de pré-fluxo de maior etiqueta, de Goldberg & Tarjan [23], foi escolhido para calcular os fluxos máximos dentro do algoritmo de Gusfield.

A fim de ilustrar o método proposto, segue um exemplo de sua aplicação. Seja *G* a rede da Figura 4.3.

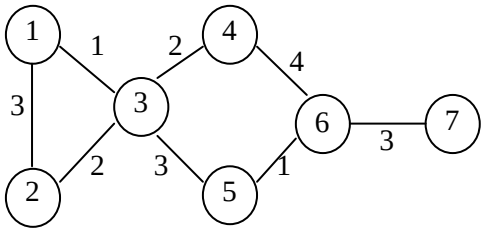


Figura 4.3: Rede *G*.

Primeiramente, o algoritmo acha os nós de corte 3 e 6 em *G*. Em seguida, identifica três componentes biconexas em *G* (linha 1), que são exibidas na Figura 4.4.

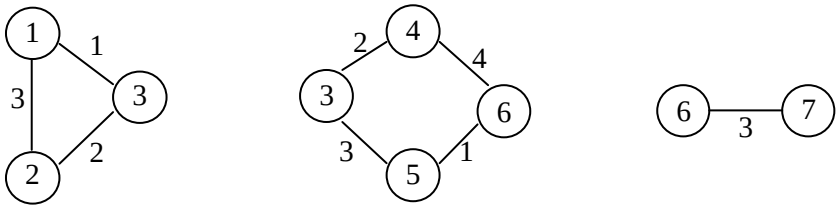


Figura 4.4: Componentes biconexas de *G*.

Como nenhuma componente biconexa de *G* possui mais do que cinco nós (linha 2), o algoritmo calcula, através do método de Gusfield, a árvore de cortes de cada componente biconexa (linha 3). As respectivas árvores de cortes estão ilustradas na Figura 4.5.

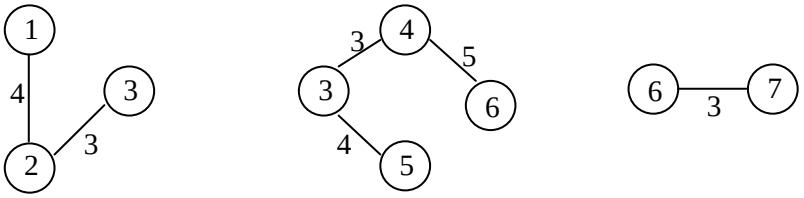


Figura 4.5: Árvores de cortes das componentes biconexas de *G*.

Por último, todas as árvores de cortes das componentes biconexas são unidas para formar a árvore de cortes de *G* (linha 4), como está na Figura 4.6.

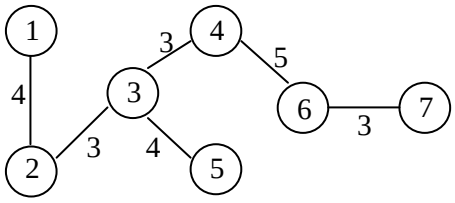


Figura 4.6: Árvore de cortes de *G*.

4.1
Experimentos computacionais

Nesta seção, nós relatamos experimentos computacionais para definir a condição da linha 2 de CN e comparar o algoritmo CN, proposto aqui, com o algoritmo de Gusfield (GUS) implementado por Skorobohatyj [24].

Os dois algoritmos foram implementados em linguagem C com o formato de arquivo de entrada e a estrutura de dados tal como em Skorobohatyj [24]. Os

algoritmos foram compilados com Mingw através do software Dev-C++ 5.10 e os testes foram executados em um computador 64-bit de processador Intel (R) Core (TM) i3 de frequência 3.50 GHz, 4 GB de memória RAM em um sistema operacional Windows 8.

Para os experimentos computacionais, as instâncias de teste foram criadas pelos geradores PATHGEN e TREEGEN, de Goldberg & Tsioutsiouliklis [8], e PARTEDGEN, CACTUSGEN e TESTGEN, especialmente desenvolvidos para este experimento. As características desejáveis das instâncias de teste são: a presença de nós de corte ou possibilidade elevada de possuir nós de corte nos grafos gerados.

Os seguintes parâmetros são utilizados pelos cinco geradores: n o número de nós no grafo, d a densidade do grafo fornecida em termos de porcentagem, P a capacidade das arestas, e S a semente de aleatoriedade do gerador. Como segue, nós os apresentamos brevemente.

Dado o comprimento do caminho (parâmetro k), PATHGEN constrói um caminho de $k-1$ arestas e conecta os remanescentes $n-k$ nós aos nós do caminho aleatoriamente. Em seguida, adiciona arestas aleatoriamente para alcançar o número desejado de arestas e tornar a instância mais difícil em relação ao problema do corte mínimo. Para PATHGEN, o valor de k determina a forma do caminho. Por exemplo, se $k = n$ então se obtém um caminho através de todos os nós; se $k = 1$, então o caminho é uma estrela.

Dado a forma da árvore (parâmetro k), o gerador TREEGEN constrói uma árvore conectando o nó i, $i = 2,..., n$, a um nó aleatoriamente escolhido em $\{1, \min\{i-1, k\}\}$. Em seguida, adiciona arestas aleatoriamente para alcançar o número desejado de arestas e tornar os problemas de corte mínimo mais difíceis. O valor de k determina a forma da árvore. Por exemplo, se $k = 1$ então a árvore é uma estrela. Se $k = n-1$, a árvore é obtida conectando cada nó, exceto o primeiro, a um nó precedente escolhido aleatoriamente.

Dado o número de componentes biconexas (parâmetro k), PARTEDGEN constrói um grafo com $k-1$ nós de corte e componentes biconexas de mesmo tamanho. Após construir um caminho através de todos os nós no primeiro passo, adiciona arestas estratégicas para criar k ciclos disjuntos em arestas de dimensão aproximadamente n/k. Por último, adiciona arestas aleatoriamente, em cada componente biconexa, para alcançar o número desejado de arestas.

Para explicar CACTUSGEN, a seguinte definição é necessária.

Definição 25. *Um grafo conexo onde quaisquer dois ciclos distintos possuem no máximo um nó em comum é um* ***grafo cacto***.

Dado o número de ciclos (parâmetro *k*), o gerador CACTUSGEN constrói um grafo cacto com *k* ciclos. Dois tipos do gerador foram criados: CACTUS_PATHGEN e CACTUS_STARGEN. No primeiro, um caminho através de todos os nós é construído e, em seguida, *k* arestas são adicionadas para formar *k* ciclos disjuntos em arestas. No segundo, os *k* ciclos possuem o mesmo tamanho e um nó em comum. Em ambos, o parâmetro de densidade *d* não é considerado, uma vez que *k* define o número de arestas *m*.

Dado o tamanho de uma componente biconexa (parâmetro *k*), TESTGEN constrói um grafo em que uma de suas componentes biconexas possui tamanho aproximado de $n * k$, sendo *k* uma porcentagem. Após construir um caminho através de todos os nós no primeiro passo, adiciona uma aresta estratégica para criar um ciclo de dimensão aproximadamente $n * k$. Por último, marca os nós do ciclo com a cor 1 e o restante dos nós com a cor 2, e adiciona arestas aleatoriamente, conectando nós de mesma cor, para alcançar o número desejado de arestas.

Nos experimentos computacionais, são consideradas até três sementes distintas (S = 1, 2, 3), exceto para a geração das instâncias PARTED. As capacidades das arestas são escolhidas segundo uma distribuição uniforme de probabilidade a partir do intervalo [1, ..., 100P]. P = 1 é definido para todas as instâncias geradas assim como n = 1000. Este valor de n foi estabelecido por ser relativamente grande em relação aos utilizados em Goldberg & Tsioutsiouliklis [8].

Para estimar o tamanho máximo de uma componente biconexa, na condição da linha 2 do algoritmo CN, foram realizados testes computacionais, em instâncias da família TEST, com duas variações do algoritmo CN definidas a seguir:

- CN_0: CN implementado com a dimensão (0,0*n) na condição da linha 2, ou seja, para esta variante, a condição nunca é satisfeita.

- CN_1: CN implementado com a dimensão (1,0**n*) na condição da linha 2, ou seja, para esta variante, a condição sempre é satisfeita.

Os tempos de execução obtidos por CN_0 e CN_1 estão relatados na Tabela 1. Os tempos de execução obtidos pelos algoritmos CN e GUS para as instâncias geradas por PARTEDGEN e PATHGEN estão resumidos nas Tabelas 2 e 3, respectivamente, enquanto que, para as instâncias geradas por TREEGEN, os resultados estão na Tabela 4. Tabelas 5 e 6 expõem os tempos de execução de CN e GUS para as instâncias CACTUS_PATH e CACTUS_STAR, respectivamente. Para cada instância, o tempo de execução se refere a mediana de cinco execuções do algoritmo dado em microssegundos (μs). O símbolo * significa que a instância possui uma componente biconexa com mais de 80% de seus nós.

Tabela 1. Tempo de execução dos algoritmos CN_0 e CN_1 para as instâncias TEST.

TEST (*n* = 1000, *m* = 3497)								
	***k* = 99**	***k* = 95**	***k* = 90**	***k* = 85**	***k* = 80**		***k* = 75**	
					***S* = 1**	***S* = 2**	***S* = 1**	***S* = 2**
CN_0	120440	123423	121045	115754	123865	120208	115726	116518
CN_1	195909	184355	158692	146726	133504	128467	119713	115259

Tabela 2. Tempo de execução para as instâncias PARTED com *n* = 1000, *m* = 3497 e *k* = 2, 4, 8, 16.

PARTED (*n* = 1000, *m* = 3497)				
	***k* = 2**	***k* = 4**	***k* = 8**	***k* = 16**
CN	71204	29685	15059	8394
GUS	97514	95450	94860	94339

Tabela 3. Tempo de execução para as instâncias PATH com *n* = 1000, *m* = 1399, *k* = 250, 500, 750 e *S* = 1, 2, 3.

PATH (*n* = 1000, *m* = 1399)									
	***k* = 250**			***k* = 500**			***k* = 750**		
	***S* = 1**	***S* = 2**	***S* = 3**	***S* = 1**	***S* = 2**	***S* = 3**	***S* = 1***	***S* = 2***	***S* = 3***
CN	38200	35789	35667	49642	49253	50138	60114	61432	59639
GUS	55046	54261	55628	58469	58095	58162	57825	58748	57192

Tabela 4. Tempo de execução para as instâncias TREE com *n* = 1000, *m* = 1549, *k* = 250, 500, 750 e *S* = 1, 2, 3.

TREE (*n* = 1000, *m* = 1549)									
	***k* = 250**			***k* = 500**			***k* = 750**		
	***S* = 1**	***S* = 2**	***S* = 3**	***S* = 1**	***S* = 2***	***S* = 3**	***S* = 1***	***S* = 2***	***S* = 3***
CN	47976	45278	47024	50595	60095	55410	60904	62053	60074
GUS	58001	57821	56908	58532	57835	59569	58825	59864	58519

Tabela 5. Tempo de execução para as instâncias CACTUS_PATH com *n* = 1000, *k* = 10, 20 e *S* = 1, 2, 3.

CACTUS_PATH (*n* = 1000)						
	***k* = 10 (*m* = 1009)**			***k* = 20 (*m* = 1019)**		
	***S* = 1**	***S* = 2**	***S* = 3**	***S* = 1**	***S* = 2**	***S* = 3**
CN	8901	9047	8905	4812	4867	4777
GUS	68897	67814	70287	59783	57929	61114

Tabela 6. Tempo de execução para as instâncias CACTUS_STAR com *n* = 1000, *k* = 10, 20 e *S* = 1, 2, 3.

CACTUS_STAR (*n* = 1000)						
	***k* = 10 (*m* = 1009)**			***k* = 20 (*m* = 1019)**		
	***S* = 1**	***S* = 2**	***S* = 3**	***S* = 1**	***S* = 2**	***S* = 3**
CN	9079	9090	9261	5313	5246	5320
GUS	69456	65624	65816	57762	55744	56923

Analisando os resultados, chegou-se às seguintes conclusões:

1. O único valor de *k*, nas instâncias TEST, em que CN_1 possui desempenho similar a CN_0, é *k* = 75. Por esta razão, a condição da linha 2 de CN foi fixada em 80%;
2. Para as instâncias PARTED, quando o parâmetro *k* aumenta, o desempenho de CN se torna muito melhor que o de GUS;
3. Em ambas PATH e TREE, quando o parâmetro *k* aumenta, o desempenho de CN diminui;

4. O tempo de execução de CN para PATH foi até 35% (k = 250, S = 3) mais baixo que o de GUS. Para TREE, o tempo de execução de CN foi até 21% (k = 250, S = 2) mais baixo que o de GUS;
5. Para as instâncias geradas com uma componente biconexa com mais nós que 0,8n, CN tem desempenho muito próximo ao do algoritmo de GUS, como esperado, visto que o algoritmo de identificação de componentes biconexas possui complexidade linear;
6. Para CACTUS_PATH e CACTUS_STAR, CN supera o algoritmo GUS, ainda melhor quando k aumenta.

Uma possível razão para explicar o bom desempenho de CN quando o parâmetro k aumenta nas instâncias PARTED, CACTUS_PATH e CACTUS_STAR é que as componentes biconexas dos grafos se tornam menores. Em PATH e TREE, o oposto deve ocorrer: quando k aumenta, as componentes biconexas se tornam maiores.

Em relação ao bom desempenho geral de CN, observamos que o método executa $n-1$ algoritmos de fluxo máximo em subgrafos do grafo original, enquanto que GUS aplica $n-1$ algoritmos de fluxo máximo no grafo original.

5
Algoritmos para calcular árvores de cortes dinamicamente

Após a teoria apresentada acerca da análise de sensibilidade em árvores de cortes, nós propomos aqui algoritmos para calcular árvores de cortes dinamicamente quando variações de capacidade ocorrem nas arestas. Para as alterações crescentes de capacidade, o primeiro algoritmo proposto resulta de uma modificação no método de Gomory e Hu. A modificação proposta visa, como primeiro passo, coletar aquelas arestas que não pertencem a $\bigcup_{x=1}^{k} P_{i,j}^{x}$ e reutilizá-las como cortes mínimos entre suas extremidades. Depois disso, o algoritmo de Gomory e Hu procede normalmente.

Para as alterações decrescentes de capacidade, o segundo algoritmo a ser proposto resulta de uma generalização do método desenvolvido por Hartmann & Wagner [13], no qual se considera uma variação decrescente de capacidade em apenas uma aresta na rede. Nesse algoritmo, ilustrado na Figura 5.1, os cortes da antiga árvore de cortes são marcados como válidos para a nova árvore de cortes. O procedimento termina quando todos os cortes, ou arestas, estão válidos. Cabe notar que, durante o método, as extremidades de uma aresta válida podem mudar, o que não altera o corte representado por ela.

Procedimento HW (G, $e = (i, j)$, CT^{α}, $\Delta = \alpha - \lambda$);

1 **se** (i, j) é uma ponte **então** substitua, em CT^{α}, α por λ em $c(e)$; **retorne** CT^{λ};

2 Diminua a capacidade das arestas em $P_{i,j}$ por Δ e marque as arestas como válidas;

3 $Q \leftarrow$ arestas não válidas em ordem não-crescente de capacidade;

4 **enquanto** $Q \neq \emptyset$ **faça**

5 $[u, v] \leftarrow$ aresta não válida de maior capacidade com v em $P_{i,j}$;

6 $N_P \leftarrow$ nós adjacentes a v em $P_{i,j}$; $\mu \leftarrow \min_{g \in N_P} \{c[g, v]\}$;

7 **se** $\mu \geq c[u, v]$ ou (u, v) é uma ponte em G **então**

8 Marque $[u, v]$ como uma aresta válida;

9 Considere a subárvore U enraizada em u com $v \notin U$;

10 Marque todas as arestas de U válidas, remova as arestas válidas de Q;

11 Vá para a linha 4;

12 $(U, \overline{U}) \leftarrow$ corte $(u - v)$ mínimo em G^{λ} com $u \in U$;

13 Marque $[u, v]$ como uma aresta válida, remova $[u, v]$ de Q;

14 **se** $c[u, v] = c(U, \overline{U})$ **então** vá para a linha 9;

15 $c[u, v] \leftarrow c(U, \overline{U})$;

16 $N \leftarrow$ nós adjacentes a v;

17 **para todo** $g \in N$ **faça**

18 **se** $g \in U$ **então** reconecte g a u;

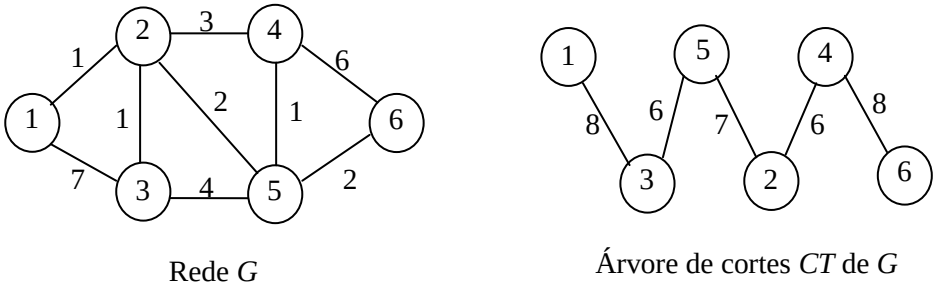
19 **finalize enquanto**

20 **retorne** CT^{λ};

Finalize HW;

Figura 5.1: Pseudocódigo do algoritmo de Hartmann e Wagner (HW).

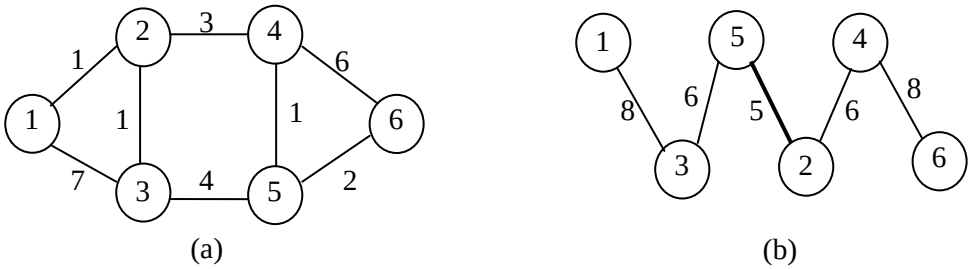
Como demonstração, o algoritmo HW será aplicado na rede G e na sua árvore de cortes CT, ambas ilustradas na Figura 5.2.



Rede *G* Árvore de cortes *CT* de *G*

Figura 5.2: Rede *G* (esquerda) e sua árvore de cortes *CT* (direita).

Suponha que a aresta (2, 5) de *G* seja excluída, ou igualmente falando, tenha sua capacidade decrescida em duas unidades. Esta nova rede está ilustrada na Figura 5.3(a). O algoritmo HW verifica inicialmente se a aresta (2, 5) é uma ponte na rede (linha 1). Como a condição da linha 1 não é satisfeita, o algoritmo executa o comando da linha 2, ou seja, diminui em duas unidades as capacidades dos cortes em $P_{2,5}$ (neste caso apenas um corte), e marca os cortes como válidos. A árvore intermediária resultante após a linha 2 encontra-se ilustrada na Figura 5.3(b), onde cortes válidos estão representados por linhas em negrito.



(a) (b)

Figura 5.3: Nova Rede *G* (a) e primeira árvore de cortes intermediária em HW (b).

Na linha 3, HW compõe o conjunto Q = {[1, 3], [4, 6], [3, 5], [2, 4]}. No comando seguinte, verifica-se a condição do laço **enquanto** (linha 4) e inicia-se a sua execução pela seleção da aresta [3, 5], com u = 3 e v = 5 (linha 5). Na linha 6, o método obtém N_P = {2} e μ = 5, e assim a condição da linha 7 não é satisfeita, pois $\mu < c[3, 5]$ e (3, 5) não é uma ponte em *G*. Um corte mínimo entre os nós 3 e 5 é então calculado na nova rede (linha 12), tendo como resultado U = {1, 3} e $c\left(U, \overline{U}\right)$ = 6. Na linha 13, o corte representado pela aresta [3, 5] é marcado como válido e removido de *Q*. Como $c[3, 5] = c\left(U, \overline{U}\right)$, a condição da linha 14 é verdadeira, e o algoritmo executa o comando da linha 9. Após ter identificado a subárvore U = {[1, 3]} na linha 9, o corte representado pela aresta [1, 3] é

marcado como válido e removido do conjunto Q (linha 10). Na linha 11, o laço **enquanto** é processado novamente e a primeira iteração chega ao fim. A Figura 5.4 ilustra a árvore de cortes intermediária resultante da primeira execução do laço **enquanto**.

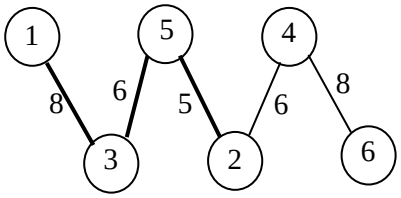


Figura 5.4: Árvore de cortes intermediária resultante da primeira execução do laço **enquanto**.

O algoritmo verifica novamente a condição do laço **enquanto** da linha 4, e inicia a segunda execução, pois neste momento Q = {[4, 6], [2, 4]}. Na linha 5, a aresta [2, 4] é selecionada, com u = 4 e v = 2, e na linha 6 N_P = {5} e μ = 5 são obtidos. Pelo fato de $\mu < c$[2, 4] e (2, 4) não ser uma ponte em G, a condição da linha 7 não é satisfeita, e o procedimento então calcula um corte mínimo entre os nós 2 e 4 na nova rede, obtendo U = {1, 3, 4, 5, 6} e $c(U, \overline{U})$ = 5 (linha 12). Na linha 13, a aresta [2, 4] é marcada como válida e removida de Q. Na linha 14, como c[2, 4] > $c(U, \overline{U})$, a condição não é verdadeira, e o algoritmo executa o comando da linha 15, atualizando a capacidade da aresta [2, 4]. O conjunto N = {5} é definido (linha 16), e o nó 5 é reconectado ao nó 4 na árvore de cortes intermediária (linhas 17 e 18). A Figura 5.5 ilustra a árvore de cortes intermediária após a segunda execução do laço **enquanto**.

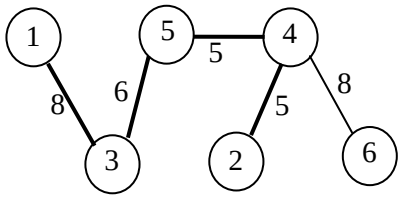


Figura 5.5: Árvore de cortes intermediária resultante da segunda execução do laço **enquanto**.

A terceira e última execução se inicia, pois neste momento Q = {[4, 6]}, e os resultados dela são: aresta [4, 6] selecionada, com u = 6 e v = 4 (linha 5); N_P = {2, 5} e μ = 5 (linha 6); condição não satisfeita (linha 7); U = {6} e $c(U, \overline{U})$ = 8 (linha 12); aresta [4, 6] marcada como válida e removida de Q (linha 13); condição satisfeita (linha 14); subárvore U = Ø (linhas 9 e 10). Como agora

$Q = \emptyset$, o laço **enquanto** não será mais executado, e o procedimento termina. A Figura 5.6 ilustra a árvore de cortes final após a terceira execução do laço **enquanto**.

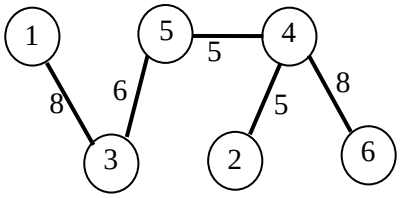


Figura 5.6: Árvore de cortes da nova rede.

O algoritmo HW lida com pontes nas linhas 1 e 7. O leitor pode entender estes comandos recordando o Lema 6, pois uma ponte é uma componente biconexa. No algoritmo geral que será aqui proposto, ao invés de pontes, se utilizará o conceito de componentes biconexas, visto que uma rede pode ter duas ou mais componentes biconexas sem conter uma ponte.

O comando da linha 2 de HW provém do Teorema 4, e o comando da linha 14 do Lema 3. Para a propriedade utilizada nas linhas 6 e 7, bem como para a explicação de se selecionar a aresta não válida de maior capacidade na linha 5, recomenda-se consultar Hartmann & Wagner [13].

As operações das linhas 17 e 18 de HW são similares às do algoritmo de Gusfield, pois obtêm cortes mínimos que não se intersectam através de cortes calculados na rede original (não contraída). A boa definição dos comandos 17 e 18 é embasada no Teorema 6, no qual se demonstra que existem cortes mínimos na nova rede que não intersectam os cortes mínimos da rede antiga.

Teorema 6 (Hartmann e Wagner [13]). *Seja G uma rede com uma aresta* $e = (i, j)$ *de capacidade* $c(e) = \alpha$. *Seja* $(X, \overline{X})$ *um corte* $(x - y)$ *mínimo em G com* $x \in X$, $y \in \overline{X}$ *e* $\{i, j\} \subseteq \overline{X}$. *Seja ainda* $(U, \overline{U})$ *um corte que separa i e j. Para* $c(e) = \lambda < \alpha$:

(i) *se* $(U, \overline{U})$ *separar x e y com* $x \in U$, *então* $c(U \cup X, \overline{U \cup X}) \leq c(U, \overline{U})$;

(ii) *se* $(U, \overline{U})$ *não separar x e y com* $x \in \overline{U}$, *então* $c(U \setminus X, \overline{U \setminus X}) \leq c(U, \overline{U})$.

Prova. Para esta prova consulte o trabalho de Hartmann & Wagner [13].

A Figura 5.7 ilustra as duas situações mencionadas no Teorema 6.

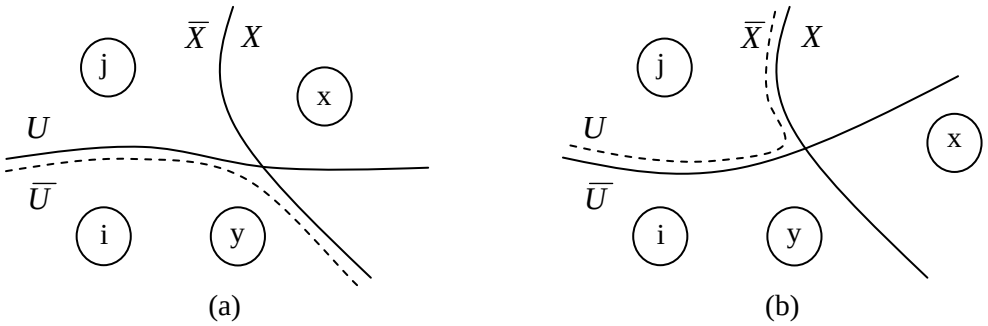


Figura 5.7: Teorema 6(i) (a) e Teorema 6(ii) (b).

Para calcular uma árvore de cortes dinamicamente quando uma alteração decrescente de capacidade ocorre em mais de uma aresta, devemos utilizar o Teorema 5 e o Lema 4 em HW. O resultado desta modificação em HW é o algoritmo Geral de Hartmann e Wagner (GHW), ilustrado na Figura 5.8. Observe que, em GHW, a condição $\bigcap_{x=1}^{k} P_{i,j}^{x} \neq \emptyset$ é necessária. Outros algoritmos devem ser desenvolvidos, caso contrário.

Procedimento GHW (G, k, $e_x = (i_x, j_x)$, CT^{α}, $\Delta_x = \alpha_x - \lambda_x$);

1 **se** $\bigcap_{x=1}^{k} P_{i,j}^{x} = \emptyset$ **então retorne** CT^{λ} calculada por GH ou GUS;

2 $B \leftarrow$ componente biconexa de G que contém as arestas e_x;

3 **para** $x = 1, \ldots, k$ **faça** diminua a capacidade das arestas em $P_{i,j}^{x}$ por Δ_x;

4 Marque as arestas em $\bigcap_{x=1}^{k} P_{i,j}^{x}$ como válidas;

5 $Q \leftarrow$ arestas não válidas em ordem não-crescente de capacidade;

6 **enquanto** $Q \neq \emptyset$ **faça**

7 $[u, v] \leftarrow$ aresta não válida de maior capacidade com v sendo uma extremidade de uma aresta válida;

8 **se** $u \notin B$ **então**

9 Marque $[u, v]$ como uma aresta válida;

10 Considere a subárvore U enraizada em u com $v \notin U$;

11 Marque todas as arestas de U válidas, remova as arestas válidas de Q;

12 Vá para a linha 5;

13 $(U, \overline{U}) \leftarrow$ corte (u - v) mínimo em G^{λ} com $u \in U$;

14 Marque $[u, v]$ como uma aresta válida, remova $[u, v]$ de Q;

15 **se** $c[u, v] = c(U, \overline{U})$ **então**

16 Considere a subárvore U enraizada em u com $v \notin U$;

17 $L_{(u,v)} \leftarrow \{x : [u,v] \in P_{i,j}^{x}\}$;

18 Marque como válidas todas as arestas e de U na qual $L_e = L_{(u,v)}$, remova as arestas válidas de Q;

19 Vá para a linha 5;

20 $c[u, v] \leftarrow c(U, \overline{U})$;

21 $N \leftarrow$ nós adjacentes a v;

22 **para todo** $g \in N$ **faça**

23 **se** $g \in U$ **então** reconecte g a u;

24 Vá para a linha 15;

25 **finalize enquanto**

26 **retorne** CT^{λ};

Finalize GHW;

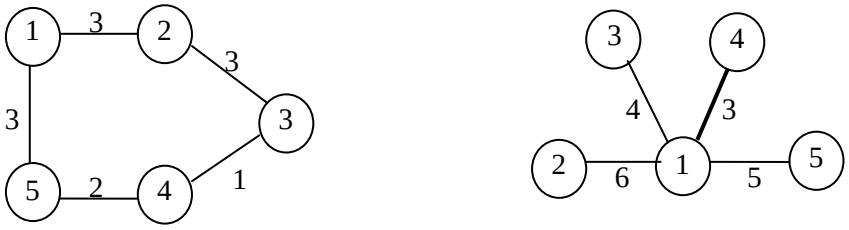
Figura 5.8: Pseudocódigo do algoritmo Geral de Hartmann e Wagner (GHW).

A fim de exemplificar GHW, segue sua aplicação na rede *G* e na sua árvore de cortes *CT*, ilustradas na Figura 5.9.

Rede *G* Árvore de cortes *CT* de *G*

Figura 5.9: Rede *G* (esquerda) e sua árvore de cortes *CT* (direita).

Suponha que as arestas $e_1 = (3, 4)$ e $e_2 = (4, 5)$ de *G* tenham suas capacidades reduzidas para um e dois, respectivamente. A nova rede encontra-se ilustrada na Figura 5.10(a). No primeiro comando de GHW, o algoritmo verifica que a interseção entre os caminhos $P_{3,4}$ e $P_{4,5}$, é um conjunto não-vazio, e o método prossegue para a linha 2. Caso a interseção fosse vazia, CT^{λ} seria construída através dos algoritmos tradicionais de Gomory e Hu ou Gusfield. Na linha 2, o algoritmo identifica a própria rede como a componente biconexa *B* que contém as arestas (3, 4) e (4, 5). Em seguida, GHW diminui em duas unidades as capacidades dos cortes em $P_{3,4}$, e, em uma unidade as capacidades dos cortes em $P_{4,5}$ (linha 3). Na linha 4, o corte representado pela aresta resultante da interseção dos caminhos $P_{3,4}$ e $P_{4,5}$, no caso a aresta [1, 4], é marcado como válido. A Figura 5.10(b) ilustra a árvore de cortes intermediária após a execução da linha 4.



(a) (b)

Figura 5.10: Nova Rede *G* (a) e primeira árvore de cortes intermediária em GHW (b).

Nos comandos seguintes, o algoritmo compõe o conjunto $Q = \{[1, 2], [1, 5], [1, 3]\}$ (linha 5), inicia o laço **enquanto** (linha 6), e seleciona a aresta [1, 2], sendo $u = 2$ e $v = 1$ (linha 7). A condição da linha 8, obviamente, não é

satisfeita, passando-se a execução do comando na linha 13 que calcula um corte mínimo entre os nós 1 e 2 na nova rede e obtém $U = \{2, 3\}$ e $c(U, \overline{U}) = 4$. Na linha 14, a aresta [1, 2] é marcada como válida e removida de Q. Na linha 15, a condição não é verdadeira, pois $c[1, 2] > c(U, \overline{U})$, e o método atualiza $c[1, 2]$ na linha 20. O conjunto $N = \{3, 4, 5\}$ é constituído, e o nó 3 é reconectado ao nó 2 na árvore de cortes intermediária (linhas 21, 22 e 23). Do comando da linha 24 segue-se ao comando da linha 16, em que é identificada a subárvore $U = \{(2, 3)\}$. O conjunto obtido na execução do comando 17 é $L_{(u,v)} = \{1\}$, significando que o corte representado pela aresta [1, 2] contém a aresta $e_1 = (3, 4)$. Como o conjunto $L_{(2,3)} = \{1\}$, a aresta [2, 3] (antiga aresta [1, 3]) contida na subárvore U é marcada como válida e removida de Q (linha 18). A linha 19 é então processada e o laço **enquanto** é executado novamente. A Figura 5.11 ilustra a árvore de cortes intermediária após a primeira execução do laço **enquanto**.

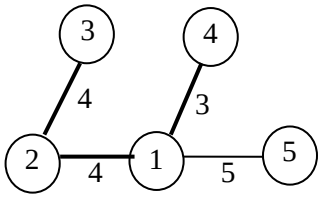


Figura 5.11: Árvore de cortes intermediária em GHW após a primeira execução do laço **enquanto.**

A segunda execução do laço **enquanto** se inicia, visto que $Q = \{[1, 5]\}$. Os resultados desta última iteração são: aresta [1, 5] selecionada, com $u = 5$ e $v = 1$ (linha 7); condição não satisfeita (linha 8); $U = \{4, 5\}$ e $c(U, \overline{U}) = 4$ (linha 13); aresta [1, 5] marcada válida e removida de Q (linha 14); condição não satisfeita (linha 15); $c[1, 5]$ atualizada (linha 20); $N = \{2, 4\}$ (linha 21); nó 4 reconectado ao nó 5 (linhas 22 e 23); subárvore $U = \{(4, 5)\}$ não possui arestas não válidas (linhas 16 a 19); condição não satisfeita, pois $Q = Ø$ (linha 6); procedimento termina. A Figura 5.12 ilustra a árvore de cortes final após a segunda iteração.

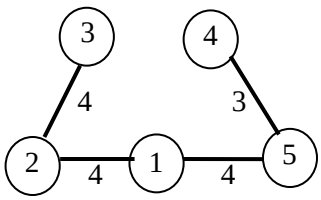


Figura 5.12: Árvore de cortes da nova rede.

Com base no Lema 6, é possível afirmar que todas as arestas de um corte mínimo pertencem a uma mesma componente biconexa. Portanto, uma vez que GHW exige $\bigcap_{x=1}^{k} P_{i,j}^{x} \neq \emptyset$, o comando da linha 2 está correto.

Na linha 4 de GHW encontra-se a aplicação do Teorema 5, nas linhas 8 a 11 a do Lema 6, e nas linhas 16 a 18 a do Lema 4. A propriedade empregada em HW nas linhas 6 e 7 não foi utilizada em GHW, pois, à medida que o número de arestas paramétricas aumenta, aplicar esta propriedade torna-se custoso. Uma futura pesquisa pode verificar a eficácia desta propriedade para um certo número de arestas paramétricas.

Seguindo a ideia central do Teorema 6, o Teorema 7 a seguir permite a execução das linhas 22 e 23 em GHW.

Teorema 7. *Seja G uma rede com k arestas* $e_x = (i_x, j_x)$*, onde* $x = 1, 2, ..., k$*, de capacidades* $c(e_x) = \alpha_x$*. Seja* $(X, \overline{X})$ *um corte* (*g* - *h*) *mínimo em G com* $g \in X$, $h \in \overline{X}$ *e* $\forall x, \{i_x, j_x\} \not\subset X$*. Seja ainda* $(U, \overline{U})$ *um corte que separa pelo menos um par* i_x *e* j_x *tal que* $\{i_x, j_x\} \subseteq \overline{X}$*. Para* $c(e_x) = \lambda_x < \alpha_x$:

(i) *se* $(U, \overline{U})$ *separar g e h com* $g \in U$*, então* $c\left(U \cup X, \overline{U \cup X}\right) \leq c(U, \overline{U})$;

(ii) *se* $(U, \overline{U})$ *não separar g e h com* $g \in \overline{U}$*, então* $c\left(U \setminus X, \overline{U \setminus X}\right) \leq c(U, \overline{U})$.

Prova. Para esta prova denota-se c^{Θ} como a capacidade do corte na nova rede, quando $c(e_x) = \lambda_x < \alpha_x$. Para demonstrar o Teorema 7(i) considere a capacidade dos seguintes cortes:

- $c(X, \overline{X}) = c(U \cap X, \overline{X}) + c(\overline{U} \cap X, \overline{X})$
- $c\left(U \cap X, \overline{U \cap X}\right) = c(U \cap X, \overline{X}) + c(U \cap X, \overline{U} \cap X)$

Observe que ambos os cortes, $(X, \overline{X})$ e $\left(U \cap X, \overline{U \cap X}\right)$, separam *g* e *h*. Como $(X, \overline{X})$ é um corte (*g* - *h*) mínimo, tem-se que $c(\overline{U} \cap X, \overline{X}) \leq c(U \cap X, \overline{U} \cap X)$. De acordo com a condição do teorema de que $\forall x, \{i_x, j_x\} \not\subset X$, $c^{\Theta}(U \cap X, \overline{U} \cap X) = c(U \cap X, \overline{U} \cap X)$, portanto:

(1) $c^{\Theta}(\overline{U} \cap X, \overline{X}) \leq c^{\Theta}(U \cap X, \overline{U} \cap X)$.

Considere agora a capacidade dos seguintes cortes na nova rede:

- $c^{\Theta}(U, \overline{U}) = c^{\Theta}(\overline{U} \cap \overline{X}, U) + c^{\Theta}(U \cap X, \overline{U} \cap X) + c^{\Theta}(U \cap \overline{X}, \overline{U} \cap X)$
- $c^{\Theta}(U \cup X, \overline{U \cup X}) = c^{\Theta}(\overline{U} \cap \overline{X}, U) + c^{\Theta}(\overline{U} \cap \overline{X}, \overline{U} \cap X)$

A partir da equação (1) e de $(\overline{U} \cap \overline{X}, \overline{U} \cap X) \subseteq (\overline{U} \cap X, \overline{X})$, conclui-se que $c^{\Theta}(U \cup X, \overline{U \cup X}) \leq c^{\Theta}(U, \overline{U})$.

Para demonstrar o Teorema 7(ii) considere a capacidade dos seguintes cortes:

- $c(X, \overline{X}) = c(\overline{U} \cap X, \overline{X}) + c(U \cap X, \overline{X})$
- $c(\overline{U} \cap X, \overline{\overline{U} \cap X}) = c(\overline{U} \cap X, \overline{X}) + c(\overline{U} \cap X, U \cap X)$

Observe que ambos os cortes, $(X, \overline{X})$ e $(\overline{U} \cap X, \overline{\overline{U} \cap X})$, separam *g* e *h*. Como $(X, \overline{X})$ é um corte (*g* - *h*) mínimo, tem-se que $c(U \cap X, \overline{X}) \leq c(\overline{U} \cap X, U \cap X)$. De acordo com a condição do teorema de que $\forall x, \{i_x, j_x\} \not\subset X$, $c^{\Theta}(\overline{U} \cap X, U \cap X) = c(\overline{U} \cap X, U \cap X)$, portanto:

(2) $c^{\Theta}(U \cap X, \overline{X}) \leq c^{\Theta}(\overline{U} \cap X, U \cap X)$.

Considere agora a capacidade dos seguintes cortes na nova rede:

- $c^{\Theta}(U, \overline{U}) = c^{\Theta}(U \cap \overline{X}, \overline{U}) + c^{\Theta}(\overline{U} \cap X, U \cap X) + c^{\Theta}(U \cap X, \overline{U} \cap \overline{X})$
- $c^{\Theta}(U \setminus X, \overline{U \setminus X}) = c^{\Theta}(U \cap \overline{X}, \overline{U}) + c^{\Theta}(U \cap X, U \cap \overline{X})$

A partir da equação (2) e de $(U \cap X, U \cap \overline{X}) \subseteq (U \cap X, \overline{X})$, conclui-se que $c^{\Theta}(U \setminus X, \overline{U \setminus X}) \leq c^{\Theta}(U, \overline{U})$. □

6
Identificando complexos de proteína

Neste capítulo, abordamos um problema de identificação de complexos de proteínas e o método de resolução utilizando árvores de cortes, de Mitrofanova et al. [14]. A seguir, aplicamos o algoritmo GHW, juntamente com novos resultados teóricos acerca de cortes reutilizáveis, no algoritmo de Mitrofanova et al. [14] com o objetivo de reduzir seu esforço computacional.

Um problema de clustering, no campo da Biologia, é o de identificar corretamente complexos de proteínas em redes de interação proteína-proteína (PPI) de levedura de dois híbridos (Y2H). Como os resultados de experimentos Y2H sofrem com uma alta taxa de erros, chamados Falso Positivo (FP) e Falso Negativo (FN), a procura por esses complexos se torna um desafio.

Redes Y2H PPI são compostas por proteínas (nós) e interações (arestas) do tipo Y2H. Desta forma, uma rede PPI é não direcionada e não capacitada. Com fins matemáticos, neste capítulo consideramos capacidade unitária para as arestas de uma rede não capacitada. Quando duas proteínas adjacentes não pertencem a um mesmo complexo proteico, diz-se que ocorreu um FP. Quando duas proteínas do mesmo complexo não compartilham uma aresta, diz-se que ocorreu um FN.

Métodos de identificação baseados em grafos e redes geralmente procuram por complexos de proteína que possuem alta conectividade ou que são do tipo clique (Definição 11). Logo, uma taxa alta de FP e FN pode interferir ou até mesmo impossibilitar a procura por tais complexos, seja por criar regiões de "falsa" densidade, através de FP, ou por ocultar complexos do tipo clique, através de FN.

Para superar as dificuldades relativas a FP e FN, Mitrofanova et al. [14] utilizaram a estrutura das árvores de cortes de Gomory e Hu para identificar os complexos de proteína nas redes Y2H PPI. Eles observaram que, mesmo quando algumas arestas de um complexo são perdidas, as proteínas do complexo permanecem conectadas por uma quantidade suficiente de caminhos na rede. Deste modo, eles usaram o número de caminhos disjuntos em arestas como medida de conectividade entre as proteínas. Numa rede Y2H PPI não direcionada e com arestas de capacidade unitária, medir esta conectividade, entre um par de

proteínas, equivale a calcular o fluxo máximo entre elas, e, entre todos os pares das *n* proteínas, a calcular uma árvore de cortes sobre as *n* proteínas.

O algoritmo desenvolvido por Mitrofanova et al. [14] procede, primeiramente, calculando uma árvore de cortes para a rede Y2H PPI. Em seguida, remove as arestas de menor capacidade da árvore de cortes, ou seja, os cortes, representados por elas, na rede. Por último, estes dois passos são aplicados, recursivamente, em cada componente conexa do grafo induzida pela remoção dos cortes. O algoritmo termina quando não há mais arestas nas componentes conexas.

A título de exemplo, o método de Mitrofanova et al. [14] será aplicado na rede *G* do tipo Y2H PPI, ilustrada na Figura 6.1, onde também se encontra o resultado do primeiro passo, sua árvore de cortes *CT*.

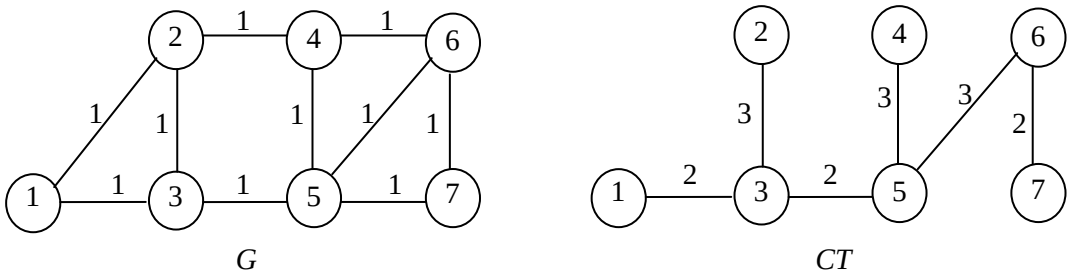


Figura 6.1: Rede *G* do tipo Y2H PPI (esquerda) e sua árvore de cortes *CT* (direita).

No segundo passo, o algoritmo identifica as arestas [1, 3], [3, 5] e [6, 7] como arestas de corte mínimo em *CT*. Portanto, são removidas de *G* as arestas (1, 2), (1, 3), (2, 4), (3, 5), (5, 7) e (6, 7), gerando as componentes conexas G_1 e G_2, estas ilustradas na Figura 6.2, juntamente com suas árvores de cortes CT_1 e CT_2. Neste ponto, o algoritmo é executado em G_1 e G_2, e finalizando quando não houver mais componentes conexas com arestas.

Figura 6.2: Componentes conexas G_1 e G_2 induzidas de *G* (esquerda) e suas árvores de cortes CT_1 e CT_2 (direita).

Em cada fase, o algoritmo de Mitrofanova et al. [14] calcula a árvore de cortes de cada componente conexa sem reutilizar cortes (*s* - *t*) mínimos de árvores calculadas previamente. Vimos no Capítulo 3 que algumas afirmações foram feitas sobre cortes reutilizáveis quando a capacidade de algumas arestas diminui ou algumas arestas são excluídas da rede. Em seguida, expomos um novo resultado acerca de cortes reutilizáveis, quando especificamente se considera a exclusão de todas as arestas de um corte (*s* - *t*) mínimo em uma rede não direcionada e não capacitada (capacidade unitária nas arestas), como ocorre no algoritmo de Mitrofanova et al. [14]. Para tal, é necessária a introdução de uma definição e de alguns lemas prévios.

Definição 26. *Um* ***corte minimal*** *é um corte que desconecta o grafo em exatamente duas componentes conexas.*

Seja a rede *G* da Figura 6.3. Um exemplo de corte minimal em *G* é $(X, \overline{X})$ = {(1, 3),(2, 4)}, em que X = {1, 2} e $\overline{X}$ = {3, 4, 5, 6}. Observe que, a remoção das arestas de $(X, \overline{X})$ desconecta *G* em duas componentes conexas. Um exemplo de corte não minimal em *G* é $(X_1, \overline{X}_1)$ = {(1, 3), (2, 4), (3, 5), (4, 6)}, em que X_1 = {1, 2, 5, 6} e $\overline{X}_1$ = {3, 4}. Note que, a remoção das arestas de $(X_1, \overline{X}_1)$ desconecta *G* em três componentes conexas.

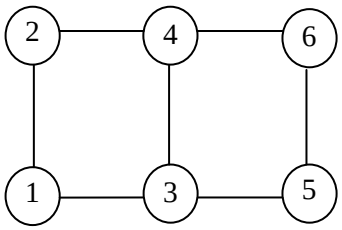


Figura 6.3: Rede *G*.

Através da Definição 26, podemos afirmar que:

Corolário 1. *Seja G uma rede conexa em que nenhuma aresta possui capacidade nula. Todo corte* (*s* - *t*) *mínimo em G é um corte minimal.*

Prova: Assuma que um corte (*s* - *t*) mínimo *C* da rede *G* não seja um corte minimal. Por consequência, *C* desconecta *G* em pelo menos três componentes conexas, Z_1, Z_2, …, Z_y, em que $3 \leq y \leq n$, sendo *n* o número de nós de *G*. Sem perda de generalidade, sejam $s \in Z_1$, $t \in Z_2$ e *Z* o conjunto de todas as componentes conexas excluindo Z_1 e Z_2. Considere *E* o conjunto não-vazio de arestas em *C* que conectam uma componente conexa de *Z* a outras componentes conexas. O corte (*C* – *E*) ainda separa *s* e *t* e possui capacidade menor do que *C*, levando a uma contradição, uma vez que assumimos que *C* era um corte (*s* - *t*) mínimo. □

Para exemplificar o Corolário 1, considere a rede *G* da Figura 6.4(a) e o corte não minimal *C* = {(s, 2), (1, t), (2, 3), (t, 4)} em *G*, que separa os nós s e t. Sejam Z_1, Z_2, e Z_3 as componentes conexas, ilustradas na Figura 6.4(b), geradas a partir da remoção de *C* de *G*, na qual Z_1 contém os nós s e 1, Z_2 contém os nós 2 e t, e Z_3 contém os nós 3 e 4. Para este exemplo, o conjunto *E* será *E* = {(2, 3), (t, 4)}, logo, (*C* – *E*) = {(s, 2), (1, t)}. O corte (*C* – *E*) separa os nós s e t, e possui capacidade menor que o corte *C*.

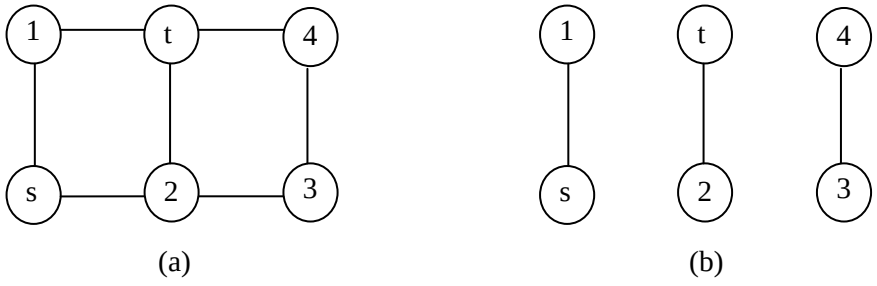


Figura 6.4: Rede *G*(a) e componentes conexas resultantes após a remoção de *C* (b).

O seguinte lema diz que, pelo Teorema 5, apenas um corte mínimo seria reutilizável se excluíssemos todas as arestas de um corte (*s* - *t*) mínimo.

Lema 7. *Sejam G uma rede conexa com arestas de capacidade não-nula e CT sua árvore de cortes. Seja C um corte mínimo de CT composto por k arestas* $e_x = (i_x, j_x)$, *em que* $x = 1, 2, \ldots, k$. *Se* $P_{i,j}^x$ *é o caminho entre* i_x *e* j_x *em CT, o único corte mínimo em* $\bigcap_{x=1}^{k} P_{i,j}^x$ *é C.*

Demonstração: **A** remoção do corte *C* desconecta *G* em duas componentes conexas Z_1 e Z_2. Sejam *s* e *t* um par de nós de Z_1. Seja C_1 um corte que separa *s* e *t*. Se $C \subset C_1$, então a remoção de C_1 desconecta *G* em pelo menos três componentes conexas, e portanto, C_1 não é um corte minimal. Desta forma, conforme o Corolário 1, C_1 não pode ser um corte (*s* - *t*) mínimo, e consequentemente, não pode pertencer a *CT*. □

Sejam *G* a rede da Figura 6.5 e *C* o corte minimal em *G* composto pelas arestas (1, 3) e (2, 4). A remoção de *C* desconecta *G* em duas componentes conexas Z_1 e Z_2, sendo Z_1 a que contém os nós s e t. Considerando C_1 um corte que separa os nós s e t, e $C \subset C_1$, dois possíveis exemplos de C_1 são: 1) {(s, t), (1, 2), (1, 3), (2, 4), (3, 4)}; 2) {(s, t), (t, 2), (1, 3), (2, 4)}. Observe que, a remoção do exemplo 1, desconecta *G* em quatro componentes conexas, e a remoção do exemplo 2, em três.

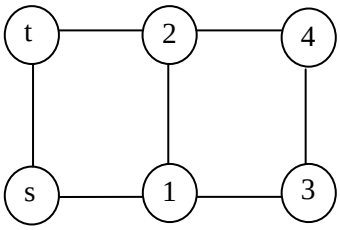


Figura 6.5: Rede *G*.

Os próximos dois lemas ajudarão a provar o Teorema 8.

Lema 8. *Seja CT uma árvore de cortes de uma rede G conexa e com capacidade unitária nas arestas. Sejam C e C_1 dois diferentes cortes mínimos de CT. Se C é composto de k arestas, então C_1 possui no máximo $\lfloor k/2 \rfloor$ arestas em comum com C.*

Demonstração: Assuma $|C \cap C_1| > k/2$. Baseado no Lema 7, $|C \cap C_1| < k$. Sejam *s* e *t* as extremidades de C_1 em *CT*. Observe que C_1 não intersecta *C* (Definição 22) visto que ambos estão em *CT*. Denote por E_1 o conjunto de arestas $(C_1 - (C_1 \cap C))$, por E_2 o conjunto de arestas $(C_1 \cap C)$, e por E_3 o conjunto de arestas $(C - (C_1 \cap C))$. Como $|E_2| > k/2$, temos que $|E_2| > |E_3|$, assim, uma vez

que todas as arestas de *G* possuem capacidade unitária, o corte $(E_1 \cup E_3)$, que também separa *s* e *t*, possui capacidade menor do que $C_1 = (E_1 \cup E_2)$. Portanto C_1 não é um corte (*s* - *t*) mínimo, contradizendo a hipótese. □

Com o objetivo de esclarecer a demonstração do Lema 8, sejam a rede com capacidade unitária nas arestas *G* da Figura 6.6(a) e os cortes $C = \{(1, s), (2, 3), (4, t)\}$ e $C_1 = \{(s, 3), (2, 3), (4, t)\}$ em *G*, representados por linhas tracejadas. Desta forma, tem-se que, $k = 3$, $k/2 < |C \cap C_1| < k$, C e C_1 não se intersectam, C_1 separa os nós s e t, e, $E_1 = \{(s, 3)\}$, $E_2 = \{(2, 3), (4, t)\}$, $E_3 = \{(1, s)\}$. O corte $(E_1 \cup E_3) = \{(s, 3), (1, s)\}$ também separa os nós s e t e possui menos arestas que o corte C_1. A Figura 6.6(b) ilustra a rede *G* com os cortes *C* e $(E_1 \cup E_3)$.

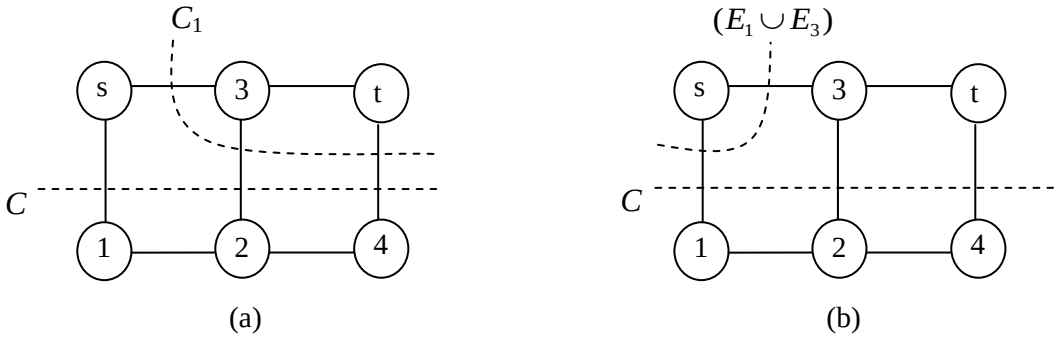


Figura 6.6: Rede *G* com cortes *C* e C_1 (a), e com cortes *C* e $(E_1 \cup E_3)$ (b).

Lema 9. *Seja G uma rede conexa que admite arestas com capacidade nula. Seja C um corte não minimal em G que separa o par de nós s e t. Seja c(C) a capacidade de C. Então existe um corte minimal* C_1 *em G que separa s e t, que* $C_1 \subset C$ *e que* $c(C_1) \leq c(C)$.

Demonstração: Considere o método usado na prova do Corolário 1. Considere que o conjunto *E*, para o caso do Lema 9, possa conter arestas com capacidade nula. Portanto, se aplicarmos o método recursivamente a (*C* – *E*), o resultado é um corte minimal C_1 que separa *s* e *t* e possui capacidade $c(C_1) \leq c(C)$. □

Como exemplo da demonstração do Lema 9, considere as componentes conexas da Figura 6.7(a) como o resultado da remoção do corte $C = \{(s, 2), (1, t),$

(2, 3), (t, 4), (3, 5), (4, 5)} da rede *G*. Considere que *G* contenha apenas arestas com capacidade unitária. Conforme o método da prova do Corolário 1, escolhemos o conjunto E = {(3, 5), (4, 5)}. O corte ($C - E$) desconecta *G* em três componentes conexas, ilustradas na Figura 6.7(b). Aplicando novamente o método, agora para ($C - E$), tem-se o conjunto E_1 = {(2, 3), (t, 4)}. O corte (($C - E$) – E_1) = {(s, 2), (1, t)} é minimal, separa os nós s e t, e possui capacidade menor que *C*.

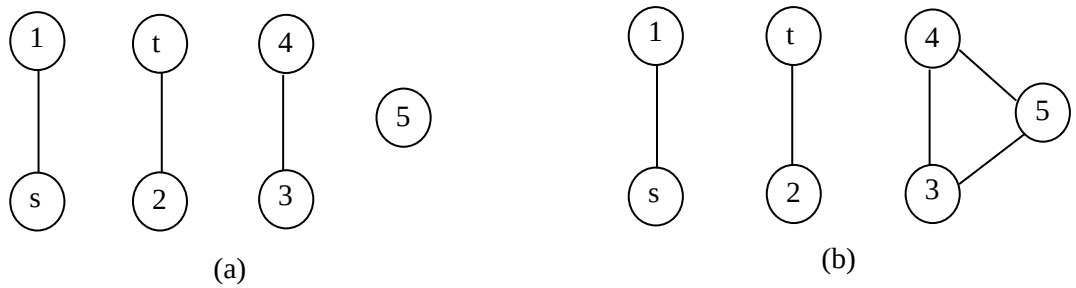


Figura 6.7: Componentes resultantes após a remoção de *C* em *G* (a) e componentes resultantes após a primeira aplicação do método (b).

O próximo Teorema é o principal resultado referente a cortes reutilizáveis a ser incorporado no algoritmo de Mitrofanova et al. [14].

Teorema 8. *Seja CT uma árvore de cortes de uma rede G conexa e com capacidade unitária nas arestas. Sejam C e* C_1 *dois diferentes cortes mínimos de CT e k o número de arestas em C. Sejam s e t as extremidades de* C_1 *em CT. Seja* G^0 *a rede obtida quando todas as arestas de C possuem capacidade nula. Se* C_1 *possui* $\lfloor k/2 \rfloor$ *arestas em comum com C, então* C_1 *é um corte* (*s* - *t*) *mínimo em* G^0 *com capacidade decrescida de* $\lfloor k/2 \rfloor$.

Prova: Baseado no Lema 9, consideraremos apenas cortes minimais nesta prova. Assuma $|C \cap C_1| = \lfloor k/2 \rfloor$. Seja C_2 um corte em *G* que separa *s* e *t* e não intersecta *C*. C_2 pode ser de dois tipos: tipo 1) $|C \cap C_2| \leq \lfloor k/2 \rfloor$. Para este caso, é fácil ver que, em G^0, a capacidade de C_2 será maior ou igual do que a de C_1; tipo 2) $k/2 < |C \cap C_2| < k$. Para este segundo caso, conforme a prova do Lema 8, deve haver um corte C_3 em *G* com capacidade menor que C_2 em que $|C \cap C_3| \leq \lfloor k/2 \rfloor$.

C_2 e C_3 terão capacidade igual em G^0. Portanto, como C_3 é um corte tipo 1, C_2 possuirá capacidade maior ou igual que C_1 em G^0. □

A Figura 6.8 ilustra a situação, da prova do Teorema 8, em que o corte C_2 é do tipo 2. Representados na Figura 6.8(a), sejam, em G, os cortes C = {(1, s), (2, 3), (4, t)}, C_2 = {(s, 3), (2, 3), (4, t)}, e, através da demonstração do Lema 8, C_3 = {(1, s), (s, 3)}. Em G^0, ou seja, com a remoção do corte C de G, as capacidades de C_2 e C_3 se tornam iguais, conforme ilustrado na Figura 6.8(b).

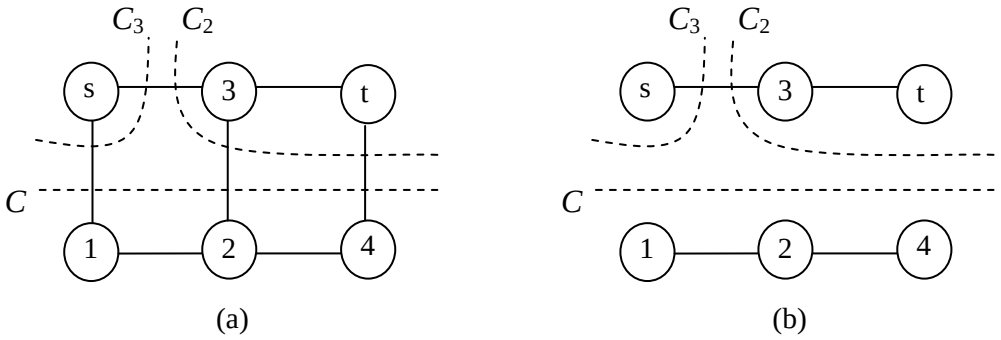


Figura 6.8: Cortes C, C_2 e C_3 em G (a) e em G^0 (b).

Analisando o Teorema 8, se $k = 1$, então $\lfloor k/2 \rfloor = 0$, significando que todo C_1 é reutilizável em G^0, uma vez que, pelo Lema 7, $|C \cap C_1| < k$. Esta situação foi mencionada em Mitrofanova et al. [14]. Se $k = 2$ ou $k = 3$, então $\lfloor k/2 \rfloor = 1$, deste modo, se $|C \cap C_1| = 1$, C_1 é um corte reutilizável em G^0 com capacidade decrescida de um. Para $k = 4$ ou $k = 5$, se $|C \cap C_1| = 2$, C_1 é reutilizável com capacidade decrescida de dois, e assim por diante. A Figura 6.9 ilustra uma árvore de cortes CT, de uma rede com capacidade unitária nas arestas G, em que C = {(1, 5), (2, 5), (3, 7), (4, 7)}, ou seja, $k = 4$. Os cortes representados pelas arestas [2, 3] e [5, 7] de CT são mínimos em G^0, pois possuem duas arestas em comum com C em G.

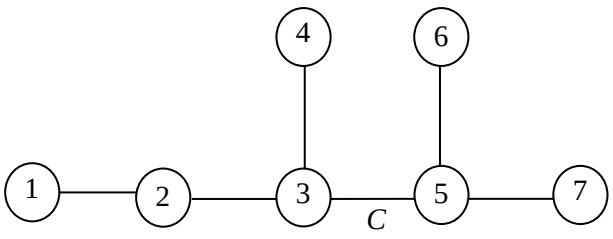


Figura 6.9: Árvore de cortes CT.

Portanto, através do Teorema 8, e também dos Lemas 4 e 6, podemos poupar cálculos de corte mínimo no algoritmo de Mitrofanova et al. [14]. O procedimento proposto é:

1) adaptar o algoritmo GHW para incluir os cortes reutilizáveis abordados no Teorema 8;

2) modificar o algoritmo de Mitrofanova et al. [14] para remover, um de cada vez, todos os menores cortes mínimos da árvore de cortes, ao invés de remover todos de uma só vez;

3) após a remoção de cada corte (s - t) mínimo da árvore de cortes, aplicar o algoritmo GHW em ambas árvores de cortes resultantes, calculando os cortes (s - t) mínimos (linha 13 de GHW) nas componentes conexas correspondentes.

Note que, o passo 2 do procedimento é necessário para que se cumpra o requisito do algoritmo GHW, de que $\bigcap_{x=1}^{k} P_{i,j}^{x} \neq \emptyset$. É óbvio dizer que, no algoritmo de Mitrofanova et al. [14], a primeira árvore de cortes calculada pode ser obtida através dos métodos tradicionais de Gomory e Hu ou de Gusfield.

7
Conclusões

Neste trabalho foi proposto um estudo da teoria da análise de sensibilidade de fluxos máximos multiterminais com o objetivo de propor métodos para acelerar a computação de árvores de cortes de Gomory e Hu. Experimentos computacionais com um dos métodos propostos mostraram o potencial quando aplicado a instâncias que possuem nós de corte. Ainda, aplicamos novos resultados teóricos sobre cortes mínimos reutilizáveis no algoritmo desenvolvido por Mitrofanova et al. [14] com o objetivo de torná-lo mais rápido na identificação de complexos de proteínas em redes Y2H PPI.

Primeiramente, foi realizada uma introdução à teoria de grafos e de fluxos em redes. O problema multiterminal foi apresentado formalmente, e o método de resolução de Gomory e Hu foi descrito em detalhes. Com a observação da existência de $n-1$ cortes mínimos que não se intersectam, Gomory e Hu desenvolveram a técnica de contração de nós que possibilitou a aplicação do algoritmo de fluxo máximo a redes menores do que a original.

Seguindo esta primeira etapa, foi apresentada a teoria da análise de sensibilidade em redes, em que o foco do estudo são os efeitos nos fluxos máximos multiterminais quando as capacidades das arestas variam. Em um algoritmo que gera árvores de cortes em sequência, esta teoria nos mostra que podemos utilizar as informações de uma árvore de cortes existente para calcular eficazmente a próxima, tanto para o caso de somente uma aresta paramétrica, como para o caso de várias arestas paramétricas.

Uma propriedade relacionando nós de corte e fluxos multiterminais foi estabelecida, permitindo desenvolver um método alternativo para calcular árvores de cortes de redes. Este método foi testado computacionalmente, sendo comparado com o método tradicional, em instâncias das famílias PATH, TREE, PARTED e CACTUS. Os resultados numéricos mostraram que, quando as redes possuem nós de corte, a computação de árvores de cortes utilizando o método proposto é bastante eficaz.

Adiante foi apresentado o algoritmo de Hartmann e Wagner (HW), que calcula árvores de cortes dinamicamente quando uma alteração decrescente ocorre

na capacidade de uma aresta da rede. Em seguida, baseado em resultados teóricos deste trabalho, uma adaptação no algoritmo HW foi realizada, a fim de considerar variações decrescentes de capacidade em mais de uma aresta.

Por último, introduzimos um resultado teórico e o algoritmo GHW num método existente que identifica clusters de proteínas, no campo da Biologia. Especificamente, abordamos o problema de identificação de complexos de proteínas em redes Y2H PPI. Mitrofanova et al. [14] desenvolveram um algoritmo eficiente e robusto que identifica os complexos de proteínas através de uma sequência de cálculos de árvores de cortes de Gomory e Hu numa rede não direcionada e não capacitada. As modificações propostas no método existente são capazes de reduzir o cômputo de cortes mínimos necessários na identificação desses complexos.

8
Recomendações para trabalhos futuros

A partir dos resultados apresentados, as seguintes questões permanecem em aberto, dando oportunidade para a continuação da pesquisa neste tema:

- Para qual número de arestas paramétricas a computação dinâmica de árvores de cortes se mantém eficaz?
- Qual o impacto na árvore de cortes quando há simultaneamente variações crescentes e decrescentes de capacidade na rede?
- Quais outras características de grafos, além de nós de corte, podem facilitar a construção de árvores de cortes?
- Para redes capacitadas, existem cortes reutilizáveis quando são excluídas todas as arestas de um corte (s - t) mínimo?
- Em quais circunstâncias se pode obter novos cortes reutilizáveis?

9
Referências bibliográficas

[1] M. Diallo. *Méthodes d'Optimisation Appliquées aux Réseaux de Flots et Télécoms.* Editions universitaires europeennes, 2011.

[2] R. K. Ahuja, T. L. Magnanti e J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New Jersey, 1993.

[3] K. Saisyu, K. Sato e R. Onodera. Comments on computer generation of trees co-trees in a cascade of multi-terminal networks. *IEEE Transactions on Circuits and systems,* cas-28(7): 747-748, Junho de 1981.

[4] K. Agarwal e S. R. Arora. Synthesis of multi-terminal communication nets: Finding one or all solutions. *IEEE Transactions on Circuits and Systems,* 23(3), 141-146, Março de 1976.

[5] L.R. Ford e D.R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1973.

[6] R.E. Gomory e T.C. Hu. Multi-terminal network flows. *SIAM Journal of Computing,* 9(4) :551-570, Dezembro de 1961.

[7] D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM Journal of Computing,* 19: 143-155, 1990.

[8] A.V. Goldberg e K. Tsioutsiouliklis. Cut Tree Algorithms: An experimental study. *Journal of Algorithms,* 38: 51-83, 2001.

[9] A. Bhalgat, R. Hariharan, T. Kavitha e D. Panigrahi. An $\tilde{O}(mn)$ Gomory-Hu Tree Construction Algorithm for Unweighted Graphs. *STOC'07*, 605-614, Junho, 2007.

[10] S.E. Elmaghraby. Sensitivity analysis of multi-terminal flow networks. *Operations Research,* 12(5): 680-688, 1964.

[11] P. Berthomé, M. Diallo e A. Ferreira. Generalized parametric multi-terminal flows problem. *Graph-theoretic concepts in computer science,* Lecture Notes in Computer Science 2880, 71-80, 2003.

[12] D. Barth, P. Berthomé, M. Diallo e A. Ferreira. Revisiting parametric multi-terminal problems: Maximum flows, minimum cuts and cut-tree computations. *Discrete Optimization,* 3: 195-205, 2006.

[13] T. Hartmann e D. Wagner. Dynamic Gomory-Hu Tree Construction – fast and simple. *CoRR,* volume abs/1310.0178, 2013.

[14] A. Mitrofanova, M. Farach-Colton e B. Mishra. Efficient and robust prediction algorithms for protein complexes using Gomory-Hu Trees. *Pacific Symposium on Biocomputing*, 14: 215-226, 2009.

[15] T.H. Cormen, C.E. Leiserson e R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambrigde, MA, 1990.

[16] T.C. Hu e M. T. Shing. *Combinatorial Algorithms, Enlarged 2nd Ed.* Dover Publications, INC, Mineola, New York, 2002.

[17] F. Harary. *Graph Theory*. Addison–Wesley, Reading, MA, 274 pp, 1969.

[18] D. Gusfield e D. Naor. Efficient algorithms for generalized cut trees. *ACM Symposium On Discrete Algorithms*, 422-433, 1990.

[19] C. P. Schnorr. Bottlenecks and edge connectivity in unsymmetrical networks. *SIAM Journal of Computing,* 8(2): 265-274, 1979.

[20] Benczúr. Counterexamples for directed and node capacitated cut-trees. *SIAM Journal of Computing,* 24(3): 505-510, 1995.

[21] M. Scutellà. A note on the parametric maximum flow problem and some related reoptimization issues. *Annals of Operations Research* (in press). Versão da conferência apareceu em *INOC*, 516-520, 2003.

[22] J. Hopcroft e R. Tarjan. Efficient algorithms for graph manipulation. *Communications of the ACM*, 16 (6): 372–378, 1973.

[23] A.V. Goldberg e R.E. Tarjan. A new approach to the maximum flow problem. *ACM Symposium On Theory of Computing,* 136-146, Berkeley, California, Maio de 1986.

[24] G. Skorobohatyj. Solver for the "all-pairs" minimum cut problem in undirected graphs. *Zuse Institute Berlin*. http://ftp.zib.de/pub/Packages/mathprog/mincut/. Último acesso em 7 de Janeiro de 2011.