O PROBLEMA-SANDUÍCHE PARA

CONJUNTOS HOMOGÊNEOS EM GRAFOS

Vinícius Gusmão Pereira de Sá

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

_____

Profª. Celina Miraglia Herrera de Figueiredo, D.Sc.

_____

Prof. Nelson Maculan Filho, D.Habil.

_____

Profª. Simone Dantas de Souza, D. Sc.

_____

Prof. Carlos Alberto de Jesus Martinhon, D.Sc.

_____

Prof. Marcus Vinicius Soledade Poggi de Aragão, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

MAIO DE 2003

SÁ, VINÍCIUS GUSMÃO PEREIRA DE

O Problema-Sanduíche para Conjuntos Homo - gêneos em Grafos [Rio de Janeiro] 2003

IX, 112 p. 29,7 cm (COPPE/UFRJ, D.Sc., En - genharia de Sistemas e Computação, 2003)

Tese – Universidade Federal do Rio de Janei - ro, COPPE

1. Grafos

2. Conjuntos Homogêneos

3. Problema-Sanduíche

I. COPPE/UFRJ II. Título (série)

A meus pais Ilydio e Celia,

que despertaram em mim,

desde a infância,

o gosto pelas ciências

e pelas artes.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

O PROBLEMA-SANDUÍCHE PARA CONJUNTOS HOMOGÊNEOS EM GRAFOS

Vinícius Gusmão Pereira de Sá

Maio/2003

Orientadora: Celina Miraglia Herrera de Figueiredo

Programa: Engenharia de Sistemas e Computação

O conceito de conjunto homogêneo é de grande valia para a elaboração de procedimentos de decomposição de grafos. Grafos-sanduíche são obtidos a partir de dois grafos pré-definidos que lhes emprestam arestas, entre obrigatórias e opcionais. No Problema-Sanduíche para Conjuntos Homogêneos (PSCH), deseja-se descobrir se há, para um par de grafos dado, um grafo-sanduíche que possua algum conjunto homogêneo. Não obstante haver algoritmos lineares para a determinação de conjuntos homogêneos em um grafo isolado, o PSCH continua a ser objeto de pesquisa, dadas as atuais complexidades dos algoritmos existentes e a importância crescente das modelagens grafo-sanduíche para inúmeras aplicações práticas. Este trabalho reúne a evolução das técnicas de resolução deste problema e os esforços do autor no sentido de solucioná-lo eficientemente, culminando na correção do limite superior até então aceito para sua complexidade temporal.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

THE HOMOGENEOUS SET SANDWICH GRAPH PROBLEM

Vinícius Gusmão Pereira de Sá

May/2003

Advisor: Celina Miraglia Herrera de Figueiredo

Department: Systems and Computers Engineering

The concept of homogeneous sets has proved itself useful for the construction of graph decomposition procedures. Sandwich graphs are obtained from two pre-defined graphs that provide them with both mandatory and optional edges. In the Homogeneous Set Sandwich Problem (HSSP), one wants to find out whether there exists an instance of a sandwich graph, for a given pair of graphs, which contains some homogeneous set. Notwithstanding the existence of linear-time algorithms for solving the problem of finding homogeneous sets in a single graph, the HSSP still remains a subject for research, not only because of the rather high time complexities of current algorithms but also motivated by the increasing importance of sandwich-graph modelling to many practical applications. This thesis presents the development of techniques for solving this problem, as well as the author's efforts to find a more efficient solution to it, which culminate in the correction of the established upper bounds for its time complexity.

Conteúdo

Capítulo 1. Introdução 1

Capítulo 2. O Problema-Sanduíche Genérico 4

Capítulo 3. Decomposição Modular e Conjuntos Homogêneos 11

Capítulo 4. O Problema-Sanduíche para Conjuntos Homogêneos 15

4.1. Existência de Conjuntos Homogêneos Sanduíche (Decisão) 16

4.2. Listagem dos Conjuntos Homogêneos Sanduíche (Enumeração) 17

4.3. Testemunhas 18

Capítulo 5. Polinomialidade do Problema de Decisão 23

5.1. Um algoritmo $O(n^5)$ 24

5.2. Um algoritmo $O(n^4)$ 30

5.3. Um algoritmo incorreto $O(\Delta n^2)$ 36

Capítulo 6. Repensando o Problema 46

6.1. Um novo algoritmo $O(n^4)$ 51

Capítulo 7. Novo Limite Superior para o PSCH-D 56

7.1. Um algoritmo $O\{ [\Delta + \varphi (G_1, G_2)] n^2] \} = O(mn^2)$ 56

Capítulo 8. Exponencialidade do Problema de Enumeração 65

Capítulo 9. Conclusão 68

9.1. Retrospectiva 68

9.2. Pesquisa Futura 68

Bibliografia 81

Apêndice 1. Grafos 84

A1.1. O que é um grafo? 84

A1.2. Representação geométrica de um grafo 85

A1.3. Isomorfismo 86

Apêndice 2. Grafos e Computadores 88

A2.1. Estruturas de Dados: Armazenando Grafos no Computador 88

A2.2. Interface Gráfica: Visualizando o Grafo 92

Apêndice 3. Digrafos e Componentes Fortemente Conexos 93

A3.1. Algoritmo para Determinação dos CFCs 94

A3.2. Sumidouros 101

Apêndice 4. Figuras Especiais 103

Lista de Símbolos

Ao longo de todo este trabalho, utilizaremos os seguintes símbolos e operadores:

- $C_{p,q}$ – número de combinações de p elementos, q a q;
- K_n – grafo completo com n vértices;
- m – número de arestas do grafo considerado (ou número de arestas obrigatórias, quando se tratar de uma instância grafo-supergrafo para um problema-sanduíche);
- m_O – número de arestas obrigatórias;
- m_P – número de arestas proibidas;
- n – número de vértices do grafo considerado;
- $N^+(..)$ – conjunto de vizinhos de saída, no digrafo considerado, do vértice em questão (ex.: $N^+(x)$, ou "o conjunto dos vertices v tais que $x \rightarrow v$ é aresta do digrafo");
- $N^-(..)$ – conjunto de vizinhos de entrada, no digrafo considerado, do vértice em questão (ex.: $N^-(x)$, ou "o conjunto dos vertices v tais que $v \rightarrow x$ é aresta do digrafo");
- $N_1(..)$ – conjunto de vértices adjacentes (vizinhos), no grafo G_1, ao vértice em questão (ex.: $N_1(x)$, ou "o conjunto dos vértices adjacentes a x, em G_1");

- $N_2(..)$ – conjunto de vértices adjacentes (vizinhos), no grafo G_2, ao vértice em questão (ex.: $N_2(x)$, ou "o conjunto dos vértices adjacentes a x, em G_2");
- $[u, v]$ – aresta não-orientada incidente aos vértices u e v;
- $u \rightarrow v$ – aresta orientada do vértice u ao vértice v;
- Δ – grau máximo do grafo considerado (ou grau máximo do menor grafo, quando se tratar de uma instância grafo-subgrafo para um problema-sanduíche);
- ..\.. – diferença de dois conjuntos (ex.: $A \setminus B$, ou "a diferença entre A e B");
- | – "tal que"
- |..| – cardinalidade de um conjunto (ex.: $|X|$, ou "a cardinalidade de do conjunto X");
- $\leftarrow$ – atribuição (ex.: $x \leftarrow 0$, ou "x recebe o valor zero").

CAPÍTULO 1

Introdução

Sejam dois grafos G_1 (V, E_1) e G_2 (V, E_2) tais que compartilhem um mesmo conjunto V de vértices e que apresentem a característica de o conjunto de arestas E_1 do primeiro ser subconjunto do conjunto de arestas E_2 do segundo. Um grafo-sanduíche para o par (G_1, G_2) é um grafo G_S constituído do mesmo conjunto V de vértices e de um conjunto de arestas E_S que contenha E_1 e esteja contido em E_2.

Os problemas-sanduíche em grafos surgem como uma generalização relaxada dos problemas de reconhecimento. Nestes últimos, estamos interessados em descobrir se um grafo G pertence ou não a uma determinada *classe* de grafos, isto é, se possui ou não determinada propriedade. Nos primeiros, objetiva-se responder se há alguma instância de grafo-sanduíche, para o par de grafos G_1 e G_2 dado, que pertença à classe (ou possua a propriedade) desejada. É evidente que um problema de reconhecimento tradicional para uma propriedade Π pode ser visto como o caso particular do problema-sanduíche correspondente (para a propriedade Π) onde $G_1 = G_2 = G$.

Entende-se por *módulo* um subconjunto M dos vértices de um grafo formado por elementos que apresentem a mesma adjacência externa a M. Em outras palavras, todos os demais vértices do grafo (que não são pertencentes a M) encaixam-se em um dos seguintes casos: (i) são

adjacentes a todos os vértices de *M*; ou (ii) não são adjacentes a vértice algum de *M*.

Um *conjunto homogêneo* é um módulo não-trivial, isto é, um módulo que não é nem vazio, nem unitário (constituído por um único vértice) e nem máximo (contendo todos os vértices do grafo).

O Problema-Sanduíche para Conjuntos Homogêneos (PSCH) é aquele em que se quer determinar, caso exista, um grafo-sanduíche para o par de grafos dado que possua um conjunto homogêneo.

É interessante destacar que vários problemas-sanduíche, mesmo para propriedades bastante simples, são NP-completos. É o caso, por exemplo, do Problema-Sanduíche para Grafos de Permutação. Por outro lado, há decerto problemas-sanduíche polinomiais, como o é o Problema-Sanduíche para Grafos-Split, e como vem a ser também o PSCH, como veremos neste trabalho.

A propriedade de possuir um conjunto homogêneo está relacionada à substituição de grafos, que tem especial importância na teoria de Grafos Perfeitos, já que é sabido que a composição de grafos por substituição, assim como por identificação de cliques e por outras técnicas, preserva a perfeição.

Apresentamos neste trabalho a evolução e os resultados dos principais esforços que foram até então envidados por outros pesquisadores no sentido de solucionar eficientemente o Problema-Sanduíche para Conjuntos Homogêneos, acrescentando, outrossim, nossa parcela contributiva.

Nos Capítulos 2 e 3, delineamos os conceitos de problema-sanduíche e conjunto homogêneo, convergindo na enunciação do PSCH, no Capítulo 4. O Capítulo 5 percorre desde a primeira idéia de algoritmo polinomial para o problema até o que era considerado, desde sua

publicação por TANG *et al.* (2001), o algoritmo mais eficiente que se conhecia para sua solução, e que provamos ser incorreto. No Capítulo 6, oferecemos uma correta caracterização para conjuntos homogêneos sanduíche e levantamos, para em seguida negar, algumas conjecturas que possibilitariam a redenção do algoritmo de TANG *et al* (2001). O Capítulo 7 propõe um novo algoritmo, cuja complexidade temporal assintótica passa a ser o limite superior conhecido para a versão de decisão do PSCH. No Capítulo 8, propalamos a exponencialidade da versão de enumeração do PSCH e concluímos, no Capítulo 9, com sugestões para futuros trabalhos.

Como leitura complementar, para fins de completividade, o Apêndice 1 resume conhecimentos básicos de Teoria de Grafos, sem os quais pouco compreender-se-ia do texto. Ainda para este propósito, encontram-se, no Apêndice 2, informações adicionais sobre representação, manipulação e visualização de grafos em computador. O Apêndice 3 apresenta o algoritmo de determinação de componentes fortemente conexos preconizado por TARJAN (1972), além de alguns resultados importantes sobre sumidouros e conexidade em digrafos necessários à compreensão de algumas passagens.

Por razões espaciais, algumas figuras não foram inseridas no corpo principal do texto e encontram-se reunidas no Apêndice 4, que encerra este trabalho.

CAPÍTULO 2

O Problema-Sanduíche Genérico

Em um *problema de reconhecimento*, deseja-se determinar se um grafo *G* satisfaz ou não determinada propriedade (conectividade, cordalidade, perfeição etc.), ou, equivalentemente, se *G* pertence a uma determinada *família* ou *classe* de grafos. Muitas famílias de grafos já se provaram úteis a diversas aplicações. Além disso, muitos problemas de otimização que são NP-difíceis em grafos genéricos podem ser resolvidos polinomialmente em algumas dessas famílias.

Na prática, pode acontecer de se desejar executar um algoritmo de domínio restrito tendo como entrada um grafo que não pertença à família desejada, mas que esteja "próximo" a ela. É possível que haja, então, interesse em se permitir uma relaxação na condição de entrada do algoritmo para que se aceite também estes grafos até certo ponto "próximos" da família em questão. Isto leva diretamente ao Problema da Compleição em Arestas: "Dado um grafo *G* e um inteiro *k*, é possível que se acrescentem no máximo *k* arestas a *G* de forma que este passe a fazer parte da família desejada?"

Além da simples compleição em arestas, muitos casos há em que faz sentido a determinação de *obrigatoriedade ou proibição* de certas arestas, enquanto outras seriam opcionalmente acrescentáveis ao grafo original. Desta necessidade nasceu o conceito de *problema-sanduíche*.

Dizemos que o grafo G_2 (V_2, E_2) é *supergrafo* para o grafo G_1 (V, E_1) se $V_2 = V$ e $E_1 \subseteq E_2$. Dados dois grafos G_1 e G_2, o grafo G_S (V, E_S) é chamado de *grafo-sanduíche* para o par (G_1, G_2) se $E_1 \subseteq E_S \subseteq E_2$. GOLUMBIC *et al.* (1995) definem o problema-sanduíche para a propriedade Π como se segue:

Problema-Sanduíche Para a Propriedade Π (PS-Π):

Entrada: Dois grafos G_1 e G_2 tais que G_2 seja um supergrafo de G_1.

Questão: Existe um grafo-sanduíche para o par (G_1, G_2) que satisfaça a propriedade Π?

Pode-se preferir adotar a tripla (V, E_O, E_P) para a entrada de um problema-sanduíche, onde V é o conjunto de vértices, E_O é o conjunto de arestas *obrigatórias* e E_P é o conjunto de arestas *proibidas*. Em comparação com a representação anterior, que se utiliza de um grafo original G_1 (V, E_1) e um supergrafo G_2 (V, E_2) seu, temos que E_O é o próprio conjunto E_1 de arestas de G_1, enquanto E_P é o conjunto de todas as arestas que pertencem ao grafo completo com $|V|$ vértices e que não pertencem a E_2. Dessa forma, o conjunto de arestas de um grafo-sanduíche para a tripla (V, E_O, E_P) deverá conter todas as arestas de E_O (daí o termo *obrigatórias*), nenhuma aresta de E_P (daí o termo *proibidas*) e quaisquer outras de $E_N = K_{|V|} \setminus (E_P \cup E_O)$, que serão designadas arestas *opcionais* (e correspondem às arestas de $E_2 \setminus E_1$, na primeira representação). Na verdade, uma instância de entrada para um problema-sanduíche pode ser entendida como um grafo G (V, E) tal que E se encontra particionado em $\{E_O, E_P \text{ e } E_N\}$, de forma que o armazenamento

de V e de qualquer par dos conjuntos que particionam E é suficiente para sua completa representação.

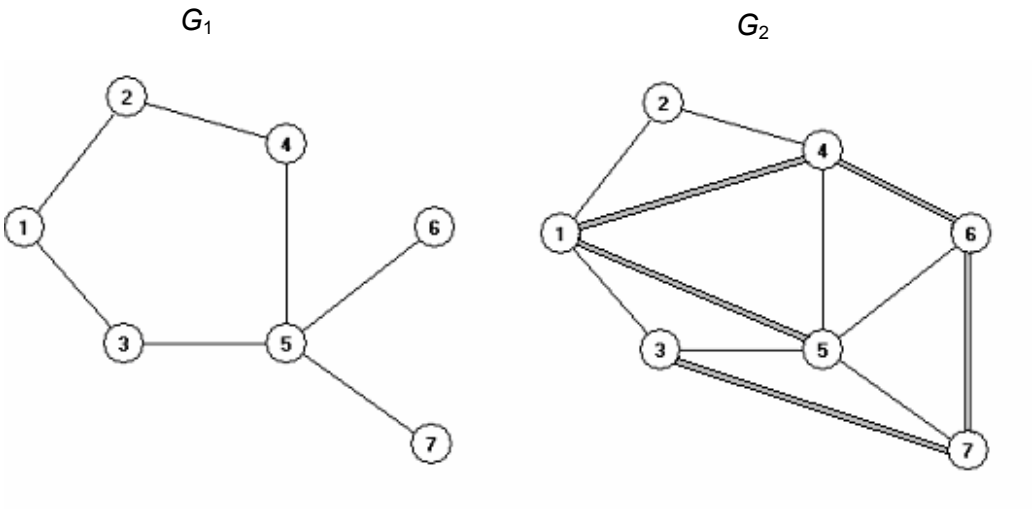


Figura 1 – Exemplo de instância de entrada para um problema-sanduíche

A *Figura 1* ilustra um exemplo de entrada para um problema-sanduíche. O grafo G_1 apresenta todas as arestas obrigatórias – que, lembramos, deverão necessariamente constar de qualquer grafo-sanduíche do par (G_1, G_2) – e o grafo G_2 apresenta, além das arestas obrigatórias, também as arestas opcionais (em destaque, na figura), que poderão fazer parte ou não de um grafo-sanduíche. Já as arestas que não constam de G_2 (proibidas) definitivamente não poderão estar em nenhum grafo-sanduíche para esse par. Observe-se que a tripla (V, E_O, E_P) soaria ligeiramente mais econômica para a representação desta instância, uma vez que o número de arestas obrigatórias ou proibidas $|E_O| + |E_P|$ é igual a 7 + 9 = 16, menor, portanto, que o total de arestas presentes na representação pelo par (G_1, G_2), que é 7 + 12 = 19.

Se, para o exemplo da *Figura 1*, o modo de representação da instância de entrada é pouco importante (no que concerne ao número de

arestas a serem armazenadas), para grafos maiores e possivelmente muito densos ou muito esparsos, no entanto, a maneira de distinguir os três diferentes tipos de aresta, na estrutura de dados utilizada para representar computacionalmente o grafo (vide Apêndice 2), pode ser bem relevante. Convém que se avalie em cada caso, portanto, de acordo com a natureza das possíveis entradas para o problema-sanduíche que se deseja resolver, qual o melhor modo de representá-las.

É interessante notarmos o surgimento natural de problemas-sanduíche a partir de situações onde existam pontos de indefinição quanto à presença ou não de alguns relacionamentos ou interdependências entre elementos do domínio estudado. Quando este domínio é modelado por meio de um grafo, a incerteza quanto à existência de um ou mais relacionamentos (arestas) entre seus elementos (vértices) poderia, em alguns casos, comprometer a qualidade da solução do problema em questão. Em outros, essa mesma incerteza não chega a inviabilizar o estudo do caso desejado, mas propõe, ao contrário, o problema de se avaliar o que ocorreria caso uma ou mais das interdependências duvidosas de fato existissem. Pode sugerir, ainda, que se determine quantas ou quais das arestas duvidosas seriam necessárias para que tal ou qual propriedade fosse satisfeita. Estaríamos, nesses casos, diante de um típico problema-sanduíche.

Os problemas-sanduíche mais interessantes são os que concernem a propriedades não-hereditárias (do contrário G_1 seria sempre solução, em havendo uma) e não-ancestrais (ou G_2 seria sempre solução, em havendo uma). Esta característica está certamente presente no Problema-Sanduíche para Cordalidade (PSC), por exemplo. Sejam os grafos G_1 e G_2 da *Figura 1* a entrada de um PSC. O grafo G_1 é visivelmente não-cordal, ou seja, possui algum ciclo que não apresenta *corda* (aresta unindo dois

de seus vértices não-consecutivos). G_2 também não é cordal, pois o ciclo 1-4-6-7-3-1 não possui corda alguma. O grafo-sanduíche $G_S = G_2 \setminus \{ [3,7] \}$ é, no entanto, cordal, de modo que a saída deste PS-C apresentaria G_S como atestado da sanduíche-cordalidade do par de entrada. Da mesma forma, o Problema-Sanduíche para Conjuntos Homogêneos (PSCH) não é um problema-sanduíche trivial (CERIOLI *et al.*, 1998).

Apresentamos, a seguir, alguns exemplos de importantes problemas-sanduíche que surgiram – ou estão surgindo – na prática:

- Mapeamento de DNA (CARRANO, 1988): na Biologia Molecular, a informação sobre a existência ou não de interseção entre pares de segmentos originários de uma cadeia de DNA é conhecida ou obtida experimentalmente. O problema consiste em se dispor estes segmentos como intervalos sobre uma linha (a cadeia de DNA que se está reconstituindo) de forma que as interseções ou não-interseções desses intervalos, dois a dois, correspondam aos dados experimentais. A modelagem usual se utiliza de um grafo cujos vértices representam os segmentos. Uma aresta será considerada obrigatória (pertencente a E_1) caso os vértices a ela incidentes representem dois segmentos que sabidamente se interceptem, e proibida (não pertencente nem a E_1, nem a E_2) caso os dois segmentos sejam sabidamente não-interceptos. Como, tipicamente, a informação sobre estas interseções é apenas parcialmente conhecida, as arestas opcionais (pertencentes a $E_2 \setminus E_1$, no par de entrada) estarão presentes na modelagem e se estará diante, então, do Problema-Sanduíche para Grafos de Intervalo, que é NP-completo.
- Árvores Filogenéticas: BUNEMAN (1974) provou que o Problema da Filogenia Perfeita pode ser reduzido ao Problema da

Triangulação de Grafos Coloridos, que é uma restrição do Problema-Sanduíche para Cordalidade, NP-completo.

- Sistemas Esparsos de Equações Lineares: considere o sistema de equações $Ax = b$, onde A (n x n) é esparsa, simétrica e definida positiva. Na fase da eliminação gaussiana, uma escolha arbitrária de pivôs pode resultar no preenchimento de posições inicialmente nulas com valores não-nulos, reduzindo a esparsidade da matriz. Dado A, defina um grafo $G(A) = (V, E)$ onde $|V| = n$ e $[v_i, v_j] \in E$ se, e somente se, $a_{ij} \neq 0$ e $i \neq j$. ROSE (1972) provou que encontrar a seqüência de pivôs que minimiza o preenchimento de posições nulas é equivalente a encontrar o conjunto mínimo de arestas que tornam o grafo $G(A)$ cordal, problema este que foi provado ser NP-completo por YANNAKAKIS (1981). Determinar uma seqüência de pivôs que induza um preenchimento de posições nulas com valores não-nulos apenas em posições específicas de A é equivalente a resolver um Problema-Sanduíche para Cordalidade no qual $G_1 = G$ e $[v_i, v_j] \in E_2 \setminus E_1$ se e somente se (i) $i \neq j$, (ii) $a_{ij} = 0$ e (iii) é permitido que a posição (i, j) na matriz se torne não-nula durante a eliminação gaussiana. Este problema surge na prática quando se deseja manter e explorar uma estrutura especial de zeros na matriz durante a eliminação.

Há, entre os problemas-sanduíche, aqueles sabidamente polinomiais, outros sabidamente NP-completos e ainda aqueles que se encontram, até o momento, em aberto (GOLUMBIC *et al.*, 1995; DANTAS *et al.*, 2002):

Polinomiais: PS-Árvore, PS-Split, PS-Cografo, PS-Threshold, PS-Bipartido, PS-(k,l) para $k + l \leq 2$;

NP-Completos: PS-Cordal, PS-Co-Cordal, PS-Comparabilidade, PS-Co-Comparabilidade, PS-Arco Circular, PS-Caminho Não-Direcionado, PS-Caminho Direcionado, PS-Intervalo, PS-Co-Intervalo, PS-Permutação, PS-(k,l) para $k + l > 2$;

Em aberto: PS-Perfeito, PS-Fortemente Cordal, PS-Cordal Bipartido.

Do conjunto dos problemas sabidamente polinomiais faz parte o Problema-Sanduíche para Conjuntos Homogêneos, que é o objeto de nosso estudo.

CAPÍTULO 3

Decomposição Modular e Conjuntos Homogêneos

Seja G (V, E) um grafo. Dizemos que um vértice $x \in V$ *vê parcialmente* um conjunto $C \subset V$ se existe um vértice $p \in C$ tal que $[x, p] \in E$ e existe um vértice $q \in C$ tal que $[x, q] \notin E$.

Um *módulo* de um grafo G (V, E) é um conjunto M de vértices tal que, para todo vértice $x \in V \setminus M$, todos os elementos de M são adjacentes a x ou nenhum membro de M é adjacente a x. Em outras palavras, M é um módulo se, e somente se, nenhum vértice de $V \setminus M$ vê parcialmente M. Os módulos triviais são o conjunto vazio, o próprio conjunto V e os conjuntos unitários que contêm apenas um vértice de V.

A *decomposição modular* de um grafo é a apresentação dos módulos de um grafo na forma de uma árvore, única, cujas folhas são os vértices do grafo e cujos nós internos, exceto a raiz, representam módulos não-triviais. Esta decomposição é muito útil para a solução eficiente de importantes problemas combinatórios em algumas classes de grafos, como a determinação de clique e conjunto independente máximos, emparelhamento máximo e colorações mínimas, entre outros.

A *Figura 2* apresenta um grafo G (V, E) e sua árvore de decomposição modular. Há uma bijeção entre as folhas da árvore e os

vértices de *G*. Cada nó *x* da árvore ou (i) é uma folha e corresponde a um módulo trivial; ou (ii) é nó interno e corresponde ao módulo que é constituído por todos os vértices de *G* representados pelas folhas que descendem de *x*; ou (iii) é a raiz e corresponde ao módulo trivial *V*.

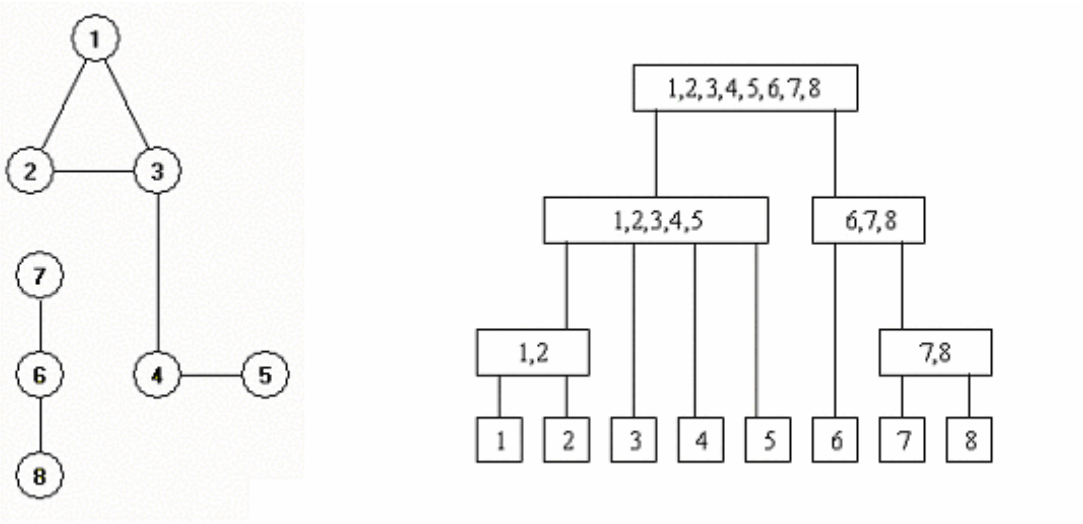


Figura 2 – Um grafo e sua árvore de decomposição modular

Um *conjunto homogêneo* é simplesmente um módulo não-trivial. Isto é, um conjunto *H* de vértices de um grafo *G* (*V, E*) é um conjunto homogêneo se, e somente se, cada vértice em *V* \ *H* – não-vazio – for adjacente a todos os vértices de *H* ou a nenhum dos vértices de *H* e, além disso, *H* possuir pelo menos dois vértices.

Na *Figura 3* estão ilustrados os conceitos de módulo e conjunto homogêneo em um grafo *G*. Alguns módulos encontram-se envolvidos por uma linha pontilhada. Os conjuntos de vértices {1, 2, 5} e {4, 7} são módulos não-triviais e, portanto, conjuntos homogêneos de *G*.

Note-se que há módulos não identificados na figura, a saber: { }, {1}, {2}, {4}, {5}, {7}, {1, 2, 3. 4, 5, 6, 7} e {1, 2, 5, 6}, dos quais o último é conjunto homogêneo.

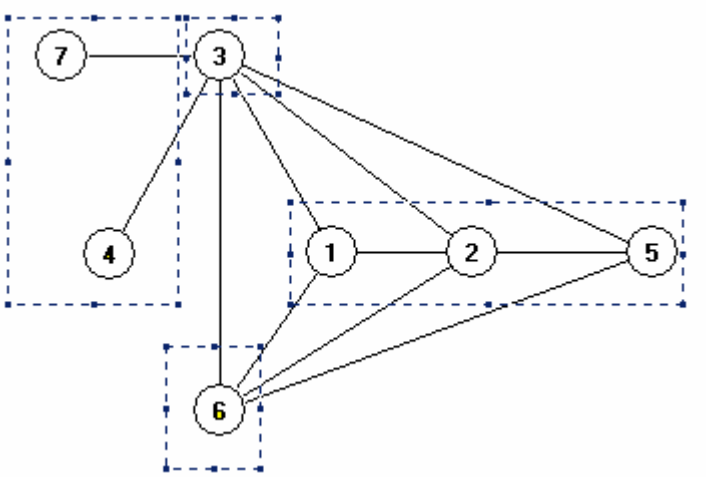


Figura 3 – Módulos e Conjuntos Homogêneos

A existência de um conjunto homogêneo *H* em um grafo *G* (V, *E*) faz com que possamos particionar os vértices de *V* \ *H* em dois conjuntos *A* e *N* tais que:

A = {*v* ∈ *V* | *v* é adjacente a todos os vértices de *H*}

N = {*v* ∈ *V* | *v* é não é adjacente a vértice algum de *H*}

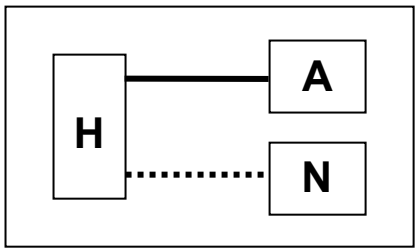


Figura 4 – Diagrama de um grafo com um conjunto homogêneo H

Podemos representar um grafo *G* com um conjunto homogêneo *H* pelo diagrama da *Figura 4*, onde a linha contínua representa a propriedade de cada vértice de *H* ser adjacente a todos os vértices de *A*, enquanto a linha pontilhada indica que nenhum vértice de *H* é adjacente a algum vértice de *N*.

A propriedade de possuir um conjunto homogêneo é de particular interesse no âmbito dos Grafos Perfeitos, remontando a LOVÁSZ (1972), onde é descrito um procedimento de decomposição de grafos que preserva a perfeição e que é baseado no conceito de conjuntos homogêneos.

CAPÍTULO 4

O Problema-Sanduíche para Conjuntos Homogêneos

Determinar um conjunto homogêneo de um grafo não é um problema difícil. McCONNELL e SPINRAD (1999) oferecem um algoritmo que atinge este objetivo em tempo O(*m*). Encontram-se, igualmente, outros algoritmos eficientes em REED (1986), MULLER e SPINRAD (1994), SPINRAD (1992), COURNIER (1993) e GAREY e JOHNSON (1979). Em se tratando do Problema-Sanduíche para Conjuntos Homogêneos (PSCH), entretanto, tal simplicidade não é verificada e muita pesquisa tem sido realizada no sentido de se tentar melhorar o estado da arte dos algoritmos existentes no quesito complexidade temporal. Note-se que o número de grafos-sanduíche para um par de grafos dado é exponencial na diferença entre as cardinalidades dos conjuntos de arestas dos grafos desse par, de forma que elimina-se *a priori* a idéia de se executar, para cada possível instância de grafo-sanduíche dos grafos dados, um dos algoritmos eficientes supracitados para o problema (não-sanduíche) da determinação de conjuntos homogêneos. Não obstante, verifica-se ser possível resolver este problema em tempo polinomial, como será abordado no Capítulo 5.

4.1 – Existência de Conjuntos Homogêneos Sanduíche (Decisão)

A versão mais comum do PSCH é a da simples determinação de *um* grafo-sanduíche para os grafos dados que possua um conjunto homogêneo, caso exista algum que satisfaça esta propriedade. Chama-la-emos versão de *decisão* (PSCH-D), onde se desejará, portanto, tão-somente atestar se o par de grafos dado admite ou não um grafo-sanduíche que possua um conjunto homogêneo.

A *Figura 5* ilustra uma instância de entrada para o PSCH-D.

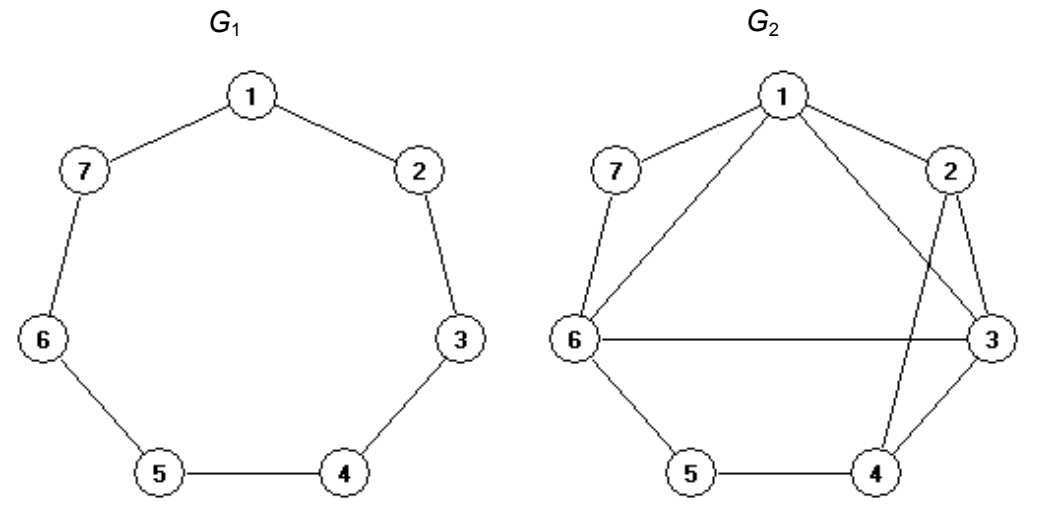


Figura 5 – Exemplo de entrada para o PSCH-D

É interessante observar, no exemplo da *Figura 5*, que, muito embora G_1 e G_2, isoladamente, não contenham conjuntos homogêneos, existe uma instância de grafo-sanduíche para este par que possui esta característica. A *Figura 6* ilustra a saída esperada para este PSCH-D, na qual se vê o grafo G_S, obtido a partir de G_1 com a adição das arestas [1, 3]

e [2, 4], presentes em G_2, e um conjunto homogêneo $H = \{2, 3\} \subset G_S$, respondendo, portanto, afirmativamente à questão "admite o par (G_1, G_2) um grafo-sanduíche que contenha algum conjunto homogêneo?". Um tal conjunto homogêneo é dito, nesse caso, *conjunto homogêneo sanduíche* (CHS) do par (G_1, G_2).

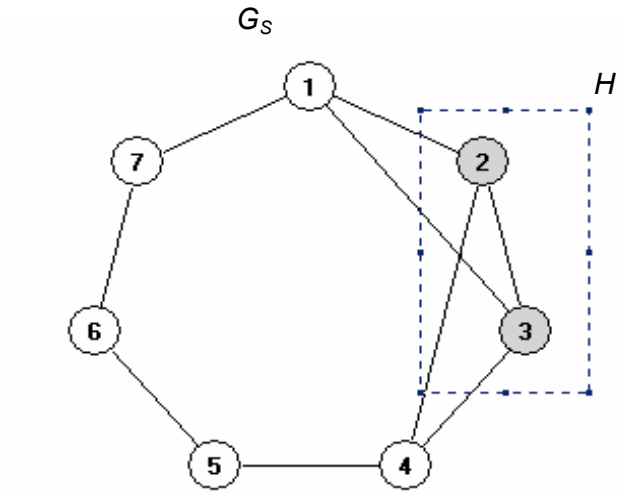


Figura 6 –Exemplo de saída para o PSCH-D

4.2 – Listagem dos Conjuntos Homogêneos Sanduíche (Enumeração)

A versão de *enumeração* do Problema-Sanduíche para Conjuntos Homogêneos (PSCH-E) propõe a obtenção de uma lista contendo todos os conjuntos homogêneos encontráveis em algum grafo-sanduíche para o par de grafos dado. TANG *et al.* (2001) julgaram obter um algoritmo que os enumerasse a todos em tempo polinomial, mas isto foi provado impossível por HABIB *et al.* (2002), como será visto no Capítulo 8.

4.3 – Testemunhas

Antes que se proceda aos capítulos seguintes, onde serão apresentados diversos algoritmos para a solução do PSCH, definiremos alguns conceitos e procedimentos que serão utilizados no decorrer do texto e enunciaremos dois importantes lemas.

Seja a entrada para o PSCH os grafos G_1 (V, E_1) e G_2 (V, E_2), onde G_2 é supergrafo de G_1. Como já o introduzíramos no Capítulo 2, estão reservados os termos *arestas obrigatórias* às arestas de E_1, *arestas proibidas* a todas as arestas que não pertencem a E_2 e *arestas opcionais* às arestas de $E_2 \setminus E_1$. Qualquer grafo-sanduíche G_S (V, E_S) do par (G_1, G_2) atende, por definição, à condição $E_1 \subseteq E_S \subseteq E_2$, devendo conter, portanto, todas as arestas obrigatórias, nenhuma aresta proibida e qualquer número de arestas opcionais.

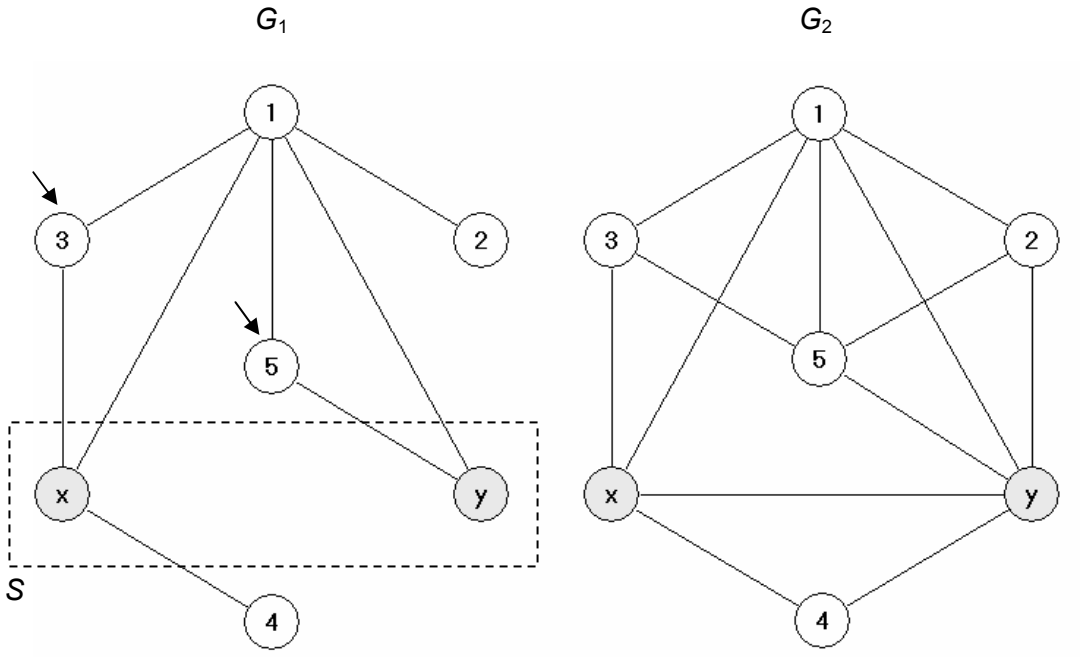


Figura 7 – Exemplo de conjunto-testemunha

Um vértice $t \in V$ será chamado *testemunha* de um conjunto de vértices $S \subseteq V \setminus \{ t \}$ se existir uma aresta obrigatória entre t e algum vértice $x \in S$ e existir, igualmente, uma aresta proibida entre t e algum vértice $y \in S$. Chamaremos *conjunto-testemunha* do conjunto $S \subseteq V$ com relação aos grafos G_1 e G_2 ao conjunto $T \subseteq V \setminus S$ que contém todas as testemunhas de S.

A *Figura 7* nos mostra o conjunto $S = \{x, y\}$, cujo conjunto-testemunha é {3, 5}, uma vez que: o vértice 1 não é testemunha, pois não há arestas proibidas nem entre 1 e *x,* nem entre 1 e *y;* o vértice 2 não é testemunha, pois não há arestas obrigatórias nem entre 2 e *x*, nem entre 2 e *y*; o vértice 3 é testemunha, pois a aresta [3, *x*] é obrigatória e a aresta [3, *y*] é proibida; o vértice 4 não é testemunha, pois, embora haja a aresta obrigatória [4, *x*], a aresta [4, *y*] é opcional, não sendo, portanto, proibida; o vértice 5 é testemunha, pois há a aresta obrigatória [5, *y*] e a aresta proibida [5, *x*].

Ainda a respeito de vértices testemunha, importa que apresentemos o *Lema 1* abaixo, que, embora bastante simples, será lembrado adiante como parte de algumas demonstrações.

Lema 1: *Se* t *é testemunha de um conjunto* S *em relação ao par de grafos* (G_1, G_2), *então* t será *também testemunha, com relação ao mesmo par, de qualquer superconjunto* $S^* \supseteq S$ *que não contenha* { t }.

Prova: se *t* é testemunha de um subconjunto *S* de *S**, então, por definição, existe uma aresta obrigatória de *t* a algum vértice de *S* (e, portanto, de *S**) e uma aresta proibida de *t* a algum vértice de *S* (e, portanto, de *S**), o que

é suficiente para que *t* seja também testemunha de *S** desde que atenda ao requisito extra (ainda pela própria definição de vértice testemunha) de não pertencer a *S**.

Procedimento 1: *Verifica Testemunha* (x, S, G_1, G_2)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$; um conjunto $S \subseteq V$; um vértice $x \subseteq V \setminus S$.

Saída: SIM, se x é testemunha de S com relação a G_1 e G_2; NÃO, caso contrário.

Variáveis: *HaArestaObrigatoria, HaArestaProibida* : booleanas; *y*: vértice.

Método:

1. *HaArestaObrigatoria* ← FALSO

2. *HaArestaProibida* ← FALSO

3. Para cada vértice $y \in S$ faça

3.1. Se $[x, y] \in E_1$ então

3.1.1. *HaArestaObrigatoria* ← VERDADEIRO

3.2. senão se $[x, y] \notin E_2$ então

3.2.1. *HaArestaProibida* ← VERDADEIRO

3.3. Se *HaArestaObrigatoria* e *HaArestaProibida* então

3.3.1. Retorne SIM. Fim.

4. Retorne NÃO. Fim.

Assim como os conceitos recém-introduzidos, os procedimentos *Verifica Testemunha* (que responde se um vértice é ou não testemunha para um conjunto dado) e *Encontra Conjunto-Testemunha* (que retorna o conjunto-testemunha para um conjunto dado) também serão utilizados no

decorrer deste trabalho, como sub-rotinas de alguns algoritmos que serão estudados.

É fácil ver que o procedimento *Verifica Testemunha* leva tempo O(|*S*|) para ser executado, uma vez que o laço da linha 3 realiza no máximo |*S*| iterações, cada uma das quais realizando um número constante de verificações de pertinência de aresta (o que pode ser conseguido em tempo constante, utilizando-se uma estrutura de armazenamento adequada, como, por exemplo, a Matriz de Adjacências – vide Apêndice 2).

Procedimento 2: *Encontra Conjunto-Testemunha* (S, G_1, G_2)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$; um conjunto $S \subseteq V$.

Saída: O conjunto $T \subseteq V \setminus S$, conjunto-testemunha de S com relação a G_1 e G_2;

Variáveis: T : conjunto de vértices; x : vértice.

Método:

1. $T \leftarrow \varnothing$

2. Para cada vértice $x \in V \setminus S$ faça

2.1. Se *Verifica Testemunha* (x, S, G_1, G_2) = SIM então

2.1.1. $T \leftarrow T \cup \{ x \}$

3. Retorne T. Fim.

Verifica-se, trivialmente, que o tempo de execução do procedimento *Encontra Conjunto-Testemunha* é $O(|V \setminus S| . |S|) = O(n^2)$, já que realiza

$|V \setminus S| = O(n)$ chamadas ao procedimento *Verifica Testemunha*, executado em tempo linear $O(|S|) = O(n)$.

Lema 2: *Se* t *é testemunha de um conjunto* S *em relação ao par de grafos* G_1 (V, E_1) e G_2 (V, E_2), *então* (G_1, G_2) *não admite um conjunto homogêneo sanduíche* H *que contenha* S *e que não contenha* { t }.

Prova: Se $H \subset V$ contém S e não contém t, então o fato de t ser testemunha de S implica, pelo *Lema 1*, que t seja também testemunha de H. Pela definição de vértice-testemunha, existe uma aresta obrigatória de t a algum vértice de H e também uma aresta proibida de t a algum vértice de H. Como qualquer grafo-sanduíche G_S de (G_1, G_2) deve conter todas as arestas de E_1 (obrigatórias), t será adjacente, em G_S, a pelo menos um vértice de H. Analogamente, por não poder conter aresta alguma que não pertença a E_2 (aresta proibida), t será, em G_S, necessariamente não-adjacente a algum vértice de H, de forma que H não satisfaz à condição necessária para ser um conjunto homogêneo de G_S, qual seja a de que todos os seus vértices possuam adjacências externas a H idênticas.

CAPÍTULO 5

Polinomialidade do Problema de Decisão

A versão de decisão do Problema Sanduíche para Conjuntos Homogêneos (PSCH-D) admite solução em tempo polinomial no tamanho do conjunto de vértices dos grafos de entrada, como provado por CERIOLI *et al.* (1998), onde foi sugerido um algoritmo de tempo $O(n^4)$. Uma idéia bastante engenhosa de TANG *et al.* (2001), que utiliza a execução do algoritmo de detecção de componentes fortemente conexos de TARJAN (1972) em um grafo auxiliar, resultou em um interessante algoritmo, cuja complexidade de tempo $O(\Delta n^2)$ permanecia, desde sua publicação e até o momento, como o limite superior aceito para este problema. Provaremos, ainda neste capítulo, que infelizmente este algoritmo não funciona. Também em (TANG *et al.*, 2001) encontra-se o que os autores julgaram ser o algoritmo $O(n^4)$ de CERIOLI *et al.* (1998) reescrito de maneira simplificada, mas que é, na verdade, uma simplificação ligeiramente distorcida daquele algoritmo, que imputa um aumento em sua complexidade temporal para $O(n^5)$. Embora apresentada erroneamente à guisa de simplificação da escrita do algoritmo de CERIOLI *et al.* (1998), esta versão será também aqui estudada, tanto por ser um bom ponto de partida para o entendimento do problema em si quanto por ajudar na compreensão do refinamento, presente em (CERIOLI *et al.*, 1998), desta

que pode ser considerada a idéia inicial para a solução do PSCH-D, e que passou desapercebido na releitura exposta por TANG *et al.* (2001).

5.1 – Um algoritmo O(n^5)

O primeiro algoritmo para o PSCH-D foi sugerido por CERIOLI *et al.* (1998) e estende uma idéia de REED (1986) para o teste de existência de um conjunto homogêneo em um grafo isolado. O fato de que a cardinalidade mínima de um conjunto homogêneo é igual a dois sugere que se verifique, para todo par de vértices x e y dos grafos $G_1 = (V, E_1)$ e $G_2 = (V, E_2)$, se existe um grafo sanduíche $G_S = (V, E_S)$ para o par (G_1, G_2) que admita um conjunto homogêneo contendo x e y.

Para esta verificação, parte-se do candidato a conjunto homogêneo H contendo apenas dois vértices x e y quaisquer. Tendo em mente que a existência de uma testemunha t em relação a H garante a impossibilidade de haver um grafo-sanduíche para o par de entrada que admita algum conjunto homogêneo contendo H e não contendo $\{ t \}$ (*Lema 2* – vide Capítulo 4), o que se faz é determinar o conjunto T de testemunhas em relação a H. Caso T não seja vazio, adiciona-se a H todos os vértices de T, e a determinação do conjunto de testemunhas para cada novo candidato a conjunto homogêneo H prossegue, sucessivamente, até que o conjunto T de testemunhas de H seja vazio (caso em que se terá obtido um conjunto homogêneo para algum grafo-sanduíche do par de entrada) ou que H já tenha incorporado todos os vértices de V (caso em que estará atestada a impossibilidade da existência de qualquer grafo-sanduíche que admita um conjunto homogêneo contendo x e y, quando então deve-se reconstruir H a partir de um novo par de vértices iniciais $\{x', y'\}$ e iniciar

nova verificação, prosseguindo, eventualmente, até que todos os possíveis pares de vértices tenham sido utilizados para a construção inicial de *H*).

Procedimento 3: *Monta Grafo-Sanduíche* (G_1, G_2, H)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$; um conjunto homogêneo sanduíche $H \subseteq V$ para o par (G_1, G_2).

Saída: Um grafo-sanduíche G_S, do qual H é conjunto homogêneo.

Variáveis: *EnxergaCHS* : booleana; h, x : vértices.

Método:

1. $E_S \leftarrow E_1$

2. Para cada vértice $x \in V \setminus H$ faça

2.1. *EnxergaCHS* $\leftarrow$ FALSO

2.2. Para cada vértice $h \in H$ faça

2.2.1. Se $[h, x] \in E_1$ então

2.2.1.1. *EnxergaCHS* $\leftarrow$ VERDADEIRO

2.2.1.2. Sai do laço *{ Segue em 2.3. }*

2.3. Se *EnxergaCHS* então

2.3.1. Para cada vértice $h \in H$ faça

2.3.1.1. $E_S \leftarrow E_S \cup \{ [h, x] \}$

3. Retorne G_S (V, E_S). Fim.

Caso um conjunto homogêneo sanduíche H tenha sido encontrado, o algoritmo se completa com a chamada ao procedimento *Monta Grafo-Sanduíche*, que procede à construção de um grafo-sanduíche G_S (V, E_S) do qual H seja conjunto homogêneo. Um tal grafo-sanduíche é por meio

deste procedimento obtido adicionando-se ao conjunto de arestas E_S (inicialmente vazio) tanto as arestas de E_1 (obrigatórias) quanto as arestas $[h, x]$ tais que $h \in H$, $x \in V \setminus H$ e exista pelo menos uma aresta $[h', x] \in E_1$ (aresta obrigatória) para algum $h' \in H$. Ou seja, se existe uma aresta obrigatória de um vértice x para algum vértice de H, então em G_S deverão estar presentes também, além desta aresta, todas as arestas de x aos demais vértices de H (para que H seja conjunto homogêneo de G_S), o que é sempre possível, pois nenhuma de tais arestas pode ser proibida, do contrário x seria testemunha de H, o que contradiria o fato de H ter sido apontado como conjunto homogêneo sanduíche do par (G_1, G_2).

A complexidade temporal do procedimento *Monta Grafo-Sanduíche* é, nitidamente, $O(|V \setminus H| \, . \, |H|) = O(n^2)$ e não onera, portanto, os algoritmos que o utilizam e que já demandam, sem exceção, tempo acima do quadrático no tamanho de V.

O *Algoritmo 1* a seguir é, na realidade, a versão $O(n^5)$ apresentada por TANG *et al.* (2001) do algoritmo $O(n^4)$ presente em (CERIOLI *et al.*, 1998), como já o adiantáramos no começo deste capítulo.

Adiante apresentaremos o *Algoritmo 2*, reescrito a partir de CERIOLI *et al.* (1998), cuja sutil porém importante diferença em relação ao *Algoritmo 1* é o fato de que os conjuntos-testemunha não são calculados do zero a cada iteração, o que consumiria tempo $O(n^2)$, mas recalculados em função do conjunto-testemunha da iteração precedente e de alguns conjuntos auxiliares de vértices que são mantidos para este fim, fazendo com que esta etapa de determinação de conjuntos-testemunha possa ser executada em tempo $O(n)$.

Algoritmo 1: *Admite CHS 1* (G_1, G_2)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$.

Saída: SIM, apresentando um grafo-sanduíche G_S(V, E_S) e um conjunto homogêneo H de G_S, caso exista; ou, caso contrário, NÃO, atestando-se a inexistência de um tal conjunto.

Variáveis: T : conjunto de vértices; G_S (V, E_S) : grafo; x, y : vértices;

Método:

1. Para cada par de vértices x e y de G faça

1.1. $H \leftarrow \{x, y\}$ *{ Inicia a iteração com um novo par em* H *}*

1.2. Enquanto $H \neq V$ faça

1.2.1. $T \leftarrow$ *Encontra Conjunto-Testemunha* (H, G_1, G_2)

1.2.2. Se $T = \varnothing$ então vá para **3.** *{* H *é um CHS }*

1.2.3. $H \leftarrow H \cup T$

2. Retorne NÃO. Fim. *{ Exauriu todos os possíveis pares }*

3. $G_S \leftarrow$ *Monta Grafo-Sanduíche* (G_1, G_2, H)

4. Retorne SIM. Apresente G_S e H. Fim.

***Prova de Corretude do* Algoritmo 1**

Provaremos a corretude do *Algoritmo 1* separadamente para os casos em que a saída é "SIM" (onde mostraremos que o conjunto retornado é de fato um conjunto homogêneo sanduíche do par de grafos da entrada) e em que a saída é "NÃO" (em que provaremos a real ausência de conjuntos homogêneos sanduíche para o par de entrada).

Se o *Algoritmo 1* retorna "SIM" e apresenta o conjunto H como conjunto homogêneo do grafo-sanduíche G_S, então necessariamente foi

satisfeita a condição da linha 1.2.2 na última iteração do laço mais interno (linha 1.2) executada pelo algoritmo, de forma que o conjunto-testemunha T de H em relação aos grafos de entrada G_1 (V, E_1) e G_2 (V, E_2) é vazio e H não contém todos os vértices de V (do contrário a condição da linha 1.2 não teria sido atendida e o algoritmo já teria parado ou passado à próxima iteração do laço principal, reinicializando H com um novo par inicial de vértices). Se T é, portanto, vazio, então nenhum vértice de $V \setminus H$ é testemunha de H, o que é o mesmo que dizer, para todo $x \in V \setminus H$, que pelo menos um dos seguintes fatos é verdadeiro: (i) não existe vértice $h \in H$ tal que [x, h] é aresta obrigatória; ou (ii) não existe vértice $h \in H$ tal que [x, h] é aresta proibida. Se (i) é verdade, então, por construção (procedimento *Monta Grafo-Sanduíche*), G_S não conterá arestas de x para vértice algum de H; por outro lado, se apenas (ii) é verdade, então, por construção, G_S conterá arestas de x para todos os vértices de H, de forma que, em ambos os casos, x não vê parcialmente H (vide Capítulo 3). Por ser esse raciocínio aplicável a cada um dos vértices de G_S não pertencentes a H, e como $2 \leq |H| < n$, temos que H é conjunto homogêneo de G_S.

Se o *Algoritmo 1* retorna "NÃO" (linha 2), então necessariamente o laço principal (linha 1) foi executado para todos os $C_{n,2}$ pares de vértices de V, e, além disso, cada execução terminou com $H = V$ (do contrário, ter-se-ia obtido, em algum momento, um conjunto-testemunha vazio e o algoritmo teria retornado SIM). Utilizaremos, a partir de agora, a notação $H_k^{x,y}$ para designar o conjunto H obtido no final da k-ésima iteração (linha 1.2.3) do laço interno do algoritmo (iniciado na linha 1.2), durante a iteração do laço principal que partiu do conjunto $H = \{x, y\} = H_0^{x,y}$. Suponhamos, então, como em (CERIOLI *et al.*, 1998), que exista um conjunto homogêneo sanduíche F para o par (G_1, G_2). Como F possui necessariamente um

mínimo de dois vértices, sejam u e v dois dos vértices de F e seja $H_0^{u,v} = \{u, v\}$ o par de vértices de partida de uma dada iteração do laço principal do algoritmo (linha 1.1). Se o laço interno do algoritmo não foi quebrado por ter sido encontrado um conjunto-testemunha vazio (linha 1.2.2) para algum $H_k^{u,v}$, então a seqüência ($H_0^{u,v}$, $H_1^{u,v}$, $H_2^{u,v}$...) parou, necessariamente, em algum $H_q^{u,v} = V$. Como $H_0^{u,v} \subseteq F \subset H_q^{u,v}$, é preciso haver um índice i ($0 < i \leq q$) para o qual $H_{i-1}^{u,v} \subseteq F$ mas $H_i^{u,v} \not\subset F$. Por construção (linha 1.2.3), $H_i^{u,v} \setminus H_{i-1}^{u,v}$ é o conjunto-testemunha não-vazio T de $H_{i-1}^{u,v}$. Uma vez que qualquer vértice de $H_i^{u,v} \setminus F$ (que é não-vazio) está presente em $H_i^{u,v} \setminus H_{i-1}^{u,v}$ (ou seja, foi adicionado a H na i-ésima iteração), existe, portanto, pelo menos um vértice $t \in T$ que não pertence a F. Ora, se $t \notin F$ é testemunha de um subconjunto de F (qual seja o conjunto $H_{i-1}^{u,v}$), então, pelo *Lema 1*, t é necessariamente testemunha de F, o que é uma contradição pois, por hipótese, F é conjunto homogêneo.

Análise de Complexidade do Algoritmo 1

O *Algoritmo 1* para reconhecimento de conjuntos homogêneos sanduíche possui um laço principal (linha 1) que executa, no pior caso, $C_{n,2}$ iterações. Cada uma dessas iterações consiste em um laço interno (linha 1.2) que executa $O(n)$ iterações (pois $2 = |H_0^{x,y}| < |H_k^{x,y}| < |H_{k+1}^{x,y}| \leq n$, $k \geq 1$), cada uma das quais consumindo tempo $O(n^2)$ na chamada ao procedimento *Encontra Conjunto-Testemunha*, o que nos remete a um tempo total $O(n^5)$.

5.2 – Um algoritmo O(n^4)

De (CERIOLI *et al.,* 1998) reescreveremos, na forma do *Algoritmo 2* que se segue, o que foi na verdade o primeiro algoritmo publicado para o PSCH-D, e que pode ser entendido como um refinamento da idéia do *Algoritmo 1* onde os conjuntos-testemunha não são recalculados do zero, mas obtidos dinamicamente com o emprego de conjuntos auxiliares de vértices, como já o disséramos.

Tal como no *Algoritmo 1*, o *Algoritmo 2* parte de um candidato a conjunto homogêneo sanduíche *H* contendo apenas um par de vértices (o que será repetido, no pior caso, para todos os possíveis pares) e segue determinando seu conjunto-testemunha *T*, cujos elementos vão sendo sucessivamente adicionados a *H* até que *H* possua todos os vértices do grafo ou até que *H* não possua mais testemunhas.

Para um dado candidato a CHS *H* e seu conjunto-testemunha *T*, definiremos os seguintes conjuntos auxiliares de vértices, que particionam $V \setminus (H \cup T)$:

- *A* (para "Adjacentes") – conterá todo vértice *a* de $V \setminus H$ que deverá ser, forçosamente, adjacente a todos os vértices de *H*, em G_S, caso *H* venha a ser conjunto homogêneo sanduíche do par de entrada. Em outras palavras, *A* conterá todo vértice $a \in V \setminus H$ tal que [*a*, *h*] seja aresta obrigatória para algum $h \in H$ e não exista aresta proibida entre *a* e vértice algum de *H*;
- *N* (para "Não-Adjacentes") – conterá todo vértice *b* de $V \setminus H$ que não poderá ser adjacente a vértice algum de *H*, em G_S, caso *H* venha a ser conjunto homogêneo sanduíche do par de entrada, ou seja, *N* conterá todo vértice $b \in V \setminus H$ tal que [*b*, *h*] seja aresta

proibida para algum $h \in H$ e não exista aresta obrigatória entre b e vértice algum de H;

- D (para "Duvidosos") – conterá todo vértice d de $V \setminus H$ que poderá ser, em G_S, adjacente a todos os vértices de H ou não-adjacente a todos os vértices de H, isto é, conterá todo vértice $d \in V \setminus H$ tal que $[d, h]$ é aresta opcional para todo vértice $h \in H$.

Procedimento 4: *Atualiza Conjunto-Testemunha* (G_1, G_2, T, A, N, D)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$; os conjuntos de vértices T, A, N, D.

Saída: Atualização dos conjuntos T, A, N e D

Variáveis: T', A', N' : conjuntos de vértices; x :vértice.

Método:

1. $T' \leftarrow \varnothing$; $A' \leftarrow \varnothing$; $N' \leftarrow \varnothing$

2. Para cada vértice $x \in T$ faça

2.1. $T' \leftarrow T' \cup (N_1(x) \cap N) \cup (A \setminus N_2(x))$

2.2. $A' \leftarrow A' \cup (N_1(x) \cap D)$

2.3. $N' \leftarrow N' \cup (D \setminus N_2(x))$

3. $A \leftarrow (A \setminus T') \cup (A' \setminus (A' \cap N'))$

4. $N \leftarrow (N \setminus T') \cup (N' \setminus (A' \cap N'))$

5. $D \leftarrow D \setminus (A' \cup N')$

6. $T \leftarrow T' \cup (A' \cap N')$

7. Fim.

Algoritmo 2: *Admite CHS 2* (G_1, G_2) (CERIOLI *et al.*, 1998)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$

Saída: SIM, apresentando um grafo-sanduíche $G_S(V, E_S)$ e um conjunto homogêneo H de G_S, caso exista; ou, caso contrário, NÃO, atestando-se a inexistência de um tal conjunto.

Variáveis: *A, N, D, T, V** : conjuntos de vértices; G_S (V, E_S) : grafo.

Método:

1. Para cada par de vértices x e y de G faça

1.1 $H \leftarrow \{x, y\}$ *{ Inicia a iteração com um novo par em* H *}*

1.2. $V^* \leftarrow V \setminus H$

1.3. $A \leftarrow [\, N_1(x) \cup N_1(y) \,] \cap [\, N_2(x) \cap N_2(y) \,]$

1.4. $N \leftarrow [\, V^* \setminus (N_1(x) \cup N_1(y)) \,] \cap [\, (V^* \setminus N_2(x)) \cup (V^* \setminus N_2(y)) \,]$

1.5. $D \leftarrow [\, V^* \setminus (N_1(x) \cup N_1(y)) \,] \cap [\, N_2(x) \cap N_2(y) \,]$

1.6. $T \leftarrow V^* / (A \cup N \cup D)$

1.7. Enquanto $H \neq V$ faça

1.7.1. Se $T = \varnothing$ então vá para **3.** *{* H *é um CHS }*

1.7.2. $H \leftarrow H \cup T$

1.7.3. *Atualiza Conjunto-Testemunha* (G_1, G_2, T, A, N, D)

2. Retorne NÃO. Fim. *{ Exauriu todos os possíveis pares }*

3. $G_S \leftarrow$ *Monta Grafo-Sanduíche* (G_1, G_2, H)

4. Retorne SIM. Apresente G_S e H. Fim.

O *Algoritmo 2* tem formulação idêntica à do já estudado *Algoritmo 1*, exceto pela determinação dos conjuntos-testemunha T, feita através da

chamada ao procedimento *Atualiza Conjunto-Testemunha*, que não apenas recalcula *T* como sendo o conjunto-testemunha do novo candidato a conjunto homogêneo sanduíche *H*, mas também atualiza os conjuntos auxiliares *A*, *N* e *D* de forma a manter a consistência entre seus elementos e suas definições. A manutenção dinâmica da consistência desses conjuntos estabelece que o conjunto *V* de todos os vértices do par de entrada permaneça particionado em: *H* (candidato a CHS), *T* (conjunto-testemunha de *H*), *A* (vértices forçosamente adjacentes a todos os vértices de *H*, em G_S), *N* (vértices forçosamente não-adjacentes a todos os vértices de *H*, em G_S) e *D* (vértices ‘‘duvidosos’’, que não estão obrigatoriamente nem em *A*, nem em *N*, mas que não são testemunhas de *H*).

Prova de Corretude do Algoritmo 2

Para a prova de corretude do *Algoritmo 2*, que tem funcionamento análogo ao do já provado correto *Algoritmo 1*, é suficiente provarmos que o conjunto-testemunha *T* do candidato inicial $H_0^{x,y}$ de cada iteração do laço principal do algoritmo é corretamente calculado (linhas 1.2 a 1.6) e que o procedimento *Atualiza Conjunto-Testemunha* calcula também de maneira correta os conjuntos-testemunha subseqüentes, em cada iteração do laço interno do algoritmo (linhas 1.7 a 1.7.3).

O conjunto-testemunha *T* para um candidato $H = \{x, y\}$ é obtido após a determinação dos conjuntos iniciais *A*, *N* e *D* (linhas 1.3 a 1.5), que obedece precisamente às definições apresentadas à página 28. Como todos os vértices do grafo que não estão em *T* ou *H* pertencem necessariamente a um dos conjuntos *A*, *N* ou *D*, então *T* fica determinado pela diferença $V \setminus (H \cup A \cup N \cup D)$, o que ocorre à linha 1.6 do algoritmo.

Na k-ésima iteração do laço interno do algoritmo (linhas 1.7 a 1.7.3), ocorre a atualização do candidato a CHS H_{k-1} por meio da adição dos elementos de seu conjunto-testemunha T_{k-1} (linha 1.7.2), determinando, assim, o novo candidato H_k. Queremos provar que é correta a obtenção do conjunto-testemunha T_k pela chamada a *Atualiza Conjunto-Testemunha* (*Procedimento 4*).

Ao procedimento *Atualiza Conjunto-Testemunha* são passados como parâmetros os conjuntos T_{k-1}, A_{k-1}, N_{k-1} e D_{k-1} remanescentes da iteração anterior do laço interno do *Algoritmo 2*, que serão, então, recalculados em função dos elementos recém-adicionados a H_k (quais sejam exatamente aqueles elementos presentes no conjunto-testemunha $T_{k-1} = H_k \setminus H_{k-1}$), obtendo os novos T_k, A_k, N_k e D_k. Isto é atingido com a introdução dos conjuntos auxiliares *A'*, *N'* e *T'*. De fato, o conjunto *T'* conterá os elementos que pertenciam a A ou N e que passaram a ser, após a adição de T_{k-1} a H_{k-1} obtendo H_k, testemunhas do candidato H_k em relação a G_1 e G_2, ou seja, aqueles vértices $a \in A_{k-1}$ para os quais existe agora uma aresta proibida $[a, t]$, $t \in T_{k-1}$, e também aqueles vértices $b \in N_{k-1}$ para os quais existe agora uma aresta obrigatória $[b, t]$, $t \in T_{k-1}$. O conjunto *A'* conterá os vértices *a'* que sairão de D (ou seja, que estarão em $D_k \setminus D_{k-1}$), por existir agora uma aresta obrigatória $[a', t]$ para algum $t \in T_{k-1}$, ou seja, entre *a'* e algum vértice t recém-adicionado a H (pertencente, portanto, a $H_k \setminus H_{k-1}$). Nas linhas 3 e 6 do *Procedimento 4* os elementos *a'* de *A'* são redirecionados a T_k ou A_k, em função de haver ou não, respectivamente, uma aresta proibida $[a', y]$ para algum $y \in T_{k-1}$, ou seja, de *a'* ser ou não testemunha de H_k em relação aos grafos G_1 e G_2. Analogamente, o conjunto *N'* conterá os vértices *b'* que sairão de D, por haver agora alguma aresta proibida $[b', t]$ entre *b'* e algum t recém-

adicionado ao candidato a CHS *H*, e que serão incluídos (linhas 4 e 6) nos conjuntos T_k ou N_k, respectivamente a existir ou não uma aresta obrigatória [*b'*, *y*] para algum *y* pertencente ao novo *H* (H_k). O novo conjunto-testemunha T_k consistirá, finalmente, tanto dos elementos de *A'* e *N'* (aqueles recém-tirados de D_{k-1}) que passaram a ser testemunhas de H_k quanto das novas testemunhas de H_k (ex-elementos de A_{k-1} e N_{k-1}) presentes em *T'*, o que completa a correta obtenção de *T* e a prova.

Análise de Complexidade do **Algoritmo 2**

A existência dos conjuntos auxiliares dinamicamente mantidos *A*, *N* e *D* permite que cada conjunto-testemunha T_k seja determinado após $|T_{k-1}|$ iterações do laço iniciado na linha 2 do *Procedimento 4*, cada uma das quais realizando um número constante de uniões e interseções de conjuntos de tamanho O(*n*). Utilizando listas de adjacência e conjuntos de vértices ordenados, estas operações podem ser realizadas em tempo linear no tamanho dos conjuntos, ou seja, O(*n*), de forma que todo o procedimento *Atualiza Conjunto-Testemunha* consegue ser executado em tempo $O(n \,.|T|)$, inferior ao tempo $O(n^2)$ do procedimento *Encontra Conjunto-Testemunha*, que obtém, para o *Algoritmo 1*, cada conjunto-testemunha a partir do zero. O tempo total gasto por todas as chamadas ao procedimento *Atualiza Conjunto-Testemunha* (linha 1.7.3) será, então, $O\,[n\,.\,(|T_0| + |T_1| + |T_2| + ...)]$. Como cada vértice de *V* poderá aparecer em no máximo um T_k, temos que $|T_0| + |T_1| + |T_2| + ... = O(n)$, remetendo-nos à complexidade temporal total de $O(n^2)$ para o laço interno do *Algoritmo 2*, e de $O(n^4)$ para todo o *Algoritmo 2*, uma vez que seu laço principal (linha 1) executa $O(n^2)$ iterações, no pior caso.

5.3 – Um algoritmo incorreto O(Δn^2)

Inspirados na mesma idéia básica de que a existência de uma testemunha $t \in V \setminus H$ para um conjunto $H \subseteq V$ é proibitivo a que H seja um conjunto homogêneo sanduíche ou esteja contido em um conjunto homogêneo sanduíche de $G_1 (V, E_1)$ e $G_2 (V, E_2)$ que não contenha { t }, TANG *et al.* (2001) desenvolveram um algoritmo bastante interessante, onde as testemunhas de todos os pares de vértices de V são determinadas, *a priori*, e então utilizadas na construção de um digrafo auxiliar que expõe todas as relações par-testemunha presentes em (G_1, G_2), o que permitiria, como se supunha, que o problema da localização de um conjunto homogêneo sanduíche fosse reduzido ao da determinação, neste digrafo, de um sumidouro fortemente conexo, formado da união de pares de vértices constituindo subconjuntos sem testemunhas. Veremos, no entanto, que esta redução não é possível.

Definiremos o *grafo-testemunha* de um par de grafos G_1 (V, E_1) e G_2 (V, E_2) – onde G_2 é supergrafo de G_1 – como sendo o digrafo G_T (V_T, E_T) tal que: V_T apresenta um vértice rotulado $\langle u,v \rangle$ (ou simplesmente $\langle uv \rangle$) para cada par não-ordenado de vértices distintos u, $v \in V$; E_T apresenta duas arestas orientadas $\langle u, v \rangle \rightarrow \langle u, w \rangle$ e $\langle u, v \rangle \rightarrow \langle v, w \rangle$ para cada vértice $\langle u, v \rangle$ de V_T e as testemunhas w do conjunto $H = \{u, v\} \in V$ com relação a (G_1, G_2).

A *Figura 8* mostra os grafos G_1 e G_2 (G_2 supergrafo de G_1) ao lado de seu grafo-testemunha G_T. Note-se, por exemplo, a existência das arestas $\langle a, b \rangle \rightarrow \langle a, c \rangle$ e $\langle a, b \rangle \rightarrow \langle b, c \rangle$, oriundas do fato de ser o vértice c testemunha do conjunto {a, b} com relação a (G_1, G_2). Da mesma forma, estão presentes as arestas $\langle a, d \rangle \rightarrow \langle a, e \rangle$ e $\langle a, d \rangle \rightarrow \langle d, e \rangle$ em virtude de

o vértice *e* ser testemunha do conjunto {*a*, *d*} com relação aos grafos G_1 e G_2. Importa, igualmente, que se atente à existência de vértices de G_T, como ⟨*b*, *c*⟩ e ⟨*d*, *e*⟩, que não possuem arestas de saída, indicando que os conjuntos-testemunha dos pares de vértices de *V* por eles representados são vazios.

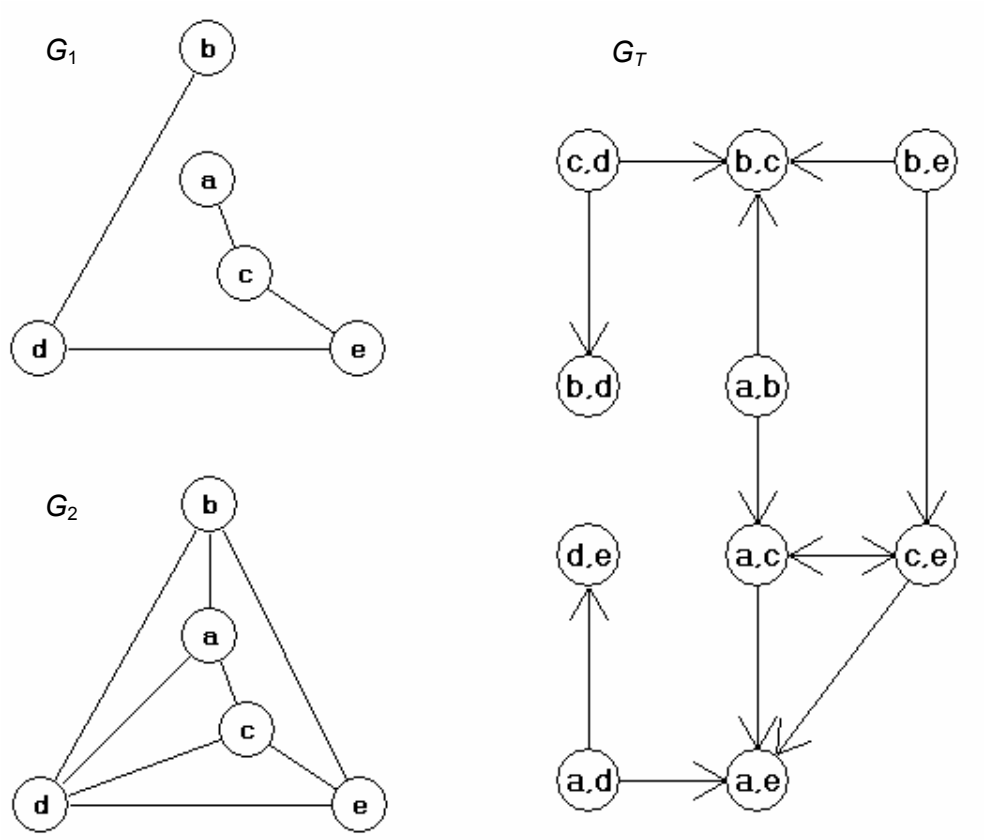


Figura 8 – Um par (grafo, supergrafo) e seu grafo-testemunha

O procedimento *Constrói Grafo-Testemunha* (*Procedimento 5*) retorna o digrafo G_T, grafo-testemunha dos grafos $G_1(V, E_1)$ e $G_2(V, E_2)$ da entrada. Seus dois primeiros laços aninhados definem o conjunto V_T dos vértices do digrafo, consumindo tempo $O(|V \times V|) = O(n^2)$ nesta etapa. A etapa seguinte, onde são adicionadas as arestas ao conjunto E_T

Procedimento 5: *Constrói Grafo-Testemunha* (G_1, G_2)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$

Saída: G_T (V_T, E_T), grafo-testemunha de (G_1, G_2)

Variáveis: G_T (V_T, E_T) : digrafo; x, y, z : vértices.

Método:

1. $V_T \leftarrow \varnothing$; $E_T \leftarrow \varnothing$

2. Para cada vértice $x \in V$ faça

2.1. Para cada vértice $y \in V \mid y \neq x$ faça

2.1.1. Se $\langle y, x\rangle \notin V_T$ então $V_T \leftarrow V_T \cup \{\langle x, y\rangle\}$

3. Para cada vértice $\langle x, y\rangle \in V_T$ faça

3.1. Para cada vértice $z \in N_1(x) \cup N_1(y)$ faça

3.1.1. Se *Verifica Testemunha* (z, {x, y}, G_1, G_2) = SIM então

3.1.1.1. $E_T \leftarrow E_T \cup \{\langle x, y\rangle \rightarrow \langle x, z\rangle, \langle x, y\rangle \rightarrow \langle y, z\rangle\}$

4. Retorne G_T. Fim.

---

inicialmente vazio, consiste de um laço principal (linha 3) que executa $O(n^2)$ iterações, uma para cada elemento de V_T. Cada uma dessas iterações tem por objetivo determinar o conjunto-testemunha de um par de vértices {x, y} $\in V$, o que é conseguido com a chamada ao procedimento *Verifica Testemunha* (*Procedimento 1*) para cada vértice $z \in V$ tal que qual exista pelo menos uma aresta obrigatória [z, x] ou [z, y] em G_1, ou seja, *Verifica Testemunha* não será chamado para vértices que não pertençam a $N_1(x) \cup N_1(y)$, uma vez que já não poderão mesmo satisfazer uma das condições necessárias para ser testemunha de {x, y}, qual seja a de possuir uma adjacência obrigatória com x ou y no grafo-testemunha

G_S. Como | $N_1(x) \cup N_1(y)$ | = $O(\Delta)$, a complexidade temporal de todo o procedimento *Constrói Grafo-Testemunha* é $O(\Delta n^2)$.

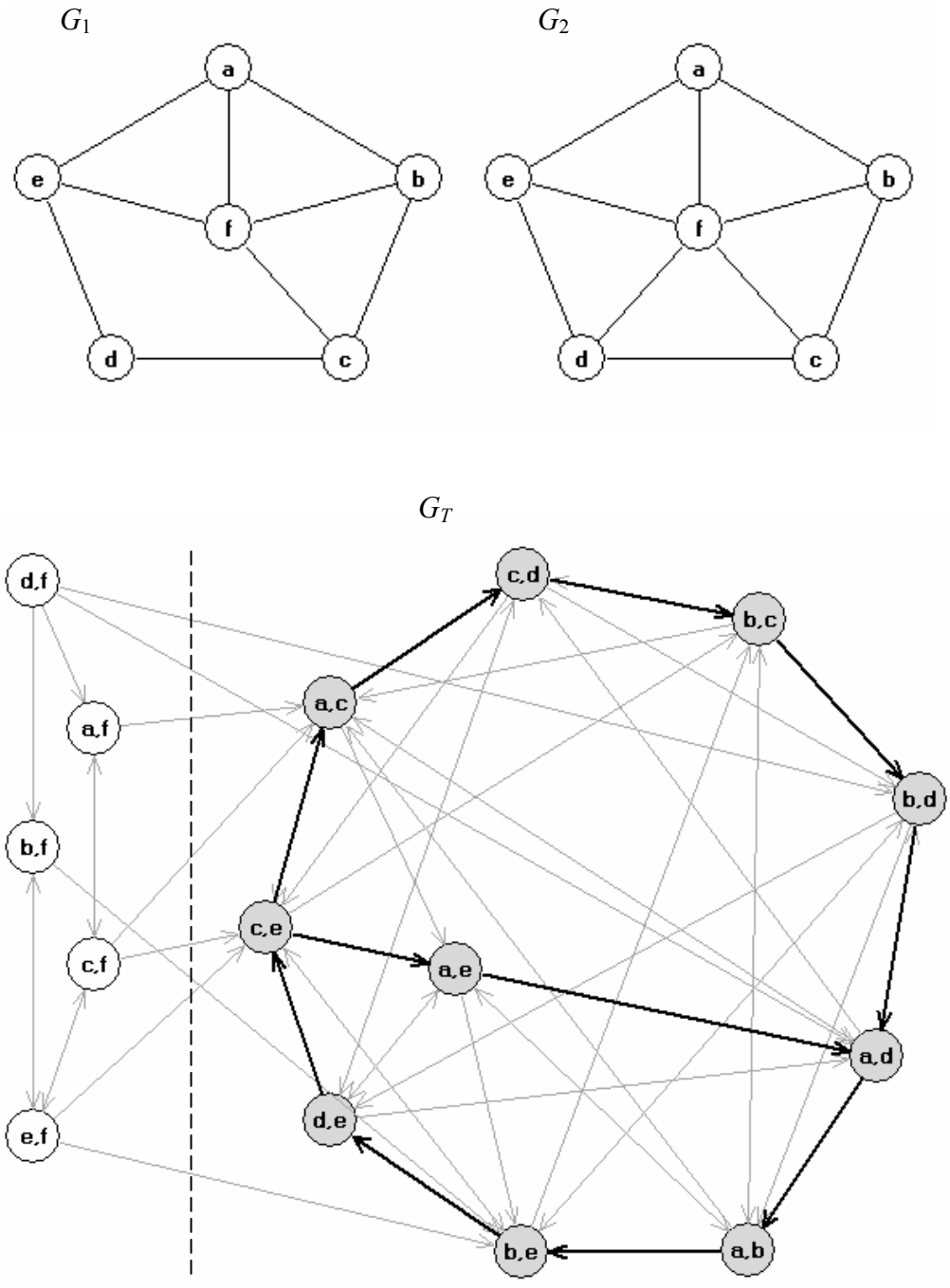


Figura 9 – Componentes e sumidouros fortemente conexos

Um grafo direcionado (digrafo) é *fortemente conexo* se existe um caminho de *u* para *v* (dizemos que *u atinge v*) e também um caminho de *v* para *u* para todo par de vértices *u*, *v* (HARARY, 1969). Um *componente fortemente conexo* (CFC) de um digrafo é um subgrafo maximal fortemente conexo desse digrafo. Um *sumidouro* é um subgrafo que não apresenta arestas de saída, ou seja, é constituído por vértices que apenas atingem outros vértices deste mesmo subgrafo, não podendo atingir, por conseguinte, vértices que dele não façam parte. Um *sumidouro fortemente conexo* (SFC) é um CFC que é um sumidouro. Um *sumidouro fortemente conexo próprio* (SFCP) de um digrafo é um SFC que não contém todos os seus vértices.

A *Figura 9* apresenta o grafo-testemunha G_T para os grafos G_1 e G_2, onde se nota a presença do SFCP formado pelos vértices em destaque (à direita da linha tracejada). É fácil verificarmos que se trata, de fato, de um sumidouro fortemente conexo, pois há caminhos de um vértice a qualquer outro (o que é suficientemente verificado observando-se as arestas em destaque na figura, as quais perfazem um ciclo com um vértice em seu interior) e, além disso, não há arestas de saída de nenhum de seus vértices para qualquer outro que dele não faça parte (o que pode ser facilmente percebido, já que todas as arestas que atravessam a linha tracejada são orientadas da esquerda para a direita, ou seja, ‘‘entram’’ no SFC). O conjunto { $\langle a, f\rangle$, $\langle b, f\rangle$, $\langle c, f\rangle$, $\langle e, f\rangle$ } é também um componente fortemente conexo. Não é, no entanto, um sumidouro fortemente conexo, pois apresenta arestas de saída de alguns de seus vértices a vértices não-pertencentes a este conjunto, como $\langle c, f\rangle \rightarrow \langle c, e\rangle$, por exemplo.

TARJAN (1972) fornece um algoritmo O(n' + m') para encontrar todos os componentes fortemente conexos de um digrafo com n' vértices e

m' arestas, que apresentamos sob o título de *Particiona Digrafo CFCs* (*Procedimento 10*) no Apêndice 3 deste trabalho.

Procedimento 6: *Encontra SFCPs* (*D*)

Entrada: Um digrafo *D* (*V*, *E*)

Saída: *C*, contendo todos os SFCPs de *D*, se existir algum; ou ∅, caso contrário.

Variáveis: *S* conjunto de vértices; *C* : conjunto de conjuntos de vértices; *v*, *y* : vértices; *TemArestaSaidaCFC* : booleano;

Método:

1. *C* ← *Particiona Digrafo CFCs* (*D*)

2. Se |*C*| = 1 então retorne ∅; Fim.

3. Para cada componente fortemente conexo $S \in C$ faça

3.1. *TemArestaSaidaCFC* ← FALSO

3.2. Para cada vértice $v \in S$ faça

3.2.1. Para cada vértice $y \in N^+(v)$ faça

3.2.1.1 Se $y \notin S$ então

3.2.1.1.1. *TemArestaSaidaCFC* ← VERDADEIRO

3.2.1.1.2. Sai do laço. *{ Segue na linha 3.2.2 }*

3.2.2. Se *TemArestaSaidaCFC* então

3.2.2.1. Sai do laço. *{ Segue na linha 3.3 }*

3.3. Se *TemArestaSaidaCFC* = VERDADEIRO então

3.3.1 *C* ← *C* \ *S*

4. Retorne *C*. Fim.

O procedimento *Encontra SFCPs* (*Procedimento 6*) utiliza o particionamento de um digrafo em componentes fortemente conexos de TARJAN (1972) e retorna os CFCs que são também sumidouros fortemente conexos. Caso não encontre nenhum, retorna $\varnothing$.

A etapa de determinação dos CFCs (chamada a *Particiona Digrafo CFCs*) consome tempo $O(n' + m')$. Como cada vértice do digrafo pode estar em no máximo um componente fortemente conexo, a linha 3.2.1 será executada, no pior caso, $O(n')$ vezes, cada uma das quais computando no máximo $| N^+(v) | = O(\Delta')$ iterações, donde uma complexidade temporal total do procedimento *Encontra SFCPs* será $O(n' + m' + \Delta' n') = O(\Delta' n')$, onde, resaltamos, Δ', n' e m' referem-se respectivamente ao grau máximo e tamanhos dos conjuntos de vértices e arestas do digrafo D, dado como entrada para o procedimento.

Reescrevemos na forma do *Algoritmo 3* o algoritmo proposto por TANG *et al.* (2001), cuja complexidade temporal $O(\Delta n^2)$ era considerada limite superior para o PSCH-D até a presente data.

De forma sucinta, o que se faz é determinar, caso exista, um sumidouro fortemente conexo $H^* \neq V_T$ do grafo-testemunha G_T (V_T, E_T) para os grafos G_1,(V, E_1) e G_2 (V, E_2) da entrada do PSCH-D, o que é conseguido pela chamada ao procedimento *Encontra SFCPs*, cuja complexidade temporal $O(\Delta' n')$ traduz-se por $O(\Delta n^2)$, uma vez que o número de vértices n' de G_T é $O(n^2)$, ou seja, quadrático no tamanho n do conjunto V, e o grau máximo Δ' de G_T é da ordem do grau máximo Δ de G_1, pois um par de vértices não pode apresentar mais testemunhas do que possui adjacências obrigatórias. O conjunto H, formado pelos vértices $v \in V$ presentes no rótulo de algum vértice de H^* (leia-se $\langle v, w \rangle$ ou $\langle w, v \rangle$, para

algum *w*), seria, segundo a *Alegação 1*, conjunto homogêneo sanduíche do par (G_1, G_2).

Alegação 1: *O conjunto de vértices retornado pelo* Algoritmo 3 *é conjunto homogêneo sanduíche para o par de entrada.* (TANG *et al.*, 2001)

Em sua prova para a *Alegação 1*, TANG *et al.* (2001) afirmam, sem provar, que "se H^* é sumidouro fortemente conexo próprio de G_T então é vazio o conjunto-testemunha do conjunto H de todos os vértices do par de entrada que aparecem em algum rótulo de H^* ", considerando, provavelmente, que este fosse um fato auto-evidente. Mostraremos a seguir que esta afirmação é falsa e, por conseguinte, incorreto o algoritmo que nela se baseia.

A *Figura E1a* (presente no Apêndice 4), apresenta os grafos G_1 (V, E_1) e G_2 (V, E_2) como exemplo de entrada para o PSCH. Esta instância do problema, quando submetida ao *Algoritmo 3*, produz o grafo-testemunha G_T (V_T, E_T) da *Figura E1b* (também no Apêndice 4), que possui um SFCP H^* (constituído pelos vértices em destaque) correspondente ao conjunto $H = \{1, 2, 3, 4, 5, 6, 7\} \subset V$. (Verifica-se facilmente ser H^* um SFCP de G_T através da observação dos ciclos formados pelas arestas em destaque e do fato de não possui H^* aresta alguma de saída.) Ocorre que, pelo *Lema 2*, *H não pode ser* conjunto homogêneo sanduíche de (G_1, G_2), pois o vértice $8 \in V \setminus H$ é testemunha do par $\{1, 2\} \subset H$, contradizendo a *Alegação 1*.

O engano cometido por TANG *et al.* foi, possivelmente, crer que todo SFCP $H^* \subset V_T$ (associado ao conjunto $H \subset V$) seria o que

---

Algoritmo 3: *Admite CHS 3* (G_1, G_2) (TANG *et al.*, 2001)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$

Saída: SIM, apresentando um grafo-sanduíche $G_S(V, E_S)$ e um conjunto homogêneo H de G_S, caso exista; ou, caso contrário, NÃO, atestando-se a inexistência de um tal conjunto.

Variáveis: G_T (V_T, E_T): digrafo; G_S (V, E_S): grafo; H e H^* : conjuntos de vértices; C : conjunto de conjuntos de vértices; v, w : vértices.

Método:

1. $G_T \leftarrow$ *Constrói Grafo-Testemunha* (G_1, G_2)

2. $C \leftarrow$ *Encontra SFCPs* (G_T)

3. Se $C = \varnothing$ então

3.1. Retorne NÃO. Fim.

4. $H^* \leftarrow$ algum elemento do conjunto C *{ um SFCP de* G_T *}*

5. $H \leftarrow \{ v \mid \{\langle v, w\rangle, \langle w, v\rangle\} \cap H^* \neq \varnothing, w \in V \}$

6. $G_S \leftarrow$ *Monta Grafo-Sanduíche* (G_1, G_2, H)

7. Retorne SIM. Apresente G_S e H. Fim.

---

chamaremos de *fechado por pares*, ou seja, conteria os vértices $\langle v_i, v_j\rangle$ associados a todo par $\{v_i, v_j\}$ de vértices de H. Isto não é, no entanto, necessário, como vemos no contra-exemplo da *Figura E1b*, onde o vértice $\langle 1,2\rangle \in V_T$ não pertence ao SFCP H* associado ao conjunto H = {1, 2, 3, 4, 5, 6, 7} $\supset$ {1, 2}. Esta não-pertinência do vértice $\langle 1,2\rangle$ ao conjunto H^* é a responsável por fazer com que a existência do vértice 8, testemunha de {1, 2} e portanto de H, não seja "sentida" por H^*, ou seja, não implique a

existência de uma aresta de saída de um vértice v de H^* para algum vértice $\langle v,8\rangle \in V_T \setminus H^*$.

O contra-exemplo das *Figuras E1a* e *E1b* (chama-lo-emos *Contra-exemplo 1*) invalida, portanto, a saída do *Algoritmo 3*, onde um SFCP *qualquer* de G_T é utilizado para gerar um suposto CHS para os grafos da entrada. Conseqüentemente, um novo limite superior para o PSCH-D precisa ser estabelecido.

CAPÍTULO 6

Repensando o Problema

Foi visto, no Capítulo 5, que o algoritmo de TANG *et al.*(2001), considerado até então o método mais eficiente para a solução do PSCH-D, é incorreto.

O *Teorema 1*, de nossa autoria, dá uma correta caracterização de conjuntos homogêneos sanduíche, utilizando, ainda, o conceito de grafo-testemunha indroduzido por TANG *et al.* (2001), mas corrigindo o que os mesmos autores apresentaram como *Corolário 6* de seu artigo (TANG *et al.*, 2001) e que, baseado na veracidade da *Alegação 1* (que já provamos ser falsa, no Capítulo 5), erroneamente caracteriza os CHSs do par G_1 (V, E_1) e G_2 (V, E_2) como subconjuntos de V associados à união de um CFC F e de todos os outros CFCs atingíveis por F no grafo-testemunha G_T do par de entrada.

Teorema 1: *Um conjunto* $H \subset V$ *é conjunto homogêneo sanduíche para os grafos* $G_1 (V, E_1)$ *e* $G_2 (V, E_2)$ *se, e somente se, o conjunto fechado por pares* $K = \{ \langle x, y \rangle \mid x, y \in H \} \subset V_T$ *for um sumidouro do grafo-testemunha* $G_T (V_T, E_T)$ *para o par* (G_1, G_2).

Prova: se K é um sumidouro (não necessariamente fortemente conexo) do digrafo G_T e é fechado por pares, então não há aresta de saída de nenhum vértice $\langle x, y\rangle \in K$ para vértice algum de $V_T \setminus K$, de forma que não existe um par {x, y} $\subseteq H$ que possua testemunha não-pertencente a H. Em outras palavras, não há vértice algum de $V \setminus H$ que apresente adjacência obrigatória com algum elemento de H e que também apresente adjacência proibida com algum outro elemento de H. De forma que sempre é possível construir um grafo-sanduíche do par (G_1, G_2) para o qual H é conjunto homogêneo, como por exemplo o grafo construído por uma chamada ao procedimento *Monta Grafo-Sanduíche* (G_1, G_2, H), em que estão presentes, além das arestas obrigatórias, apenas as arestas opcionais que objetivam ligar um vértice $w \in V \setminus H$ *a* todos os elementos de H caso w seja obrigatoriamente adjacente a algum elemento de H. Para a prova de necessidade, seja H é um CHS e t uma testemunha para o par $\{x, y\} \in H$. Pelo *Lema* 2, $t \in H$ (ou H não seria CHS). Uma vez que $K \subset V_T$ contém vértices associados a *todos* os pares de vértices de $H \subset V$, K conterá necessariamente os vértices $\langle x, t\rangle$ e $\langle y, t\rangle$ (pois x, y e t pertencem a H), o que implica que as arestas $\langle x, y\rangle \rightarrow \langle x, t\rangle$ e $\langle x, y\rangle \rightarrow \langle y, t\rangle$ sejam internas a K, donde K é sumidouro.

Uma outra correta caracterização para conjuntos homogêneos sanduíche é dada por HABIB *et al.* (2002). Esta última, no entanto, não utiliza o conceito de grafo-testemunha e se baseia, na realidade, na idéia de CERIOLI *et al.* (1998), com vistas a ser explorada pela versão de enumeração do PSCH (PSCH-E), não embasando, contudo, nenhuma alternativa algorítmica mais eficiente para o PSCH-D.

É fato que o *Teorema 1* não sugere, igualmente, algoritmo eficiente para o PSCH-D, uma vez que não se tem um método para a localização de todos os sumidouros que não sejam necessariamente fortemente conexos. Contudo, algumas conjecturas foram levantadas na tentativa de fazer ainda valer o método de TANG *et al.* (2001) – se necessário, com algumas modificações – para o problema de decisão.

Tentativa 1: *Um par grafo-supergrafo* (G_1, G_2) *admite CHS se, e somente se, seu grafo-testemunha* G_T *apresenta algum SFCP.*

Se o enunciado pela *Tentativa 1* fosse correto, poder-se-ia utilizar o algoritmo $O(\Delta n^2)$ de TANG *et al.* (2001), sem maiores modificações, para resolver o PSCH-D, desde que não houvesse a necessidade da apresentação de um CHS como atestado da saída "SIM".

Infelizmente, a *Tentativa 1* falta com a verdade, como o comprova o *Contra-exemplo 2*, apresentado nas *Figuras E2a, E2b* e *E2c* (Apêndice 4). Temos, nessa instância, o caso de um grafo-testemunha que possui dois sumidouros fortemente conexos próprios (condensados na *Figura E2b* na forma dos vértices de rótulos *S* e *S'*, mas ilustrados por completo na *Figura E2c*) para um par grafo-supergrafo que não admite conjunto homogêneo sanduíche algum. Atente-se para as arestas em destaque, suficientes para mostrar a inexistência de outros sumidouros fortemente conexos, além de *S* e *S'*, que, por sua vez, não correspondem a CHSs devido à existência dos vértices 8 e 8', testemunhas respectivas dos subconjuntos de *V* associados a *S* e *S'*.

A busca pela viabilidade da utilização da idéia do *Algoritmo 3* continua na forma da *Tentativa 2*, cuja veracidade permitiria que o PSCH-D fosse resolvido por uma variante do *Algoritmo 3*, na qual ter-se-ia adicionado uma etapa final de verificação onde cada um dos conjuntos *H* obtidos dos SFCP *H** correspondentes seria testado quanto a ser ou não CHS do par de entrada. Caso algum desses conjuntos viesse a ser de fato um CHS, o algoritmo modificado retornaria "SIM", apresentando-o como atestado. Do contrário (caso nenhum dos SFCP de G_T gerasse um CHS válido), o algoritmo terminaria retornando "NÃO".

Tentativa 2: *Se um par grafo-supergrafo* (G_1, G_2) *admite algum CHS, então existe um SFCP* H* *de seu grafo-testemunha* G_T *que corresponde a algum CHS* H.

Mais uma vez, um contra-exemplo especialmente construído para este fim vem alterar nossos mais bem-intencionados planos. A partir do par de grafos G_1 (V, E_1) e G_2 (V, E_2) da *Figura E3a* (*Contra-exemplo 3*), é gerado o grafo-testemunha G_T ilustrado pela *Figura E3b*. Por possuir 253 vértices e mais de mil arestas, em sua representação optamos por condensar, na forma do vértice *H,* o subgrafo induzido de G_T que contém os cento e cinqüenta e três vértices associados aos pares contidos no subconjunto {1, 2, 3, 4, 5, 6, 7, 8, 9, 1', 2', 3', 4' ,5', 6', 7', 8', 9'} de *V*, e cuja topologia é idêntica à do digrafo da *Figura E2b*. Da mesma forma, nem todas as arestas estão desenhadas na figura, mas sim subentendidas, de acordo com explicação constante no próprio Apêndice 4, em seguida à figura em questão. Esta instância, que acreditamos ser a menor possível contraditória à *Tentativa 2*, possui apenas dois sumidouros fortemente

conexos, quais sejam *S* e *S'*, presentes no interior do subgrafo condensado no vértice *H*. Não é difícil verificar a corretude desta afirmativa, uma vez que no desenho da *Figura E3b* todos os vértices atingem *H*, que não apresenta arestas de saída. *H*, como se sabe, não é fortemente conexo, possuindo dois SFCPs *S* e *S'*, cada um dos quais associados a um subconjunto de *V* (a saber: {1, 2, 3, 4, 5, 6, 7} e {1', 2', 3', 4', 5', 6', 7'}, respectivamente) que *não é* CHS do par de entrada. Não obstante, os grafos G_1 e G_2 *admitem* o CHS {1, 2, 3, 4, 5, 6, 7, 8, 9, 1', 2', 3', 4', 5', 6', 7', 8', 9'}, isto é, o conjunto de todos os vértices de *V* que rotulam os vértices de *H*, *embora* H *não seja um sumidouro fortemente conexo de* G_T.

Dada a não confirmação da *Tentativa 2*, esvaem-se as esperanças de se utilizar o *Algoritmo 3*, ainda que com a inclusão da etapa de verificações, para a resolução do PSCH-D, uma vez que a existência de SFCPs em G_T não garante a localização de algum CHS para o par de entrada, que poderia, nesse caso: (i) possuir um CHS correspondente a algum dos SFCPs; (ii) possuir outro CHS, não- correspondente a SFCP algum e sem que algum SFCP corresponda a um CHS; ou ainda (iii) não possuir CHS algum. Em resumo, se algum SFCP é localizado, a *versão modificada* do *Algoritmo 3* (chamaremos assim ao *Algoritmo 3* acrescido da etapa final que verifica, para cada SFCP encontrado, se está associado ou não a um CHS) pode retornar tanto "SIM", com a exibição do grafo-sanduíche G_S contendo um CHS válido obtido de algum SFCP que passou na verificação, quanto "INCONCLUSIVO", significando que nenhum dos SFCPs correspondia a algum CHS do par de entrada, embora seja possível a existência de algum CHS que não corresponda a um SFCP de G_T.

Diante da invalidação do *Algoritmo 3* tal como proposto por TANG *et al.* (2001) e do fracasso da tentativa de redimi-lo por meio da adição da etapa final de verificação (que não evita, em alguns casos, uma saída inconclusiva), o *Algoritmo 2*, reescrito de (CERIOLI *et al*., 1998) voltaria a ser, portanto, o algoritmo correto mais eficiente para a solução do PSCH-D e sua complexidade de tempo, $O(n^4)$, o limite superior conhecido para o PSCH-D.

6.1 – Um novo algoritmo $O(n^4)$

Ainda que a versão original do *Algoritmo 3*, como apresentada por TANG *et al.* (2001), possa dar resultados incorretos para algumas entradas (como as dos *Contraexemplos 1* e *2*) e que mesmo uma eventual versão modificada que submeta os SFCPs a uma etapa de verificação não apresente uma saída conclusiva para todas as instâncias (como no caso do *Contraexemplo 3*), a idéia que o consubstanciou é por demais interessante e o esforço computacional – $O(\Delta n^2)$ – envolvido até a determinação dos SFCPs baixo o suficiente para que pensemos em administrá-la de alguma forma na elaboração de novos algoritmos.

Se o problema do *Algoritmo 3* modificado reside nos casos em que o grafo-testemunha G_T não é fortemente conexo (isto é, apresenta algum SFCP), por outro lado, nos casos em que G_T não possui SFCPs a saída conclusiva "NÃO" dada em tempo $O(\Delta n^2)$ é correta. Isto é garantido pelo *Corolário 1*, advindo diretamente do *Teorema 1* e do *Lema 5* (vide Apêndice 3, item A3.2), que enunciaremos a seguir.

Corolário 1: *Um par grafo-supergrafo admite CHS somente se seu grafo-testemunha possui algum SFCP.*

Prova: Seja o digrafo G_T (V_T, E_T) o grafo-testemunha do par de entrada (G_1, G_2) para o PSCH. Se G_T não possui SFCPs, então, pelo *Lema 5*, G_T não possui qualquer sumidouro próprio, com que então o *Teorema 1* garante a impossibilidade de haver algum CHS para o par (G_1, G_2).

Impulsionados pelo *Corolário 1*, que ratifica a corretude da saída "NÃO" dada pelo método de TANG *et al.* (2001) nos casos em que o grafo-testemunha não possui SFCPs, e tendo em mente que entradas que não admitam CHSs e cujo grafo-testemunha apresente SFCPs parecem ser bastante incomuns (dada a característica toda especial que é preciso haver em tais instâncias – cujo menor exemplo acreditamos ser o *Contraexemplo 2*), soa natural a idealização de um novo possível algoritmo, bastante simples, no qual uma dada instância de entrada seria primeiramente submetida ao *Algoritmo 3* modificado e, apenas em caso de não se haver chegado a uma resposta conclusiva, seria então passada ao método de CERIOLI *et al.* (1998), de forma que, intuitivamente, a complexidade de caso médio seja inferior à do algoritmo de CERIOLI *et al.* (1998) puro (*Algoritmo 2*), que teria voltado a ser o algoritmo mais eficiente para o PSCH-D.

O *Algoritmo 4* nada mais é, portanto, que uma combinação dos esforços de CERIOLI *et al.* (1998) e TANG *et al.* (2001). É determinado, em primeiro lugar, o conjunto *C* de todos os SFCPs do grafo-testemunha G_T para o par de entrada. Se *C* é vazio, O *Algoritmo 4* retorna "NÃO". Caso contrário, o algoritmo prossegue testando, para cada um dos SFCPs

---

Algoritmo 4: *Admite CHS 4* (G_1, G_2)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$

Saída: SIM, apresentando um grafo-sanduíche $G_S(V, E_S)$ e um conjunto homogêneo H de G_S, caso exista; ou, caso contrário, NÃO, atestando-se a inexistência de um tal conjunto.

Variáveis: G_T (V_T, E_T): digrafo; G_S (V, E_S): grafo; H e H^*: conjuntos de vértices; C: conjunto de conjuntos de vértices; v, w : vértices.

Método:

1. $G_T \leftarrow$ *Constrói Grafo-Testemunha* (G_1, G_2) *{ TANG* et al. *}*

2. $C \leftarrow$ *Encontra Todos SFCPs* (G_T)

3. Se $C = \varnothing$ então retorne NÃO. Fim.

4. Enquanto $C \neq \varnothing$ faça *{ Etapa de verificações }*

4.1. $H^* \leftarrow$ algum elemento do conjunto C *{ um SFCP de* G_T *}*

4.2. $H \leftarrow \{ v \mid \{\langle v, w\rangle, \langle w, v\rangle\} \cap H^* \neq \varnothing, w \in V \}$

4.3. Se *Encontra Conjunto-Testemunha* (H, G_1, G_2) = $\varnothing$ então

4.3.1 $G_S \leftarrow$ *Monta Grafo-Sanduíche* (G_1, G_2, H)

4.3.2. Retorne SIM. Apresente G_S e H. Fim.

4.4. $C \leftarrow C \setminus H$

5. Retorne *Admite CHS 2* (G_1, G_2). Fim. *{ CERIOLI* et al. *}*

H^* encontrados, se o subconjunto H de V utilizado para rotular os vértices de H^* é, de fato, um CHS de (G_1, G_2). Caso um deles o seja, o *Algoritmo 4* retorna "SIM", exibindo-o, junto ao grafo-sanduíche construído pelo procedimento *Monta Grafo-Sanduíche*, como atestado. Caso contrário, os grafos G_1 e G_2 são passados como parâmetros à chamada do *Algoritmo 2*,

que retorna, conclusivamente, um CHS, se existir algum para o par de entrada, ou ‘‘NÃO’’, caso não exista nenhum.

Prova de Corretude do Algoritmo 4

Caso uma saída ‘‘NÃO’’ seja apresentada como decorrência da ausência de SFCPs em G_T (linha 3), o *Corolário 1* garante sua exatidão. Se um CHS *H* é apresentado pela linha 4.3.2, a corretude do procedimento *Encontra Conjunto-Testemunha* atesta a inexistência de vértices-testemunha para *H*, donde será *H*, como já visto, conjunto homogêneo do grafo G_S retornado pelo procedimento *Monta Grafo-Sanduíche*. Finalmente, caso nenhum CHS tenha sido ainda encontrado e o método de CERIOLI *et al.* (1998) seja então disparado à linha 5 do *Algoritmo 4*, a prova de corretude do *Algoritmo 2* avaliza a saída que será apresentada.

Análise de Complexidade do Algoritmo 4

A determinação do grafo-testemunha e de seus SFCPs (linhas 1 e 2) é feita em tempo $O(\Delta n^2)$, como no *Algoritmo 3*. O laço da linha 4, que compreende a etapa onde é verificada a correspondência de cada SFCP de G_T a um eventual CHS do par de entrada, executa, no máximo $\varphi(G_1, G_2) = O(n^2)$ iterações, onde $\varphi(G_1, G_2)$ é o número de sumidouros fortemente conexos do grafo-testemunha de (G_1, G_2) que não são fechados por pares, ou seja, o tamanho da maior seqüência de SFCPs que não corresponderão, segundo o *Teorema 1*, a um CHS da entrada. (Obs.: o limite $O(n^2)$ para $\varphi(G_1, G_2)$ é trivial, uma vez que um sumidouro fortemente conexo é um CFC e os CFCs particionam G_T, digrafo com $O(n^2)$ vértices. A justeza deste limite será, no entanto, discutida mais adiante, por se tratar do ponto crítico da análise de complexidade do *Algoritmo 5* que

iremos propor.) Cada uma das iterações do laço da linha 4 consome tempo $O(n^2)$ na chamada a *Encontra Conjunto-Testemunha*. O tempo computacional empregado das linhas 1 a 4.4 – que consistem no que apelidáramos de *versão modificada* do *Algoritmo 3* de TANG *et al.* (2001) – é, portanto, O [n^2 ($\Delta + \varphi$ (G_1, G_2)], igual ou inferior à complexidade de pior caso $O(n^4)$ da chamada ao *Algoritmo 2* de CERIOLI *et al.* (1998) que ocorre na linha 5, complexidade esta – $O(n^4)$ – que constituirá a complexidade temporal de pior caso para todo o *Algoritmo 4*.

Finalmente, apresentaremos no próximo capítulo o *Algoritmo 5*, uma evolução natural do *Algoritmo 4*, que resolve, tão eficientemente quanto se saiba ser possível, a versão de decisão do Problema-Sanduíche para Conjunto Homogêneos.

CAPÍTULO 7

Novo Limite Superior para o PSCH-D

Foi visto, nos Capítulos 5 e 6, que o algoritmo introduzido por TANG *et al.* (2001) para a solução do PSCH-D, embora criativo e de bastante interesse teórico, falha em alguns casos em que SFCPs do grafo-testemunha não correspondem a CHSs do par de entrada. Sua complexidade de tempo $O(\Delta n^2)$, em conseqüência, viria novamente dar lugar à complexidade $O(n^4)$ do algoritmo de CERIOLI *et al.* (1998), que o precedeu, como limite superior para o problema.

7.1 – Um algoritmo $O\{ [\Delta + \varphi (G_1, G_2)] n^2 \} = O(mn^2)$

Com a mesma idéia do *Algoritmo 4*, qual seja a de submeter uma instância do PSCH-D ao método de TANG *et al.* (2001) com a etapa de verificações e só então, no caso em que nenhum CHS tenha sido encontrado a partir de um SFCP do grafo-testemunha, submetê-la ao método de CERIOLI *et al.* (1998) que liquidaria o assunto em tempo $O(n^4)$, foi construído o *Algoritmo 5*, o qual não se utiliza do método de CERIOLI *et al.* (1998) original (que parte de cada um dos possíveis pares de vértices de V como candidato a CHS), mas sim de uma versão também modificada desse método, onde os conjuntos iniciais a serem investigados e sujeitos

ao processo de incorporação sucessiva de testemunhas seriam justamente os SFCPs provenientes da etapa de verificação do *Algoritmo 4*, nenhum dos quais correspondente a um CHS (do contrário, o algoritmo já teria parado). Isto é possível graças ao *Corolário 2*, que não é mais que conseqüência direta do *Teorema 1* (Capítulo 6) e do *Lema 4* (Apêndice 3).

Corolário 2: *Se* H *é CHS de um par grafo-supergrafo, então o conjunto fechado por pares* $K = \{ \langle x, y \rangle \mid x, y \in H \} \subset V_T$ *ou é SFCP ou contém propriamente um SFCP do grafo-testemunha* $G_T (V_T, E_T)$ *para esse par.*

Prova: Se *H* é CHS, então, pelo *Teorema 1*, *K* é sumidouro, de forma que o *Lema 4* garante que $K \supseteq S$, para algum SFCP $S \subset V_T$.

Em verdade, o *Corolário 2* nos fornece mais do que a garantia de que um CHS será encontrado, caso exista algum, pelo algoritmo de CERIOLI *et al.* (1998) modificado que parta apenas dos conjuntos de vértices de *V* associados a SFCPs do grafo-testemunha. Como, *para um dado candidato a CHS*, a complexidade de tempo da etapa de testes ao final do método de TANG *et al.* (2001) modificado é a mesma complexidade $O(n^2)$ da etapa de incorporação de testemunhas do método de CERIOLI *et al.*(1998) modificado, e não mais baixa, deixa de fazer sentido que primeiramente se teste um SFCP *H** para se descobrir se *H** corresponde ou não a um CHS e só então, caso não se verifique esta correspondência, o conjunto $H \subset V$ obtido a partir de $H^* \subset V_T$ seja utilizado para iniciar o processo de incorporação sucessiva de testemunhas. O *Corolário 2* permite, sem perda de completude ou aumento de

complexidade, que os SFCPs encontrados em G_T sejam passados *diretamente* à etapa de incorporação de testemunhas, o que acarreta uma certa simplificação – e, como veremos, *redução da complexidade temporal assintótica* – deste novo algoritmo que se está apresentando em relação ao *Algoritmo 4* que o antecedeu, uma vez que no *Algoritmo 4* era justificada a existência da etapa de verificação dos SFCPs, passível de encontrar um CHS em tempo – $O[\varphi(G_1,G_2)n^2]$ – possivelmente inferior (e nunca superior) ao da etapa de incorporação de testemunhas – $O(n^4)$ – seguinte.

Para que sejam determinados os conjuntos auxiliares iniciais *A*, *N* e *D*, bem como o primeiro conjunto-testemunha *T* para cada candidato inicial a ser submetido ao método de CERIOLI *et al.* (1998) – que, em sua versão modificada pode ter tamanho maior que dois – será utilizada a rotina *Obtém Conjuntos Auxiliares Iniciais* (*Procedimento 7*) que apresentamos logo em seguida.

Sintetizando o funcionamento do *Procedimento 7*: o que se faz é adicionar ao conjunto *A*, inicialmente vazio, todos os vértices de $V \setminus H$ que tem adjacência obrigatória com algum elemento *x* de *H* (linha 1.1). Em seguida, serão retirados os elementos que, além de alguma adjacência obrigatória, apresentarem também alguma adjacência proibida com algum elemento de *H* (linha 2.1), pois estes serão, na realidade, testemunhas de *H*. O mesmo é feito para a construção do conjunto *N*, nas linhas 1.2 e 3.1 (apenas invertam-se as palavras *obrigatória* e *proibida* na explicação acima). Em seguida, o já conhecido procedimento *Encontra Conjunto-Testemunha* (*Procedimento 2*) determina o conjunto-testemunha *T* inicial e, finalmente, *D* consistirá, segundo a linha 5, de todo vértice de *V* que não pertencer a *A*, *N*, *T* ou *H*, completando sua partição. Todos os laços são, claramente, executados em tempo $O(n^2)$ e dominam a complexidade do

Procedimento 2, O($|V \setminus H|$. $|H|$), de forma que os conjuntos auxiliares iniciais ficam todos determinados em tempo O(n^2).

Procedimento 7: *Obtém Conjuntos Auxiliares Iniciais* (H, G_1, G_2, A, N, D, T)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2); um conjunto H, candidato a CHS; os conjuntos A, N, D e T, como parâmetros de saída.

Saída: Atualização dos conjuntos A, N, D e T.

Variáveis: x : vértice.

Método:

1. $A \leftarrow \varnothing$; $N \leftarrow \varnothing$

1. Para cada vértice $x \in H$ faça

1.1. $A \leftarrow A \cup N_1(x) \setminus H$

1.2. $N \leftarrow N \cup (V \setminus N_2(x)) \setminus H$

2. Para cada vértice $x \in A$ faça

2.1. $A \leftarrow A \setminus (V \setminus N_2(x))$

3. Para cada vértice $x \in N$ faça

3.1. $N \leftarrow N \setminus N_1(x)$

4. $T \leftarrow$ *Encontra Conjunto-Testemunha* (H, G_1, G_2)

5. $D \leftarrow V \setminus (A \cup N \cup T \cup H)$

6. Fim.

Apresentamos, a seguir, o pseudo-código para o novo *Algoritmo 5*, que vem baixar, por um fator de pelo menos ($n^2 \div m$), o limite superior conhecido para o problema de decisão.

Algoritmo 5: *Admite CHS 5* (G_1, G_2)

Entrada: Dois grafos G_1 (V, E_1) e G_2 (V, E_2), onde $E_1 \subseteq E_2$

Saída: SIM, apresentando um grafo-sanduíche $G_S(V, E_S)$ e um conjunto homogêneo H de G_S, caso exista; ou, caso contrário, NÃO, atestando-se a inexistência de um tal conjunto.

Variáveis: G_T (V_T, E_T): digrafo; G_S (V, E_S): grafo; A, D, N, T, H e H^* : conjuntos de vértices; C : conjunto de conjuntos de vértices; v, w : vértices.

Método:

1. $G_T \leftarrow$ *Constrói Grafo-Testemunha* (G_1, G_2) *{ TANG* et al. *}*

2. $C \leftarrow$ *Encontra Todos SFCPs* (G_T)

3. Se $C = \varnothing$ então retorne NÃO. Fim.

4. Enquanto $C \neq \varnothing$ faça *{ CERIOLI* et al. *modificado }*

4.1. $H^* \leftarrow$ algum elemento do conjunto C *{ um SFCP de* G_T *}*

4.2. $H \leftarrow \{ v \mid \{\langle v, w\rangle, \langle w, v\rangle\} \cap H^* \neq \varnothing, w \in V \}$

4.3. *Obtém Conjuntos Auxiliares Iniciais* (H, G_1, G_2, A, N, D, T)

4.4. Enquanto $H \neq V$ faça

4.4.1. Se $T = \varnothing$ então *{* H *é um CHS }*

4.4.1.1 $G_S \leftarrow$ *Monta Grafo-Sanduíche* (G_1, G_2, H)

4.4.1.2. Retorne SIM. Apresente G_S e H. Fim.

4.4.2. $H \leftarrow H \cup T$

4.4.3. *Atualiza Conjunto-Testemunha* (G_1, G_2, T, A, N, D)

4.5. $C \leftarrow C \setminus H^*$

5. Retorne NÃO. Fim. *{ Exauriu todos os SFCNFs }*

Prova de Corretude do Algoritmo 5

O funcionamento do *Algoritmo 5* é, até o ponto (linha 4) onde começa o método de CERIOLI *et al.* (2001) modificado, idêntico ao do método de TANG *et al.* (2001) original, ou seja, é determinado o grafo-testemunha para o par de entrada, após o que são localizados seus sumidouros fortemente conexos próprios, se existirem. Caso não existam, o *Corolário 1* garante que o algoritmo possa parar, retornando "NÃO". Caso o grafo-testemunha não seja fortemente conexo (isto é, caso possua algum SFCP), a computação chega à linha 4, onde os subconjuntos de *V* associados aos SFCPs são submetidos diretamente ao processo de identificação e incorporação de testemunhas. Se há algum CHS *F* para o par de entrada, então, segundo o *Corolário 2*, *F* contém (é possível que não propriamente) o conjunto *H* de vértices que rotulam um SFCP *H** do grafo testemunha. Dito isto, e como garante a prova de corretude do algoritmo de CERIOLI *et al.* (1998) puro (*Algoritmo 2*), o CHS *F* (ou outro CHS $F' \subset$ F) será localizado pelo algoritmo se um subconjunto de *F* (como o é o conjunto *H*) for utilizado como candidato inicial para iniciar o processo de incorporação de testemunhas, o que conclui a prova.

Análise de Complexidade do Algoritmo 5

As únicas e importantes diferenças do *Algoritmo 5* em relação ao *Algoritmo 4* são: (i) não há mais a etapa de verificações, e a parte que cabe ao método de TANG *et al.* (2001) retorna "NÃO" e pára, caso o grafo-testemunha não apresente SFCPs, ou passa diretamente os SFCPs encontrados à etapa que concerne ao método de CERIOLI *et al.* (1998) modificado; e (ii) a etapa da incorporação sucessiva de testemunhas de CERIOLI *et al.* (1998) não partirá de todos os pares de *V*, mas investigará

apenas os conjuntos associados a SFCPs de G_T. A primeira parte (linhas 1 a 3) é executada, como já o sabemos, em tempo $O(\Delta n^2)$. A segunda (laço iniciado na linha 4) consome tempo $O(n^2)$ *para cada conjunto candidato inicial* (vide Análise de Complexidade do *Algoritmo 2*, no Capítulo 5). Portanto, a complexidade temporal de pior caso do *Algoritmo 5* depende da quantidade de candidatos iniciais a serem investigados no pior caso, ou seja, do tamanho da maior seqüência de SFCPs do grafo-testemunha G_T *que não corresponderão a CHSs*, pois o algoritmo pára quando algum CHS é encontrado. Como, pelo *Teorema 1*, todo SFCP que é fechado por pares corresponde a um CHS, então o tamanho máximo dessa seqüência será o número de SFCNFs de G_T.

A função φ (G_1, G_2) retorna o número de sumidouros fortemente conexos não-fechados por pares (SFCNFs) do grafo testemunha G_T para o par (G_1, G_2). Podemos escrever a complexidade total do *Algoritmo 5* como sendo, portanto, $O[\Delta n^2 + \varphi(G_1,G_2)n^2]$, ou $O\{ [\Delta+\varphi(G_1,G_2)] n^2 \}$.

O *Teorema 2* refina a análise dando um limite superior para φ (G_1, G_2) abaixo do trivial $O(n^2)$ (limite para o número de vértices e CFCs do grafo-testemunha), sendo, portanto, fundamental para o estabelecimento do novo limite superior para o PSCH-D.

Teorema 2: *O número* φ (G_1, G_2) *de sumidouros fortemente conexos próprios que não são fechados por pares, no grafo-testemunha* G_T (V_T, E_T) *para os grafos* G_1 (V, E_1) *e* G_2 (V, E_2), *é* $O(m)$.

Prova: Seja G_T (V_T, E_T) o grafo-testemunha para os grafos G_1 (V, E_1) e G_2 (V, E_2). Seja S um sumidouro fortemente conexo não-fechado por pares

(SFCNF) de G_T e seja $\langle x, y \rangle$ um vértice de S. É preciso que $\langle x, y \rangle$ tenha alguma aresta de saída, do contrário $\langle x, y \rangle$ seria um sumidouro e não poderia estar contido propriamente em S. (Além disso, S \ { $\langle x, y \rangle$ } é não-vazio, pois { $\langle x, y \rangle$ } *é fechado por pares*). Seja $\langle x, y \rangle \rightarrow \langle x, t \rangle$ uma aresta de S. O vértice $t \in V$ é, portanto, testemunha do par $\{x, y\} \subset V$, de forma que uma das arestas $[x, t]$ ou $[y, t]$ é obrigatória e a outra, proibida. Sem perda de generalidade, seja $[x, t]$ a aresta obrigatória. Rotulemos o SFCNF S por meio desta aresta, ou seja, *Rótulo* (S) = "$[x, t]$". Nenhum outro SFCNF S' poderá ser rotulado por esta mesma aresta $[x, t]$. Do contrário, teria que haver, pela maneira como o rótulo é atribuído, uma aresta $\langle x, w \rangle \rightarrow \langle x, t \rangle$ (ou $\langle t, w \rangle \rightarrow \langle x, t \rangle$) em S', para algum $w \in V$. Por S' ser SFC, $\langle x, t \rangle$ deve pertencer a S'. O que é absurdo, pois a interseção de dois SFCs é vazia (um SFC é um CFC, e CFCs estabelecem uma partição nos vértices de um digrafo) e $\langle x, t \rangle$ já pertence, por hipótese, a S, que é um SFC. Como todo SFCNF de G_T pode ser rotulado por uma aresta obrigatória do par (G_1, G_2) segundo o método supracitado, e por dois SFCNFs não poderem ter o mesmo rótulo, conclui-se que o número de SFCNFs de G_T não pode ser maior do que o número de arestas obrigatórias. Um raciocínio análogo, mas que utilizasse as arestas proibidas no lugar das obrigatórias para rotular os SFCNFs, inferiria que o número de SFCNFs do grafo-testemunha também não pode ser maior que o número de arestas proibidas do par de entrada. De forma que chegamos ao limite O(Min $\{m_O, m_P\}$) = O(m) para φ (G_1, G_2), onde $m_O = |E_1|$ é o número de arestas obrigatórias e $m_P = C_{n,2} - |E_2|$ é o número de arestas proibidas do par de entrada.

O *Teorema 2* permite-nos, finalmente, reescrever a complexidade temporal de pior caso do *Algoritmo 5* como $O([\Delta + \text{Min}\ \{m_O, m_P\}]\ n^2) = O(mn^2)$, que passa a ser considerada, a partir de agora, o limite superior conhecido para o PSCH-D.

CAPÍTULO 8

Exponencialidade do Problema de Enumeração

A versão de enumeração do Problema-Sanduíche para Conjuntos Homogêneos (PSCH-E) pode aparecer, na prática, da necessidade de se encontrar o *melhor* conjunto homogêneo sanduíche dentre todos aqueles que um par grafo-supergrafo possa admitir. Em aplicações onde os vértices e/ou arestas são dotados de pesos (ou "custos"), por exemplo, pode-se desejar determinar o CHS que possua o menor custo total. Para este fim, e na ausência de um método que o atinja direta e eficientemente, a enumeração de todos os CHSs, objetivo do PSCH-E, surge como uma estratégia natural. Não obstante, a enumeração irrestrita da totalidade de CHSs possivelmente admitidos por um par (G_1, G_2) pode levar à listagem de um número de CHSs exponencial no tamanho da entrada, já que, mesmo no caso particular onde $G_1 = G_2$, este número pode chegar a esta magnitude. Basta observarmos, por exemplo, o grafo completo $K_{|V|}$, onde qualquer subconjunto próprio não-unitário de seus vértices é conjunto homogêneo.

Na tentativa de se evitar a possibilidade de uma tal enumeração, foi introduzido, com sucesso, o conceito de *conjunto homogêneo forte* (HABIB *et al.*, 2002). Um conjunto homogêneo H de um grafo G (V, E) é conjunto homogêneo forte se, e somente se, qualquer conjunto homogêneo H' de G tal que $H \cap H' \neq \varnothing$ satisfaz $H \subset H'$ ou $H' \subset H$. Os conjuntos homogêneos

fortes de um grafo são os módulos não-triviais que aparecem em sua árvore de decomposição modular (vide Capítulo 3), de forma que existe um número O(*n*) de conjuntos homogêneos fortes *para um grafo isolado*. Em se tratando de *grafos-sanduíche*, no entanto, nem mesmo que se restrinja a saída do PSCH-E para apenas conjuntos homogêneos fortes presentes em algum grafo-sanduíche do par de entrada (*conjuntos homogêneos sanduíche fortes*, ou CHSF) estará eliminada a possibilidade de haver um número exponencial de conjuntos, como comprova o exemplo a seguir.

Seja a tripla (V, E_O, E_P) a entrada para o PSCH-E, onde E_O e E_P representam (vide Capítulo 2), respectivamente, as arestas obrigatórias e proibidas. Na ilustração da *Figura 10*, as arestas desenhadas em linha contínua são as arestas de E_O, enquanto as desenhadas em linha tracejada são as de E_P.

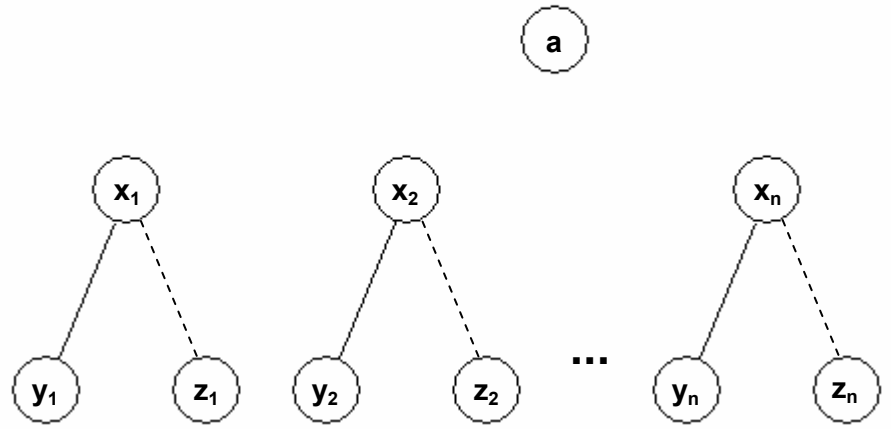


Figura10 – Instância contendo um número exponencial de CHSFs.

Enumeremos, agora, todos os CHSFs do exemplo da *Figura 10* que contêm o vértice *a*. Para cada um dos conjuntos { x_i, y_i, z_i } podem ser encontrados três CHSs contendo *a*: $M_{i,1}$ = { *a*, x_i }, que é forte no grafo-sanduíche $G_{i,1}$ = { V, $E_{i,1}$ } onde $E_{i,1}$ = $E_O \cup$ { [*a*, y_i] } e, similarmente,

$M_{i,2}$ = { a, y_i } e $M_{i,3}$ = { a, x_i, y_i }, que são também fortes nos grafos-sanduíche (minimais em arestas) que os contêm.

Qualquer união arbitrária de dois ou mais desses CHSFs associados aos conjuntos { x_i, y_i, zi } dá origem a um novo CHSF, como mostrado por HABIB *et al.* (2002), totalizando, portanto, 4^n CHSFs, o que evidencia a existência de instâncias para o PSCH-E que admitem um número exponencial de CHSFs.

CAPÍTULO 9

Conclusão

9.1 – Retrospectiva

Neste trabalho, abordamos o Problema-Sanduíche para Conjuntos Homogêneos em grafos. Primeiramente apresentamos os conceitos de problema-sanduíche e conjunto homogêneo e em seguida traçamos um histórico das técnicas que foram desenvolvidas para solucioná-lo. Como contribuição teórica, provamos, com contra-exemplos, ser incorreto o algoritmo que vinha sendo há dois anos considerado o mais eficiente para sua versão de decisão e cuja complexidade temporal respondia por seu limite superior conhecido. Adicionalmente, propusemos um novo algoritmo que passa a ser o mais eficiente para o problema e que lhe estabelece um novo limite superior.

9.2 – Pesquisa Futura

Para a análise de complexidade do *Algoritmo 5*, provamos que o número φ (G_1, G_2) de sumidouros fortemente conexos não-fechados por pares (SFCNFs) do grafo-testemunha G_T para um par grafo-supergrafo (G_1, G_2) é O(Min{m_O, m_P}) = O(m). É possível, no entanto, que este limite

não seja justo, uma vez que não encontramos um exemplo que alcançasse esta marca.

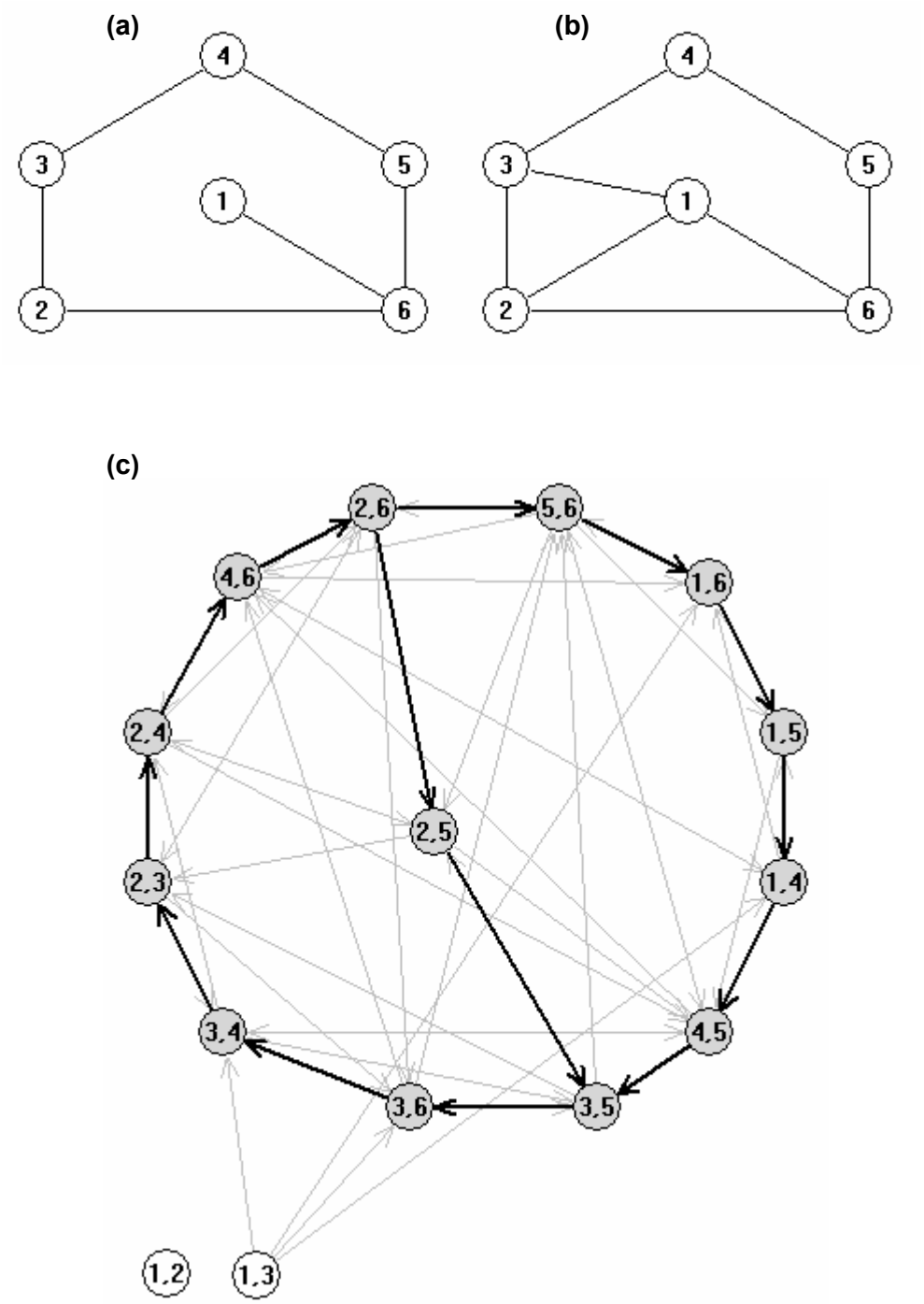


Figura 10 – Sumidouro Fortemente Conexo Não-Fechado por Pares

Uma sugestão de pesquisa futura útil compreende a busca de uma instância que possua, de fato, O(*m*) SFCNFs, ou de um refinamento deste limite.

Verifica-se, por exaustão, que a menor estrutura (ou subgrafo induzido) que gera um SFCNF em G_T, quando presente nos grafos de entrada, é aquela baseada no ciclo de cinco vértices e cinco arestas obrigatórias, com a presença de um vértice adicional ligado ao ciclo por uma aresta obrigatória, duas opcionais e duas proibidas, como mostram as *Figuras 10a* e *10b*. A *Figura 10c* apresenta todos os vértices de G_T correspondentes aos pares de vértices dessa estrutura, onde se nota a existência de um SFCNF (em destaque, na figura).

Existem, decerto, infinitas estruturas como esta (que veio a ser, inclusive, o ponto de partida para a criação do *Contraexemplo 1* – vide Capítulo 5) e que produzem SFCNFs no grafo-testemunha. Todo ciclo sem cordas de tamanho maior ou igual a cinco, por exemplo, induzido em G_1 e G_2 e com a presença de um vértice externo especial que apresente adjacências entre obrigatórias, proibidas e opcionais a vértices do ciclo (como o vértice 1 da *Figura 10*) gerarão SFCNFs. Utilizaremos o termo *geradores de SFCNFs* para referimo-las.

Os vértices de um *gerador de SFCNFs* aparecerão, portanto, nos rótulos tanto de vértices de G_T que farão parte de um sumidouro fortemente conexo, quanto de vértices (pelo menos um) que não façam parte do sumidouro, do contrário o sumidouro seria fechado por pares.

Ocorre que, muito embora um mesmo vértice v possa aparecer nos rótulos de dois vértices $\langle v, x\rangle, \langle v, y\rangle \in V_T$ que estejam em CFCs distintos ou mesmo em SFCNFs distintos de G_T, cada *par de vértices* $\{v, w\} \subset V$ dá origem ao vértice $\langle v, w\rangle \in V_T$ que só pode aparecer em *um único* CFC de

G_T (pois os CFCs de um digrafo constituem uma partição) e, portanto, em um único SFCNF. É verdade que pares de vértices de um gerador de SFCNFs não apenas rotulam, em G_T, vértices confinados em um SFCNF, mas também pelo menos um vértice *externo* ao sumidouro (do contrário, o sumidouro seria fechado por pares). No entanto, é suficiente observarmos que um gerador de SFCNFs produzirá pelo menos um vértice $\langle x, y \rangle \in V_T$ que *estará* confinado no sumidouro (do contrário, o sumidouro seria vazio) para inferirmos que, ainda que geradores de SFCNFs (como o da *Figura 10a* e *b*) presentes num par grafo-supergrafo não sejam necessariamente disjuntos, todo gerador conterá algum *par de vértices exclusivo* $\{v, w\} \in V$ que não estará contido em nenhum outro gerador (pela existência do vértice $\langle v, w \rangle \in V_T$ que fará parte de um sumidouro – SFCNF – e, portanto, de *apenas um* SFCNF). Em verdade haverá, certamente, bem mais que um par de vértices exclusivos por gerador, pois, como se vê, o menor gerador de SFCNFs (*Figura 10*) confina 13, dos 15 vértices que rotula, no SFCNF que gera.

O fato de exigir uma estrutura toda especial para gerar um SFCNF nos faz crer, intuitivamente, que o limite O(m) para o número de SFCNFs é bastante alto. Na verdade, decidimos investigar a possibilidade de se quebrar o limite O(n) para o número de geradores de SFCNFs num par grafo-supergrafo, ordem de grandeza essa que seria de fato um limitante caso fosse impossível existir geradores que não contivessem vértices exclusivos, ou seja, vértices que não fizessem parte de outros geradores (pois haveria uma bijeção entre cada gerador e algum de seus vértices exclusivos).

Apresentamos, agora, o resultado dessa pesquisa, que terminou por fornecer um exemplo de grafo-testemunha que contém $\Omega(n \log n)$ SFCNFs.

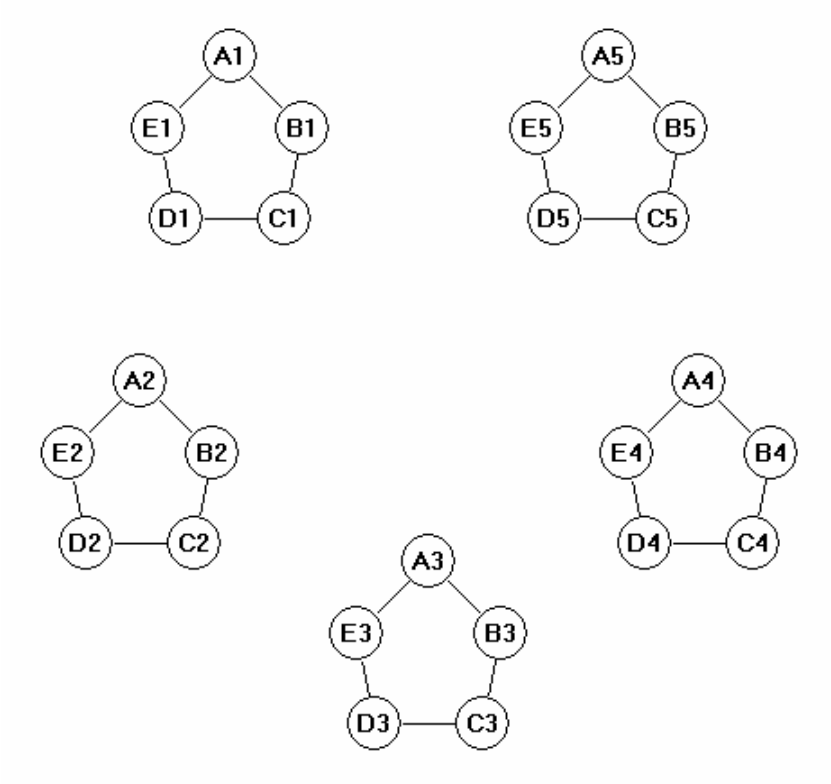


Figura 11 – Núcleos de Geradores de SFCNFs

Seja a entrada de um PSCH-D os grafos G_1 (V, E_1) e G_2 (V, E_2) tais que contenham $g = \Omega(n \div 5)$ ciclos disjuntos de 5 vértices com arestas obrigatórias e cordas proibidas, como na *Figura 11*. Chama-lo-emos *núcleos*. Cada um desses núcleos gera, no grafo-testemunha, um SFC *fechado* por pares. Contudo, a adição de um simples vértice externo $t \in V$ ligado aos vértices de cada núcleo por uma aresta obrigatória, duas opcionais e duas proibidas (como no exemplo da *Figura 10*) transforma estes SFCs fechados por pares em SFCNFs, de forma que nossa estratégia consistirá em gerar, por meio da adição de arestas mas não de

vértices, o maior número possível de núcleos, sem que jamais se crie, no grafo-testemunha, uma aresta de saída de um SFCNF que já esteja sendo (ou venha a ser) gerado.

Separemos os *g* geradores disjuntos de *V* em grupos de 5, ainda como no exemplo da *Figura 11*.

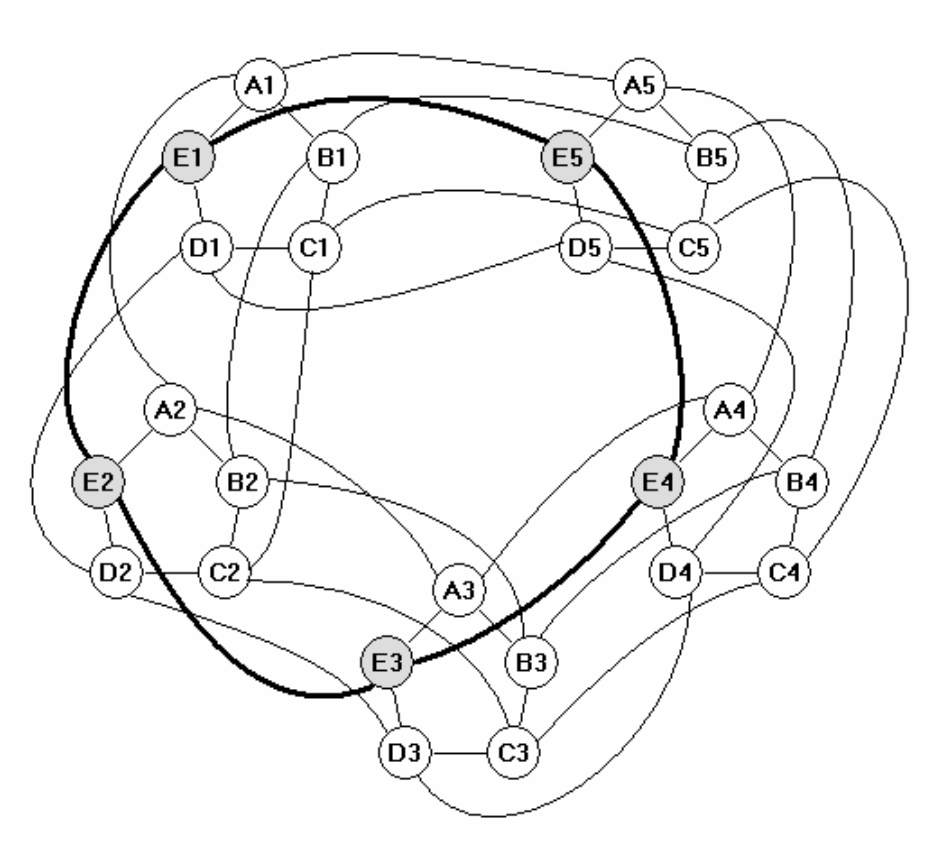


Figura 12 – Criação de novos núcleos de geradores de SFCNFs

Seja um grupo de *5* geradores. Como um gerador não pode compartilhar mais que um vértice de seu núcleo com o núcleo de outro gerador, então precisaremos de um vértice de cada gerador do grupo para constituir um novo núcleo. Atribuamos os rótulos *Ai*, *Bi*, *Ci*, ..., *Ei* (*i* = 1,...,5) aos vértices de cada gerador V_i deste grupo. Construiremos 5 novos núcleos apenas com a inclusão das arestas obrigatórias [*Ai*, *A*(*i*+1)]

(i = 1,..., 4) e [$A5$, $A1$], [Bi, $B(i+1)$] (i = 1,..., 4) e [$B5$, $B1$], ..., [Ei, $E(i+1)$] (i = 1,..., 4) e [$E5$, $E1$], como ilustrado pela *Figura 12* para o caso-exemplo introduzido pela *Figura 11*.

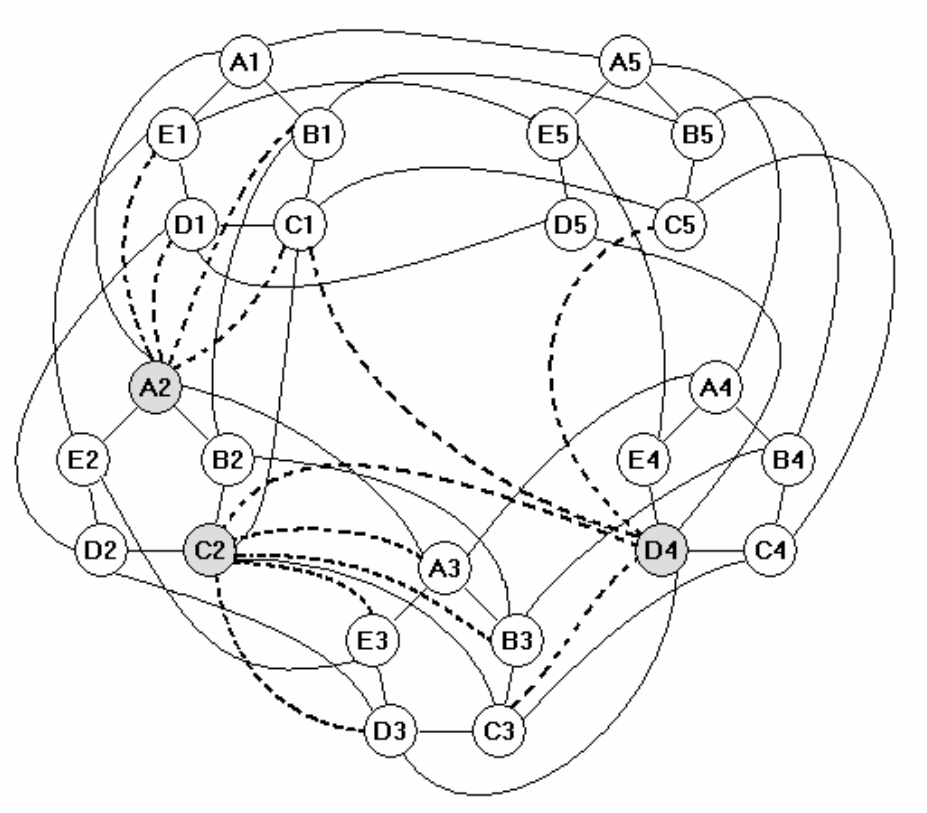


Figura 13 – Adição de arestas opcionais

Não permitiremos que algum par de vértices {x, y} de um núcleo L_i possua uma testemunha externa pertencente a outro núcleo $L_j \neq L_i$. Isso será conseguido com a adição de diversas arestas opcionais, que impedirão, justamente, a presença de testemunhas externas a algum par de vértices dos núcleos. Explicitamente, as arestas opcionais que se deve acrescentar, sempre para evitar que uma aresta obrigatória recém-introduzida gere uma testemunha indesejada, são (os exemplos referem-se às ilustrações da *Figura 13*, onde apenas algumas arestas-exemplo

foram acrescentadas ao desenho da *Figura 12* – a linha tracejada indica aresta opcional):

(Obs.: a função *f*(*k*) retorna a *k*-ésima letra do alfabeto.)

- [*f*(*j*)*i*, *f*(*j'*)(*i*+*1*)], (*j'*≠ *j*; *i* = 1,...,4) e [*f*(*j*)5, *f*(*j'*)1] (ex.: [*C*2, *A*3], [*C*2, *B*3], [*C*2, *D*3] e [*C*2, *E*3]);
- [*f*(*j*)*i*, *f*(*j'*)(*i*-*1*)], (*j'*≠ *j*; *i* = 2,...,5) e [*f*(*j*)1, *f*(*j'*)5] (ex.: [*A*2, *B*1], [*D*4, *C*1], [*D*4, *C*2] e [*C*3, *C*5]);
- [*f*(*j'*)*i*, *f*(*j*)*i'*] ([*f*(*j'*)*i*, *f*(*j*)*i*] é aresta obrigatória; *i'* ≠ *i*) (ex.: [*D*4, *C*1], [*D*4, *C*2], [*D*4, *C*3] e [*D*4, *C*5]).

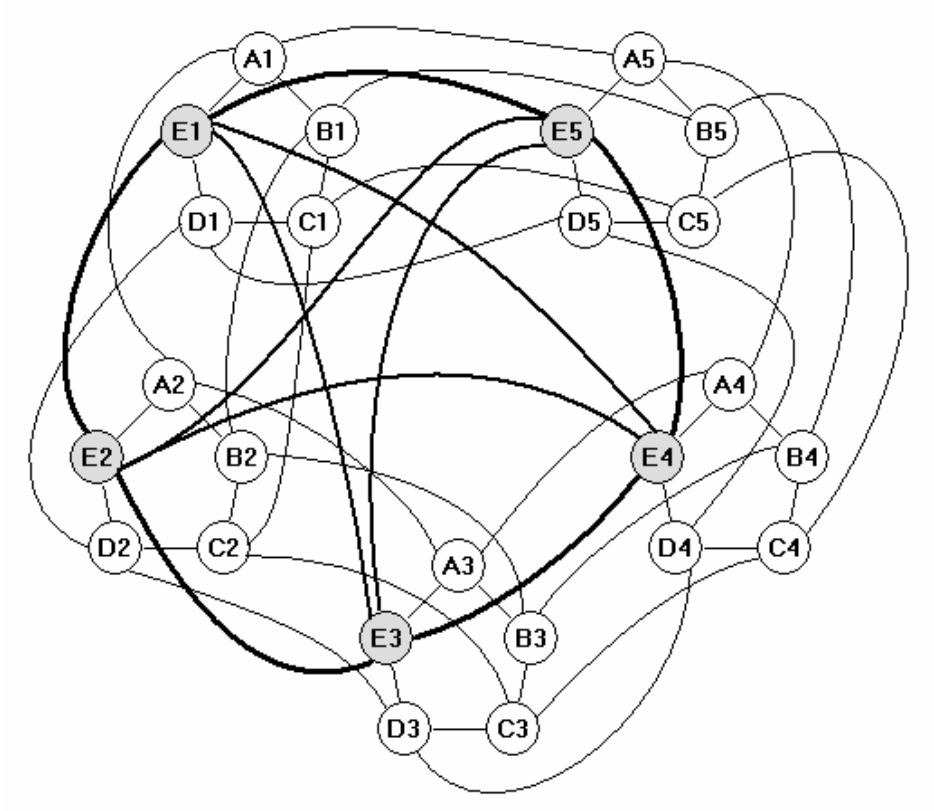


Figura 14 – Adição de arestas proibidas

Assim como há a necessidade de serem inseridas arestas opcionais para manter a integridade dos núcleos, isto é, para impedir que passem a existir testemunhas externas para algum de seus pares, também não devemos esquecer de impedir cordas nos núcleos, de forma a eliminar a possibilidade de que algum núcleo venha a conter algum conjunto-homogêneo (que corresponderia, pelo *Teorema* 1, a um sumidouro fechado por pares e não poderia, portanto, estar contido em outro sumidouro, que é o SFCNF que se intenta gerar). Explicitamente, as arestas proibidas que se deve acrescentar, após a criação dos 5 novos núcleos paralelos, são (os exemplos referem-se às arestas proibidas, algumas das quais representadas, em destaque, no interior do ciclo *E*1-*E*2-*E*3-*E*4-*E*5, na *Figura 14* – atenção: as arestas *do ciclo*, evidentemente, não são proibidas, mas obrigatórias):

- [*f*(*j*)1, *f*(*j*)3], [*f*(*j*)1, *f*(*j*)4], [*f*(*j*)2, *f*(*j*)4], [*f*(*j*)2, *f*(*j*)5], [*f*(*j*)3, *f*(*j*)5]
  (*j* = 1, …, 5) (ex.: [*A*4, *E*1], [*A*4, *E*2], [*A*4, *E*3] e [*A*4, *E*5]).

Após todas as inserções necessárias de arestas opcionais e proibidas, é impossível a obtenção de algum outro núcleo reutilizando vértices dos núcleos de *um mesmo* grupo de 5 geradores. A Matriz de Adjacências da *Figura 15* apresenta todas as arestas do exemplo das *Figuras 11*, *12*, *13* e *14*, após a construção dos 5 novos núcleos. Foi utilizada a letra O para indicar aresta obrigatória, P para proibida e N para opcional. As cinco letras T presentes (em destaque) indicam uma "tentativa" de ser criado um novo núcleo utilizando-se apenas vértices deste grupo de núcleos, e a letra X (também em destaque) ilustra a razão disso não ser possível. Qualquer outro núcleo que se tente gerar *após* a construção dos 5 núcleos paralelos produz um conflito de definições para uma mesma aresta. Neste exemplo, após a tentativa de ser estabelecido o

novo núcleo *A*3-*C*1-*E*4-*B*2-*D*5, a aresta [*A*3, *E*4] precisa ser proibida (para impedir a criação de uma corda no ciclo do núcleo mínimo), mas já havido sido designada como opcional, para que o vértice *A*3 não se tornasse testemunha do par {*E*3, *E*4}.

Os primeiros 5 novos núcleos paralelos foram obtidos com o encadeamento de vértices dos geradores V_i, sem perda de generalidade, na ordem crescente do subscrito, ou seja, segundo a permutação circular *p* = (1, 2, 3, 4, 5) dos índices. Devido à inclusão das arestas opcionais [*f*(*j*)*i*, *f*(*j'*)(*i+1*)] e [*f*(*j*)*i*, *f*(*j'*)(*i-1*)], não é mais possível a criação de arestas obrigatórias para o encadeamento de outros vértices dos mesmos geradores *nesta mesma ordem*. De forma que uma nova permutação circular *p'* = (1, 3, 5, 2, 4) dos índices deve ser seguida para que possa haver a possibilidade de um novo núcleo ser formado com sucesso. Uma vez que o novo núcleo tenha sido estabelecido, uma aresta proibida terá que ser acrescentada entre cada par de vértices não adjacentes desse núcleo. Mas vértices não-adjacentes do núcleo recém-formado segundo a nova permutação *p'* são, necessariamente, vértices de grupos V_i e V_j onde *i* e *j* são consecutivos na permutação circular original *p*, de forma que já se há necessariamente incluído entre eles uma aresta opcional (vide *Figura 13* e texto referente à inclusão de arestas opcionais).

Com isso, está mostrada a impossibilidade de se criar novos núcleos, *desta forma e neste exemplo em particular*, pelo simples re-encadeamento de vértices *de um mesmo grupo* W_i *de cinco núcleos* $V_{i,1}$, $V_{i,2}$, $V_{i,3}$, $V_{i,4}$ *e* $V_{i,5}$. Na possibilidade de se poder combinar vértices de grupos diferentes, todavia, observemos que, em cada eventual novo núcleo que se forme, apenas poderá estar presente *um único* vértice de cada grupo W_i. Suponha que dois vértices *f*(*j*)*i* e *f*(*j'*)*i'* de um mesmo grupo venham a fazer parte de um novo núcleo obtido com vértices de mais de

um grupo. Como a topologia de um núcleo compreende 5 arestas obrigatórias e 5 proibidas, a aresta [*f*(*j*)*i*, *f*(*j'*)*i'*] deve ser: (i) *obrigatória*, o que implicaria a presença, entre outras, de uma aresta opcional [*f*(*j*)*i*, *f*(*j*)*i'*] – para que o vértice *f*(*j*)*i* não seja testemunha do par { *f*(*j*)*i'*, *f*(*j'*)*i'* } – embora uma tal aresta já haja sido forçosamente designada como proibida (vide *Figura 14* e texto referente à inclusão de arestas proibidas), descartando, portanto, esta primeira possibilidade; ou (ii) *proibida*, o que implicaria, por sua vez, que a aresta *f*(*j*)*i* constituísse uma testemunha externa do par { *f*(*j'*)*i'* , *w* } para algum *w* ligado a *f*(*j*)*i* por meio de aresta obrigatória.

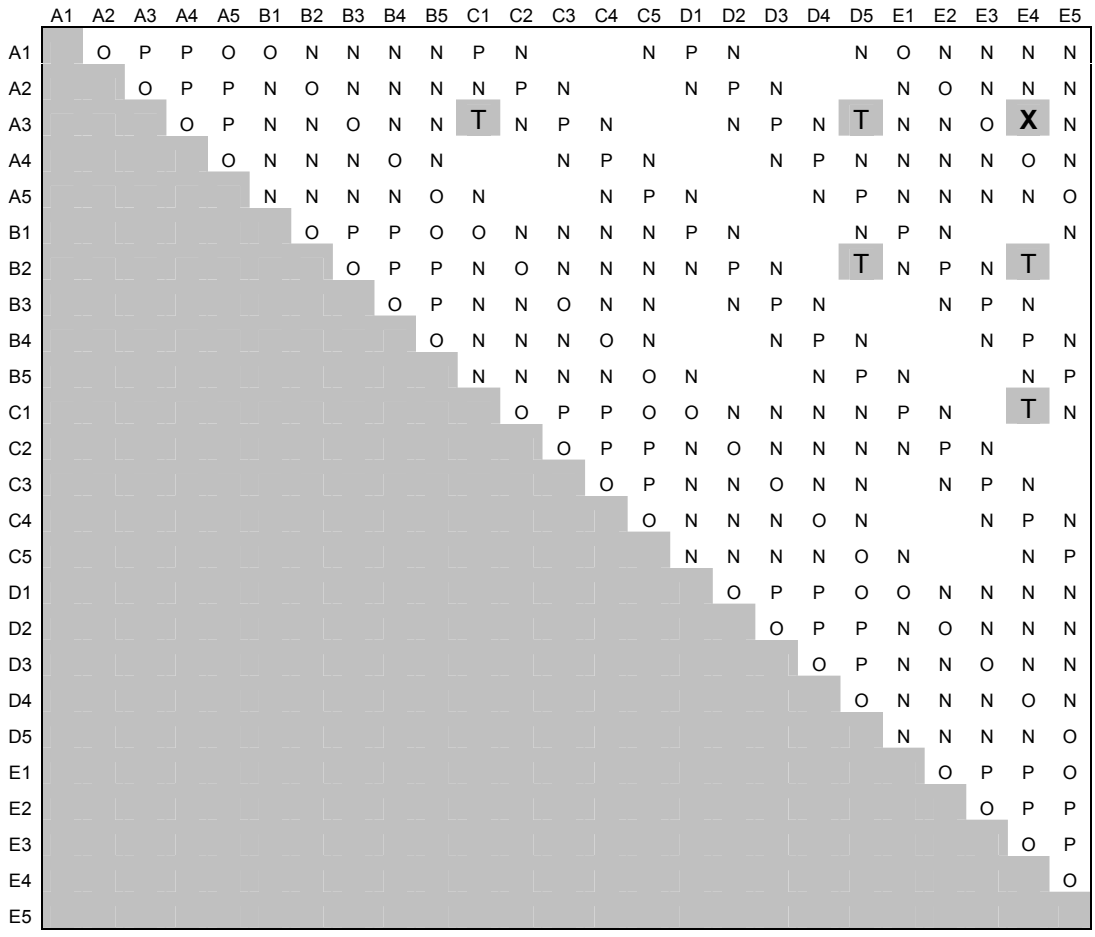


Figura 15 – Matriz de Adjacências p/ instância com núcleos não-disjuntos

Tomando-se, portanto apenas um vértice de cada grupo de 5 núcleos para a constituição de um novo gerador de SFCNFs, faz-se necessária a existência de 5 grupos, que passarão a constituir um supergrupo, contendo 25 núcleos com vértices exclusivos, originalmente.

Como é, de fato, possível que cada grupo contribua com um vértice por vez para a construção de novos núcleos, teremos conseguido 25 novos núcleos, para cada um dos $\Omega(n \div 25)$ supergrupos.

Um raciocínio análogo ao exposto para os grupos mostra que agora cada um dos supergrupos poderá contribuir com um de seus vértices por vez para a formação de novos núcleos, com que então cada 5 supergrupos poderão dar origem a 125 novos núcleos e assim por diante.

Totalizando o número g^* de núcleos obtidos (incluindo os originais, de vértices exclusivos), teremos:

$$g^* = g + \Omega[\, 5(n \div 5) + 25(n \div 25) + 125(n \div 125) + \ldots + 5^z(n \div 5^z)\,],$$

para $z = \log_5 n$ e $g = \Omega(n \div 5) = \Omega(n)$. Portanto,

$$g^* = \Omega[\, n + n \,.\, (1 + 1 + 1 + \ldots)\,] = \Omega(n \log n).$$
$$\uparrow \log_5 n \text{ vezes} \uparrow$$

Adicionando-se, como já o adiantáramos, um único vértice t com as arestas apropriadas ligando-o aos vértices de cada um dos $\Omega(n \log n)$ núcleos, é possível obter, neste exemplo, $\Omega(n \log n)$ SFCNFs no grafo-testemunha.

Com isso, superamos o limite de $O(n)$ que poderia se insinuar como teto para o número de SFCNFs. Permanece, contudo, um espaço entre o maior número – $\Omega(n \log n)$ – de SFCNFs que já se conseguiu, na prática, gerar, e o limite superior $O(m)$ que já se conseguiu, na teoria, provar, sendo, portanto, uma área ainda aberta à pesquisa. Seriam benvindos tanto um exemplo que apresentasse um número de SFCNFs que não

fosse O(n log n) quanto uma prova (baseada, talvez, na generalização das impossibilidades de criação de novos núcleos no exemplo dado) que provasse um limite superior mais justo.

Bibliografia

BUNEMAN, P., 1974, "A characterization of rigid circuit graphs", *Discrete Math. 9*, pp. 205-212.

CARRANO, A. V., 1988, "Establishing the order of human chromosome-specific DNA fragments". In A. D. Woodhead and B. J. Barnhart (eds), *Biotechnology and the Human Genome*, Plenum Press, pp. 37-55.

CERIOLI, M. R., EVERETT, H., FIGUEIREDO, C. M. H., *et al.*, 1998, "The homogeneous set sandwich problem", *Information Processing Letters 67,* pp. 31-35.

COURNIER, A., 1993, *Sur quelques algorithmes de décomposition de graphes*, Thése, Université Montpellier II.

DANTAS, S., FARIA, L., FIGUEIREDO, C. M. H., 2002, "On the complexity of (*k*,*l*)-graph sandwich problems", *Proceedings of WG 2002*, *Lecture Notes in Computer Science 2573*, pp. 92-101.

GAREY, M. R., JOHNSON, D. S., 1979, *Computers and Intractability, a Guide to The Theory of NP-completeness*, Freeman, San Francisco, pp.198-199.

GOLDBARG, M. C., LUNA, H. P. L., 2000, *Otimização Combinatória e Programação Linear*, Editora Campus, Rio de Janeiro.

GOLUMBIC, M. C., KAPLAN, H., SHAMIR, R., 1995, "Graph sandwich problems", *Journal of Algorithms 19*, pp. 449-473.

HABIB, M., LEBBAR, E., PAUL, C., 2002, "A note on finding all Homogeneous Set Sandwiches". Technical Report RR-LIRMM-02141, LIRMM, Université de Montpellier. Aceito para publicação em Information Processing Letters.

HARARY, F., 1969, *Graph Theory*, Addison-Wesley, Reading, MA.

LOVÁSZ, L., 1972, "Normal hypergraphs and the perfect graph conjecture", *Discrete Math. 2*, pp. 253-267.

McCONNELL, R. M., SPINRAD, J., 1999, "Modular Decomposition and transitive orientations", *Discrete Math. 201*, pp. 189-241.

REED, B., 1986, *A semi-strong perfect graph theorem*. Ph.D. thesis, School of Computer Science, McGill University, Montreal.

REED, B., SBIHI, N., 1995, "Recognizing bull-free perfect graphs", *Graphs and Combinatorics 11*, pp.171-178.

ROSE, J. D., 1972, "A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations." In R. C. Reed (ed), *Graph Theory and Computing*, Academic Press, NY, pp. 183-217.

SÁ, V. G. P., 2001, *Goonie: uma ferramenta para representação e algoritmação em grafos & O Algoritmo Dos Caminhos Disjuntos: uma alternativa para o problema da determinação de ciclos de suporte a anéis virtuais em grafos não-direcionados*, Projeto Final de Curso, Instituto de Matemática, Universidade Federal do Rio de Janeiro.

SZWARCFITER, J. L., 1984, *Grafos e algoritmos computacionais*, Editora Campus, Rio de Janeiro.

TANG, S., YEH, F., WANG, Y., 2001, "An efficient algorithm for solving the homogeneous set sandwich problem", *Information Processing Letters 77,* pp.17-22.

TARJAN, R. E., 1972, "Depth-first search and linear graph algorithms", *SIAM J. Comput. 1 (2)*, pp. 146-160.

YANNAKAKIS, M., 1981, "Computing the minimum fill-in is NP-complete", *SIAM Journal of Alg. Disc. Meth. 2*, pp. 77-79.

APÊNDICE 1

Grafos

A1.1 – O que é um grafo?

Embora seja deveras comum a associação da palavra ‘‘grafo’’ a um conjunto de linhas e pontos desenhados num plano, este tipo de esquema constitui apenas uma maneira de se representar aquilo que é, assim como um número ou um conjunto, entidade puramente abstrata. Esta utilização distorcida do termo, em geral consciente e voluntária, deve-se principalmente ao fato de permitirem os grafos uma *representação gráfica* bastante confortável. Em muitos casos, contudo, esta representação não é capaz de formalizar, completa e satisfatoriamente, toda a estrutura imaginada, remetendo-nos, então, à sua definição analítica:

‘‘Um grafo é uma estrutura de abstração que representa um conjunto de elementos denominados *vértices* e suas relações de interdependência ou *arestas*.’’ (GOLDBARG e LUNA, 2000)

Desta forma, todo conjunto entre cujos elementos existam relacionamentos binários de qualquer natureza pode ser enxergado como um grafo. Há, hoje, uma infinidade de aplicações práticas de grafos em áreas como Sociologia, Telecomunicações, Computação, Contabilidade e Ciências Matemáticas em geral, onde não apenas se representam as mais diversas estruturas sob a forma característica dos grafos como também se

pode contar com um sem-número de técnicas e algoritmos estudados em Teoria dos Grafos como auxílio à resolução de seus problemas.

Um grafo pode ser *simples* ou *direcionado*, segundo tenham ou não suas arestas uma orientação definida. Grafos direcionados são também chamados *digrafos*.

A notação matemática emprega, tipicamente, os símbolos G (V, E) para representar um grafo simples G constituído de um conjunto V de vértices e um conjunto E de arestas. O número de vértices do grafo é normalmente denotado por $n = |V|$, e o de arestas por $m = |E|$. Os símbolos v_i ($i = 1, \ldots, n$) designam, usualmente, os vértices, enquanto os pares não-ordenados (ou ordenados) $[v_i, v_j]$ descrevem cada aresta do grafo simples ou direcionado, respectivamente.

A1.2 – Representação geométrica de um grafo

Unicamente com o objetivo de facilitar a visualização dos vértices e arestas de um grafo, usa-se representá-los por pontos e linhas, respectivamente. Dois vértices $v_i , v_j \in V$ são ditos *adjacentes* quando existir, no grafo, a aresta $[v_i , v_j]$. Graficamente, uma linha unindo os pontos (círculos ou outras figuras quaisquer) que representam os vértices v_i e v_j representará a existência da aresta $[v_i , v_j]$ no grafo.

O estudo dos grafos está tão fortemente ligado a sua representação geométrica característica que toda uma classe de grafos é definida em função deste tipo de representação: a classe dos grafos *planares*, que são aqueles que admitem pelo menos uma representação plana, ou seja, uma que, ao ser traçada no plano, não contenha interceptação de arestas

exceto em suas extremidades (vértices). Os grafos planares têm grande importância na modelagem de problemas de topologia plana. Redes de estradas, mapas, circuitos eletrônicos, etc., configuram, na maioria dos casos, grafos planares.

A *Figura A1* apresenta exemplos da representação geométrica usual de grafos.

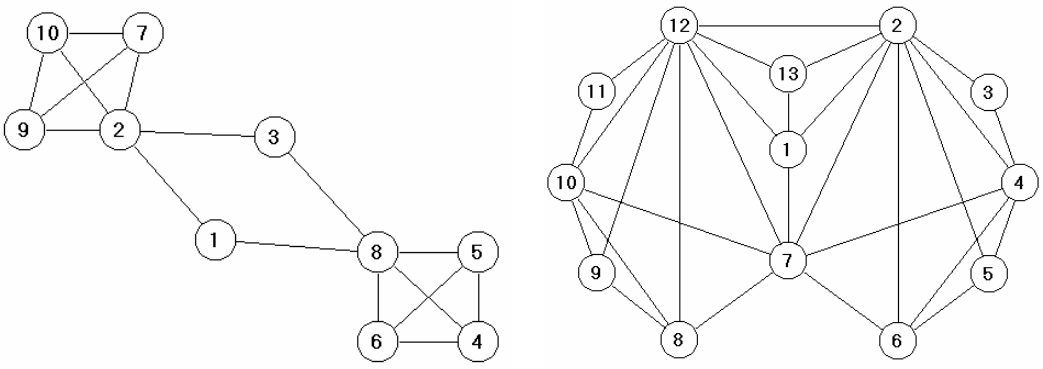


Figura A1 – Exemplos de grafos simples (não-direcionados)

A1.3 – Isomorfismo

Um interessante aspecto dos grafos é que um mesmo modelo das relações vértices X arestas pode ser representado graficamente de diversas formas, ou seja, podemos encontrar grafos *desenhados* diferentemente mas que representam exatamente o mesmo fenômeno de associação vértices X arestas, sendo, portanto, analiticamente idênticos. Isto constitui o chamado *isomorfismo*, que pode ser definido formalmente assim:

"Dois grafos G_1 (V_1, E_1) e G_2 (V_2, E_2) são ditos *isomorfos* entre si quando $|V_1| = |V_2|$ e existe uma função unívoca $f : V_1 \rightarrow V_2$ tal que $[v, w] \in$

E_1 se e somente se [$f(v)$, $f(w)$] $\in$ E_2, para todo v, w $\in$ V_1.” (SZWARCFITER, 1984)

Temos, na *Figura A2*, representações geométricas de dois grafos isomorfos.

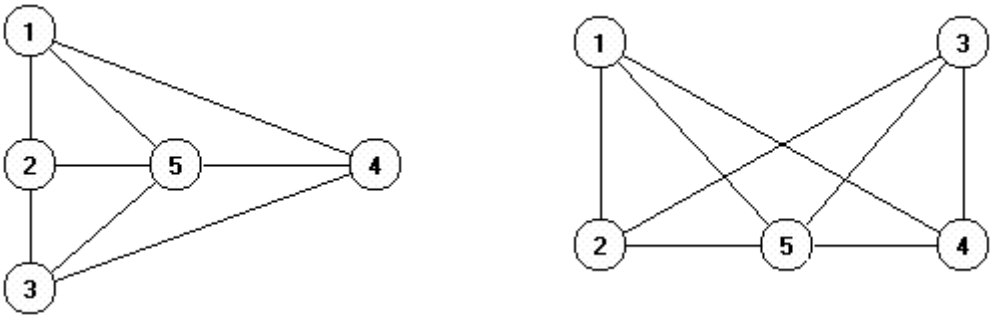


Figura A2 – Grafos isomorfos

APÊNDICE 2

Grafos e Computadores

A2.1 – Estruturas de dados: armazenando grafos no computador

Para que seja dado tratamento computacional aos problemas em grafos, necessário se faz um modo de codificá-los de modo apropriado. A facilidade (ou a dificuldade) que se sentirá quando da implementação dos algoritmos voltados à solução de um determinado problema é bastante sensível à maneira como se tratou o problema da representação dos grafos em estruturas de dados mais ou menos adequadas.

Além da representação geométrica (tão propícia à ilustração dos vértices e arestas no que concerne à inteligibilidade das estruturas por parte dos humanos, dotados de olhos), várias outras formas de representação permitem descrever completamente um grafo, sendo, contudo, indubitavelmente mais apropriadas a seu armazenamento e manipulação por parte de um computador (que, se não dispõe de olhos, lida com matrizes e listas encadeadas milhões de vezes mais rápido do que qualquer humano jamais o faria!) Vejamos algumas das mais comumente empregadas.

Representação através da Matriz de Adjacências

Trata-se de uma representação bastante simples. O grafo é expresso por uma matriz $A = [a_{ij}]$ contendo os vértices e suas relações de vizinhança. As linhas e as colunas da matriz estão associadas aos vértices do grafo. A matriz é normalmente booleana, ou seja, seus elementos são 0 ou 1. (Obs.: um tipo especial de grafo, chamado *multigrafo*, aceita mais de uma aresta unindo o mesmo par de vértices. Nesse caso, o valor de a_{ij} pode representar o número de arestas entre esses vértices.)

Seja A (n x n) = $[a_{ij}]$ a matriz de adjacências do grafo $G = (V, E)$, como exemplificado pela *Figura A3*.

Teremos:

$$\begin{cases} a_{ij} = 1 \leftrightarrow (v_i , v_j) \in E \\ a_{ij} = 0 \leftrightarrow (v_i , v_j) \notin E \end{cases}$$

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	0	0	1	0	1	0	0	0
2	0	0	1	1	0	0	1	0	0
3	0	1	0	1	0	0	1	0	1
4	1	1	1	0	1	1	0	0	1
5	0	0	0	1	0	1	0	1	0
6	1	0	0	1	1	0	0	1	0
7	0	1	1	0	0	0	0	0	0
8	0	0	0	0	1	1	0	0	0
9	0	0	1	1	0	0	0	0	0

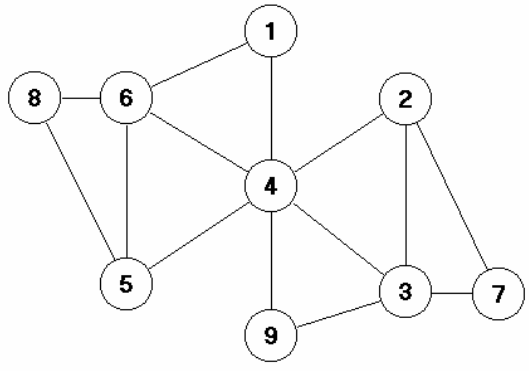


Figura A3 – Um grafo e sua Matriz de Adjacências

Implementa-se facilmente, no computador, esta representação. Um inconveniente é, no entanto, a grande quantidade de memória despendida com o armazenamento de ‘‘zeros’’, já que, em geral, essas matrizes são significativamente esparsas. Além disso, outras informações sobre os elementos do grafo (como, por exemplo, os rótulos dos vértices ou os custos das arestas) precisariam ser guardadas em outras estruturas.

Representação através da Matriz de Incidências

Nesta representação, as colunas correspondem às arestas do grafo e as linhas aos vértices.

Seja a matriz n X m $A = [a_{ij}]$ a matriz de incidências do grafo $G = (V, E)$. Então, para toda aresta j que liga os vértices v_p e vq, temos:

$$\begin{cases} a_{ij} = +1 \leftrightarrow i = p \\ a_{ij} = -1 \leftrightarrow i = q \text{ (para grafos direcionados; senão, } a_{ij} = 1 \text{ neste caso e no anterior)} \\ a_{ij} = 0 \text{ nos outros casos} \end{cases}$$

Implementa-se, também facilmente, esta representação. Os inconvenientes são, contudo, os mesmos da representação por matriz de adjacências.

Representação através de listas encadeadas

A representação computacional de grafos utilizando listas encadeadas é extremamente conveniente para a eficiência de alguns algoritmos, bem como para a economia de memória de armazenamento. A mais tradicional representação por listas encadeadas possui uma

configuração baseada nos vértices do grafo e em suas vizinhanças, estabelecendo-se que cada vértice encabece uma lista contendo todos os vértices que lhe são adjacentes. A *Figura A4* exemplifica esta representação.

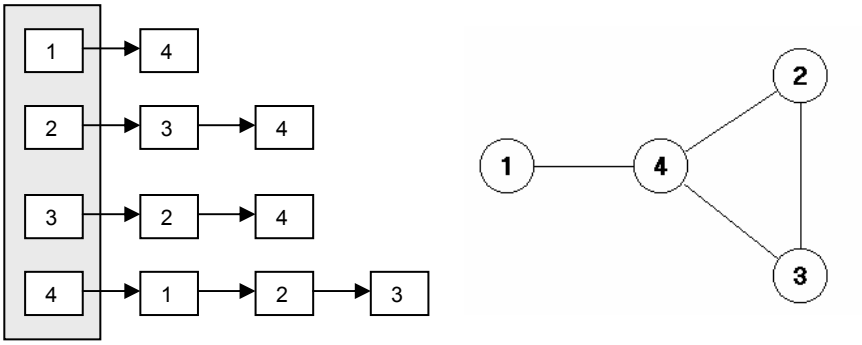


Figura A4 – Representação por listas encadeadas

Representação vetorial

Pode-se simular a representação por listas encadeadas de maneira ainda mais simples (e eficiente, no caso de grafos esparsos), utilizando-se apenas dois vetores: o primeiro armazena, na posição *x*, o número de vértices adjacentes a v_X , enquanto o segundo os enumera. A *Figura A5* ilustra a representação vetorial do grafo da *Figura A4*.

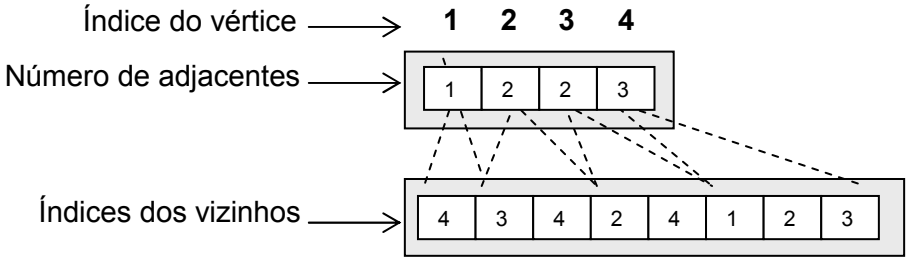


Figura A5 – Representação vetorial

A2.2 – Interface gráfica: visualizando o grafo

Uma vez que se tenha adotado uma maneira de *representar* um grafo no computador, utilizando estruturas de dados apropriadas, já é possível, em princípio, a implementação de qualquer algoritmo que o manipule. Qualquer que seja a maneira pela qual um computador represente internamente um grafo, esta diferirá drasticamente daquela a que nós, humanos, estamos mais familiarizados e com a qual nos sentimos mais confortáveis para visualizar e compreender as estruturas envolvidas (inclusive aquelas retornadas como saída de muitos algoritmos), que é a representação gráfica.

A tarefa de "traduzir" as informações numéricas da representação computacional de um grafo em objetos de tela é a *raison d'être* da interface gráfica de um programa de manipulação de grafos. À cada vértice deve estar associada uma coordenada da tela, com o centro na qual deve-se desenhar uma circunferência (ou outra figura qualquer) que o represente. Estas coordenadas servirão também de guia para as linhas que representarão as arestas.

Por admitir um grafo infinitas representações gráficas distintas, é conveniente que uma ferramenta computacional que o represente permita que o usuário determine a configuração gráfica que mais lhe aprouver, "movendo" os vértices livremente na tela.

A prática saudável da Engenharia de Sofware recomenda que métodos de tratamento gráfico (métodos de interface) não sejam misturados a métodos de manipulação de dados, de forma que, para a construção correta de um programa de manipulação de grafos, deve-se encapsular tudo o que trate exclusivamente de interface separadamente das estruturas de armazenamento desses.

APÊNDICE 3

Digrafos e Componentes Fortemente Conexos

Digrafos são grafos direcionados, ou seja, aqueles cujas arestas possuem uma orientação bem definida. Enquanto nos grafos simples (não-direcionados), cada aresta é simplesmente *incidente* a dois vértices e tem função idêntica para cada um deles, qual seja a de representar sua interdependência, nos digrafos cada aresta é dita *aresta de saída* de seu vértice-origem e *de entrada* com relação a seu vértice-destino. Na *Figura A6a*, temos um exemplo de digrafo, no qual a aresta 8→10 é aresta de saída do vértice 8, sendo, para o vértice 10, aresta de entrada. Pode-se perceber, ainda no exemplo da *Figura A6a*, a presença de arestas sobrepostas com orientações contrárias, como as arestas 7→11 e 11→7, por exemplo, formando um *ciclo* de tamanho dois (isto é, formado por dois vértices).

Um digrafo é *fortemente conexo* se existe um caminho de *u* até *v* e de *v* até *u* para qualquer par distinto de vértices *u* e *v*. Se um digrafo não é fortemente conexo, cada um de seus subgrafos fortemente conexos maximais (isto é, que não contenham propriamente nenhum subgrafo fortemente conexo) é chamado *componente fortemente conexo* (CFC).

Considere o digrafo ilustrado na *Figura A6a*. Não é difícil verificar que ele possui quatro componentes fortemente conexos: {1, 2, 3, 4}, {6}, {5, 8, 9, 10} e (7, 11). O *grafo de condensação* G_C (V_C, E_C) de um digrafo

D (V, E) é aquele que apresenta todos os vértices v_1, v_2, ..., de um mesmo componente fortemente conexo de D agrupados em um único vértice seu, rotulado $\langle v_1, v_2, ... \rangle$. G_C possuirá, também, uma aresta direcionada $\langle x_1, x_2, ... \rangle \rightarrow \langle y_1, y_2, ... \rangle$ se, e somente se, existir uma aresta $x_i \rightarrow y_j \in E$ para algum par (i, j), ou seja, se o componente $\{y_1, y_2, ...\}$ for *atingível* pelo componente $\{x_1, x_2, ...\}$.

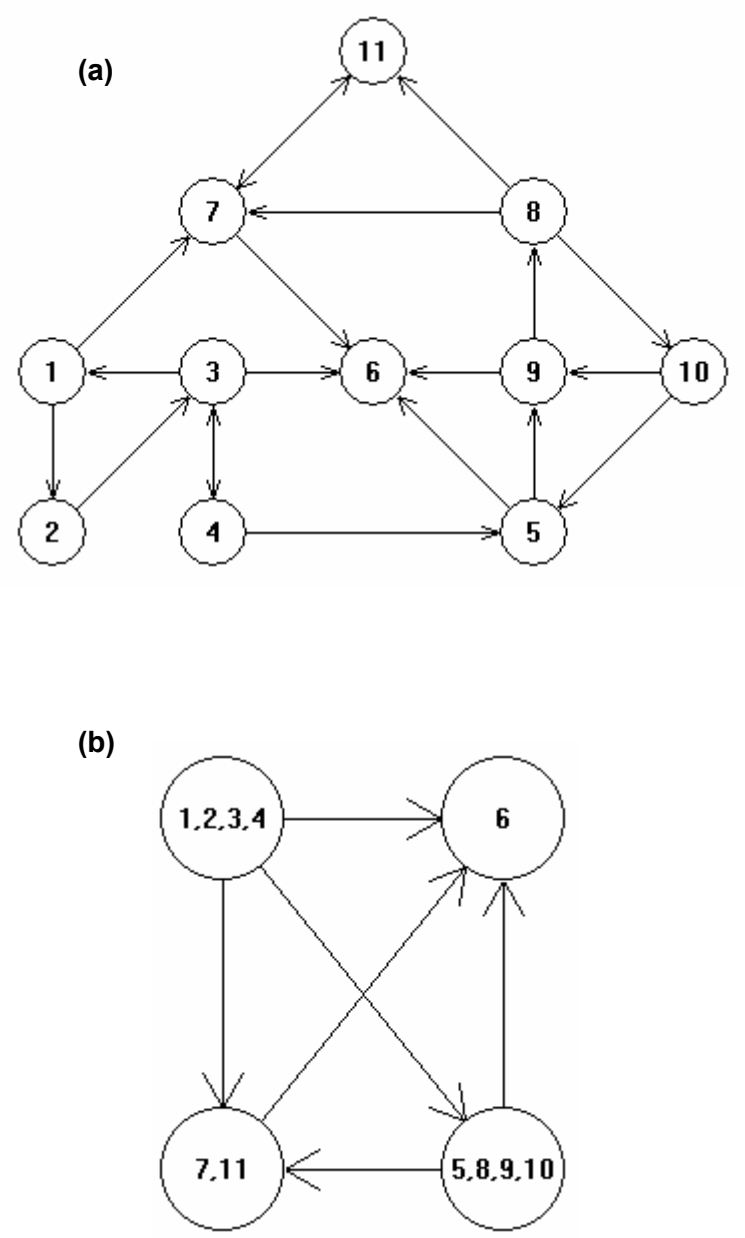


Figura A6 - Um digrafo e seu grafo de condensação

A3.1 – Algoritmo para determinação dos CFCs

É possível se determinar, em tempo O(n + m), a partição de um digrafo em seus componentes fortemente conexos (TARJAN, 1972).

Procedimento 8: *Busca Profundidade* (*v*, *D*, *F*, *Visitado*[ ], *Posição*[ ])

Entrada: Um vértice *v*; os digrafos *D* (*V*, *E*) e *F* (*V*, E_F); um vetor de |*V*| booleanos *Visitado*[ ]; um vetor de |*V*| inteiros *Posição*[ ]

Saída: atualização de *F* (*V*, E_F), floresta de busca de *D*; atualização de *Visitado*[*v*] e *Posição* [*v*].

Variáveis: *PosiçãoAtual* : inteiro (global); *w* : vértice.

Método:

1. *Visitado*[*v*] $\leftarrow$ VERDADEIRO

2. Para cada vértice *w* tal que v$\rightarrow w \in E$ faça

2.1 Se *Visitado*[*w*] = FALSO então

2.1.1. $E_F \leftarrow E_F \cup \{ v \rightarrow w \}$

2.1.2. *Busca Profundidade* (*w*, *D, F, Visitado*[ ], *Posição*[ ])

3. Se *Posição*[ ] $\neq$ NULO então

3.1. *PosiçãoAtual* $\leftarrow$ *PosicaoAtual* + 1

3.2. *Posição*[*v*] $\leftarrow$ *PosiçãoAtual* *{ Pós-ordem }*

5. Fim.

O algoritmo que para a identificação de todos os componentes fortemente conexos de um digrafo *D* (*Procedimento 10*) consiste em:

1) Realizar uma busca em profundidade em *D*, associando a cada vértice *v* um inteiro *Posição*[*v*] que indica a posição do vértice *v* na seqüência de visitação da árvore de busca em percurso pós-ordem.

2) Construir o digrafo *D'*, idêntico a *D*, apenas com todas as suas arestas orientadas no sentido contrário às de *D*.

3) Realizar uma busca em profundidade em *D'*, começando do vértice *v* com o maior valor *Posição*[*v*] associado. Recomeçar o processo, sempre a partir do vértice de maior *Posição* dentre aqueles que ainda não tenham sido visitados, sempre que se chegar ao final de uma árvore na floresta de profundidade e ainda houver vértices de *D'* não visitados.

4) Retornar o conjunto dos vértices presentes em cada árvore obtida no passo 3 como um componente fortemente conexo de *D*.

Procedimento 9: *Busca* (*D*, *Posição*[ ])

Entrada: Um digrafo *D* (*V*, *E*); um vetor de |*V*| inteiros *Posição*[ ]

Saída: *F* (*V*, E_F), floresta de busca de *D* ; atualização de Posição [ ].

Variáveis: *F* (*V*, E_F) : digrafo; *Visitado* [ ] : vetor de |*V*| booleanos;

v : vértice; *PosiçãoAtual*: inteiro (global).

Método:

1. *PosiçãoAtual* ← 0

2. Para cada vértice $v \in V$ faça

2.1. *Visitado* [*v*] ← FALSO

3. Para cada vértice $v \in V$ faça

3.1 Se *Visitado* [*v*] = FALSO então

3.1.1 *Busca-Profundidade* (*v*, *D, F, Visitado*[ ], *Posição*[ ])

4. Retorne *F*. Fim.

Procedimento 10: *Particiona Digrafo CFCs* (*D*) (TARJAN, 1972)

Entrada: Um digrafo *D* (*V*, *E*)

Saída: O conjunto $C = \{C_1,..., C_k\}$ contendo os $k \geq 1$ componentes fortemente conexos de *D*.

Variáveis: *Posição*[ ] : vetor de |*V*| inteiros; *C* : conjunto de conjuntos de vértices; *F* (*V*, E_F) e *D'* (*V*, $E_{D'}$): digrafos; *Visitado*[ ] : vetor de |*V*| booleanos; *x*, *v* : vértices.

Método:

1. $F \leftarrow$ *Busca* (*D*, *Posicao*[ ])

2. Para cada aresta $v \rightarrow w \in E$ faça

2.1. $E_{D'} \leftarrow E_{D'} \cup \{ w \rightarrow v \}$

3. Para cada vértice $v \in V$ faça

3.1. *Visitado*[*v*] $\leftarrow$ FALSO

4. $F \leftarrow (V, \varnothing)$

5. Enquanto houver vértice *x* tal que Visitado[*x*] = FALSO faça

5.1. *PosNaoVisitados* $\leftarrow$ { *Posição*[*x*] | *Visitado*[*x*] = FALSO }

5.2. $v \leftarrow x \in V$ tal que *Posição*[*x*] = MAX {*PosNaoVisitados*}

5.3. *Busca Profundidade* (*v*, *D'*, *F*, *Visitado*[ ], NULO);

6. $C \leftarrow \varnothing$

7. Para cada árvore $T_K \in F$ faça

7.1. $C \leftarrow C \cup \{ \{ x \mid x$ é vértice de $T_K \} \}$

8. Retorne *C*. Fim.

A título de exemplo, acompanharemos a execução do algoritmo *Particiona Digrafo CFCs* (*Procedimento 10*) aplicado ao digrafo *D* (*V*, *E*) da *Figura A6a*.

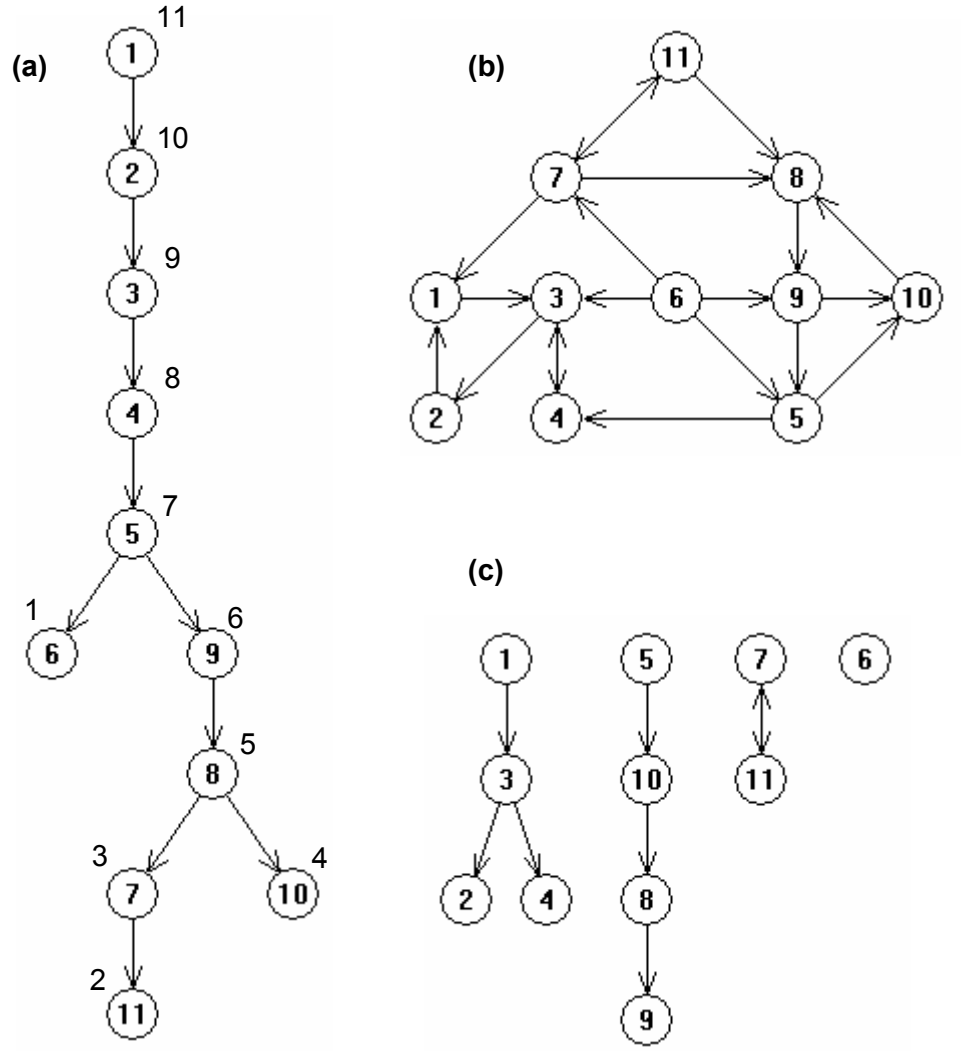


Figura A7 – Exemplo de execução do algoritmo de determinação de CFCs

A *Figura A7a* ilustra a árvore de profundidade obtida na linha 1 do *Procedimento 10*, bem como o valor de *Posição* [*v*] para cada vértice *v* do digrafo. Na *Figura A7b* se encontra o digrafo *D'* obtido no laço iniciado na linha 2 do algoritmo. Em *D'* é então executada uma busca em profundidade, sempre a partir do vértice de maior *Posição*, gerando a floresta representada na *Figura A7c*, que apresenta exatamente uma árvore para cada CFC do digrafo original *D*, como era de se esperar.

***Prova de Corretude do* Procedimento 10**

Para a prova de corretude do *Procedimento 10*, mostraremos que dois vértices *u* e *v* estão no mesmo componente fortemente conexo de *D* se, e somente se, *u* e *v* aparecem na mesma árvore da floresta de profundidade obtida após o término do laço iniciado à linha 5 do algoritmo.

Suponhamos que *u* e *v* estão no mesmo componente fortemente conexo. Por definição, haverá um caminho de *u* até *v* e vice-versa em *D.* Como *D'* é obtido invertendo-se a orientação de todas as arestas de *D*, a seqüência das arestas que, em *D*, constitui um caminho de *u* a *v* será, em *D'*, a seqüência inversa das arestas que constituirão um caminho de *v* a *u*, e vice-versa, de forma que *u* e *v*, sendo mutuamente atingíveis em *D'*, estarão na mesma árvore da busca em profundidade realizada neste digrafo (laço da linha 5).

Para a suficiência, seja *v* um vértice de uma árvore de raiz $r \neq v$ da floresta de busca em *D'*. Isso implica que existe um caminho de *r* até *v* no digrafo *D'* e, portanto, de *v* até *r* no digrafo *D* (cujas arestas, lembramos, são todas orientadas contrariamente às de *D'*). Como *r* é raiz da árvore onde se encontra *v*, teremos, necessariamente, *Posição*[*r*] > *Posição*[*v*]. Consideremos agora as possíveis posições relativas de *r* e *v* na árvore que corresponde à busca em profundidade realizada em *D* (linha 1). Três são, inicialmente, as possibilidades:

1) *r* é ancestral de *v*;

2) *r* é descendente de *v*;

3) *r* não é nem ancestral nem descendente de *v*.

Consideremos o segundo caso: se *r* é descendente de *v*, teríamos forçosamente *Posição*[*r*] < *Posição*[*v*], o que, já o sabemos, não é o caso, eliminando esta possibilidade.

Avaliemos a terceira possibilidade: como *Posição*[*r*] > *Posição*[*v*], o vértice *r* foi visitado depois de *v* e está, portanto, à direita de *v* na árvore. O caminho de *v* até *r*, em *D* (e é certo existir um, pois *r* atinge *v*, em *D'*) não pode ser como o ilustrado na *Figura A8a*, onde não existe um vértice que seja ancestral de ambos. Se existisse um tal caminho, *r* seria descendente de *v* na árvore de busca (em profundidade) de *D*. Portanto, o caminho de *v* até *r*, em *D*, deve conter um vértice intermediário *x*, ancestral comum de *v* e *r* na árvore de busca (*Figura A8b*). Porém, a existência de um tal vértice *x*, ancestral de *r*, implica a existência de um caminho de *r* a *x*, em *D'*, de forma que *x* também deve estar na mesma árvore em que se encontram *r* e *v* na floresta de busca de *D'*. Chegamos, portanto, a uma contradição, pois se *x* e *r* estão na mesma árvore de busca de *D'*, cuja raiz é *r*, temos que *Posição*[*r*] > *Posição*[*x*] (pois *r*, e não *x*, foi selecionado para iniciar a busca que originou esta árvore), o que é impossível, pois *x* é ancestral de *r* na árvore de busca de *D*.

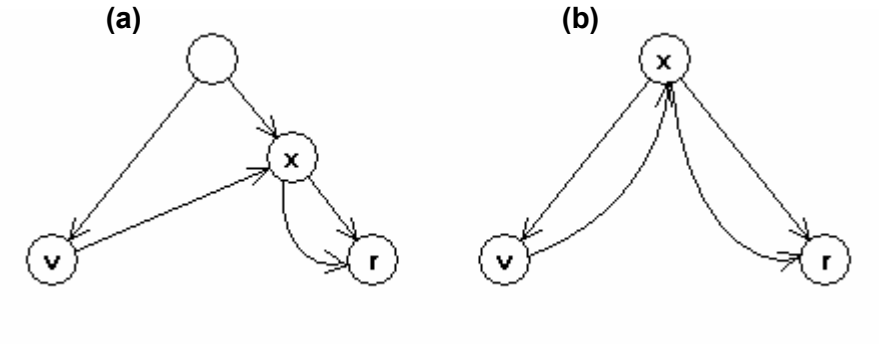


Figura A8 – Caminhos de v *a* r *na árvore de profundidade de* D

Apenas resta, portanto, a primeira possibilidade, que implica a existência de um caminho de *r* até *v*, em *D*.

Consideremos agora dois vértices *u* e *v* da mesma árvore da floresta de busca de *D'*, da qual nenhum deles é a raiz. Seja *r* a raiz dessa árvore. Já provamos que existe um caminho que vai de cada um desses vértices até a raiz, e da raiz até cada um desses vértices. Existe, portanto, um caminho de *u* até *v* passando por *r*, e vice versa.

***Análise de Complexidade do* Procedimento 10**

Não é difícil verificarmos que o *Procedimento 10* consome tempo total O(*n* + *m*), uma vez que as etapas das buscas em profundidade (em digrafos com *n* vértices e *m* arestas) e da construção do digrafo *D'* são executadas seqüencialmente e possuem, cada qual, esta mesma complexidade temporal.

A3.2 – Sumidouros

Lema 3: *Um digrafo* D (V, E) *é fortemente conexo se, e somente se, não contém propriamente nenhum sumidouro*.

Prova: Se existe um sumidouro $S \subset V$, subconjunto de vértices do digrafo que não apresenta aresta partindo de um de seus vértices e incidindo em algum outro vértice que não lhe seja pertencente, existe, então, algum vértice $w \in (V \setminus S) \neq \varnothing$ que não é atingido por nenhum vértice de *S*, donde *D* não pode ser fortemente conexo. Por outro lado, se *D* não é fortemente conexo, então existe um par de vértices $x, y \in V$ para os quais não há caminho de *x* até *y*. Dessa forma, o conjunto *R*(*x*) formado por *x* e todos os vértices atingíveis por *x* estará contido propriamente em *V* (pois não pode

conter *y*) e não terá arestas de saída para nenhum vértice $z \in V \setminus R(x)$, do contrário *z* seria atingível por *x* e deveria pertencer a *R*(*x*), que será, portanto, sumidouro de *D*.

Lema 4: *Seja* D (V, E) *um digrafo. Se* S $\subset$ D *é sumidouro, então* S $\supseteq$ S* *para algum SFCP* S* d*e* D.

Prova: Se *S* é fortemente conexo, a prova é trivial (*S'* = *S* é SFCP de *D*). Se *S*, por outro lado, não é fortemente conexo, então, pelo *Lema 3*, *S* contém propriamente algum sumidouro *S'*. Suponhamos, por absurdo, que *S* não contenha nenhum SFC. Conseqüentemente, *S'* não pode ser fortemente conexo e precisa conter propriamente (novamente pelo *Lema 3*) um sumidouro *S''*, que, por sua vez, não pode ser fortemente conexo. E assim sucessivamente. O que é impossível, pois o conjunto de vértices de *D* é finito. De forma que *S* contém necessariamente um SFC *S**. Finalmente, $S^* \subseteq S \subset D$ é, portanto, SFCP de *D*.

Lema 5: *Um digrafo* D (V, E) *contém propriamente um sumidouro se, e somente se, contém algum* SFCP.

Prova: A necessidade é conseqüência direta do *Lema 4* que diz que todo sumidouro próprio de *D* contém propriamente ou é ele próprio um SFCP de *D*. A suficiência é trivial, uma vez que se *D* contém algum SFCP *S* então *S* é sumidouro próprio de *D*.

APÊNDICE 4

Figuras Especiais

As figuras apresentadas neste Apêndice 4 são referidas ao longo do texto e se encontram aqui destacadas em função do seu tamanho, que prejudicaria a formatação dos capítulos onde são citadas.

Observe-se que de todos os rótulos de vértices presentes em algum grafo-testemunha deste apêndice foi suprimida, por economia de espaço, a vírgula separadora, de forma que um vértice aqui rotulado "12", por exemplo, equivale ao vértice ⟨1,2⟩ do grafo-testemunha em questão.

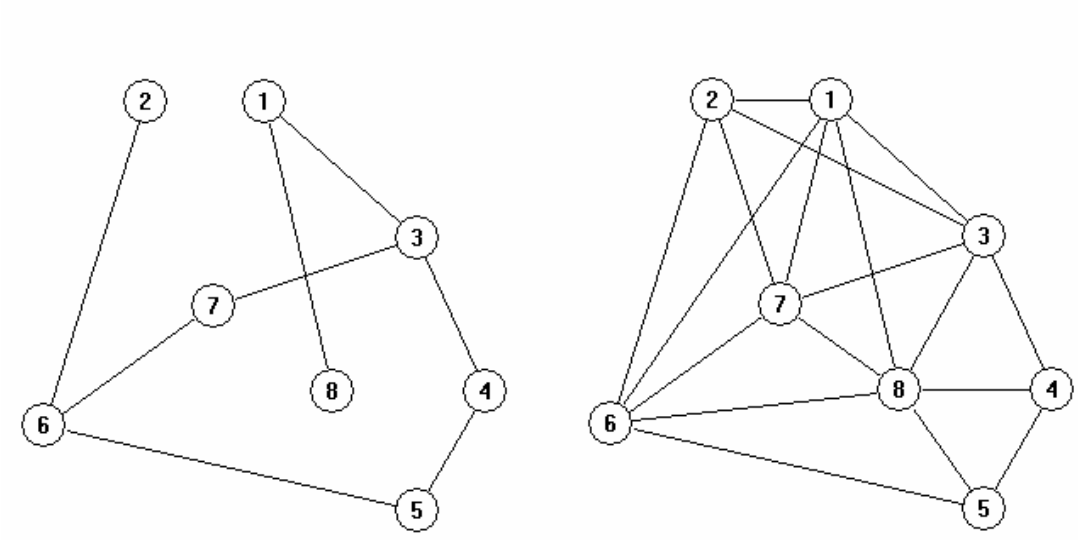


Figura E1a – Entrada para o PSCH - Contraexemplo 1

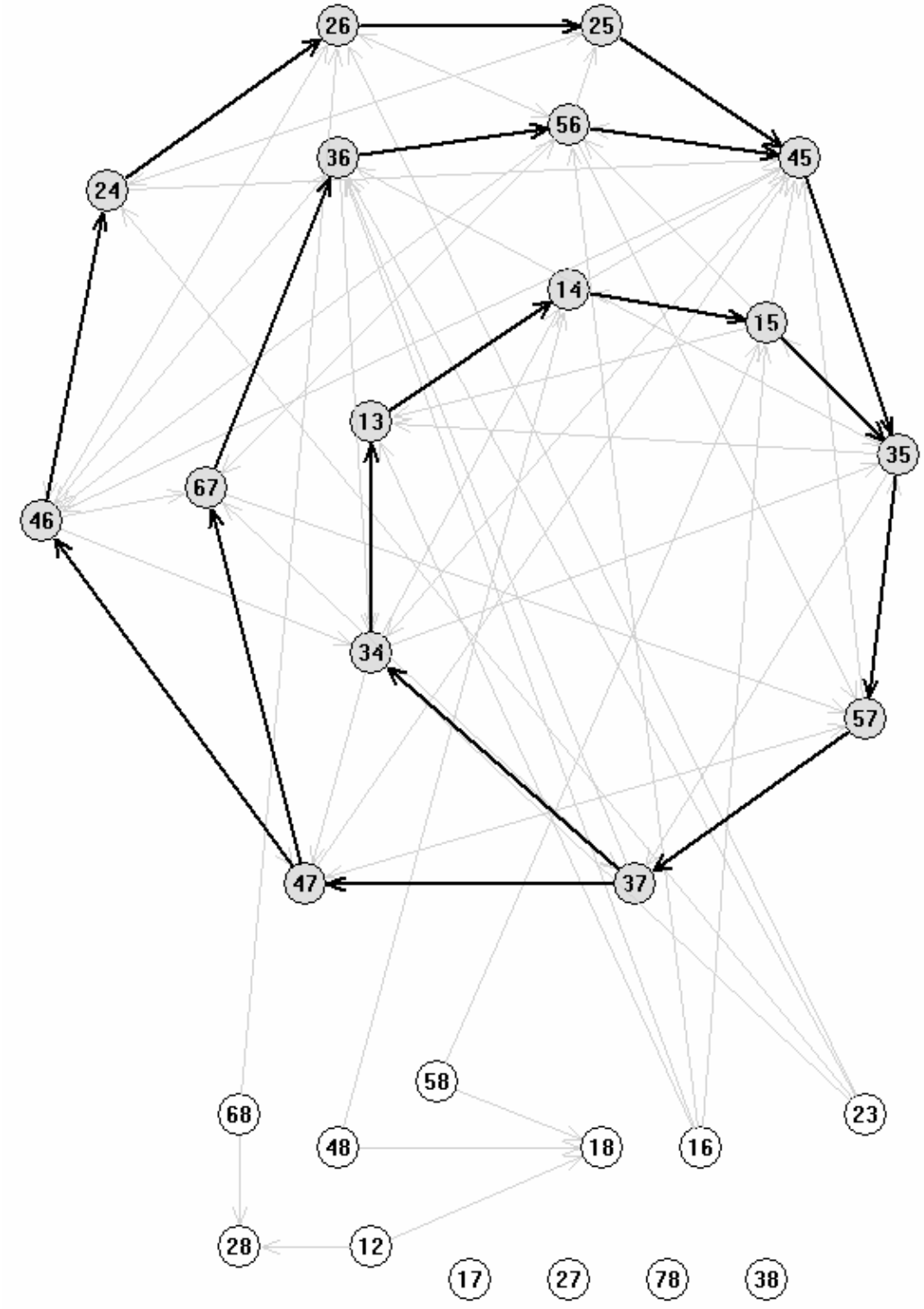


Figura E1b – Grafo-testemunha p/ o Contraexemplo 1 *(com um SFCP que não corresponde a CHS)*

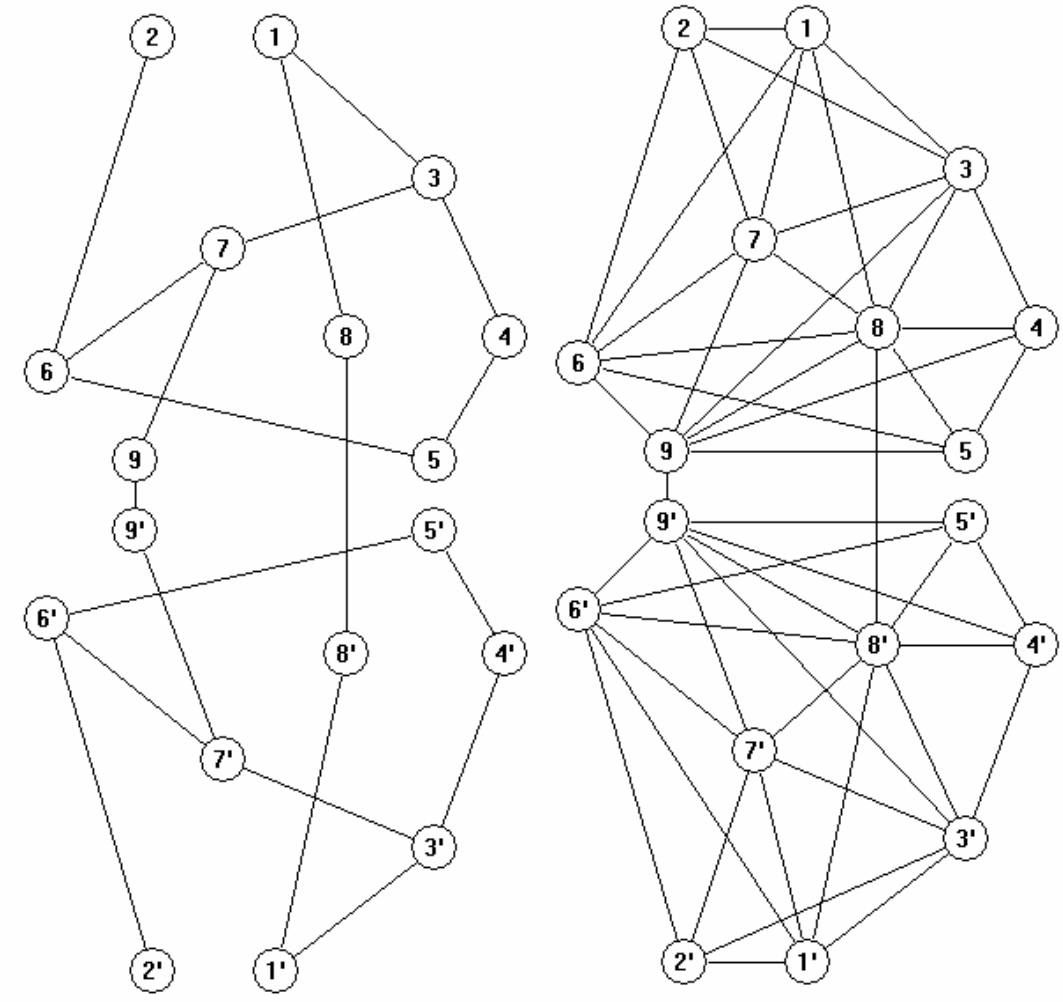


*Figura E2a – Um par de grafos que não admite conjunto homogêneo sanduíche embora seu grafo-testemunha (*Figura E2b*) apresente dois sumidouros fortemente conexos próprios* – Contraexemplo 2

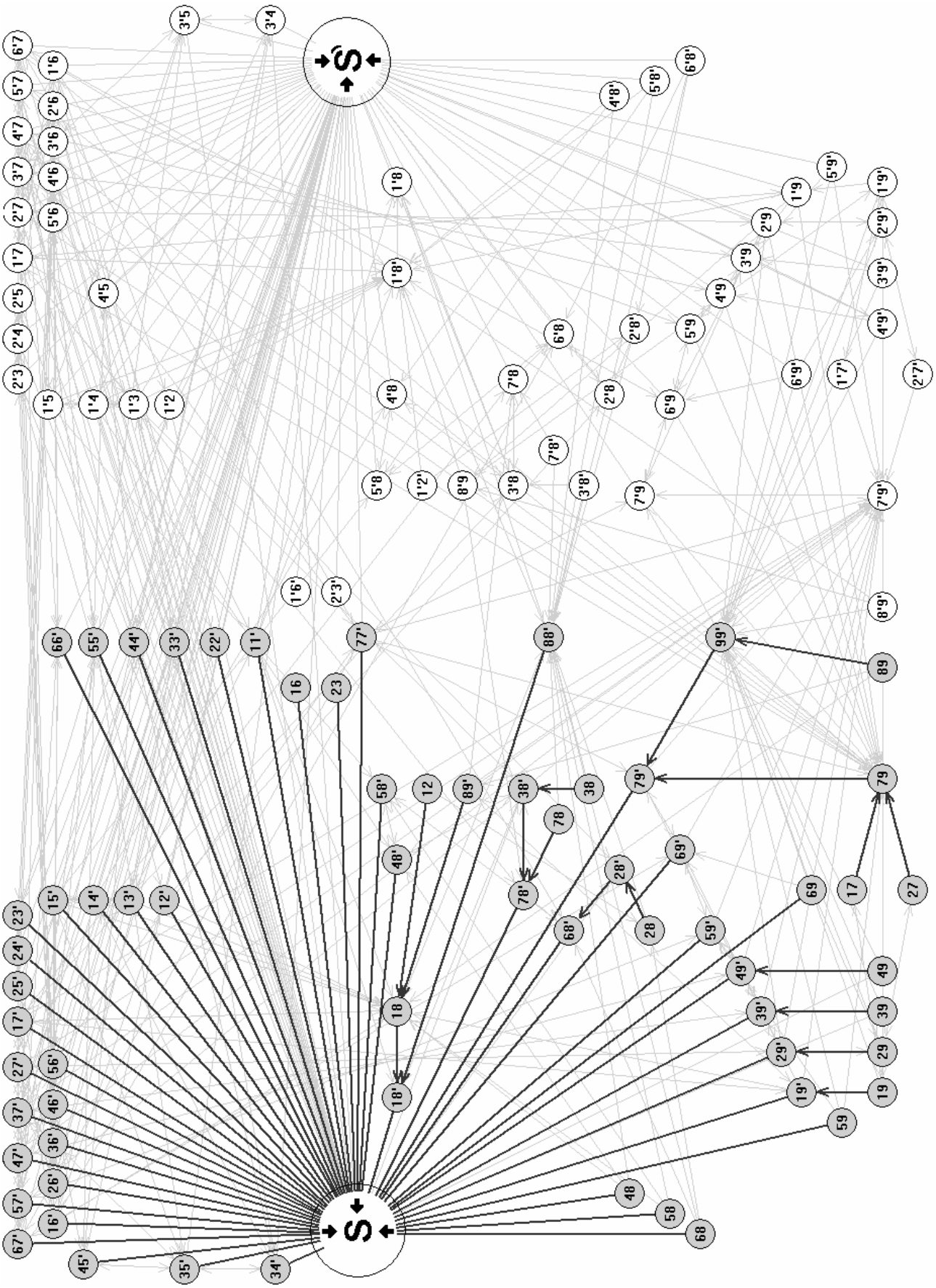

S
S
3'5
3'4
6'7
1'6
5'7
2'6
4'7
3'6
3'7
4'6
2'7
5'6
1'7
2'5
4'5
2'4
2'3
1'5
1'4
1'3
1'2
1'8
1'8'
4'8
5'8
1'2'
8'9
3'8
3'8'
7'8'
7'8
6'8
2'8
4'8'
5'8'
6'8'
1'9
5'9'
2'9
1'9'
3'9
2'9'
4'9
3'9'
5'9
4'9'
6'9
1'7'
2'7'
6'9'
7'9
7'9'
8'9'
1'6'
2'3'
66'
55'
44'
33'
22'
11'
77'
16
23
88'
99'
89
58'
12
89'
38'
38
79'
79
78
48'
69'
28'
78'
68'
28
59'
69
17
27
15'
14'
13'
12'
23'
24'
25'
17'
27'
56'
37'
46'
47'
36'
26'
57'
16'
67'
45'
35'
34'
18
18'
49'
49
39'
39
29'
29
19'
19
59
48
58
68

Figura E2b (página anterior) – Grafo-testemunha p/ o Contraexemplo 2

Obs.: foram suprimidas as pontas das flechas que indicam a orientação das arestas incidentes aos SFCPs *S* e *S'*, por questões de clareza. Ressalte-se, contudo, que a orientação de *todas* as arestas incidentes a *S* ou *S' entram* nestes conjuntos. Atente-se, igualmente, para o eixo de simetria vertical presente no centro da figura. Os destaques foram dados apenas em sua metade (simétrica) esquerda.

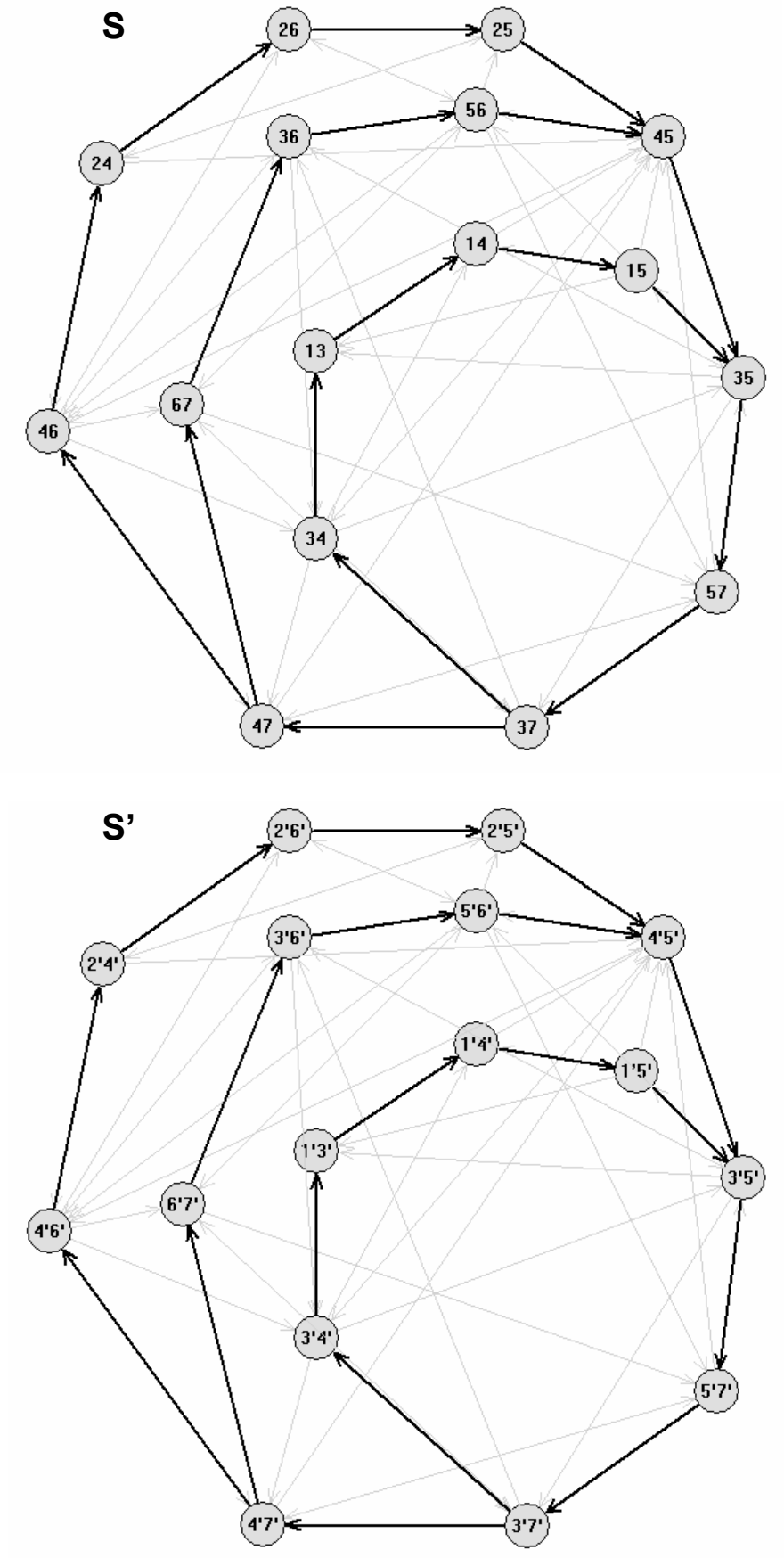


Figura E2c – Visualização completa dos SFCs da Figura E2b

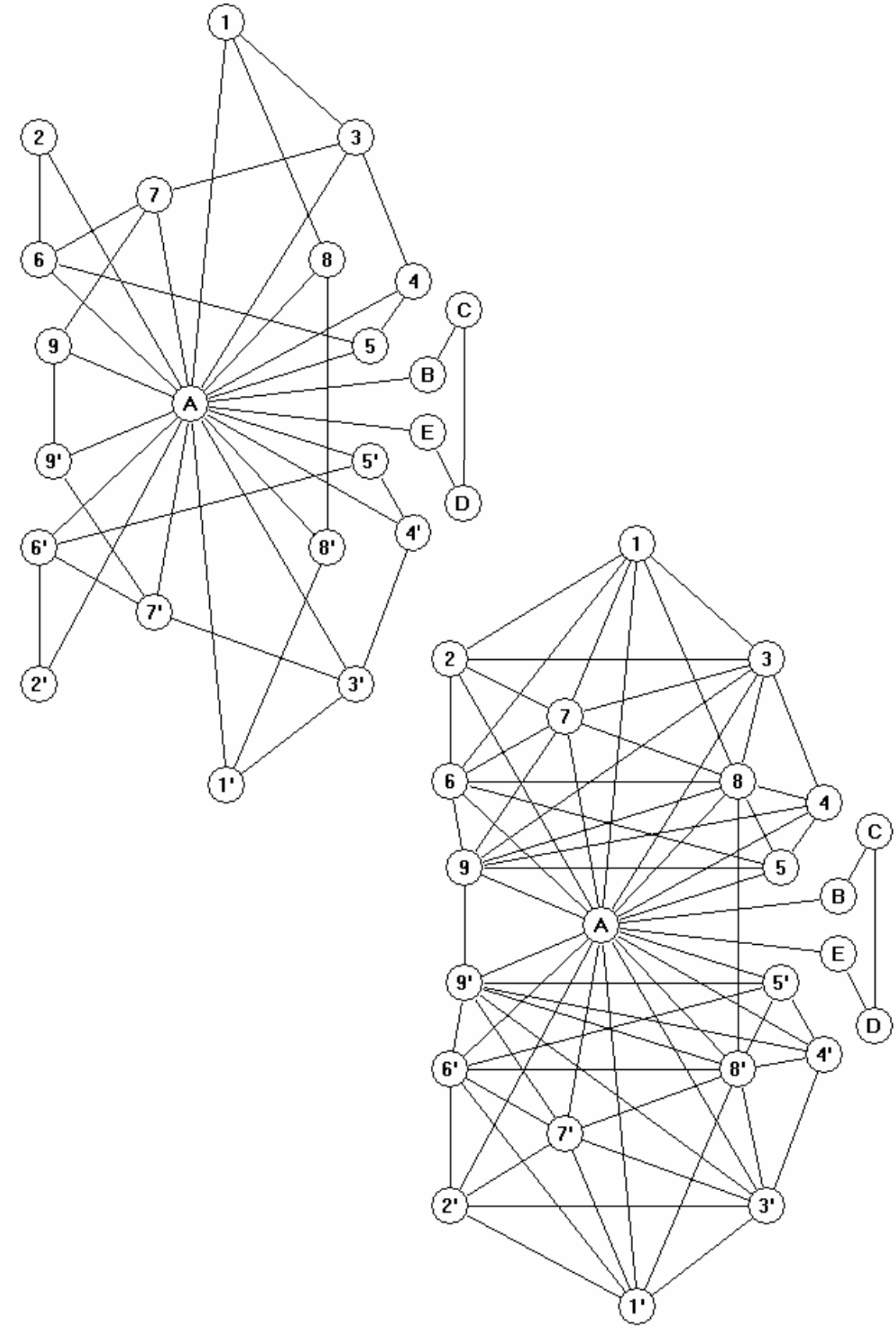


Figura E3a – Par de grafos que admite um CHS não associado a SFCP – Contraexemplo 3

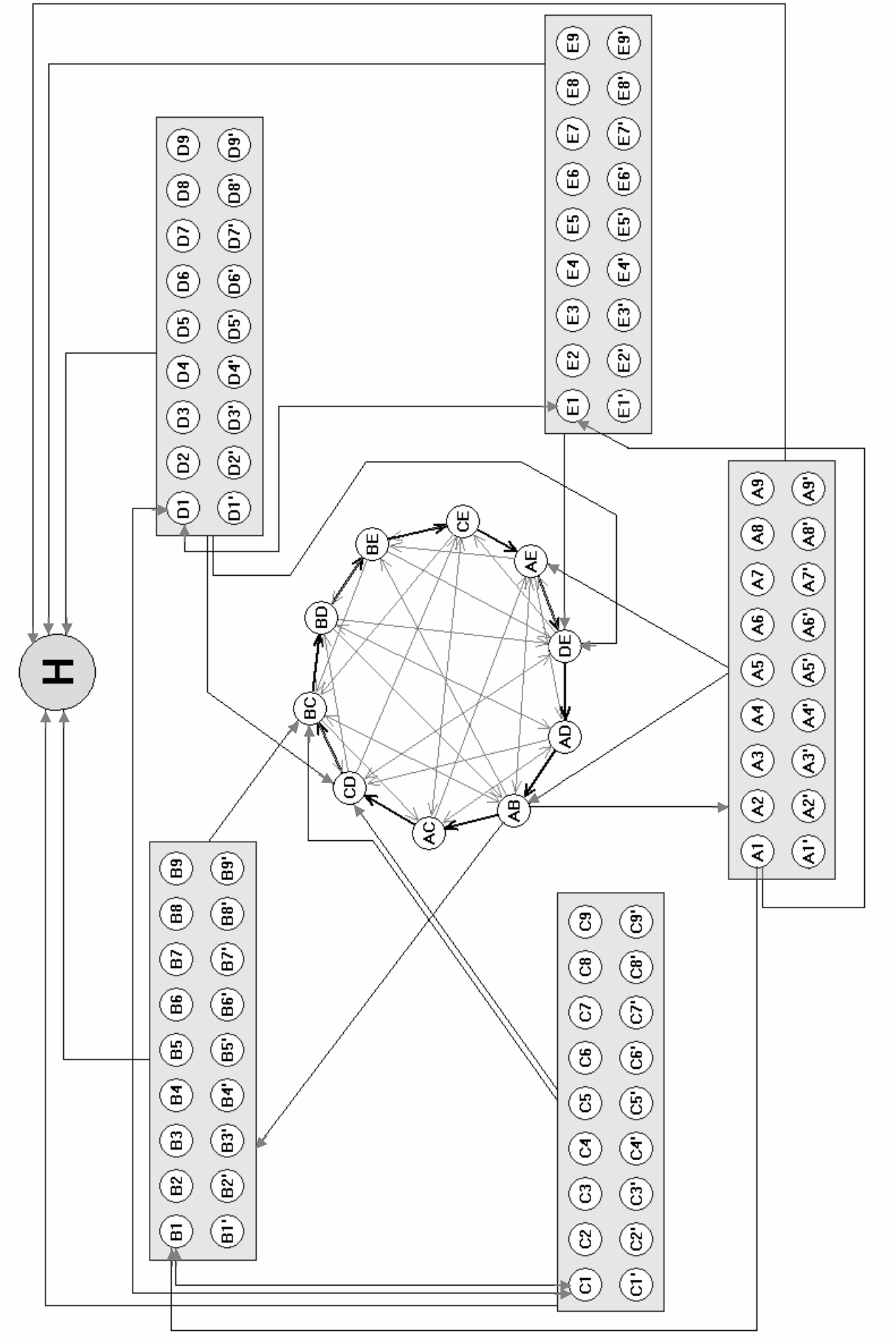
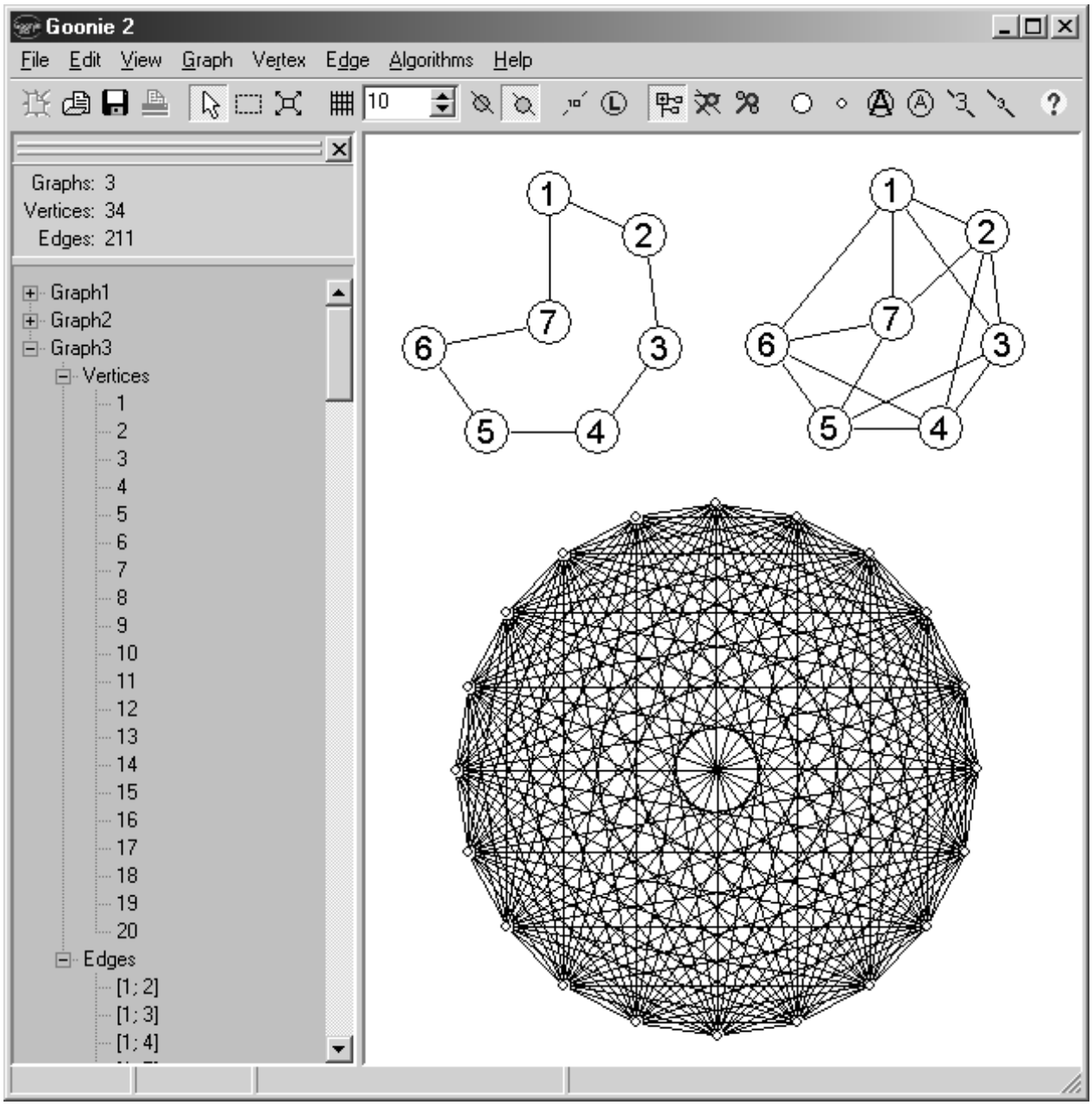

H
A1 A2 A3 A4 A5 A6 A7 A8 A9
A1' A2' A3' A4' A5' A6' A7' A8' A9'
B1 B2 B3 B4 B5 B6 B7 B8 B9
B1' B2' B3' B4' B5' B6' B7' B8' B9'
C1 C2 C3 C4 C5 C6 C7 C8 C9
C1' C2' C3' C4' C5' C6' C7' C8' C9'
D1 D2 D3 D4 D5 D6 D7 D8 D9
D1' D2' D3' D4' D5' D6' D7' D8' D9'
E1 E2 E3 E4 E5 E6 E7 E8 E9
E1' E2' E3' E4' E5' E6' E7' E8' E9'
AB AC AD AE BC BD BE CD CE DE

Figura E3b (página anterior) – Grafo-testemunha reduzido para o par de grafos da Figura E3a

Obs.: cada aresta de entrada incidente a um dos retângulos envolventes dos grupos de vértices { ⟨*A*1⟩, ⟨*A*2⟩, ... }, { ⟨*B*1⟩, ⟨*B*2⟩, ... }, ..., representa uma aresta de entrada para cada um dos vértices do grupo, todas provenientes do mesmo vértice-origem . O mesmo pode-se dizer das arestas de saída (provenientes de um dos retângulos envolventes), que representam uma aresta de saída para cada um dos vértices do grupo, todas convergindo no mesmo vértice-destino. Por outro lado, uma aresta que incida diretamente em apenas um dos vértices de um desses cinco grupos representa a existência de dezoito arestas paralelas, com vértices-origem e destino análogos aos da aresta representada. Ex.: ⟨*A*1⟩ → ⟨*B*1⟩ representa a existência de dezoito arestas, quais sejam: ⟨*A*1⟩ → ⟨*B*1⟩, ⟨*A*2⟩ → ⟨*B*2⟩, ..., ⟨*A*9'⟩ → ⟨*B*9'⟩. O conjunto *H* aglutina todo o grafo da Figura E2b, e suas arestas de entrada representam a existência de arestas de entrada para algum vértice de *H*, cuja identificação não é importante para a compreensão do que se pretende mostrar.



Todas as figuras ilustrando grafos e todas as implementações dos algoritmos foram feitas utilizando-se o programa *Goonie (Graphs Object-Oriented Novel Integrated Environment)*, criado pelo autor por ocasião do Projeto Final de Curso de sua graduação e atualizado para este trabalho.