Algoritmos e Complexidade de Decomposição em Grafos

Sulamita Klein

Tese submetida ao corpo docente da Coordenação dos programas de Pós-graduação em Engenharia da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências em Engenharia de Sistemas e Computação.

Aprovada por :

Prof. Jayme Luiz Szwarcfiter, Ph.D.
(Presidente)

Profª Celina Miraglia Herrera de Figueiredo, D.Sc.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Paulo Oswaldo Boaventura Netto, Dr.Ing.

Prof. Oscar Porto, D.Sc.

Profª Yoshiko Wakabayashi, Ph.D.

Rio de Janeiro, RJ – Brasil
Outubro de 1994

KLEIN, SULAMITA

Algoritmos e Complexidade de Decomposição em Grafos. [Rio de Janeiro] 1994.
X, 71 p., 29,7cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1994)
Tese – Universidade Federal do Rio de Janeiro, COPPE.

1 - Algoritmos
2 - Grafos Perfeitos
3 - Complexidade
4 - Decomposição
5 - Par Homogêneo
I. COPPE/UFRJ
II. Título (série).

Aos meus pais
e ao meu filho.

Agradecimentos

Ao professor Jayme Luiz Szwarcfiter por ter me dado a chance de começar de novo, me incentivado, e pela orientação competente, e segura ao longo de todo o doutorado.

À professora Celina Miraglia Herrera de Figueiredo pelo acompanhamento constante, pela leitura e crítica das primeiras versões e sugestões sempre valiosas, e principalmente pela amizade.

Aos professores Hazel Everett e Bruce Reed que me orientaram durante a minha estadia na UQAM em Montréal (no doutorado-sanduiche):

Á professora Hazel Everett pela sua orientação dedicada e interessada, e também pela acolhida carinhosa durante a minha estadia em Montréal.

Ao professor Bruce Reed pela feliz idéia de ter me sugerido trabalhar com Hazel em Montréal e pela sua efetiva participação na minha orientação com a sugestão de problemas, discussão de idéias e acompanhamento.

Ao professor Kyriakos Kilakos pelas conversas e discussoões proveitosas, e pelo companheirismo e amizade.

Ao professor René Ferland pela assistência e paciência nos meus primeiros passos no Latex, e também pela amizade.

Aos amigos Márcia Rosana Cerioli e Petrúcio Viana:

Á Marcia pelo constante apoio via e.mail enquanto eu estava em Montréal e também pela leitura e crítica de esboços iniciais dos meus trabalhos, e dicas no Latex.

Ao Petrúcio pela sua lógica irrepreensível.

Às amigas Célia, Carmen, Mônica (além de Celina e Márcia) pelo companheirismo e amizade que foram muito importantes no início do meu doutorado. Esse grupo de estudos que formamos durante um período deu incentivo e motivação ao que se seguiu.

A todos aqueles, em especial aos colegas da COPPE, que de alguma forma contribuiram para a elaboração desta tese.

Aos professores Valmir Carneiro Barbosa, Paulo Oswaldo Boaventura Netto, Oscar Porto e Yoshiko Wakabayashi, além de Celina e Jayme, pela participação na banca examinadora desta tese, pela leitura cuidadosa e pelas sugestões enriquecedoras.

E " last but not least" ao meu filho Siome, pelo carinho, pelo estímulo e incentivo constante, e também pela maioria dos desenhos que constam desta tese.

Life

Life is a chance, take it.
Life is beauty, admire it.
Life is beatitude, savour it.

Life is a dream, make it a reality.
Life is a challenge, face it.
Life is a duty, fulfil it.
Life is a game, play it.
Life is invaluable, take care of it.

Life is a wealth, preserve it.
Life is love, enjoy it.
Life is a mystery, penetrate it.

Life is a promise, keep it.
Life is sadness, get over it.
Life is an anthem, sing it.
Life is a struggle, accept it.
Life is a tragedy, seize it.

Life is adventure, dare it.
Life is happiness, earn it.
Life is life, stand up for it.

Mother Teresa.

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

Algoritmos e Complexidade de Decomposição em Grafos

Sulamita Klein

Outubro de 1994

Orientador : Jayme Luiz Szwarcfiter
Programa : Engenharia de Sistemas e Computação

Estudamos alguns tipos de decomposição em grafos.

Consideramos inicialmente a decomposição induzida pela existência de um conjunto homogêneo em um grafo. Apresentamos um algoritmo de tempo $O(n^3)$ que determina se um grafo admite um conjunto homogêneo.

A seguir consideramos a decomposição induzida pela existência de um par homogêneo em um grafo, que é uma generalização natural do conjunto homogêneo. Apresentamos um algoritmo de tempo $O(n^5)$ que determina se um grafo admite um par homogêneo.

Focalizamos também as decomposições de grafos em cliques, sujeitas a algumas restrições dadas a priori. Consideramos um caso particular, que chamamos de partição em clique-cruz e desenvolvemos um algoritmo linear para determinar se um grafo admite tal partição.

Discutimos problemas de reconhecimento de grafos com certos tipos de cortes, assim como corte clique, corte estrela, corte estável, corte assimétrico e outros. Mostramos que o problema CORTE ESTÁVEL e o problema CORTE MULTIPARTIDO COMPLETO são NP-completos.

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

Algorithms and Complexity of Graph Decomposition

Sulamita Klein

October, 1994

Thesis Supervisors : Jayme Luiz Szwarcfiter
Department : Computing and Systems Engineering

We study some types of graph decompositions.

Initially, we consider the decomposition induced by the existence of a homogeneous set in a graph. We present an $O(n^3)$ time algorithm which determines if a graph admits a homogeneous set.

Secondly, we consider the decomposition induced by the existence of a homogeneous pair in a graph. Homogeneous pairs are natural generalizations of homogeneous sets. We present an $O(n^5)$ time algorithm which determines if a graph admits a homogeneous pair.

We also study clique decompositions of a graph, with some restrictions given a priori. We consider a particular case called clique-cross partition and we develop a linear algorithm which determines if a graph admits such partition.

We discuss some recognition problems of graphs with certain cutsets, such as clique cutset, star cutset, stable cutset, skew cutset and others. We show that the problem STABLE CUTSET and the problem MULTIPARTITE CUTSET are NP-complete.

Índice

Lista de Figuras

Capítulo 1

Introdução

O estudo de algoritmos é um tópico central em Ciência da Computação. Nas últimas duas décadas ocorreram avanços significativos nesse campo. Esses avanços vão desde o desenvolvimento de algoritmos rápidos até a descoberta surpreendente de certos problemas naturais para os quais todos os algoritmos parecem ser ineficientes.

A área de Algoritmos Combinatórios lida com problemas que envolvem estruturas matemáticas finitas e discretas. É uma área nova e apenas nos últimos anos começou a emergir como um campo sistemático do conhecimento, ao invés de uma coleção de truques variados. Novos algoritmos têm surgido. Progressos rápidos têm sido feitos, principalmente de natureza matemática, especialmente na compreensão dos algoritmos, seu desenvolvimento e sua análise. Todos esses fatores fazem com que a área de Algoritmos Combinatórios tenha se tornado importante, localizando-se na fronteira da Ciência da Computação e da Matemática.

O assunto do nosso trabalho nesta tese foi justamente o desenvolvimento e análise de certos algoritmos combinatórios em uma estrutura matemática muito especial: os grafos.

O estudo de algoritmos em grafos interliga as áreas de Algoritmos Combinatórios e Teoria dos Grafos. A elaboração de novas estruturas teóricas motiva a procura de algoritmos que reconheçam essas estruturas.

Uma estratégia que tem se mostrado bastante efetiva na busca desses algoritmos é o método de Divisão e Conquista. Esse método, como o nome diz, procura dividir, isto é, decompor o grafo em uma hierarquia de componentes, resolver o problema nessas componentes e gradualmente ir juntando as soluções, de maneira que no final seja obtida a solução para o grafo original.

O uso dessa estratégia encontra forte motivação também na classe dos grafos perfeitos, uma classe de grafos em que todo subgrafo induzido ad-

mite uma coloração e uma clique com o mesmo tamanho. O problema do reconhecimento de grafos perfeitos está em aberto. Sabe-se que, juntando dois grafos perfeitos através de determinadas operações, como por exemplo substituição e identificação por clique, obtém-se um grafo também perfeito. Diz-se que essas operações preservam perfeição. Um caminho natural para testar a perfeição de um grafo, seria então decompor esse grafo através de alguma operação que preserve perfeição e testar a perfeição para cada um dos subgrafos obtidos. Infelizmente essa idéia não nos dá um teste geral para perfeição, mas nos possibilita reconhecer diversas subclasses de grafos perfeitos tais como a classe dos grafos separáveis por clique [18, 41, 38], a dos grafos i-triangularizados [41] e a dos triangularizados [41, 38]. E também como aplicação dessa técnica, podemos resolver os problemas de otimização: clique máxima e número cromático, para classes de grafos hereditários em geral, e em particular para algumas classes de grafos perfeitos, uma vez que saibamos resolver os problemas nas componentes mais baixas da hierarquia.

Neste trabalho nos preocupamos com o problema de dado um grafo, como encontrar certos conjuntos de vértices que induzem uma estrutura especial nesse grafo. Ou ainda, olhando a relação desse conjunto de vértices com o restante do grafo, como achar uma decomposição do grafo através desse conjunto.

Passamos agora a descrever a nossa contribuição e como o nosso trabalho está distribuído.

No capítulo presente, além da introdução e localização dos tópicos da tese, damos as definições básicas da Teoria dos Grafos e Complexidade, com as respectivas notações que serão usadas ao longo deste texto.

No capítulo 2, fazemos uma breve introdução à classe dos grafos perfeitos e também apresentamos em detalhes o algoritmo de 2-satisfabilidade de Aspvall, Plass e Tarjan [1], importantes para a compreensão dos algoritmos desenvolvidos nos capítulos posteriores.

No capítulo 3, consideramos dois tipos de decomposições homogêneas em um grafo, a decomposição produzida pela existência de um conjunto homogêneo e a produzida pela existência de um par homogêneo. Desenvolvemos algoritmos eficientes, i.e., polinomiais, para ambos os casos, provamos suas correções e analisamos suas complexidades. Observamos que o problema do reconhecimento de um grafo com par homogêneo era um problema em aberto, desde 1987, e pelo que é do nosso conhecimento, o primeiro algoritmo polinomial que encontra um par homogêneo (se existir) é o que apresentamos na seção 3.2.

No capítulo 4, estudamos o problema da cobertura por cliques generalizada, também chamado problema da cobertura-(H, K). Apresentamos o algoritmo desenvolvido por MacGillivray e Yu [29], que acha essa cobertura-(H, K) em tempo polinomial para um grafo H sem triângulos. Consideramos um caso particular desse problema , o qual denominamos Partição em Clique-Cruz e desenvolvemos um algoritmo (usando basicamente a mesma técnica dos algoritmos do capítulo 3) que resolve esse problema em tempo linear.

No capítulo 5, discutimos problemas de reconhecimento de grafos com determinados cortes, assim como corte clique, corte estrela, corte estável, corte multipartido completo e outros. Para os casos de corte estável e corte k-multipartido completo construímos reduções polinomiais (para cada caso) a partir de problemas NP-completos, estabelecendo assim que esses problemas são NP-completos.

Finalmente, no capítulo 6, apresentamos as conclusões desse trabalho, assim como direções para pesquisas futuras.

Os algoritmos originais desenvolvidos no capítulo 3 e no capítulo 4 foram feitos em co-autoria com os professores Hazel Everett e Bruce Reed. Os resultados originais do capítulo 5 foram obtidos em co-autoria com o professor Kyriakos Kilakos.

Os resultados desta tese foram reportados nos seguintes trabalhos [13], [14] e [25].

1.1 Definições e Notações Básicas

Nesta seção vamos estabelecer algumas definições e notações básicas que serão utilizadas ao longo desta tese.

1.1.1 Grafos

Um *grafo* é um par ordenado $G = (V, E)$ onde V é um conjunto finito não vazio de *vértices* e E é um conjunto de pares não ordenados de vértices distintos, chamados *arestas*. Denotaremos o conjunto de vértices de um grafo G por $V(G)$, ou simplesmente por V, caso não haja ambiguidade. A mesma convenção será usada para seu conjunto de arestas $E(G)$.

Um vértice x é *adjacente* a um vértice y em G se $\{x, y\}$ é uma aresta de G. Neste caso também dizemos que x e y são *vizinhos* em G. Denotamos o

conjunto dos vértices adjacentes a x por $\mathcal{N}_G(x)$, ou simplesmente por $\mathcal{N}(x)$, caso não haja dúvidas quanto ao grafo em questão.

Um vértice x de G é dito *isolado* quando $\mathcal{N}(x) = \emptyset$, e é *universal* quando $\mathcal{N}(x) = V \setminus \{x\}$.

Uma aresta $e = \{x, y\} \in E$ é *incidente* aos vértices x e y que são os *extremos* de e.

Duas arestas são *adjacentes* se possuem um vértice em comum.

O *grau* de x, denotado por $g_G(x)$ é o número de vizinhos de x, isto é, $g_G(x) = |\mathcal{N}(x)|$. O símbolo δ denota o maior grau de um vértice de G.

A *ordem* de um grafo G é o número de vértices de G, e é denotada por $|G|$ ou por $|V(G)|$. O *tamanho* de G é a soma do número de arestas de G com o número de vértices de G. Como é usual, vamos assumir que $|V(G)| = n$ e $|E(G)| = m$, onde $|E(G)|$ denota o número de arestas de G.

O *complemento* de um grafo G, denotado por $\overline{G}$, é o grafo que tem o mesmo conjunto de vértices de G e tal que dois vértices distintos são adjacentes em $\overline{G}$ se e somente se não são adjacentes em G.

Uma propriedade é dita *auto-complementar* quando é satisfeita pelo grafo e pelo seu complemento.

Um grafo $H = (V(H), E(H))$ é um *subgrafo* de um grafo $G = (V, E)$ se $V(H) \subseteq V$ e $E(H) \subseteq E$. Se $V(H) \subset V$, $V(H) \neq V$ então H é dito um *subgrafo próprio* de G. Dado um conjunto de vértices $X \subseteq V$, $X \neq \emptyset$, então o *subgrafo induzido* por X é o subgrafo H de G tal que $V(H) = X$ e $E(H)$ é o conjunto das arestas de G que têm ambos os extremos em X. O subgrafo induzido por X é denotado por $G[X]$.

Uma propriedade é dita *hereditária* caso seja satisfeita pelo grafo e por todos os seus subgrafos induzidos. Analogamente definimos uma classe de grafos hereditária.

O grafo $G \setminus v$, obtido do grafo G pela remoção de um vértice v, é o subgrafo induzido pelo conjunto $V \setminus \{v\}$. Analogamente, o grafo $G \setminus H$, obtido do

grafo G pela remoção de um subgrafo H, é o subgrafo induzido pelo conjunto $V \setminus V(H)$.

Dois grafos são *isomorfos* caso exista uma bijeção entre os seus conjuntos de vértices que preserve adjacência.

Quando um grafo G não contém subgrafo induzido isomorfo a algum grafo H, dizemos que G é um grafo *sem* H's ou *livre* de H's.

Um grafo G é *completo* se quaisquer dois vértices distintos de G são adjacentes. A menos de isomorfismo, existe um único grafo completo com n vértices. Denotamos tal grafo por K_n.

Dados dois vértices x e y de um grafo G, um *caminho* entre eles é uma seqüência da forma $P = [x = v_1, v_2, ..., v_k = y]$, onde os v_i's são vértices, dois a dois distintos, e $\{v_i, v_i + 1\} \in E$, $1 \leq i \leq k - 1$. Em particular, P é um *caminho induzido* se não existem arestas entre dois vértices não consecutivos. Denotamos por P_k um caminho induzido por k vértices. Observamos que P_k tem $k - 1$ arestas.

Um *ciclo* é uma seqüência $C = [v_1, v_2, ..., v_{k+1}]$ tal que $[v_1, v_2, ..., v_k]$ é um caminho, $v_1 = v_{k+1}$ e $k \geq 3$. Em particular C é um *ciclo induzido* ou *ciclo sem cordas* se não existem arestas entre dois vértices não consecutivos. Denotamos por C_k um ciclo induzido por k vértices. Observamos que C_k tem k arestas.

Uma *corda* em um ciclo é uma aresta no grafo, cujas extremidades são vértices não consecutivos no ciclo.

Um grafo G é *conexo* se, para todo par de vértices distintos u e v em G, existe um caminho de u a v. Caso contrário, G é *desconexo*. Uma *componente conexa* de G é um subgrafo maximal conexo de G.

Um *corte* de um grafo é um subconjunto de vértices desse grafo tal que o grafo obtido após a sua remoção é desconexo.

O *grafo de linha* $L(G)$ associado a um grafo G é o grafo cujos vértices são as arestas de G e cujas arestas são os pares de arestas adjacentes em G.

Um grafo $G = (V, E)$ é *bipartido* se V pode ser particionado em dois conjuntos X e Y ($X \cup Y = V$ e $X \cap Y = \emptyset$), tais que toda aresta tem um extremo em X e outro em Y. Os conjuntos X e Y são chamados *partes* de G. Em particular, G é *bipartido completo* se é bipartido e se todo vértice de X é adjacente a todo vértice de Y.

Um conjunto $M \subseteq E$ é um *emparelhamento* em G se cada vértice de V é extremo de no máximo uma aresta de M.

Um conjunto de vértices K de um grafo G é uma *clique* se $G[K]$ é um grafo completo.

Um conjunto de vértices S de um grafo G é um *conjunto estável* ou *conjunto independente* se os seus vértices são dois a dois não adjacentes em G, ou em outras palavras, se em $\overline{G}$ o conjunto S é uma clique.

Um grafo G é *k-multipartido*, $k \geq 2$ se V pode ser particionado em k conjuntos estáveis $V_1, V_2, ..., V_k$, não vazios, tais que se $\{x, y\} \in E$ então $x \in V_i$ e $y \in V_j$, $i \neq j$. G é *multipartido* se ele é k-multipartido para algum $k \geq 2$. Em particular um grafo *bipartido* é um grafo *2-multipartido*. G é *multipartido completo* se ele é multipartido e $\{x, y\} \in E$ se e somente se $x \in V_i$ e $y \in V_j$, $i \neq j$.

Uma *cobertura por cliques* de um grafo G é uma partição do conjunto de vértices de G onde cada classe da partição é uma clique.

Uma *cobertura por estáveis* ou *coloração* de um grafo é uma partição do conjunto de vértices desse grafo onde cada classe da partição é um conjunto estável. Uma *k-coloração* é uma cobertura por exatamente k conjuntos estáveis.

PARÂMETROS DE G:

$\omega(G)$ - *tamanho de uma maior clique*, i.e., o número de vértices de uma clique máxima de G.

$\kappa(G)$ - *tamanho de uma menor cobertura por cliques*, i.e., o número

mínimo de cliques que cobrem os vértices de G.

$\alpha(G)$ - *tamanho de um maior conjunto estável de* G, i.e., o número de vértices de um conjunto estável máximo. Também chamado *número de independência de* G.

$\chi(G)$ - *número cromático de* G, i.e., menor k para o qual existe uma k-coloração, ou ainda, o tamanho de uma menor cobertura por conjuntos estáveis.

Um grafo G é *perfeito* se para todo subgrafo induzido H de G temos $\omega(G) = \chi(G)$ (ou equivalentemente $\alpha(G) = \kappa(G)$). Grafos que não são perfeitos são ditos *imperfeitos*.

Um grafo G é *minimalmente imperfeito* se ele não é perfeito, mas todos os seus grafos induzidos próprios são perfeitos.

1.1.2 Grafos Direcionados

Um *grafo direcionado* ou *digrafo*, $\vec{G} = (V, \vec{E})$, consiste de um grafo com uma orientação no seu conjunto de arestas, i.e., cada aresta é um par ordenado de vértices distintos chamados *arestas direcionadas* ou *arcos*.

Um arco $a = (u, v)$ é *incidente* aos vértices u e v, que são os extremos de a. Além disso, dizemos que o arco a *sai* de u e *entra* em v. O *grau de entrada* de um vértice v é o número de arestas que entram em v. O *grau de saída* de um vértice v é o número de arestas que saem de v. Se o grau de entrada de v é nulo, v é chamado *fonte*. Se o grau de saída de v é nulo, v é chamado *sumidouro*.

As definições de *subgrafo, subgrafo induzido* por um conjunto de vértices são as mesmas que para grafos, resguardadas as direções das arestas.

Um *caminho direcionado* entre dois vértices u e v de $\vec{G} = (V, \vec{E})$ é uma sequência da forma $\vec{P} = [u = v_1, v_2, ..., v_k = v]$, onde os v_i's são vértices, dois a dois distintos, e $(v_i, v_i + 1) \in \vec{E}$, $1 \leq i \leq k - 1$.

Um grafo direcionado $\vec{G}$ é *fortemente conexo* se, para todo par de vértices

distintos u e v de $\vec{G}$, existe caminho direcionado de u para v e de v para u. Os subgrafos maximais fortemente conexos de um grafo direcionado não têm vértices em comum e são chamados *componentes fortemente conexas*. Dizemos que $(C_1, C_2, ..., C_k)$ é uma *ordenação topológica reversa* das componentes fortemente conexas de $\vec{G}$ se para todo par i, j, onde $1 \leq i \leq j \leq k$, não existe aresta direcionada de algum vértice de C_i para um vértice de C_j. Se C_k e C_l são duas componentes fortemente conexas tais que existe uma aresta direcionada de um vértice de C_k para um vértice de C_l então C_k é chamado de um *predecessor* de C_l e C_l é chamado de um *sucessor* de C_k.

1.1.3 As Classes P, NP, NP-completo

Um problema algorítmico costuma ser caracterizado como um conjunto de todos os possíveis dados do problema, conjunto esse denominado *conjunto de instâncias*, e uma questão solicitada, denominada *objetivo do problema*. *Resolver* um problema algorítmico consiste em desenvolver um algoritmo cuja entrada seja uma instância do problema e cuja saída responda ao objetivo do problema.

Denotamos um problema π com conjunto de instâncias D e questão Q por $\pi(D, Q)$.

Um problema $\pi(D, Q)$ é dito de *decisão* quando o objetivo consiste em decidir se a resposta a questão Q é SIM ou NÃO.

Um algoritmo é *eficiente* ou *polinomial* quando sua complexidade de tempo é uma função polinomial nos tamanhos dos dados de entrada.

Os problemas de decisão para os quais existem algoritmos eficientes que os resolvam constituem uma classe que é chamada *polinomial*, e é denotada por P.

Um problema de decisão é dito ser *não-determinístico polinomial* caso qualquer instância SIM para o problema possua um certificado sucinto, isto é, verificável em tempo polinomial no tamanho da instância. Essa classe de problemas de decisão é denotada por NP.

Sejam $\pi_1(D_1, Q_1)$ e $\pi_2(D_2, Q_2)$ dois problemas de decisão. Uma *transformação polinomial* de π_1 em π_2 é uma função $f : D_1 \rightarrow D_2$ tal que as seguintes condições são satisfeitas:

- f pode ser computada em tempo polinomial;
- para toda instância $I \in D_1$ do problema π_1, tem-se que $\pi_1(I)$ possui resposta SIM se se somente se $\pi_2(f(I))$ possui resposta SIM.

Um problema de decisão π pertence à classe *NP-Completo* quando as seguintes condições são satisfeitas:

- $\pi \in NP$;
- para todo problema $\pi' \in NP$ existe uma transformação polinomial de π' em π.

Um problema que pertence à classe P é chamado *polinomial*, e um problema que pertence à classe *NP-Completo* é chamado *NP-completo*.

Como fonte de referências sobre complexidade indicamos [17] e [36].

Capítulo 2

Preliminares

Este capítulo é dedicado à apresentação de um embasamento teórico para o nosso trabalho. Na primeira seção damos um panorama geral sobre a classe dos grafos perfeitos, motivando o próximo capítulo sobre decomposição homogênea. Na segunda seção apresentamos o algoritmo de 2-satisfabilidade, de grande importância no capítulo 3 e capítulo 4.

2.1 Grafos Perfeitos

Um grafo é *perfeito* se cada um de seus subgrafos induzidos tem número cromático igual ao tamanho de uma maior clique.

Os grafos perfeitos foram definidos em 1961 por Claude Berge [2], que na época propôs as seguintes conjecturas:

Conjectura 1 *Um grafo é perfeito se e somente se nem o grafo, nem seu complemento contém um ciclo induzido ímpar com mais de três vértices.*

Conjectura 2 *Um grafo é perfeito se e somente se seu complemento é perfeito.*

Trivialmente a Conjectura 1 implica a Conjectura 2. Por isso ficaram conhecidas como *Conjectura Forte* e *Conjectura Fraca* dos grafos perfeitos, respectivamente. Em 1972 Lovász [26] provou a Conjectura Fraca que passou então a ser chamada *Teorema dos Grafos Perfeitos*. A Conjectura Forte continua em aberto até hoje, estimulando uma grande quantidade de trabalhos com o seu desafio.

Nas últimas três décadas a Teoria de Grafos e a Combinatória tiveram um grande desenvolvimento. Ligados a esse desenvolvimento emergiram al-

gumas teorias fundamentais, assim como a teoria de *Complexidade Computacional* e a formulação geral de problemas de otimização combinatória em *Programação Linear*. O desenvolvimento de ambas está profundamente relacionado com grafos perfeitos. Os grafos perfeitos também contribuíram para o desenvolvimento da teoria dos "Anti-Blocking Polyhedra", para o estudo da estrutura das faces dos politopos e para a noção de integralidade dual total em programação linear inteira. Por outro lado, constituem uma classe razoavelmente grande de grafos para os quais o número cromático e o número de independência podem ser calculados em tempo polinomial [22]. Em diversas classes especiais de grafos perfeitos, os problemas de otimização são resolvidos polinomialmente por algoritmos rápidos, e envolvem idéias importantes de algoritmos combinatórios, tais como busca em profundidade e algoritmo guloso.

Mas, além disso, grafos perfeitos constituem também um assunto de permanente interesse para os mais tradicionais teóricos de grafos. Muitas vezes resultados sobre grafos perfeitos generalizam outros resultados importantes em diversos campos da Teoria dos Grafos.

Como fonte de consulta sugerimos [19, 28, 15].

2.1.1 O Teorema dos Grafos Perfeitos

Começamos esta seção enunciando o teorema central de grafos perfeitos.

Teorema 1 *[26] Teorema dos Grafos Perfeitos.*
Um grafo é perfeito se e somente se seu complemento é perfeito.

A peça chave na demonstração desse teorema é a *operação de substituição* que definimos a seguir:

Sejam G e H grafos e x um vértice de G. Construimos um novo grafo G' substituindo o vértice x pelo grafo H da seguinte maneira:
Tomamos a união disjunta de H e $G \setminus x$, e para todo par de vértices y e z, com $y \in G \setminus x$ e $z \in H$, adicionamos uma aresta $\{z, y\}$ se e somente se $\{x, y\}$ é uma aresta de G. Dizemos então que G' foi obtido por *substituição* de G.

A figura 2.1 apresenta um grafo G' obtido por substituição de um grafo G, onde o vértice x pertencente a G foi substituido por um grafo H, e H é um grafo induzido por um conjunto estável de dois vértices.

O próximo lema, provado por Lovász, é usado na prova do *Teorema dos Grafos Perfeitos*. Esse lema assegura que a operação de substituição por um grafo perfeito preserva a perfeição.

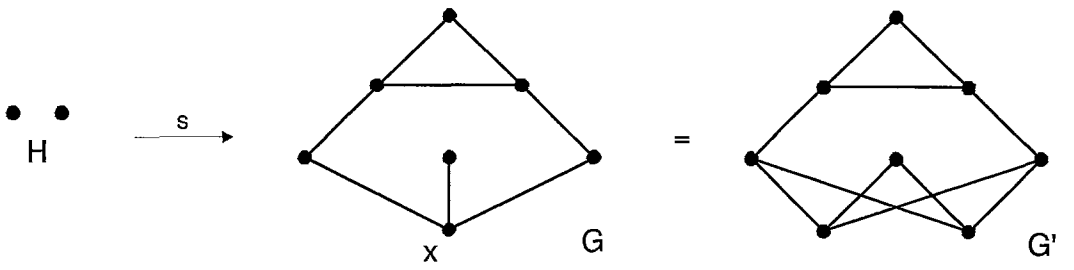


Figura 2.1: *Operação de substituição.*

Lema 1 *[26] Lema da Substituição.*
Se um vértice de um grafo perfeito é substituído por um grafo perfeito então o grafo resultante é perfeito.

Não podemos deixar de observar que Fulkerson [16] na mesma época, enquanto desenvolvia a Teoria dos "Anti-Blocking Polyhedra", havia reduzido a Conjectura Fraca à afirmação: *Substituição preserva perfeição*, chegando dessa forma bem perto da demonstração do Teorema dos Grafos Perfeitos.

Um *conjunto homogêneo* em um grafo G é um subconjunto próprio H dos vértices de G contendo pelo menos dois vértices e tal que $V(G) \setminus H$ pode ser particionado em $A = \{x | x$ é adjacente a todo vértice de $H\}$ e $N = \{x | x$ não é adjacente a nenhum vértice de $H\}$, onde A ou N são não vazios.

Na figura 2.2 apresentamos um exemplo de um grafo que contém um conjunto homogêneo.

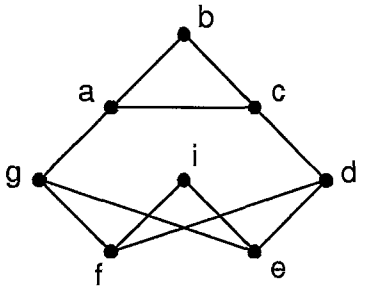


Figura 2.2: *Grafo com conjunto homogêneo* $H = \{e, f\}$.

Observamos que se G tem um conjunto homogêneo H e y é um vértice qualquer de H, então G pode ser obtido substituindo no grafo $G \setminus (H \setminus y)$, o vértice y pelo grafo induzido por H. Reciprocamente, caso G' tenha sido obtido por substituição de um vértice x em G por um grafo induzido por um conjunto de vértices H então H é um conjunto homogêneo em G' com $A = \mathcal{N}_G(x)$ e $N = \mathcal{N}_{\overline{G}}(x)$.

A operação de substituição, além de ser um conceito fundamental na demonstração do Teorema do Grafo Perfeito, tem sido usada também para provar que várias classes de grafos são perfeitas; como por exemplo, os grafos de comparabilidade [20] e os grafos sem P_4 [8].

2.1.2 Grafos Minimalmente Imperfeitos

A Conjectura Forte dos Grafos Perfeitos permanece em aberto até os dias de hoje como um problema interessante, de formulação simples, embora aparentemente muito difícil. Por tudo isso continua desafiando e estimulando uma grande quantidade de trabalhos.

Um grafo é *minimalmente imperfeito* se ele não é perfeito, mas todos os seus subgrafos induzidos próprios são perfeitos.

Esse conceito nos permite a seguinte formulação equivalente à Conjectura Forte:

Conjectura 3 *Os únicos grafos minimalmente imperfeitos são os ciclos induzidos ímpares com mais de três vértices e seus complementos.*

O Teorema dos Grafos Perfeitos pode ser também reformulado em termos desses grafos:

Teorema 2 *Um grafo é minimalmente imperfeito se e somente se seu complemento é minimalmente imperfeito.*

Um dos caminhos que tem sido muito explorado na tentativa de se provar a Conjectura Forte tem sido estudar e procurar novas propriedades dos grafos minimalmente imperfeitos. Acredita-se que se for obtida uma lista extensa dessas propriedades será possível mostrar que só estes ciclos induzidos e seus complementos satisfazem todas elas.

Nesse sentido, uma das primeiras contribuições foi a de Lovász [27], que mostrou que todo grafo G minimalmente imperfeito admite precisamente $\alpha(G).\omega(G)+1$ vértices (onde $\alpha(G)$ é o tamanho de um maior conjunto estável e $\omega(G)$ é o tamanho de uma maior clique de G). Além disso, para qualquer vértice x de G, $G \setminus x$ pode ser particionado em $\alpha(G)$ cliques de tamanho $\omega(G)$ e em $\omega(G)$ conjuntos estáveis de tamanho $\alpha(G)$. Em particular os grafos que têm essa propriedade recebem o nome de *particionáveis* ou *grafos* (α, ω). Observe que a classe dos grafos particionáveis contém a classe dos grafos minimalmente imperfeitos. Além disso, a maior parte das propriedades

estruturais dos grafos minimalmente imperfeitos são partilhadas pelos grafos particionáveis. No momento são conhecidas poucas propriedades que são satisfeitas pela classe dos grafos minimalmente imperfeitos e não são satisfeitas pela classe dos grafos particionáveis. Por isso tais propriedades adquirem especial importância e são chamadas propriedades que *separam*.

Como conseqüência dos resultados de Lovász acima mencionados, diversas propriedades dos grafos minimalmente imperfeitos, envolvendo cliques e conjuntos estáveis, foram obtidas por Padberg [32], e depois estendidas para os grafos particionáveis por Bland, Huang e Trotter [3]. Mas os estudos dos grafos minimalmente imperfeitos não têm se limitado apenas ao estudo de suas cliques e conjuntos estáveis. Certos tipos especiais de cortes também têm merecido atenção e resultados bastante interessantes acerca desses objetos têm sido obtidos. Como exemplos podemos citar os cortes clique e os cortes estrela.

Um *corte clique* em um grafo G é um corte de G que é uma clique. Na figura 2.3 apresentamos um exemplo de um grafo com um corte clique.

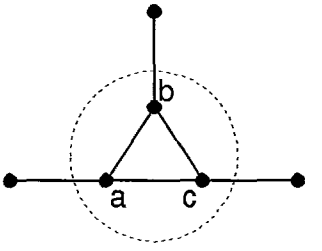


Figura 2.3: *Grafo com corte clique* $C = \{a, b, c\}$.

O lema a seguir é um resultado importante em coloração de grafos perfeitos [33].

Lema 2 *Nenhum grafo minimalmente imperfeito contém um corte clique.*

Um *corte estrela* em um grafo G é um corte de G que tem um vértice, chamado *centro da estrela*, que é adjacente a todos os outros vértices do corte. Observamos que um corte clique é um corte estrela em que todo vértice é o centro. Na figura 2.4 apresentamos um exemplo de um grafo com um corte estrela.

O conceito de corte estrela foi introduzido por Chvátal [5], que mostrou o lema a seguir, chamado *Lema do Corte Estrela*, o qual generaliza o resultado anterior sobre corte clique.

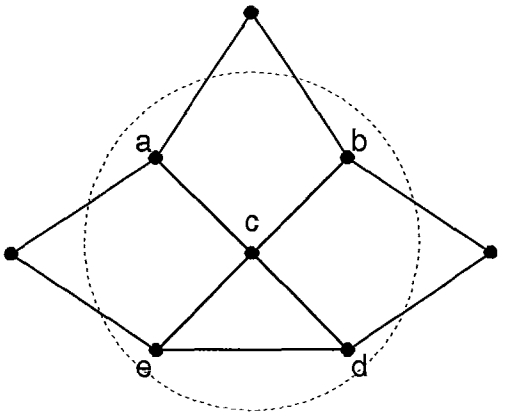


Figura 2.4: *Grafo com corte estrela* $C = \{a, b, c, d, e\}$, *onde* c *é o centro de* C.

Lema 3 *[5]*

Nenhum grafo minimalmente imperfeito contém um corte estrela.

O problema do reconhecimento de um grafo com corte clique foi resolvido por Whitesides [40] e Tarjan [38], que exibiram algoritmos polinomiais para determinar um corte clique. Já o problema do reconhecimento de um grafo com corte estrela foi resolvido pelo próprio Chvátal [5], que caracterizou grafos com corte estrela e apresentou um algoritmo polinomial decorrente dessa caracterização.

É interessante observar a estreita relação que existe entre os conceitos de corte estrela e conjunto homogêneo dada pelo lema a seguir:

Lema 4 *Se um grafo contém um conjunto homogêneo então o grafo ou o seu complemento contém um corte estrela.*

Prova. Seja G um grafo com um conjunto homogêneo H. Por definição $|H| \geq 2$ e V pode ser particionado nos conjuntos H, A e N (já definidos), onde A ou N são não vazios. Se A e N são não vazios então tomando qualquer vértice x de H, o conjunto $C = \{x\} \cup A$ é um corte estrela de G. Se $A = \emptyset$ então G é desconexo e qualquer vértice de H é um corte estrela de G. Se $N = \emptyset$ então $\overline{G}$ é desconexo e qualquer vértice de H é corte estrela de $\overline{G}$. ∎

Além disso, temos também como conseqüência do lema 1 o seguinte resultado:

Lema 5 *Nenhum grafo minimalmente imperfeito contém um conjunto homogêneo.*

Como a classe dos grafos perfeitos é auto-complementar, o conceito de corte estrela pode então ser visto como uma generalização do conceito de conjunto homogêneo.

Continuando nessa linha Chvátal [5] definiu um novo corte que chamou de *assimétrico* (skew), que por sua vez generaliza o conceito de corte estrela.

Um corte *assimétrico* em um grafo G é um corte de G que pode ser particionado em dois conjuntos V_1 e V_2 tais que se $x \in V_1$ e $y \in V_2$ então $\{x, y\} \in E$. Na figura 2.5 apresentamos um exemplo de um grafo com corte assimétrico.

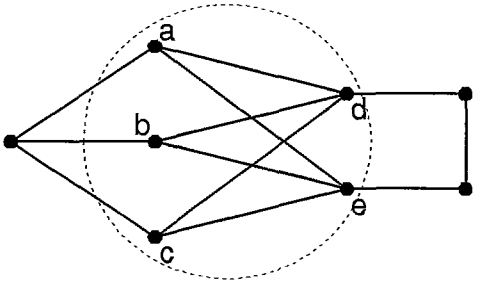


Figura 2.5: *Grafo com corte assimétrico* $C = \{a, b, c, d, e\}$.

Observe que se C é um corte assimétrico em G então $\overline{G[C]}$ é desconexo. Um corte assimétrico é uma generalização particularmente interessante de corte estrela porque temos o seguinte fato:
G tem um corte assimétrico se e somente se $\overline{G}$ tem um corte assimétrico.

Este fato sugere naturalmente a seguinte conjectura proposta por Chvátal [5]:

Conjectura 4 *Nenhum grafo minimalmente imperfeito contém um corte assimétrico.*

Essa conjectura assume uma importância especial por ser uma propriedade que separa a classe dos grafos particionáveis dos grafos minimalmente imperfeitos. Mas essa conjectura tem se mostrado mais difícil de ser provada do que se esperava. Procura-se então prová-la para casos mais simples.

Um corte *multipartido completo* em um grafo G é um corte de G que induz um grafo multipartido completo.

O corte multipartido completo é um caso particular do corte assimétrico. De fato: dado um corte multipartido completo C, tome um dos conjuntos

estáveis de C como V_1 e a união dos restantes como V_2. Temos que se $x \in V_1$ e $y \in V_2$ então $\{x, y\} \in E$ e logo C é um corte assimétrico. O exemplo da figura 2.5 é em particular um corte multipartido completo.

Recentemente Cornuejols e Reed [10] mostraram que:

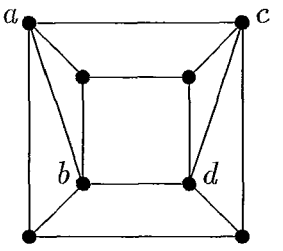
Lema 6 *Nenhum grafo minimalmente imperfeito contém um corte multipartido completo.*

Ainda a respeito de generalizações de conceitos e propriedades que separam, cabe aqui mencionar que o conceito de par homogêneo, aparece como uma generalização natural de conjunto homogêneo.

Um *par homogêneo* em um grafo $G = (V, E)$ é um par $\{Q_1, Q_2\}$ de conjuntos disjuntos de vértices desse grafo tal que:

- todo vértice de $V \setminus (Q_1 \cup Q_2)$ é adjacente a todos os vértices de Q_1 ou a nenhum vértice de Q_1;
- todo vértice de $V \setminus (Q_1 \cup Q_2)$ é adjacente a todos os vértices de Q_2 ou a nenhum vértice de Q_2;
- $|Q_1| \geq 2$ ou $|Q_2| \geq 2$;
- $|V \setminus (Q_1 \cup Q_2)| \geq 2$.

Na figura 2.6 apresentamos um exemplo de um grafo que contém um par homogêneo.



G

Figura 2.6: *Grafo G com par homogêneo* $\{\ \{a, b\}, \{c, d\}\ \}$

Note que um conjunto homogêneo Q_1 com $|V \setminus Q_1| \geq 2$, corresponde a um par homogêneo $\{Q_1, Q_2\}$ onde Q_2 é vazio. O resultado a seguir estabelece a relação entre pares homogêneos e a classe dos grafos perfeitos, assim como também generaliza o lema 5.

Lema 7 *[7] Nenhum grafo minimalmente imperfeito contém um par homogêneo.*

Os pares homogêneos foram introduzidos por Chvátal e Sbihi [7] em 1987, assim como o lema 7, para provar que os grafos Berge sem touro são perfeitos. Um grafo é *Berge* quando nem o grafo, nem seu complemento contém um ciclo ímpar induzido de tamanho pelo menos cinco. Um *grafo sem touro* é um grafo que não contém como subgrafo induzido um grafo isomorfo ao mostrado na Figura 2.7, é chamado um *touro*.

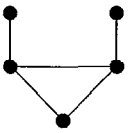


Figura 2.7: *touro*.

Como já comentamos anteriormente, a maior parte das propriedades conhecidas dos grafos minimalmente imperfeitos são também válidas para os grafos particionáveis.

Como exemplo de propriedades que são satisfeitas por estas duas classes, podemos citar a *não existência de cortes estrela* provada tanto para grafos minimalmente imperfeitos quanto para grafos particionáveis [5].

Como a existência de cortes estrela generaliza a existência de corte clique e também a existência de conjunto homogêneo (pelo lema 4), obtemos como conseqüência que a *não existência de cortes cliques* e a *não existência de conjuntos homogêneos* são propriedades que não separam.

Por outro lado, a *não existência de par homogêneo* é uma propriedade que separa, o que lhe confere um carácter especial. O mesmo se aplica a *não existência de corte assimétrico*, caso a conjectura 4 seja verdadeira.

Para mostrar que uma propriedade separa estas duas classes, basta exibir um grafo particionável para o qual a propriedade que vale para os minimalmente imperfeitos não vale para ele.

Assim, para a propriedade da não existência de corte assimétrico em um grafo minimalmente imperfeito, vamos considerar o grafo G dado na figura 2.8. G é particionável. De fato, basta observar que a remoção de qualquer vértice de G produz um grafo que pode ser particionado em três cliques de tamanho três ou em três conjuntos estáveis de tamanho três e G tem um conjunto estável de tamanho três e uma clique máxima de tamanho três. Além disso G não é um grafo minimalmente imperfeito porque contém um C_5 como subgrafo induzido próprio. Tome $V_1 = \{9, 3\}$ e $V_2 = \{8, 4\}$. $V_1 \cup V_2$ é um corte assimétrico de G.

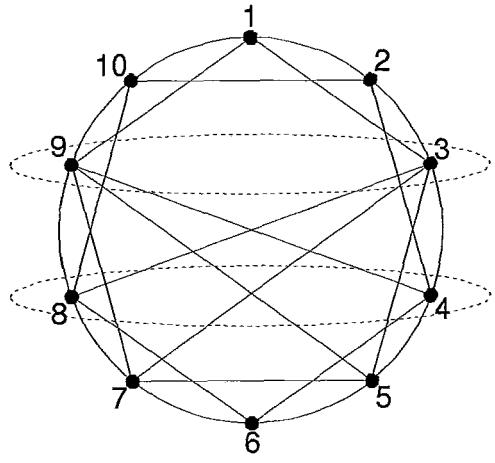


Figura 2.8: *Grafo G particionável com corte assimétrico $C = \{9, 3, 8, 4\}$.*

Agora, para mostrar que a não existência de par homogêneo é uma propriedade que separa vamos precisar da seguinte definição:

Dois vértices x e y de um grafo G são ditos *anti-gêmeos* se satisfazem:

- $G \setminus \{x, y\} = \mathcal{N}(x) \cup \mathcal{N}(y)$;
- $\mathcal{N}(x) \cap \mathcal{N}(y) = \emptyset$.

Temos então que:

Lema 8 *[7] Todo grafo com pelo menos cinco vértices que contém anti-gêmeos contém um par homogêneo.*

Prova. : Seja G um grafo com anti-gêmeos x e y e $|G| \geq 5$.

Seja $Q_1 = \mathcal{N}(x) \setminus \{x, y\}$ e $Q_2 = \mathcal{N}(y) \setminus \{x, y\}$.

Então Q_1 ou Q_2 tem pelo menos dois vértices e pela definição de anti-gêmeos, $\{Q_1, Q_2\}$ formam um par homogêneo para G. ∎

Como nenhum grafo minimalmente imperfeito contém um par homogêneo, temos que nenhum grafo minimalmente imperfeito contém um par de anti-gêmeos. Essa segunda propriedade embora saia como conseqüência da primeira, foi provada independentemente em [31].

Consideremos a seguir o grafo G dado na figura 2.9. Tal grafo é particionável. De fato, basta observar que a remoção de qualquer vértice de G produz um grafo que pode ser particionado em três cliques de tamanho três

ou em três conjuntos estáveis de tamanho três e G tem um conjunto estável de tamanho três e uma clique máxima de tamanho três. Além disso G não é um grafo minimalmente imperfeito porque contém um C_5 como subgrafo induzido próprio. Os vértices x e y constituem um par de anti-gêmeos em G, logo um par homogêneo em G. Exibimos então um grafo particionável que contém um par homogêneo, o que mostra que a não existência de par homogêneo é uma propriedade que separa.

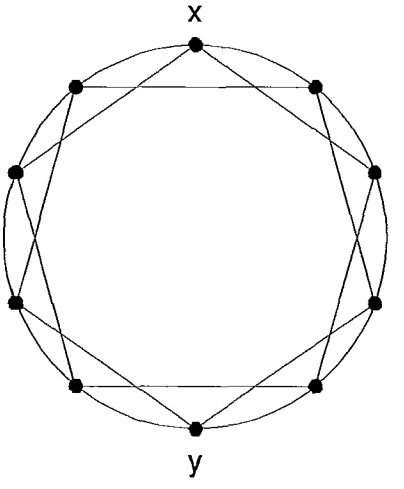


Figura 2.9: *Anti-gêmeos x e y.*

No capítulo 3 apresentamos um algoritmo polinomial para achar pares homogêneos em um grafo G, respondendo a uma questão formulada por Figueiredo [15]:

- É possível testar eficientemente a existência de um par homogêneo em um grafo?

2.2 Algoritmo 2-satisfabilidade

Seja $X = \{x_1, x_2, ..., x_n\}$ um conjunto de variáveis booleanas. Uma *atribuição de valores de verdade* para X é uma função:

$t : X \rightarrow \{verdadeiro, falso\}$.

Seja C uma fórmula booleana tal que C é uma conjunção de cláusulas, onde cada cláusula é uma disjunção de literais, e cada literal ou é uma variável x_i ou a negação de uma variável, $\overline{x_i}$. A fórmula C sobre X é *satisfazível* se e somente se existe uma atribuição de valores de verdade para X que satisfaz C.

Uma tal atribuição de valores de verdade é chamada *atribuição satisfatória* para C. O *problema de satisfabilidade* é determinar se existe uma atribuição satisfatória para C. É amplamente conhecido que o problema de satisfabilidade é NP-completo para fórmulas com três literais por cláusula (3-SAT) (veja [17]). Entretanto, o mesmo problema, para fórmulas com dois literais por cláusula (2-SAT) pode ser resolvido em tempo polinomial [17]; Even, Itai, e Shamir [12] delinearam o primeiro algoritmo de linear para o 2-SAT, e posteriormente, Aspvall, Plass e Tarjan [1] desenvolveram um algoritmo linear bem mais simples, o qual chamaram *algoritmo 2-satisfabilidade*. Vamos apresentar esse algoritmo a seguir.

Seja C uma fórmula sobre X tal que C tem no máximo dois literais por cláusula. Vamos assumir, sem perda de generalidade, que não existem cláusulas com apenas um literal (já que a cláusula u é equivalente à cláusula $u \vee u$).

Algoritmo 2-satisfabilidade
Entrada: uma fórmula C sobre $X = \{x_1, x_2, ..., x_n\}$
Saída: *SIM-C é satisfazível* ou *NÃO-C não é satisfazível.* Se a resposta for *SIM*, o algoritmo retorna também uma atribuição satisfatória para C.

Passo 1 Construa um grafo direcionado $\vec{G}_C = (V(\vec{G}_C), E(\vec{G}_C))$ com $2n$ vértices e $2|C|$ arestas como segue:

(i) Para cada variável x_i adicionamos a $V(\vec{G}_C)$ dois vértices que chamamos x_i e $\overline{x_i}$. Dizemos que um deles é o *complemento* do outro.

(ii) Para cada cláusula $u \vee v$ de C adicionamos a $E(\vec{G}_C)$ as arestas direcionadas $(\overline{u}, v)$ e $(\overline{v}, u)$. Na figura 2.10 apresentamos uma representação gráfica dessa relação.

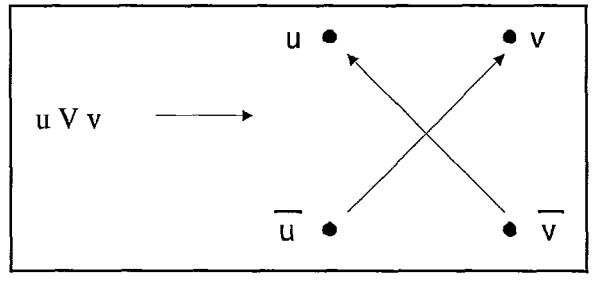


Figura 2.10: *Cláusula $u \vee v$ e sua representação gráfica.*

Passo 2 Ache as componentes fortemente conexas de $\vec{G}_C$ e retorne-as em ordem topológica reversa. Use, por exemplo, o algoritmo de Tarjan dado em [37].

Passo 3 Processe as componentes fortemente conexas (na ordem recebida) de $\vec{G}_C$ segundo o procedimento geral dado a seguir:

Se a componente S já foi marcada, não faça nada, passe para a seguinte. Se S não foi marcada, então se $S = \overline{S}$, onde $\overline{S}$ é a *componente dual* de S (a qual consiste do subgrafo induzido pelos vértices complementos dos vértices de S), retorne *NÃO-C não é satisfazível* e pare. Caso contrário ($S \neq \overline{S}$) marque S *verdadeira* e $\overline{S}$ *falsa.*

Passo 4 Retorne *SIM-C é satisfazível* e também a atribuição satisfatória para C e pare.

Observações:

Observação 1 *Por construção $\vec{G}_C$ tem a seguinte propriedade de dualidade: $\vec{G}_C$ é isomorfo ao grafo obtido dele mesmo, invertendo o sentido das arestas e complementando os nomes de todos os vértices (i.e., x passa a ser $\overline{x}$ e $\overline{x}$ passa a ser x).*

Observação 2 *Pela propriedade de dualidade observada acima, toda componente fortemente conexa S de $\vec{G}_C$ tem uma componente dual $\overline{S}$ que consiste do subgrafo induzido pelos vértices complementos dos vértices de S. Desse fato decorre que se S_1 e S_2 são duas componentes fortemente conexas de $\vec{G}_C$ e S_1 é um predecessor de S_2, então $\overline{S_1}$ é um sucessor de $\overline{S_2}$ e vice-versa. Logo temos que S_1 e S_2 são incomparáveis se e somente se $\overline{S_1}$ e $\overline{S_2}$ são incomparáveis.*

Observação 3 *Suponha que sejam atribuídos valores de verdade aos vértices de $\vec{G}_C$. Tal atribuição corresponde a um conjunto de valores de verdade para as variáveis que tornam C satisfazível se e somente se:*

(i) para todo i, vértices x_i e $\overline{x_i}$ recebem valores de verdade complementares;

(ii) nenhuma aresta (u, v) $(u \rightarrow v)$ tem u com a atribuição verdadeira *e v com atribuição* falsa. *Em outras palavras, nenhum caminho leva um vértice verdadeiro a um vértice falso.*

O teorema a seguir assegura a correção do algoritmo.

Teorema 3 *C é satisfazível se e somente se no grafo $\vec{G}_C$ nenhum vértice u_i está na mesma componente fortemente conexa que o seu complemento $\overline{u_i}$ (i.e., nenhuma componente fortemente conexa S é igual a sua componente dual $\overline{S}$).*

Prova. Suponha que existe um vértice u_i na mesma componente fortemente conexa que o seu complemento $\overline{u_i}$. Então qualquer atribuição de valores de verdade aos vértices de $\vec{G}_C$, deve violar ou a observação 3(i) ou a observação 3(ii); logo C não é satisfazível.

Para provar a recíproca, usamos o Passo 3 do algoritmo 2-satisfabilidade. Esse passo ou detecta a condição do teorema, i.e., acha algum vértice na mesma componente fortemente conexa que o seu complemento e só nesse caso pára prematuramente, ou marca as componentes fortemente conexas de $\vec{G}_C$ de forma que obtemos uma atribuição de valores de verdade para C. De fato, das propriedades de dualidade e indução decorre que toda componente marcada *verdadeira* tem apenas componentes verdadeiras como sucessoras e toda componente marcada *falsa* tem apenas componentes falsas como predecessoras. Logo, se não paramos prematuramente ao executar o Passo 3, as componentes são marcadas de tal maneira que as componentes duais têm valores complementares, e além disso, não existe nenhum caminho que vá de uma componente *verdadeira* para uma componente *falsa*. Se atribuímos a cada vértice a mesma marcação da componente que o contém, obtemos uma (atribuição de valores de verdade) satisfazendo a observação 3(i) ou a observação 3(ii). ∎

Consideremos agora um exemplo da execução do algoritmo 2-satisfabilidade aplicado a uma dada fórmula C.

Entrada: $C = \{a \vee b, b \vee \overline{c}, \overline{b} \vee \overline{d}, b \vee d, d \vee a\}$.

Passo 1: Construção do grafo $\vec{G}_C$. Na figura 2.11 exibimos o grafo $\vec{G}_C$ obtido.

Passo 2: Aplicação do algoritmo [37], que acha as componentes fortemente conexas de $\vec{G}_C$ e as retorna na ordem topológica reversa:

$S_1 = \{a\}$, $S_2 = \{\overline{c}\}$, $S_3 = \{d, \overline{b}\}$, $S_4 = \{b, \overline{d}\}$, $S_5 = \{c\}$, $S_6 = \{\overline{a}\}$. Veja figura 2.12.

Observe que $S_6 = \overline{S_1}$, $S_5 = \overline{S_2}$, $S_4 = \overline{S_3}$.

Passo 3: Processando as componentes:

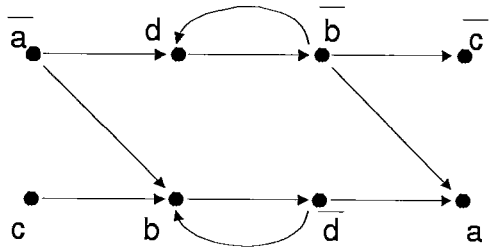


Figura 2.11: *Grafo* $\vec{G}_C$.

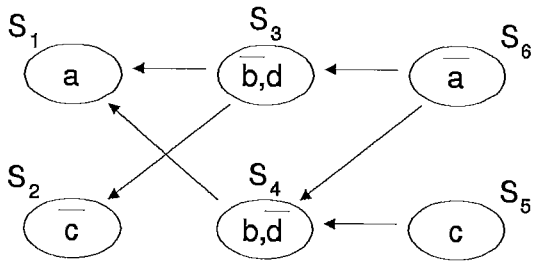


Figura 2.12: *As componentes fortemente conexas de* $\vec{G}_C$ *em ordem topológica reversa.*

$S_1 :=$ *verdadeiro* e $S_6 :=$ *falso*;
$S_2 :=$ *verdadeiro* e $S_5 :=$ *falso*;
$S_3 :=$ *verdadeiro* e $S_4 :=$ *falso*.

Passo 4: *SIM-C é satisfazível*, e a atribuição satisfatória para C é:
$a :=$ *verdadeiro*;
$b :=$ *falso*;
$c :=$ *falso*;
$d :=$ *verdadeiro*.

Capítulo 3

Decomposição Homogênea

Dizemos que um grafo admite uma *decomposição homogênea* quando o grafo possui um par homogêneo, ou em particular, um conjunto homogêneo.

O objetivo deste capítulo é apresentar dois algoritmos originais para decomposições homogêneas em um grafo. O primeiro determina se um grafo tem um conjunto homogêneo e encontra esse conjunto, caso ele exista, em tempo polinomial. O segundo, por sua vez, determina se um grafo tem um par homogêneo e encontra esse par, caso ele exista, também em tempo polinomial.

3.1 Um Algoritmo para Achar Conjuntos Homogêneos

3.1.1 Introdução

Um *conjunto homogêneo* em um grafo G é um conjunto H de vértices de G tal que:

- todo vértice de $V \setminus H$ é adjacente a todos os vértices de H ou a nenhum vértice de H;
- $|H| \geq 2$;
- $|V \setminus H| \geq 1$.

Observamos que essa definição é equivalente à definição dada na seção 2.1. De fato, a existência de um conjunto homogêneo $H \subset V$ particiona o subconjunto de vértices $V \setminus H$ em dois conjuntos, A e N, tais que:

$A = \{v \in V \mid \mathcal{N}(v) \cap H = H\}$;

$N = \{v \in V \mid \mathcal{N}(v) \cap H = \emptyset\}$.

Podemos então representar um grafo que possui conjunto homogêneo como o diagrama da figura 3.1, onde a linha contínua entre os conjuntos A e H representa a propriedade de que cada vértice de A é adjacente a cada vértice de H e a linha tracejada entre os conjuntos N e H representa o fato de que cada vértice de N não é adjacente a nenhum vértice de H.

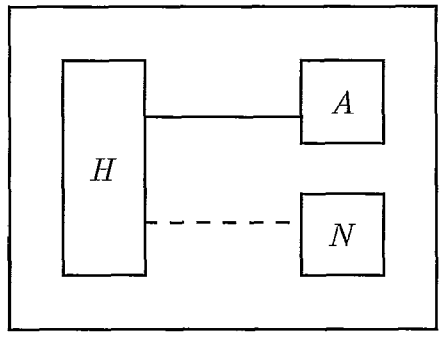


Figura 3.1: *Grafo G com conjunto homogêneo H.*

Nesse diagrama ficam bem evidenciadas as propriedades de adjacência do conjunto homogêneo H em relação aos conjuntos A e N.

Observe que conter um conjunto homogêneo é, por definição, uma propriedade auto-complementar. De fato: se o conjunto H é um conjunto homogêneo de G então H é também um conjunto homogêneo para o grafo complemento $\overline{G}$, onde os conjuntos A e N trocam de papéis, isto é:
$A = \{v \in V \mid \mathcal{N}(v) \cap H = \emptyset\}$;
$N = \{v \in V \mid \mathcal{N}(v) \cap H = H\}$.

Um resultado chave na teoria de grafos perfeitos citado na seção 2.1 é que nenhum grafo minimalmente imperfeito contém um conjunto homogêneo. De fato, esse é um lema importante usado por Lovász na prova de seu celebrado Teorema do Grafo Perfeito [26] : "G é perfeito se e somente se seu complemento é perfeito". Esse resultado tem sido usado para provar que várias classes de grafos são perfeitas, por exemplo, os grafos de comparabilidade [20] e os grafos sem P_4 [8].

Algoritmos polinomiais para achar conjuntos homogêneos são dados em [11, 30, 35]. O mais rápido deles é o algoritmo de tempo $O(m)$ desenvolvido por Spinrad (não publicado ainda; veja [35] para um algoritmo de tempo $O((m\alpha(m,n))))$.

O algoritmo que desenvolvemos para achar conjuntos homogêneos tem complexidade $O(n^3)$. Logo, comparado com alguns dos anteriores ele não é o mais eficiente. No entanto, ele é um algoritmo bastante simples e que além disso não requer uma estrutura de dados sofisticada.

Optamos por apresentá-lo por sua originalidade e também porque sua compreensão facilita o entendimento posterior do algoritmo para achar pares homogêneos, que nada mais é do que uma generalização desse algoritmo.

3.1.2 Descrição do Algoritmo Conjunto Homogêneo

Dado um grafo G, vamos descrever um algoritmo de tempo $O(n^2)$ que testa, para um vértice qualquer h de G, se G tem um conjunto homogêneo H com h em H. Começamos nosso algoritmo colocando todos os vértices do grafo G em uma lista $\mathcal{L}$. Aplicamos esse algoritmo a todos os elementos de $\mathcal{L}$, um de cada vez, a menos que o algoritmo pare antes, o que ocorre quando é encontrado um conjunto homogêneo, que nesse caso é retornado. No final, caso não seja encontrado nenhum conjunto homogêneo o algoritmo retorna *NÃO-G não tem um conjunto homogêneo.* O tempo total consumido pelo algoritmo é de $O(n^3)$.

Particionamos o conjunto de vértices $V^* = V \setminus h$ de G nos dois conjuntos seguintes:

$AH = \{x \in V^* \mid \{x, h\} \in E\}$;
$NH = \{x \in V^* \mid \{x, h\} \notin E\}$.

Considere um vértice $x \in AH$. Como x é adjacente a $h \in H$ ele não pode estar em N. Logo x só pode ser colocado no conjunto A ou no conjunto H, o que vamos indicar com a notação $x \rightarrow A$ ou $x \rightarrow H$. O conjunto AH contém então vértices de V^* que só podem estar em A ou em H. Podemos argumentar da mesma forma para um vértice $x \in NH$. Como x não é adjacente a $h \in H$, ele não pode estar em A. Portanto o conjunto NH contém vértices de V^* que só podem estar em N ou H. Temos então:

$AH \subseteq \{x \in V^* \mid x \rightarrow A \text{ ou } x \rightarrow H\}$;
$NH \subseteq \{x \in V^* \mid x \rightarrow N \text{ ou } x \rightarrow H\}$.

A idéia do algoritmo, o qual segue aproximadamente algumas idéias de Apsvall, Plass e Tarjan [1], é transformar os conjuntos AH e NH nos conjun-

tos A, N e H, especificando para cada vértice se ele deve ou não ser colocado em H. Dizemos que um vértice é INTERNO se ele é colocado em H e EXTERNO caso contrário. Então, depois que nossas escolhas foram feitas, A é a união dos vértices EXTERNOS de AH, assim como N é união dos vértices EXTERNOS de NH. Então H será formado pelos vértices INTERNOS de AH e de NH.

É fácil ver que G contém um conjunto homogêneo H contendo h se e somente se para todo par de vértices x e y as condições abaixo são satisfeitas. As condições (I)-(IV) asseguram que os vértices são colocados de maneira que todas as restrições de existência de arestas e não arestas são satisfeitas e a condição (V) assegura que H tenha no mínimo dois vértices e que $V \setminus H$ tenha pelo menos um vértice, como exige a definição de conjunto homogêneo.

(I) Se x e y são adjacentes e estão em NH então eles são ambos INTERNOS ou ambos EXTERNOS.

(II) Se x e y não são adjacentes e estão em AH então eles são ambos INTERNOS ou ambos EXTERNOS.

(III) Se x e y são adjacentes, x em NH e y em AH então se x é EXTERNO, y é também EXTERNO.

(IV) Se x e y não são adjacentes, x em NH e y em AH então se y é EXTERNO, x é também EXTERNO.

(V) Existe pelo menos um vértice EXTERNO e um vértice INTERNO.

Algoritmo Conjunto Homogêneo

Entrada: um grafo $G = (V, E)$ com $|V| \geq 3$.
Saída: *SIM-G tem um conjunto homogêneo* ou *NÃO-G não tem um conjunto homogêneo*. Se a resposta for *SIM*, o algoritmo também retorna um conjunto homogêneo H.

Passo 0 Coloque os vértices de G em uma lista $\mathcal{L}$.

Passo 1 Se $\mathcal{L}$ for vazia retorne *NÃO-G não tem um conjunto homogêneo* e pare. Caso contrário, seja h o primeiro vértice de $\mathcal{L}$. Remova h de $\mathcal{L}$.

Passo 2 Particione o conjunto $V^* = V \setminus h$ nos dois conjuntos:

$$AH = \{x \in V^* \mid \{x, h\} \in E\}$$

$$NH = \{x \in V^* \mid \{x, h\} \notin E\}.$$

Passo 3 Construa um grafo direcionado $\vec{G}_h = (V(\vec{G}_h), E(\vec{G}_h))$ como segue. O conjunto $V(\vec{G}_h) = AH \cup NH$. O conjunto $E(\vec{G}_h)$ é dado por:

(i) Para cada $\{x, y\} \in E$, se x e y pertencem ao mesmo conjunto NH, temos as arestas direcionadas (x, y) e (y, x).

(ii) Para cada $\{x, y\} \notin E$, se x e y pertencem ao mesmo conjunto AH, temos as arestas direcionadas (x, y) e (y, x).

(iii) Para cada $\{x, y\} \in E$, x em NH e y em AH corresponde a aresta direcionada (x, y).

(iv) Para cada $\{x, y\} \notin E$, x em NH e y em AH corresponde a aresta direcionada (y, x).

Passo 4 Ache as componentes fortemente conexas de $\vec{G}_h$ e retorne-as em ordem topológica reversa.

Passo 5 Processe as componentes fortemente conexas de $\vec{G}_h$ (em ordem topológica reversa) como segue. Se $\vec{G}_h$ tiver apenas uma componente fortemente conexa, então G não tem um conjunto homogêneo para esse vértice h escolhido. Neste caso, retorne para o Passo 1. Caso contrário, marque todos os vértices das componentes fortemente conexas que possuem predecessores com **E** e os vértices das outras componentes (que não possuem predecessores) com **I**. Se existem apenas componentes isoladas (i.e. componentes que não tem predecessores e nem sucessores), escolha uma delas, marque seus vértices com **I** e todos os vértices das outras componentes com **E**.

Passo 6 Faça:

$H = \{$vértices de AH marcados com **I** $\} \cup \{$vértices de NH marcados com **I**$\} \cup \{h\}$.

Retorne o conjunto H e a mensagem *SIM-G tem um conjunto homogêneo*. Pare.

3.1.3 Porque Funciona

Seja h um vértice qualquer de G. Vamos mostrar que o algoritmo marca corretamente os vértices do grafo direcionado $\vec{G}_h$ tal que as condições (I)-(V) são satisfeitas se e somente se G tiver um conjunto homogêneo H com $h \in H$. Considere as arestas direcionadas de $\vec{G}_h$. A aresta (x, y), por exemplo, onde $x \in NH$ e $y \in AH$ representa o fato de que se x for colocado em N, i.e., se x

for EXTERNO, então y deve ser colocado em A, i.e., y não pode ser INTERNO. Essa é exatamente a restrição dada pela condição (III). Logo os vértices de $\vec{G}_h$ podem ser marcados de maneira que não haja aresta (x, y) com x EXTERNO e y INTERNO se e somente se as condições (I)-(V) forem satisfeitas. Então provamos:

Lema 9 *G tem um conjunto homogêneo H com $h \in H$ se e somente se os vértices de $\vec{G}_h$ podem ser marcados ou* INTERNO *ou* EXTERNO *(ou **I** ou **E**) tal que as duas condições seguintes são satisfeitas:*

(i) nenhuma aresta direcionada (x, y), pode ter x EXTERNO *e y* INTERNO*;*

(ii) existe pelo menos um vértice EXTERNO *e um vértice* INTERNO*.*

O próximo lema é necessário para a prova de correção do algoritmo.

Lema 10 *Suponha que G tenha um conjunto homogêneo H com $h \in H$. Então cada componente fortemente conexa de $\vec{G}_h$ ou tem apenas vértices* EXTERNOS *ou apenas vértices* INTERNOS*.*

Prova. Suponha que temos na mesma componente fortemente conexa um vértice EXTERNO v e um vértice INTERNO u. Como eles estão na mesma componente, então existe um caminho direcionado de v para u e de u para v, e isso claramente implica que a condição (i) do Lema 9 é violada por alguma aresta. ■

Agora estamos prontos para enunciar o teorema que assegura a correção do algoritmo.

Teorema 4 *O algoritmo determina corretamente se G tem um conjunto homogêneo.*

Prova. Suponha que o algoritmo não retorne insucesso. Nesse caso, o algoritmo termina porque para algum vértice h, o grafo direcionado $\vec{G}_h$ tem mais de uma componente. Como todos os vértices de uma componente fortemente conexa recebem a mesma marca e apenas as componentes que não têm predecessores (considerando o caso em que as componentes não são todas isoladas) são marcadas **I**, concluímos que não há aresta direcionada (x, y) com x marcado **E** e y marcado **I**. É claro que isso é também verdade para o caso em que todas as componentes são isoladas. Como o algoritmo nos dá pelo menos uma componente marcada **E** e uma componente marcada **I** teremos pelo menos

um vértice INTERNO e um vértice EXTERNO. Então, pelo Lema 9, G tem um conjunto homogêneo.

Agora, vamos assumir que G tem um conjunto homogêneo H. Nesse caso precisamos mostrar que o nosso algoritmo realmente acha um conjunto homogêneo em G. Seja h um vértice de H. Então os vértices de $\vec{G}_h$ podem ser marcados de tal forma que as condições do Lema 9 são satisfeitas. Nós queremos mostrar que o algoritmo acha tal marcação. Suponha que o algoritmo falha em achar uma marcação. Isso implica que $\vec{G}_h$ possui apenas uma única componente fortemente conexa. De acordo com o Lema 9 e o Lema 10, os vértices dessa componente recebem todos a marca **E** ou todos a marca **I**. Se todos recebem a marca **E**, então todos os vértices serão EXTERNOS e teremos $H = \{h\}$, contradizendo a definição de conjunto homogêneo. Se todos recebem a marca **I**, então todos os vértices serão INTERNOS e teremos $V = H$, também contradizendo a definição de conjunto homogêneo. ∎

3.1.4 Complexidade

Podemos estimar a complexidade computacional do **Algoritmo Conjunto Homogêneo** como segue:

O Passo 0 pode ser implementado em tempo $O(n)$ e o Passo 1 pode ser feito em tempo constante. No Passo 2 a construção de cada um dos dois conjuntos é executada em um tempo proporcional a n. No Passo 3 a construção do grafo direcionado $\vec{G}_h$ leva claramente tempo $O(n^2)$. O Passo 4 pode ser feito usando o algoritmo de Tarjan [37], que acha componentes fortemente conexas e as retorna em ordem topológica reversa em tempo proporcional ao tamanho do grafo. Os passos 5 e 6 podem ser feitos em tempo linear. A execução dos passos 1 ao 5 requer tempo $O(n^2)$. Como os passos 1 a 5 são executados no máximo n vezes, o tempo total de execução do algoritmo é $O(n^3)$. Temos então:

Teorema 5 *O Algoritmo Conjunto Homogêneo determina se um grafo possui um conjunto homogêneo em tempo $O(n^3)$.*

3.1.5 Exemplo do Algoritmo Conjunto Homogêneo

Consideremos o grafo G dado na figura 3.2. Vamos apresentar a seguir um exemplo da execução do **Algoritmo Conjunto Homogêneo** aplicado ao grafo G.

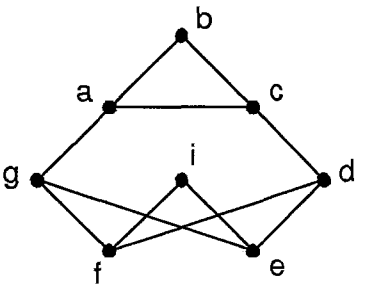


Figura 3.2: *Grafo G.*

Entrada: grafo $G = (V, E)$ dado na figura 3.2.

Passo 0: $\mathcal{L} = \{g, f, i, e, d, a, b, c\}$.

Iteração 1:

Passo 1: $h := g$; $\mathcal{L} = \{f, i, e, d, a, b, c\}$.

Passo 2: $V^* = V \setminus \{g\}$ é particionado nos conjuntos:
$AH = \{a, e, f\}$;
$NH = \{b, c, d, i\}$.

Passo 3: construção do grafo $\vec{G}_g = (V(\vec{G}_g), E(\vec{G}_g))$, onde,
$V(\vec{G}_g) = \{a, e, f, b, c, d, i\}$ e,
$E(\vec{G}_g) = \{(b, c), (c, b), (c, d), (d, c)$ (dados por (i)),
$(a, f), (f, a), (a, e), (e, a), (f, e)e(e, f)$ (dados por (ii)),
$(b, a), (d, e), (d, f), (i, e), (i, f)$ (dados por (iii)),
$(e, b), (f, b), (e, c), (f, c), (a, d), (a, i)$ (dados por (iv)) $\}$.
Na figura 3.3 apresentamos o grafo $\vec{G}_g$ obtido.

Passo 4: retorna: $C_1 = \{a, e, f, b, c, d, i\}$ (única componente fortemente conexa de $\vec{G}_g$).

Passo 5: retorna para o Passo 1.

Iteração 2:

Passo 1: $h := f$, $\mathcal{L} = \{i, e, d, a, b, c\}$.

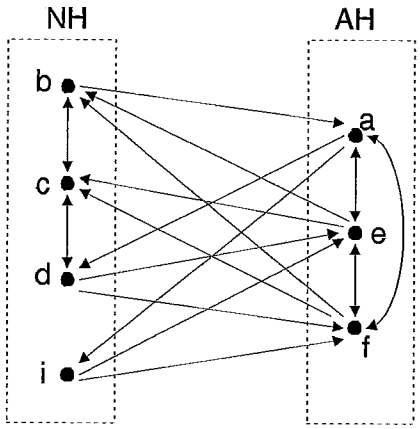


Figura 3.3: *Grafo* $\vec{G}_g$.

Passo 2: $V^* = V \setminus \{f\}$ é particionado nos conjuntos:
$AH = \{d, g, i\}$;
$NH = \{a, b, c, e\}$.

Passo 3: construção do grafo $\vec{G}_f = (V(\vec{G}_f), E(\vec{G}_f))$, onde,
$V(\vec{G}_f) = \{d, g, i, a, b, c, e\}$ e,
$E(\vec{G}_f) = \{(a,b),(b,a),(a,c),(c,a),(b,c),(c,b)$ (dados por (i)),
$(g,i),(i,g),(d,g),(g,d),(d,i),(i,d)$ (dados por (ii)),
$(a,g),(c,d),(e,g),(e,i),(e,d)$ (dados por (iii)),
$(i,a),(d,a),(g,b),(i,b),(d,b),(g,c),(i,c)$ (dados por (iv)) $\}$.
Na figura 3.4 apresentamos o grafo $\vec{G}_g$ obtido.

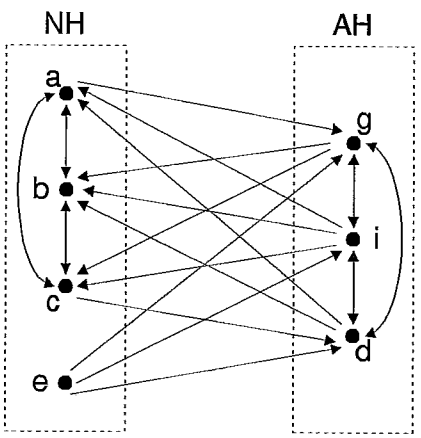


Figura 3.4: *Grafo* $\vec{G}_f$.

Passo 4: $C_1 = \{a, b, d, i, g\}$, $C_2 = \{e\}$ (na ordem topológica reversa).

A figura 3.5 nos mostra as componentes fortemente conexas obtidas em ordem topológica reversa.

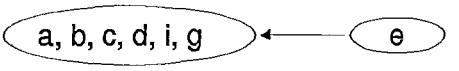


Figura 3.5: *As componentes fortemente conexas de $\vec{G}_f$ em ordem topológica reversa.*

Passo 5: $\{a^E, b^E, d^E, i^E, g^E\} \leftarrow \{e^I\}$.

Passo 6: retorna *SIM-G tem um conjunto homogêneo* $H = \{e, f\}$ e pára.

3.2 Um Algoritmo para Achar Pares Homogêneos

3.2.1 Introdução

Um *par homogêneo* em um grafo $G = (V, E)$ é um par $\{Q_1, Q_2\}$ de conjuntos disjuntos de vértices desse grafo tal que:

- todo vértice de $V \setminus (Q_1 \cup Q_2)$ é adjacente a todos os vértices de Q_1 ou a nenhum vértice de Q_1;
- todo vértice de $V \setminus (Q_1 \cup Q_2)$ é adjacente a todos os vértices de Q_2 ou a nenhum vértice de Q_2;
- $|Q_1| \geq 2$ ou $|Q_2| \geq 2$;
- $|V \setminus (Q_1 \cup Q_2)| \geq 2$.

Pares homogêneos são uma generalização de conjuntos homogêneos. Note que um conjunto homogêneo Q_1 corresponde a um par homogêneo $\{Q_1, Q_2\}$ onde Q_2 é vazio com $|V \setminus Q_1| \geq 2$.

Observe que a existência de um par homogêneo $\{Q_1, Q_2\}$ em um grafo G implica que o conjunto de vértices $V \setminus (Q_1 \cup Q_2)$ pode ser particionado em quatro conjuntos A, N, S_1 e S_2 tais que:

$A = \{v \in V \mid \mathcal{N}(v) \cap (Q_1 \cup Q_2) = Q_1 \cup Q_2\}$;
$N = \{v \in V \mid \mathcal{N}(v) \cap (Q_1 \cup Q_2) = \emptyset\}$;
$S_1 = \{v \in V \mid \mathcal{N}(v) \cap Q_1 = Q_1, \mathcal{N}(v) \cap Q_2 = \emptyset\}$;
$S_2 = \{v \in V \mid \mathcal{N}(v) \cap Q_2 = Q_2, \mathcal{N}(v) \cap Q_1 = \emptyset\}$.

onde $\mathcal{N}(v) = \{x \in V \mid \{v, x\} \in E\}$. Podemos representar um grafo G que tem um par homogêneo pelo diagrama na Figura 3.6, onde uma linha contínua entre dois conjuntos representa a propriedade de que cada vértice de um conjunto é adjacente a todos os vértices do outro conjunto. Uma linha pontilhada representa a propriedade de que nenhum vértice de um conjunto é adjacente a qualquer vértice do outro conjunto.

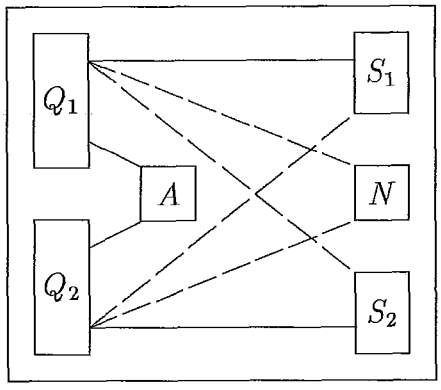


Figura 3.6: *Grafo com par homogêneo* $\{Q_1, Q_2\}$.

Chvátal e Sbihi [7] provaram que nenhum grafo minimalmente imperfeito contém um par homogêneo. Eles também mostraram que se um grafo G é um grafo Berge sem touro então G satisfaz uma das cinco propriedades seguintes:

(i) G é bipartido;

(ii) $\overline{G}$ é bipartido;

(iii) G tem um par homogêneo;

(iv) G tem um corte estrela;

(v) $\overline{G}$ tem um corte estrela.

Observamos que esse resultado implica que a Conjectura Forte dos Grafos Perfeitos vale para os grafos sem touro, ou em outras palavras, que todo grafo Berge sem touro é perfeito. De fato, suponha que isso não seja verdade. Então existe um grafo Berge sem touro G, minimalmente imperfeito. Mas G não pode ser um grafo bipartido, porque todo grafo bipartido é perfeito. Além do mais, pelo Teorema do Grafo Perfeito (teorema 1), $\overline{G}$ não pode ser bipartido. Sabemos também que G não pode conter nem um par homogêneo

e nem um corte estrela, porque essas estruturas não aparecem em grafos minimalmente imperfeitos (lema 7 e lema 3 respectivamente). E $\overline{G}$ também não pode conter um corte estrela pelo teorema 1. Logo G não satisfaz nenhuma das propriedades (i)-(v), o que é uma contradição.

Reed e Sbihi [34] desenvolveram um algoritmo polinomial para o reconhecimento da classe dos grafos Berge sem touros.

Observamos que o reconhecimento de um grafo com par homogêneo estava em aberto desde 1987.

3.2.2 Descrição do Algoritmo Par Homogêneo

Nosso algoritmo procura por um par homogêneo em dois estágios. Primeiro ele verifica se G tem um par homogêneo que é em particular um conjunto homogêneo; i.e., se G tem um conjunto homogêneo H com $|V \setminus H| \geq 2$. O lema 11 abaixo implica que podemos testar a existência de um tal H em tempo $O(m)$ usando o algoritmo de Spinrad.

Lema 11 *Se G é um grafo com $|V| \geq 4$ e H é um conjunto homogêneo de G com $|V \setminus H| = 1$ então G tem um conjunto homogêneo H' com $|V \setminus H'| \geq 2$ se e somente se o subgrafo $G[H]$ de G, induzido por H, contém um conjunto homogêneo.*

Prova. Seja G um grafo com $|V| \geq 4$, e H um conjunto homogêneo de G com $|H| = |V| - 1$, e seja $V \setminus H = \{x\}$. Nós observamos que ou x é adjacente a todos os vértices de H ou a nenhum vértice de H. Logo, se H' é um conjunto homogêneo de $G[H]$ então H' é um conjunto homogêneo de G com $|V \setminus H'| \geq 2$. Falta mostrar que se G tem um conjunto homogêneo H_1 com $|V \setminus H_1| \geq 2$ então $G[H]$ tem um conjunto homogêneo. Para isso, seja H_1 um conjunto homogêneo de G com $|V \setminus H_1| \geq 2$. Se $|H_1| \geq 3$ ou $x \notin H_1$ então, nesse caso $H_1 \cap H$ é um conjunto homogêneo de $G[H]$. Caso contrário, $H_1 = \{x, y\}$ para algum $y \in H$. Mas agora x e y são ambos adjacentes a todos os vértices de $V \setminus \{x, y\}$ ou a nenhum vértice de $V \setminus \{x, y\}$. Logo, $V \setminus \{x, y\}$ é um conjunto homogêneo de $G[H]$. ∎

No segundo estágio procuramos por um par homogêneo com ambos Q_1 e Q_2 não vazios e ou S_1 ou S_2 não vazios. Sem perda de generalidade podemos assumir que S_1 é não vazio. Começamos fazendo uma lista $\mathcal{L}$ de todas as possíveis triplas ordenadas de vértices (q_1, q_2, s_1) tais que $\{q_1, s_1\} \in E$ e $\{q_2, s_1\} \notin E$. Então G tem um par homogêneo se e somente se ele tem um par homogêneo com q_1 em Q_1, q_2 em Q_2 e s_1 em S_1 para alguma tripla de $\mathcal{L}$.

Nós descrevemos um algoritmo de tempo $O(n^2)$ que testa para uma tripla ordenada particular (q_1, q_2, s_1) se G tem um par homogêneo com q_1 em Q_1, q_2 em Q_2 e s_1 em S_1. Aplicamos esse algoritmo a todas as possibilidades, uma de cada vez. Se alguma vez é achado um par homogêneo, ele é retornado. Caso contrário, nós retornamos *NÃO-G não tem par homogêneo.* O tempo total de execução do algoritmo é de $O(n^5)$.

Nós particionamos o conjunto de vértices $V^* = V \setminus \{q_1, q_2, s_1\}$ de G nos oito conjuntos seguintes:

$AQ_1 = \{x \in V^* \mid \{x, q_1\} \in E,\ \{x, q_2\} \in E \text{ e } \{x, s_1\} \in E\}$;
$AQ_2 = \{x \in V^* \mid \{x, q_1\} \in E,\ \{x, q_2\} \in E \text{ e } \{x, s_1\} \notin E\}$;
$S_1Q_1 = \{x \in V^* \mid \{x, q_1\} \in E,\ \{x, q_2\} \notin E \text{e} \{x, s_1\} \in E\}$;
$S_2Q_1 = \{x \in V^* \mid \{x, q_1\} \notin E,\ \{x, q_2\} \in E \text{e} \{x, s_1\} \in E\}$;
$S_1Q_2 = \{x \in V^* \mid \{x, q_1\} \in E,\ \{x, q_2\} \notin E \text{ e } \{x, s_1\} \notin E\}$;
$S_2Q_2 = \{x \in V^* \mid \{x, q_1\} \notin E,\ \{x, q_2\} \in E \text{e} \{x, s_1\} \notin E\}$;
$NQ_1 = \{x \in V^* \mid \{x, q_1\} \notin E,\ \{x, q_2\} \notin E \text{e} \{x, s_1\} \in E\}$;
$NQ_2 = \{x \in V^* \mid \{x, q_1\} \notin E,\ \{x, q_2\} \notin E \text{ e } \{x, s_1\} \notin E\}$.

Considere um vértice $x \in AQ_1$. Como x é adjacente a $q_1 \in Q_1$, a $q_2 \in Q_2$ e a $s_1 \in S_1$ ele não pode estar nos conjuntos S_1 ou S_2 ou Q_2 ou N. Em outras palavras x deve ser colocado ou no conjunto A ou no conjunto Q_1, o que indicaremos com a notação $x \to A$ ou $x \to Q_1$. Dessa forma, usando uma análise similar para os outros conjuntos, temos:

$AQ_1 \subseteq \{x \in V^* \mid x \to A \text{ ou } x \to Q_1\}$;
$AQ_2 \subseteq \{x \in V^* \mid x \to A \text{ ou } x \to Q_2\}$;
$S_1Q_1 \subseteq \{x \in V^* \mid x \to S_1 \text{ ou } x \to Q_1\}$;
$S_2Q_1 \subseteq \{x \in V^* \mid x \to S_2 \text{ ou } x \to Q_1\}$;
$S_1Q_2 \subseteq \{x \in V^* \mid x \to S_1 \text{ ou } x \to Q_2\}$;
$S_2Q_2 \subseteq \{x \in V^* \mid x \to S_2 \text{ ou } x \to Q_2\}$;
$NQ_1 \subseteq \{x \in V^* \mid x \to N \text{ ou } x \to Q_1\}$;
$NQ_2 \subseteq \{x \in V^* \mid x \to N \text{ ou } x \to Q_2\}$.

A idéia do algoritmo, similar as do conjunto homogêneo e que segue aproximadamente algumas idéias de Apsvall, Plass e Tarjan [1], é transformar os conjuntos AQ_1, AQ_2, S_1Q_1, S_2Q_1, S_1Q_2, S_2Q_2, NQ_1 e NQ_2 nos conjuntos A, N, S_1, S_2, Q_1 e Q_2, especificando para cada vértice se ele deve ou não ser colocado em $Q_1 \cup Q_2$. Dizemos que um vértice é INTERNO se ele é colocado em $Q_1 \cup Q_2$; e EXTERNO, caso contrário. Então, depois que nossas escolhas foram feitas, A é a união dos vértices EXTERNOS de AQ_1 e AQ_2, S_1 é a união dos vértices EXTERNOS de S_1Q_1 e S_1Q_2, S_2 é a união dos vértices EXTERNOS

de S_2Q_1 e S_2Q_2 e N é a união dos vértices EXTERNOS de NQ_1 e NQ_2. Logo Q_1 será formado pelos vértices INTERNOS de AQ_1, S_1Q_1, S_2Q_1 e NQ_1, e Q_2 será formado pelos vértices INTERNOS de AQ_2, S_1Q_2, S_2Q_2 e NQ_2.

É fácil ver que G contém um par homogêneo com q_1 em Q_1, q_2 em Q_2 e s_1 em S_1 se e somente se para todo par de vértices x e y as seguintes condições são satisfeitas. As condições (I)-(VI) asseguram que os vértices sejam colocados de maneira que todas as restrições de existência de arestas e não arestas são satisfeitas e a condição (VII) assegura que Q_1 ou Q_2 tenha no mínimo dois vértices e que $V \setminus Q_1 \cup Q_2$ tenha pelo menos dois vértices, como exige a definição de par homogêneo.

(I) Se x e y não são adjacentes e estão em um dos conjuntos AQ_1, AQ_2, S_1Q_1 ou S_2Q_2, então eles são ambos INTERNOS ou ambos EXTERNOS.

(II) Se x e y são adjacentes e estão em um dos conjuntos S_1Q_2, S_2Q_1, NQ_1 ou NQ_2, então eles são ambos INTERNOS ou ambos EXTERNOS.

(III) Se $x \in AQ_1$, $y \in AQ_2$ e x e y não são adjacentes então eles são ambos INTERNOS ou ambos EXTERNOS. Similarmente para $x \in AQ_1$ e $y \in S_1Q_1$ ou $y \in S_1Q_2$, $x \in AQ_2$ e $y \in S_2Q_1$ ou $y \in S_2Q_2$, e $x \in S_1Q_2$ e $y \in S_2Q_1$.

(IV) Se $x \in S_1Q_1$, $y \in S_2Q_2$ e x e y são adjacentes então eles são ambos INTERNOS ou ambos EXTERNOS. Similarmente para $x \in S_1$ e $y \in NQ_2$, $x \in S_2Q_2$ e $y \in NQ_1$, $x \in S_1Q_2$ e $y \in NQ_2$, $x \in S_2Q_1$ e $y \in NQ_1$ e $x \in NQ_1$ e $y \in NQ_2$.

(V) Se $x \in AQ_1$, $y \in S_2Q_1$ e x e y não são adjacentes então se x é EXTERNO y também é EXTERNO. Similarmente para $x \in AQ_1$ e $y \in S_2Q_2$ ou $y \in NQ_1$ ou $y \in NQ_2$, $x \in AQ_2$ e $y \in S_1Q_1$ ou $y \in S_1Q_2$ ou $y \in NQ_1$ ou $y \in NQ_2$, $x \in S_1Q_1$ e $y \in S_2Q_1$ ou $y \in NQ_1$, $x \in S_1Q_2$ e $y \in S_1Q_1$ ou $y \in NQ_1$ e $x \in S_2Q_1$ e $y \in S_2Q_2$ ou $y \in NQ_2$.

(VI) Se $x \in AQ_1$, $y \in S_2Q_1$ e x e y são adjacentes então se x é INTERNO y também é INTERNO. Similarmente para $x \in AQ_1$ e $y \in S_2Q_2$ ou $y \in NQ_1$ ou $y \in NQ_2$, $x \in AQ_2$ e $y \in S_1Q_1$ ou $y \in S_1Q_2$ ou $y \in NQ_1$ or $y \in NQ_2$, $x \in S_1Q_1$ e $y \in S_2Q_1$ ou $y \in NQ_1$, $x \in S_1Q_2$ e $y \in S_1Q_1$ ou $y \in NQ_1$ e $x \in S_2Q_1$ e $y \in S_2Q_2$ ou $y \in NQ_2$.

(VII) Existe pelo menos um vértice EXTERNO e um vértice INTERNO.

Observamos que poderíamos evitar a condição (VII) tentando todas as possíveis escolhas para vértices INTERNOS e EXTERNOS em V^*, e aí então aplicar o algoritmo de Apsvall, Plass e Tarjan [1] diretamente para uma instância de 2-SAT descrevendo nossas condições. Isso nos daria um algoritmo de tempo $O(n^7)$ que com um pouco mais de cuidado pode ser melhorado para ser executado em tempo $O(n^6)$. Nós não sabemos se é possível aplicar Apsvall, Plass e Tarjan [1] para obter um algoritmo de tempo $O(n^5)$ para o nosso problema. Entretanto o nosso algoritmo segue o algoritmo deles bem de perto.

Algoritmo Par Homogêneo

Entrada: um grafo $G = (V, E)$ com $|V| \geq 4$.
Saída: *SIM-G tem um par homogêneo* ou *NÃO-G não tem um par homogêneo.* Se a resposta for *SIM*, o algoritmo também retorna um par homogêneo $\{Q_1, Q_2\}$.

Passo 0 Use o algoritmo de Spinrad de tempo $O(m)$ para testar se G tem um conjunto homogêneo. Se ele retornar um conjunto homogêneo H com $|H| \leq |V - 2|$, então retorne *SIM-G tem um par homogêneo,* $Q_1 = H$ *e* $Q_2 = \emptyset$ e pare. Se ele retornar um conjunto homogêneo H tal que $V \setminus H$ consiste de um único vértice x então aplique esse algoritmo de novo em $G[H]$. Se $G[H]$ tiver um conjunto homogêneo H' então retorne *SIM-G tem um par homogêneo,* $Q_1 = H'$ *e* $Q_2 = \emptyset$ e pare.

Passo 1 Faça uma lista $\mathcal{L}$ de todas as triplas ordenadas (a, b, c) de vértices de G de maneira que $\{a, c\} \in E$ e $\{b, c\} \notin E$.

Passo 2 Se $\mathcal{L}$ for vazia retorne *NÃO-G não tem um par homogêneo* e pare. Caso contrário, seja T a primeira tripla de $\mathcal{L}$. Seja q_1 o primeiro vértice de T, q_2 o segundo e s_1 o terceiro. Remova T de $\mathcal{L}$.

Passo 3 Particione o conjunto $V^* = V \setminus \{q_1, q_2, s_1\}$ nos oito conjuntos:

$AQ_1 = \{x \in V^* \mid \{x, q_1\} \in E, \{x, q_2\} \in E \text{ e } \{x, s_1\} \in E\}$;

$AQ_2 = \{x \in V^* \mid \{x, q_1\} \in E, \{x, q_2\} \in E \text{ e } \{x, s_1\} \notin E\}$;

$S_1Q_1 = \{x \in V^* \mid \{x, q_1\} \in E, \{x, s_1\} \in E \text{ e } \{x, q_2\} \notin E\}$;

$S_2Q_1 = \{x \in V^* \mid \{x, q_2\} \in E, \{x, s_1\} \in E \text{ e } \{x, q_1\} \notin E\}$;

$S_1Q_2 = \{x \in V^* \mid \{x, q_1\} \in E, \{x, q_2\} \notin E \text{ e } \{x, s_1\} \notin E\}$;

$S_2Q_2 = \{x \in V^* \mid \{x, q_2\} \in E, \{x, q_1\} \notin E \text{ e } \{x, s_1\} \notin E\}$;

$NQ_1 = \{x \in V^* \mid \{x, s_1\} \in E, \{x, q_1\} \notin E \text{ e } \{x, q_2\} \notin E\}$;

$NQ_2 = \{x \in V^* \mid \{x, q_1\} \notin E, \{x, q_2\} \notin E \text{ e } \{x, s_1\} \notin E\}$.

Passo 4 Construa um grafo direcionado $\vec{G}_{q_1,q_2,s_1} = (V(\vec{G}_{q_1,q_2,s_1}), E(\vec{G}_{q_1,q_2,s_1}))$ como se segue. O conjunto $V(\vec{G}_{q_1,q_2,s_1}) = AQ_1 \cup AQ_2 \cup S_1Q_1 \cup S_2Q_1 \cup S_1Q_2 \cup S_2Q_2 \cup NQ_1 \cup NQ_2$. O conjunto $E(\vec{G}_{q_1,q_2,s_1}))$ é dado pelas duas tabelas abaixo. Representamos a existência de duas arestas direcionadas (x, y) e (y, x) entre vértices x e y por $\langle x, y\rangle$. A primeira tabela mostra as arestas direcionadas dadas pela aresta $\{x, y\}$ de G, onde x pertence a um conjunto da primeira coluna e y pertence a um conjunto da primeira linha. A segunda tabela representa as arestas direcionadas dadas pela não existência de arestas $\{x, y\}$.

(arestas)	AQ_1	AQ_2	S_1Q_1	S_2Q_2	S_1Q_2	S_2Q_1	NQ_1	NQ_2
AQ_1				(y,x)		(y,x)	(y,x)	(y,x)
AQ_2			(y,x)		(y,x)		(y,x)	(y,x)
S_1Q_1		(x,y)		$\langle x,y\rangle$	(x,y)	(y,x)	(y,x)	$\langle x,y\rangle$
S_2Q_2	(x,y)		$\langle x,y\rangle$		(y,x)	(x,y)	$\langle x,y\rangle$	(y,x)
S_1Q_2		(x,y)	(y,x)	(x,y)	$\langle x,y\rangle$		(y,x)	$\langle x,y\rangle$
S_2Q_1	(x,y)		(x,y)	(y,x)		$\langle x,y\rangle$	$\langle x,y\rangle$	(y,x)
NQ_1	(x,y)	(x,y)	(x,y)	$\langle x,y\rangle$	(x,y)	$\langle x,y\rangle$	$\langle x,y\rangle$	$\langle x,y\rangle$
NQ_2	(x,y)	(x,y)	$\langle x,y\rangle$	(x,y)	$\langle x,y\rangle$	(x,y)	$\langle x,y\rangle$	$\langle x,y\rangle$

(não arestas)	AQ_1	AQ_2	S_1Q_1	S_2Q_2	S_1Q_2	S_2Q_1	NQ_1	NQ_2
AQ_1	$\langle x,y\rangle$	$\langle x,y\rangle$	$\langle x,y\rangle$	(x,y)	$\langle x,y\rangle$	(x,y)	(x,y)	(x,y)
AQ_2	$\langle x,y\rangle$	$\langle x,y\rangle$	(x,y)	$\langle x,y\rangle$	(x,y)	$\langle x,y\rangle$	(x,y)	(x,y)
S_1Q_1	$\langle x,y\rangle$	(y,x)	$\langle x,y\rangle$		(y,x)	(x,y)	(x,y)	
S_2Q_2	(y,x)	$\langle x,y\rangle$		$\langle x,y\rangle$	(x,y)	(y,x)		(x,y)
S_1Q_2	$\langle x,y\rangle$	(y,x)	(x,y)	(y,x)		$\langle x,y\rangle$	(x,y)	
S_2Q_1	(y,x)	$\langle x,y\rangle$	(y,x)	(x,y)	$\langle x,y\rangle$			(x,y)
NQ_1	(x,y)	(x,y)	(x,y)		(x,y)			
NQ_2	(x,y)	(x,y)		(x,y)		(x,y)		

Passo 5 Ache as componentes fortemente conexas de $\vec{G}_{q_1,q_2,s_1}$ e retorne-as em ordem topológica reversa.

Passo 6 Processe as componentes fortemente conexas de $\vec{G}_{q_1,q_2,s_1}$ (em ordem topológica reversa) como segue. Se $\vec{G}_{q_1,q_2,s_1}$ tiver apenas uma componente fortemente conexa, então G não tem um par homogêneo para essa tripla de vértices escolhida, logo retorne para o Passo 2. Caso contrário

marque todos os vértices das componentes fortemente conexas que possuem predecessores com **E** e os vértices das outras componentes (que não possuem predecessores) com **I**. Se existem apenas componentes isoladas (i.e. componentes que não tem predecessores e nem sucessores), escolha uma delas e marque seus vértices com **I** e marque todos os vértices das outras componentes com **E**.

Passo 7 Faça:

$Q_1 = \{$vértices de AQ_1 marcados com **I** $\} \cup \{$vértices de S_1Q_1 marcados com **I**$\} \cup \{$vértices de S_2Q_1 marcados com **I**$\} \cup \{$vértices de NQ_1 marcados com **I**$\} \cup \{q_1\}$.

$Q_2 = \{$vértices de AQ_2 marcados com **I** $\} \cup \{$vértices de S_1Q_2 marcados com **I**$\} \cup \{$vértices de S_2Q_2 marcados com **I**$\} \cup \{$vértices de NQ_2 marcados com **I**$\} \cup \{q_2\}$.

Retorne *SIM-G tem um par homogêneo* e os conjuntos Q_1, Q_2 e pare.

3.2.3 Porque funciona

Nesta seção, vamos assumir que temos uma tripla de vértices $\{q_1, q_2, s_1\}$ dadas pelo Passo 2. Vamos mostrar que o algoritmo marca corretamente os vértices do grafo direcionado $\vec{G}_{q_1,q_2,s_1}$ tal que as condições (I)-(VII) são satisfeitas se e somente se G tiver um par homogêneo com $q_1 \in Q_1$, $q_2 \in Q_2$ e $s_1 \in S_1$.

Considere as arestas direcionadas de $\vec{G}_{q_1,q_2,s_1}$. A aresta (x, y), por exemplo, onde $x \in S_2Q_2$ e $y \in AQ_1$, representa o fato de que se x for colocado em S_2, i.e., se x for EXTERNO, então y deve ser colocado em Q_1, i.e., y não pode ser INTERNO. Essa é exatamente a restrição dada pela condição (VI). Logo, os vértices de $\vec{G}_{q_1,q_2,s_1}$ podem ser marcados de maneira que não haja aresta (x, y) com x EXTERNO e y INTERNO se e somente se as condições (I) a (VI) são satisfeitas. Então provamos:

Lema 12 *G tem um par homogêneo com $q_1 \in Q_1$, $q_2 \in Q_2$ e $s_1 \in S_1$ se e somente se os vértices de $\vec{G}_{q_1,q_2,s_1}$ podem ser marcados ou* INTERNO *ou* EXTERNO *(ou **I** ou **E**) tal que as duas condições seguintes são satisfeitas:*

(i) nenhuma aresta direcionada (x, y) pode ter x EXTERNO *e y* INTERNO*;*

(ii) existe pelo menos um vértice EXTERNO *e um vértice* INTERNO*.*

O próximo lema é necessário para a prova de correção do algoritmo.

Lema 13 *Suponha que G tem um par homogêneo com $q_1 \in Q_1$, $q_2 \in Q_2$ e $s_1 \in S_1$. Então cada componente fortemente conexa de $\vec{G}_{q_1,q_2,s_1}$ ou tem apenas vértices* EXTERNOS *ou apenas vértices* INTERNOS.

Prova. Suponha que temos na mesma componente fortemente conexa um vértice EXTERNO v e um vértice INTERNO u. Como eles estão na mesma componente, existe um caminho direcionado de v para u e de u para v, e isso claramente implica que a condição (i) do lema 12 é violada por alguma aresta. ∎

Teorema 6 *O algoritmo determina corretamente se G tem um par homogêneo.*

Prova. Suponha que o algoritmo não retorne fracasso. Pelo lema 11 sobre conjuntos homogêneos, que são em particular pares homogêneos, se o algoritmo termina no Passo 0, então ele de fato retorna um par homogêneo. Caso contrário, o algoritmo termina porque para alguma tripla (q_1, q_2, s_1), o grafo direcionado $\vec{G}_{q_1,q_2,s_1}$ tem mais de uma componente. Como todos os vértices de uma componente fortemente conexa recebem a mesma marca, e apenas os vértices das componentes que não têm predecessores (considerando o caso em que as componentes não são todas isoladas) são marcados **I**, temos que existe aresta direcionada (x, y), com x marcado **E** e y marcado **I**. É claro que isso é também verdade para o caso em que todas as componentes são isoladas. Como temos pelo menos uma componente marcada **E** e uma componente marcada **I**, teremos pelo menos um vértice interno e um vértice externo. Então pelo lema 12, G tem um par homogêneo.

Agora, vamos assumir que G tem um par homogêneo $\{Q_1, Q_2\}$. Nesse caso precisamos mostrar que o nosso algoritmo realmente acha um par homogêneo em G. Se para esse par homogêneo um dos conjuntos Q_1, Q_2, ou $Q_1 \cup Q_2$ for um conjunto homogêneo contendo menos de $|V| - 1$ vértices, então pelo lema 11 sobre conjuntos homogêneos, nós acharemos no Passo 0 um tal conjunto homogêneo, o qual também é um par homogêneo. Caso contrário, G tem um par homogêneo $\{Q_1, Q_2\}$ tal que para alguma tripla de vértices (a, b, c) em G nós temos $a \in Q_1$, $b \in Q_2$, $c \in S_1$. Então os vértices de $\vec{G}_{a,b,c}$ podem ser marcados de tal forma que as condições do Lema 12 são satisfeitas. Nós queremos mostrar que o algoritmo acha tal marcação na iteração na qual $T = (a, b, c)$ (podemos assumir que o algoritmo executa essa iteração, porque senão ele pararia anteriormente e portanto teria achado um par homogêneo como requerido). Suponha que o algoritmo falha em achar uma marcação. Isso implica que $\vec{G}_{a,b,c}$ possui apenas uma única componente

fortemente conexa. De acordo com o lema 12 e lema 13, os vértices dessa componente recebem todos a marca **E** ou todos a marca **I**. Se todos recebem a marca **E**, então todos os vértices serão externos e teremos $Q_1 = q_1$ e $Q_2 = q_2$ contradizendo a definição de conjunto homogêneo. Se todos recebem a marca **I**, então todos os vértices serão internos e $V = Q_1 \cup Q_2$ também contradizendo a definição de par homogêneo. ∎

3.2.4 Complexidade

Podemos estimar a complexidade computacional do **Algoritmo Par Homogêneo** como segue:

Os Passos 0 e 1 podem ser implementados em tempos $O(m)$, $O(n^3)$ respectivamente, e o Passo 2 em tempo constante. No Passo 3 a construção de cada um dos oito conjuntos é executada em um tempo proporcional a n. No Passo 4 a construção do grafo direcionado $\vec{G}_{q_1,q_2,s_1}$ leva claramente tempo $O(n^2)$. O Passo 5 pode ser feito usando o algoritmo de Tarjan [37] que acha componentes fortemente conexas e as retorna em ordem topológica reversa em tempo proporcional ao tamanho do grafo. Os passos 6 e 7 podem ser feitos em tempo linear. A execução dos passos 2 a 5 requer tempo $O(n^2)$. Como os passos 2 a 5 são executados no máximo $\binom{n}{3}$ vezes, o algoritmo é executado em um tempo total de $O(n^5)$. Temos então:

Teorema 7 *O Algoritmo Par Homogêneo determina se um grafo possui um par homogêneo em tempo $O(n^5)$.*

3.2.5 Um Exemplo do Algoritmo Par Homogêneo

Consideremos o grafo G dado na figura 3.7. Vamos apresentar a seguir um exemplo da execução do **Algoritmo Par Homogêneo** aplicado ao grafo G.

Entrada: grafo $G = (V, E)$ da figura 3.7.

Passo 1: $\mathcal{L} = \{(e, f, h), (a, g, e), (e, b, f), (e, c, h), ...\}$

Passo 2: $q_1 := e, q_2 := f, s_1 := h$;
$\mathcal{L} = \{(a, g, e), (e, b, f), (e, c, h), ...\}$.

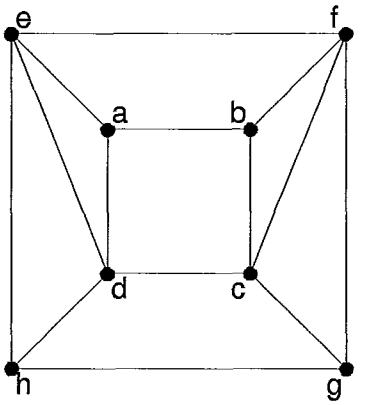


Figura 3.7: *Grafo G.*

Passo 3: $V^* = V \setminus \{e, f, h\}$ é particionado nos conjuntos:

$AQ_1 = \emptyset$;
$AQ_2 = \emptyset$;
$S_1Q_1 = \{d\}$;
$S_2Q_1 = \{g\}$;
$S_1Q_2 = \{a\}$;
$S_2Q_2 = \{b, c\}$;
$NQ_1 = \emptyset$;
$NQ_2 = \emptyset$.

Passo 4: Construção de $\vec{G}_{e,f,h} = V(\vec{G}_{e,f,h}), E(\vec{G}_{e,f,h}))$, onde,
$V(\vec{G}_{e,f,h}) = \{a, e, f, b, c, d, i\}$ e,
$E(\vec{G}_{e,f,h}) = \{(b, c), (c, b), (c, d), (d, c)$ (dados por (i)),
$(a, f), (f, a), (a, e), (e, a), (f, e)e(e, f)$ (dados por (ii)),
$(b, a), (d, e), (d, f), (i, e), (i, f)$ (dados por (iii)),
$(e, b), (f, b), (e, c), (f, c), (a, d), (a, i)$ (dados por (iv)) $\}$.
Na figura 3.8 apresentamos o grafo $\vec{G}_g$ obtido.

Passo 5: retorna:
$C_1 = \{b\}$, $C_2 = \{a, g\}$, $C_3 = \{c, d\}$.

A figura 3.9 nos mostra as componentes fortemente conexas obtidas em ordem topológica reversa.

Passo 6: retorna:

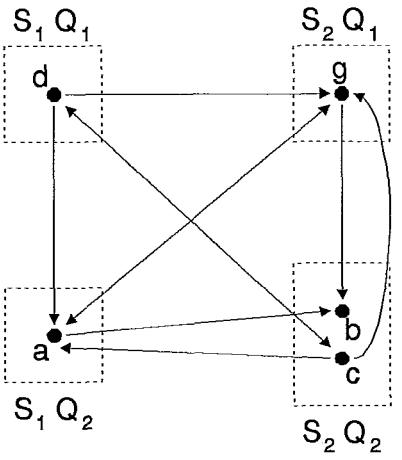


Figura 3.8: *Grafo* $\vec{G}_{e,f,h}$.

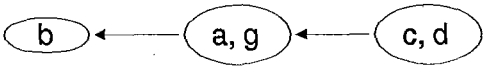


Figura 3.9: *componentes fortemente conexas de* $\vec{G}_{e,f,h}$ *em ordem topológica reversa.*

$$\{b^E\} \leftarrow \{a^E, g^E\} \leftarrow \{c^I, d^I\}.$$

Passo 7: retorna:

SIM-G tem um par homogêneo, $Q_1 = \{d, e\}$ e $Q_2 = \{c, f\}$ e pára.

Capítulo 4

Decomposição em Cliques

Nesse capítulo consideramos o problema de decompor um grafo em cliques, ou em outras palavras, como é mais conhecido, o problema de determinar uma cobertura por cliques desse grafo, obedecendo certas restrições dadas a priori.

4.1 Introdução

O problema generalizado da cobertura por cliques de um grafo G pode ser descrito assim:

Dado um grafo H, queremos saber se existe uma decomposição $V_1, V_2, \ldots, V_{|V(H)|}$ do conjunto dos vértices de G tal que cada V_i, $i = 1, 2, \ldots, |V(H)|$, é uma clique (possivelmente vazia), e o grafo $\hat{G}$ obtido pela contração de cada conjunto V_i em um único vértice v_i é um subgrafo de H. O grafo $\hat{G} = (\hat{V}, \hat{E})$ é definido por: $\hat{V} = \bigcup\{v_i\}$ $i = 1, 2, \ldots, |V(H)|$ e $\hat{E} = \{\{v_i, v_j\} \mid$ existe uma aresta entre um vértice de V_i e um vértice de $V_j\}$.

Se essa partição existe, dizemos que $V_1, V_2, \ldots, V_{|V(H)|}$ é uma *cobertura-*(H, K) *de* G, onde K é o conjunto de todos os grafos completos (observe que $G[V_i] \in K$ para todo $i = 1, 2, \ldots, |V(H)|$). Os conjuntos V_i, $i = 1, 2, \ldots, |V(H)|$, são chamados de *blocos* da cobertura.

A origem desse problema vem do problema geral COBERTURA-(H, C) de um grafo G quando C é um conjunto qualquer de grafos e H é um grafo dado. Existem alguns problemas interessantes e bastante conhecidos de cobertura de grafos onde conjuntos especiais C são considerados. Por exemplo, seja C o conjunto I dos grafos induzidos por conjuntos independentes, i.e., grafos sem arestas, incluindo o grafo vazio. O problema COBERTURA-(H, I) é justamente decidir se os vértices de G podem ser cobertos por $|V(H)|$ conjuntos independentes $\{V_h | h \in V(H)\}$, disjuntos dois a dois, tais que se $x \neq y$,

$x, y \in V(H)$, então existe uma aresta com um extremo em V_x e o outro extremo em V_y somente se $\{x, y\} \in E(H)$. Em outras palavras, COBERTURA-(H, I) é H-COLORAÇÃO [23]. Se H é em particular um grafo completo com n vértices denotado por K_n então o problema COBERTURA-(K_n, I) é decidir se os vértices de um grafo G podem ser partitionados em no máximo n conjuntos independentes, ou equivalentemente se G pode ser colorido com n cores. Logo o problema COBERTURA-(K_n, I) é exatamente o famoso problema n-COLORAÇÃO [24]. Agora, considerando C como o conjunto K de todos os grafos completos, incluindo o grafo vazio, o problema COBERTURA-(K_n, K), por sua vez, procura decidir se os vértices de um grafo G podem ser particionados em no máximo n cliques. O problema COBERTURA-(K_n, K) é também conhecido como PARTIÇÃO EM CLIQUES [24, 17].

MacGillivray and Yu [29] mostraram que se H é um grafo sem triângulos, então o problema generalizado da cobertura por cliques é polinomial, desenvolvendo um algoritmo de tempo $O(n^{|V(H)|+2})$ para resolvê-lo. Apresentamos esse algoritmo na seção 4.2. Por outro lado, se $K_3 \subset H$, esses mesmos autores acreditam que o problema seja NP-completo. No momento eles estão trabalhando nessa direção.

Na seção 4.3 consideramos um caso particular do problema generalizado da cobertura por cliques de um grafo G (cobertura-(H, K)) para H sem triângulos, o qual denominamos partição em clique-cruz, e apresentamos um algoritmo ótimo para resolvê-lo.

Consideramos apenas grafos finitos, não direcionados $G = (V, E)$. Vamos seguir a notação e algumas definições dadas por McGillivray and Yu [29].

4.2 Descrição do Algoritmo de MacGillivray e Yu

Vamos supor, daqui para a frente, que o grafo dado H é sem triângulos e $V(H) = \{1, 2, \ldots, k\}$. Dada $V_1, V_2, \ldots, V_k$ uma coleção de subconjuntos de vértices de G, disjuntos dois a dois, dizemos que $V_1, V_2, \ldots, V_k$ é *extensível* se existe uma cobertura-(H, K) dada por $V_1', V_2', \ldots, V_k'$ de G, tal que $V_i \subseteq V_i'$, $1 \leq i \leq k$. (Estamos supondo que cada V_i' será contraído a um vértice i). A cobertura $V_1', V_2', \ldots, V_k'$ é dita uma *extensão* de $V_1, V_2, \ldots, V_k$.

A idéia principal do algoritmo de MacGillivray e Yu é, uma vez dada uma coleção de subconjuntos $V_1, V_2, \ldots, V_k$, disjuntos dois a dois, determinar se essa coleção é extensível. O lema a seguir mostra que isso pode ser feito em tempo polinomial. A sua demonstração é baseada na descrição de um

algoritmo polinomial, que é justificado passo a passo.

Lema 14 (MacGillivray,Yu) *Sejam G e H dois grafos dados, $|V(H)| = k$. Seja $V_1, V_2, \ldots, V_k$ uma coleção de subconjuntos de V, disjuntos dois a dois. Se H é sem triângulos, então existe um algoritmo polinomial que decide se $V_1, V_2, \ldots, V_k$ é extensível.*

Prova. Consideremos inicialmente as seguintes observações:

Observação 1: Como cada bloco de uma cobertura-(H, K) deve induzir um grafo completo, então cada vértice de $V \setminus \bigcup_{i=1}^{k} V_i$ deve ser adjacente a todos os vértices de algum conjunto V_i, caso contrário $V_1, V_2, \ldots, V_k$ não é extensível.

Observação 2: Não existe extensão se algum dos conjuntos V_i (aumentados) não induz um grafo completo ou se existem vértices não adjacentes $i, j \in V(H)$ tal que algum vértice em V_i tem algum vizinho em V_j (de acordo com a definição de cobertura-(H, K)).

Começamos aumentando cada conjunto V_i adicionando todos os vértices que são forçados a pertencer a V_i' em qualquer extensão. Isto é feito da seguinte maneira.
Seja $v \in V \setminus \bigcup_{i=1}^{k} V_i$. Se existe um único i tal que v é adjacente a todos os vértices de V_i, então em qualquer extensão de $V_1, V_2, ..., V_k$, temos que $v \in V_i'$. Da mesma forma, se existem distintos i, j, l, tais que $\{i, j\}, \{j, l\} \in E(H)$ e v é adjacente a algum vértice de V_i, a algum vértice de V_j e a algum vértice de V_l, então, necessariamente $v \in V_j'$ em qualquer extensão de $V_1, V_2, ..., V_k$. (Isso decorre do fato de que H é livre de triângulos.) Além do mais precisamos testar se v é adjacente a todos os vértices de V_j. Caso isso não ocorra, $V_1, V_2, ..., V_k$ não admite extensão.

Agora, depois de terminada a primeira etapa, se não concluímos que $V_1, V_2, \ldots, V_k$ não admite extensão, significa que estamos numa situação em que cada V_i induz um subgrafo completo de G e onde se algum vértice em V_i é adjacente a algum vértice em V_j, para $1 \leq i \neq j \leq k$, então $\{i, j\} \in E(H)$.

Podemos então passar para a próxima e principal etapa que é reduzir o problema a uma instância de 2-SAT e assim completarmos o teste de extensão.

Seja $v \in V \setminus \bigcup_{i=1}^{k} V_i$. Suponha $t \geq 3$ e que v é adjacente a pelo menos um vértice em cada um dos conjuntos $V_{i_1}, V_{i_2}, \ldots, V_{i_t}$. Como v não foi adicionado a nenhum V_i na etapa anterior, sabemos que os vértices $i_1, i_2, \ldots, i_t$ não induzem uma estrela em H. Logo, para todo r, $1 \leq r \leq t$, existe s,

$1 \leq s \leq t$, $s \neq r$, tal que i_r e i_s não são adjacentes em H. Concluímos então que v não pode pertencer a V'_{i_r} em nenhuma extensão. Como isto vale para todo r, $1 \leq r \leq t$, segue que não existe extensão nesse caso. Assim, se existe extensão, então $t = 2$ e $\{i_1, i_2\} \in E(H)$. Para cada aresta $\{i,j\} \in E(H)$, $i < j$, seja E_{ij} o conjunto de vértices de G que são adjacentes a algum vértice de V_i e a algum vértice de V_j. Logo temos que, se $v \in E_{ij}$ e existe extensão, então ou $v \in V_i$ ou $v \in V_j$.

Suponha que $x \in E_{ij}$, $y \in E_{rs}$ e $\{x,y\} \in E$. Nesse caso temos duas possibilidades a considerar:
(i) x e y pertencem ao mesmo bloco. Isso implica que i,j,r,s não são todos distintos, i.e., ou $i = r$, ou $i = s$, ou $j = r$, ou $j = s$.
(ii) x e y estão em blocos adjacentes. Por exemplo: $x \in V_i$ e $y \in V_r$, então deve existir aresta $\{i,r\}$ em H. Analisando todas as possibilidades então podemos assumir que se i,j,r,s são todos distintos então ou $\{i,r\}$, ou $\{i,s\}$, ou $\{j,r\}$, ou $\{j,s\}$ é uma aresta de H, caso contrário não temos extensão.

Após essas considerações vamos definir uma instância de 2-SAT como se segue.

Seja $X = V \setminus \bigcup_{i=1}^{k} V_i$ o conjunto de variáveis. As cláusulas são descritas abaixo.

A: $\{\overline{x} : x \in E_{ij}, G[V_i \cup \{x\}]$ não é completo$\}$
B: $\{x : x \in E_{ij}, G[V_j \cup \{x\}]$ não é completo$\}$
C: $\{x \vee y, \overline{x} \vee \overline{y} : x, y \in E_{ij}, \{x,y\} \notin E\}$
D: $\{x \vee \overline{y} : x \in E_{ij}, y \in E_{jl}, \{x,y\} \notin E\}$
E: $\{\overline{x} \vee \overline{y} : x \in E_{ij}, y \in E_{il}, \{x,y\} \notin E\}$
F: $\{x \vee y : x \in E_{ij}, y \in E_{lj}, \{x,y\} \notin E\}$
G: $\{x \vee \overline{y}, \overline{x} \vee y : \{x,y\} \in E, x \in E_{ij}, y \in E_{rs}, i,j,r,s$ distintos, $\{i,r\}, \{j,s\} \in E\}$
H: $\{x, y : \{x,y\} \in E, x \in E_{ij}, y \in E_{rs}, i,j,r,s$ distintos, $\{i,r\} \in E, \{j,s\} \notin E\}$
I: $\{\overline{x}, \overline{y} : \{x,y\} \in E, x \in E_{ij}, y \in E_{rs}, i,j,r,s$ distintos, $\{i,r\} \notin E, \{j,s\} \in E\}$
J: $\{x \vee y, \overline{x} \vee \overline{y} : \{x,y\} \in E, x \in E_{ij}, y \in E_{rs}, i,j,r,s$ distintos, $\{i,s\}, \{j,r\} \in E\}$
K: $\{x, \overline{y} : \{x,y\} \in E, x \in E_{ij}, y \in E_{rs}, i,j,r,s$ distintos, $\{i,s\} \in E, \{j,r\} \notin E\}$
L: $\{\overline{x}, y : \{x,y\} \in E, x \in E_{ij}, y \in E_{rs}, i,j,r,s$ distintos, $\{i,s\} \notin E, \{j,r\} \in E\}$.

Falta apenas mostrar que:
$V_1, V_2, \ldots, V_k$ é extensível se e somente se existe uma atribuição de valores de verdade que é satisfatória.

Suponhamos que exista uma atribuição de valores verdade que é satisfatória. Vamos definir os conjuntos $V'_1, V'_2, \ldots, V'_k$ da seguinte maneira: para $i = 1, 2, \ldots, k$ coloque todos os vértices de V_i em V'_i. Para $v \in E_{ij}$, coloque v em V'_i se a variável v é *verdadeira* e em V'_j caso contrário. Vamos mostrar que $V'_1, V'_2, \ldots, V'_k$ é uma cobertura-(H,K) de G.

Como todas as cláusulas dos grupos A e B são satisfeitas , temos que para $i = 1, 2, \ldots, k$, todo vértice em V_i' é adjacente a todos os vértices em V_i, i.e. V_i' é uma clique.

Como todas as cláusulas dos grupos C, D, E e F são satisfeitas , temos que sempre que x e y são não adjacentes, com $x \in E_{ij}$ e $y \in E_{rs}$, onde i, j, r, s não são necessariamente distintos, os vértices x e y pertencem a blocos distintos da cobertura. Podemos então dizer que as cláusulas de A a F sendo satisfeitas garantem que cada bloco da cobertura induz um grafo completo.

Falta mostrar que se x e y são vértices adjacentes em G, pertencendo a blocos diferentes da cobertura, então esses blocos correspondem a vértices adjacentes em H. Suponha que x e y são vértices adjacentes , $x \in E_{ij}$ e $y \in E_{rs}$, onde i, j, r, s são distintos. Como observado anteriormente, podemos assumir, já que chegamos a esse estágio do algoritmo, que existe pelo menos uma das arestas $\{i, r\}$, $\{i, s\}$, $\{j, r\}$, $\{j, s\}$ (arestas de H). Além disso, como H é livre de triângulos , isso nos dá seis casos , cada um dos quais, se ocorrer, é coberto por uma cláusula em algum dos grupos G, H, I, J, K e L. Logo $V_1, V_2, \ldots, V_k$ é extensível.

Suponhamos agora, que $V_1, V_2, \ldots, V_k$ é extensível, e seja $V_1', V_2', \ldots, V_k'$ uma extensão. Como já argumentado, se $v \in E_{ij}$, então em qualquer extensão ou $v \in V_i'$ ou $v \in V_j'$. Faça a variável v *verdadeira* se $v \in V_i'$ e falsa se $v \in V_j'$. É fácil verificar que todas as cláusulas são satisfeitas.

Claramente todas as etapas que levam a redução a 2-SAT podem ser feitas em tempo polinomial, assim como a própria redução. Como 2-SAT é linear no número de cláusulas (veja seção 2.2), o algoritmo também é polinomial. ∎

Para determinar se alguma coleção é extensível, basta testar $O(n^k)$ coleções. Logo, o algoritmo completo de McGillivray e Yu tem complexidade $O(n^{k+2})$.

4.3 Um Algoritmo para Determinar Partições em Clique-Cruz

Uma *partição em clique-cruz*, que vamos abreviar por PCC, de um grafo G é uma partição dos vértices de G em quatro cliques disjuntas A, B, C, D tais que:

- se $x \in A$ e $y \in C$, então $\{x, y\} \notin E$, e
- se $x \in B$ e $y \in D$, então $\{x, y\} \notin E$.

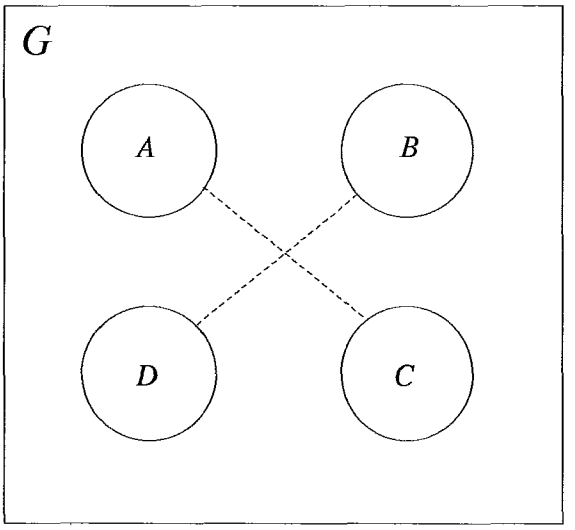


Figura 4.1: *PCC do grafo G. A, B, C e D são cliques.*

Observe que não há restrições sobre as arestas entre vértices de A e B, B e C, C e D e D e A.

Podemos representar um grafo G que possui uma PCC pelo diagrama da figura 4.1. A linha pontilhada entre A e C representa a propriedade de que nenhum vértice de A é adjacente a algum vértice de C. Similarmente para B e D. Os círculos representam cliques.

Nessa seção vamos apresentar um algoritmo linear para determinar se um grafo G admite uma partição em clique-cruz e determinar tal partição, se ela existir, em tempo proporcional ao tamanho do grafo.

O problema da partição em clique-cruz é um caso particular do problema da cobertura-(H, K), quando tomamos H como o ciclo induzido de 4 vértices, como na figura 4.2. O algoritmo de MacGillivray e Yu resolve esse problema em tempo $O(n^6)$.

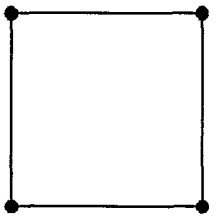


Figura 4.2: *Grafo* $H = C_4$.

É interessante notar que o problema de determinar se um grafo admite uma PCC é equivalente ao problema de determinar se um grafo pode ser

coberto por dois grafos bipartidos completos (*Um grafo G tem uma PCC se e somente se o seu complemento $\overline{G}$ tem uma cobertura por dois grafos bipartidos completos*). O problema de cobrir um grafo por três grafos bipartidos completos é NP-completo [29].

Vamos mostrar como nosso problema pode ser transformado em uma instância de 2-SAT.

4.3.1 Descrição do Algoritmo Partição em Clique-Cruz

Vamos descrever um algoritmo linear, que, dados dois vértices a e b, determina se G tem uma PCC tal que $a \in A$ e $b \in B$. Poderíamos aplicar esse algoritmo a todos os pares de vértices de G a fim de determinar se G tem uma PCC, mas esse procedimento levaria muito tempo. Em vez disso, primeiramente determinamos três vértices u, v, w tais que se G tem uma PCC então ou G tem uma PCC com $u \in A$, $v \in B$, ou G tem uma PCC com $v \in A$ e $w \in B$. Fazemos $a = u$ e $b = v$ (depois se necessário repetimos para $a = v$ e $b = w$.

Em seguida particionamos o conjunto de vértices $V^* = V \setminus \{a, b\}$ de G nos quatro conjuntos abaixo:

$$AB = \{x \in V^* \mid \{x, a\} \in E \text{ e } \{x, b\} \in E\};$$
$$BC = \{x \in V^* \mid \{x, a\} \notin E \text{ e } \{x, b\} \in E\};$$
$$CD = \{x \in V^* \mid \{x, a\} \notin E \text{ e } \{x, b\} \notin E\};$$
$$DA = \{x \in V^* \mid \{x, a\} \in E \text{ e } \{x, b\} \notin E\}.$$

Considere um vértice $x \in AB$. Como x é adjacente a $a \in A$, x não pode estar na clique C e como x é adjacente a $b \in B$, x não pode estar na clique D. Em outras palavras x deve ser designado para o conjunto A ou para o conjunto B, o que indicaremos com a notação $x \rightarrow A$ ou $x \rightarrow B$. Dessa forma, usando uma análise similar para os outros conjuntos, temos:

$$AB \subseteq \{x \in V^* \mid x \rightarrow A \text{ ou } x \rightarrow B\};$$
$$BC \subseteq \{x \in V^* \mid x \rightarrow B \text{ ou } x \rightarrow C\};$$
$$CD \subseteq \{x \in V^* \mid x \rightarrow C \text{ ou } x \rightarrow D\};$$
$$DA \subseteq \{x \in V^* \mid x \rightarrow D \text{ ou } x \rightarrow A\}.$$

É fácil ver que G contém uma PCC com $a \in A$ e $b \in B$ se e somente se as seguintes condições são satisfeitas:

(I) se x e y são vértices não adjacentes e pertencem ao mesmo conjunto AB, BC, CD ou DA, então eles são designados para cliques diferentes.

Por exemplo, se x e y estão em AB e x for designado para A, então y deve ser designado para B.

(II) se x e y são adjacentes e $x \in AB$, $y \in CD$ então se x é designado para A, y é designado para D; e se x é designado para B, y é designado para C. Similarmente para $x \in DA$ e $y \in BC$.

(III) se x e y não são adjacentes e $x \in AB$, $y \in DA$ então se x é designado para A, y é designado para D. Similarmente para $x \in BC$ e $y \in AB$, $x \in CD$ e $y \in BC$, e $x \in DA$ e $y \in CD$.

(IV) se x e y são adjacentes e $x \in AB$, $y \in DA$ então se y é designado para D, x é designado para A. Similarmente para $x \in BC$ e $y \in AB$, $x \in CD$ e $y \in BC$, e $x \in DA$ e $y \in CD$.

Agora para um dado par de vértices a,b construimos uma instância de 2-SAT onde cada vértice de G corresponde a uma variável e cada aplicação de uma das regras I-IV corresponde ou a uma ou a duas cláusulas. Então podemos aplicar o algoritmo 2-satisfabilidade (veja seção 2.2 para resolver esse problema em tempo $O(m+n)$.

Algoritmo PCC
Entrada: um grafo $G = (V, E)$
Saída: *SIM-G tem uma PCC* ou *NÃO-G não tem uma PCC*. Se a resposta for *SIM*, o algoritmo retorna também uma partição de V nas cliques A, B, C e D que constituem a PCC.

Passo 0 Execute o procedimento de inicialização dado na seção 4.3.2. Esse procedimento ou retorna *NÃO-G não tem uma PCC* ou retorna *SIM-G tem uma PCC* e também as quatro cliques A, B, C, D, ou retorna uma tripla ordenada de vértices de G. Nos dois primeiros casos o processo termina. Caso contrário, seja a o primeiro vértice desta tripla, b o segundo, e assuma $a \in A$ e $b \in B$, e vá para o Passo seguinte.

Passo 1 Particione o conjunto $V^* = V \setminus \{a, b\}$ nos quatro conjuntos AB, BC, CD e DA:

$$AB = \{x \in V^* \mid \{x, a\} \in E \text{ e } \{x, b\} \in E\};$$
$$BC = \{x \in V^* \mid \{x, a\} \notin E \text{ e } \{x, b\} \in E\};$$
$$CD = \{x \in V^* \mid \{x, a\} \notin E \text{ e } \{x, b\} \notin E\};$$
$$DA = \{x \in V^* \mid \{x, a\} \in E \text{ e } \{x, b\} \notin E\}.$$

Passo 2 Construa uma instância de 2-SAT como se segue. Para cada vértice x de V^* associamos uma variável $move_x$. A idéia é transformar os conjuntos AB, BC, CD e DA nos conjuntos A, B, C e D, respectivamente, movendo alguns vértices. Considere, por exemplo, um vértice $x \in AB$. O vértice x está no conjunto A ou no conjunto B. Se ele estiver em B então à variável $move_x$ vai ser atribuído o valor *verdadeiro*, caso contrário (i.e. se $x \in A$) vai ser atribuído o valor $falso$. A seguir construimos cláusulas que correspondem às condições (I)-(IV) dadas acima.

(i) para cada $\{x, y\} \notin E$, x e y pertencendo ao mesmo conjunto AB, BC, CD ou DA temos as cláusulas:

$$(move_x \vee move_y) \wedge (\overline{move_x} \vee \overline{move_y})$$

(ii) para cada $\{x, y\} \in E$, $x \in AB$, $y \in CD$ temos as cláusulas:

$$(move_x \vee move_y) \wedge (\overline{move_x} \vee \overline{move_y})$$

Similarmente para $\{x, y\} \in E$, $x \in DA$, $y \in BC$.

(iii) para cada $\{x, y\} \notin E$, $x \in AB$ e $y \in DA$ temos:

$$move_x \vee \overline{move_y}$$

Similarmente para $x \in BC$ e $y \in AB$, $x \in CD$ e $y \in BC$, e $x \in DA$ e $y \in CD$.

(iv) para cada $\{x, y\} \in E$, $x \in AB$ e $y \in DA$ temos:

$$\overline{move_x} \vee move_y$$

Similarmente para $x \in BC$ e $y \in AB$, $x \in CD$ e $y \in BC$, e $x \in DA$ e $y \in CD$.

Seja $\mathcal{C}$ a conjunção dessa cláusulas.

Passo 3 Aplique o algoritmo 2-satisfabilidade a $\mathcal{C}$ (veja seção 2.2).

Passo 4 Se $\mathcal{C}$ é não satisfazível então G não tem uma PCC para esse par escolhido de vértices, logo vá para o Passo 5. Caso contrário faça:

$$A = \{x \in AB \mid move_x = falso\} \cup \{x \in DA \mid move_x = verdadeiro\} \cup \{a\}$$
$$B = \{x \in BC \mid move_x = falso\} \cup \{x \in AB \mid move_x = verdadeiro\} \cup \{b\}$$

$C = \{x \in CD \mid move_x = falso\} \cup \{x \in BC \mid move_x = verdadeiro\}$

$D = \{x \in DA \mid move_x = falso\} \cup \{x \in CD \mid move_x = verdadeiro\}$

Retorne *SIM-G tem uma PCC* e os conjuntos A, B, C, D e pare.

Passo 5 Repita Passos 1 a 4 com a como o segundo vértice da tripla ordenada dada no Passo 0 e b como o terceiro vértice. Se no Passo 4 ficar estabelecido que G não tem uma PCC para esse novo par, retorne *NÃO-G não tem uma PCC* e pare.

4.3.2 O Procedimento de Inicialização

Lembremos que o propósito desse procedimento é achar três vértices de G tais que G tem uma PCC se e somente se ele tem uma PCC com um desses vértices em A e outro em B. No caso em que tais três vértices não são achados, o procedimento determina corretamente se G possui ou não uma PCC.

Começamos contando as arestas de G. Se $m < 4\binom{\lfloor n/4 \rfloor}{2}$ então evidentemente G não pode ser particionado em quatro cliques. Retorne *NÃO-G não tem uma PCC* e pare. Caso contrário, nós achamos as componentes conexas de G. Note que se G tem uma PCC então cada componente conexa contém pelo menos uma das quatro cliques. Se G tem mais do que quatro componentes então retorne *NÃO-G não tem uma PCC* e pare. Caso contrário, procuramos por um P_3: um caminho induzido com três vértices x, y, z e duas arestas $\{x, y\}$ e $\{y, z\}$. Veja figura 4.3. Observe que se G tem um P_3 então ou x, y e z estão todos em cliques diferentes ou x e y estão na mesma clique e z está em uma clique diferente. De qualquer forma, podemos assumir, sem perda de generalidade que ou $x \in A$ e $y \in B$ ou $y \in A$ e $z \in B$. Se existir algum P_3 então retorne os vértices desse P_3. Caso contrário, sejam G_1, G_2, G_3, G_4 as componentes conexas de G, onde G_2, G_3 e G_4 são possivelmente vazias, retorne *SIM-G tem uma PCC com* $A = V(G_1)$, $B = V(G_2)$, $C = V(G_3)$, $D = V(G_4)$ e pare.

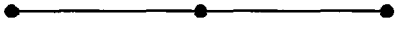


Figura 4.3: P_3

4.3.3 Complexidade

Estamos agora prontos para estimar a complexidade computacional do **Algoritmo PCC**.

Teorema 8 *O Algoritmo PCC determina se um grafo tem uma PCC em tempo $O(m+n)$.*

Prova. Está claro que G possui uma PCC se e somente se existir uma atribuição de valores verdade que é satisfatória para $\mathcal{C}$. Podemos estimar o tempo de execução do **Algoritmo PCC** da seguinte maneira. O procedimento de inicialização, Passo 0, pode ser feito em tempo $O(m+n)$. De fato, pode-se achar as componentes conexas de G em tempo linear no tamanho do grafo [37]. Determinar se G tem um P_3 pode ser feito em tempo linear no tamanho do grafo usando uma busca em largura. No Passo 1 a construção de cada um dos quatro conjuntos pode ser executada em tempo proporcional a n. No Passo 2 a construção de uma instância de 2-SAT leva tempo $O(n^2)$. O Passo 3, o algoritmo 2-satisfabilidade, pode ser executado em tempo linear no número de cláusulas [1], que nesse caso é $O(n^2)$. Claramente o Passo 4 pode ser feito em tempo linear. Então o tempo de execução dos passos 1 a 4 é $O(n^2)$. De acordo com o Passo 5, vamos aplicar Passos 1 a 4 no máximo duas vezes. Logo o tempo de execução do algoritmo é $O(n^2)$. Convém notar que no começo do procedimento de inicialização nós contamos as arestas de G e continuamos apenas no caso de m ser maior ou igual a $4\binom{\lfloor n/4 \rfloor}{2}$. Logo os passos 1 a 4 são executados somente se $m = O(n^2)$. ∎

4.3.4 Um Exemplo do Algoritmo PCC

Consideremos o grafo G dado na figura 4.4. Vamos apresentar a seguir um exemplo da execução do **Algoritmo PCC** aplicado a esse grafo.

Entrada: grafo $G = (V, E)$ da figura 4.4.

Passo 0: Suponha que tenha sido retornado a tripla ordenada (a, b, c) que induz um P_3 em G. Faça $a{:=}\ a$, $b{:=}\ b$.

Iteração 1:

Passo 1: $V^* = V \setminus \{a, b\}$ é particionado nos conjuntos:

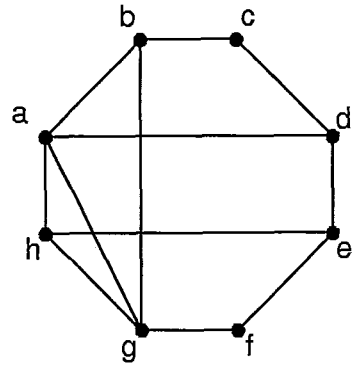


Figura 4.4: *Grafo G.*

$AB = \{g\}$;
$BC = \{c\}$;
$CD = \{e, f\}$;
$DA = \{d, h\}$.

Passo 2: A cada vértice v de V^* é associada uma variável m_v. Seja $\mathcal{C}$ o conjunto de claúsulas obtidas das condições (I) a (IV) aplicadas a V^*.

Condição (I): $(m_d \vee m_h) \wedge (\overline{m_d} \vee \overline{m_h})$.

Condição (II): $(m_g \vee m_f) \wedge (\overline{m_g} \vee \overline{m_f})$, $(m_c \vee m_d) \wedge (\overline{m_c} \vee \overline{m_d})$.

Condição (III): $m_g \vee \overline{m_d}$, $m_c \vee \overline{m_g}$, $m_e \vee \overline{m_c}$, $m_f \vee \overline{m_c}$, $m_d \vee \overline{m_f}$, $m_h \vee \overline{m_f}$.

Condição (IV): $\overline{m_g} \vee m_h$, $\overline{m_d} \vee m_e$, $\overline{m_h} \vee m_e$.

Passo 3: Aplicação do algoritmo 2-satisfabilidade [1]. Na figura 4.5 exibimos o grafo direcionado $\vec{G}_{\mathcal{C}}$ que é construído durante a execução desse algoritmo. A figura 4.6 nos mostra as componentes fortemente conexas obtidas em ordem topológica reversa. Observe que a componente C_2 contém variáveis duais. O algoritmo retorna: *$\mathcal{C}$ é não satisfazível.*

Passo 5: Faça a:= b, b:= c.

Iteração 2:

Passo 1: $V^* = V \setminus \{a, b\}$ é particionado nos conjuntos:
$AB = \emptyset$;
$BC = \{d\}$;
$CD = \{e, f, h\}$; $DA = \{a, g\}$.

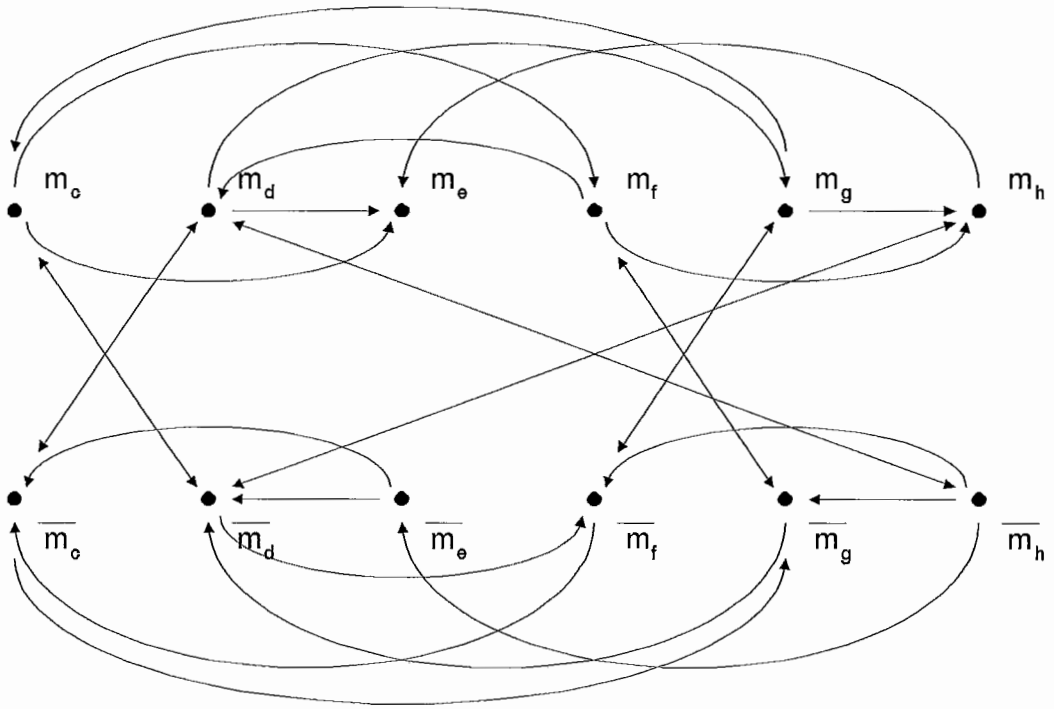


Figura 4.5: *Grafo* $\vec{G}_C$.

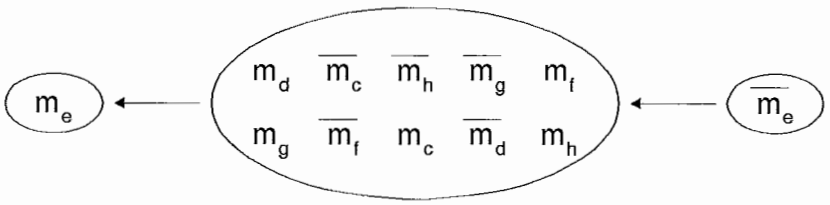


Figura 4.6: *As componentes fortemente conexas de* $\vec{G}_C$ *em ordem topológica reversa.*

Passo 2: a cada vértice v de V^* é associada uma variável m_v. Seja C o conjunto de claúsulas obtidas das condic,ões (I)-(IV) aplicadas a V^*.

Condição (I): $(m_f \vee m_h) \wedge (\overline{m_f} \vee \overline{m_h})$.

Condição (II): $(m_d \vee m_a) \wedge (\overline{m_d} \vee \overline{m_a})$.

Condição (III): $m_f \vee \overline{m_d}$, $m_h \vee \overline{m_d}$, $m_a \vee \overline{m_e}$, $m_a \vee \overline{m_f}$, $m_g \vee \overline{m_e}$.

Condição (IV): $\overline{m_e} \vee m_d$, $\overline{m_a} \vee m_h$, $\overline{m_g} \vee m_f$, $\overline{m_g} \vee m_h$.

Passo 3: Aplicação do algoritmo 2-satisfabilidade. Na figura 4.7 exibimos o grafo direcionado $\vec{G}_C$ que é construído durante a execução desse algoritmo. A figura 4.8 nos mostra as componentes fortemente conexas obtidas em ordem topológica reversa.

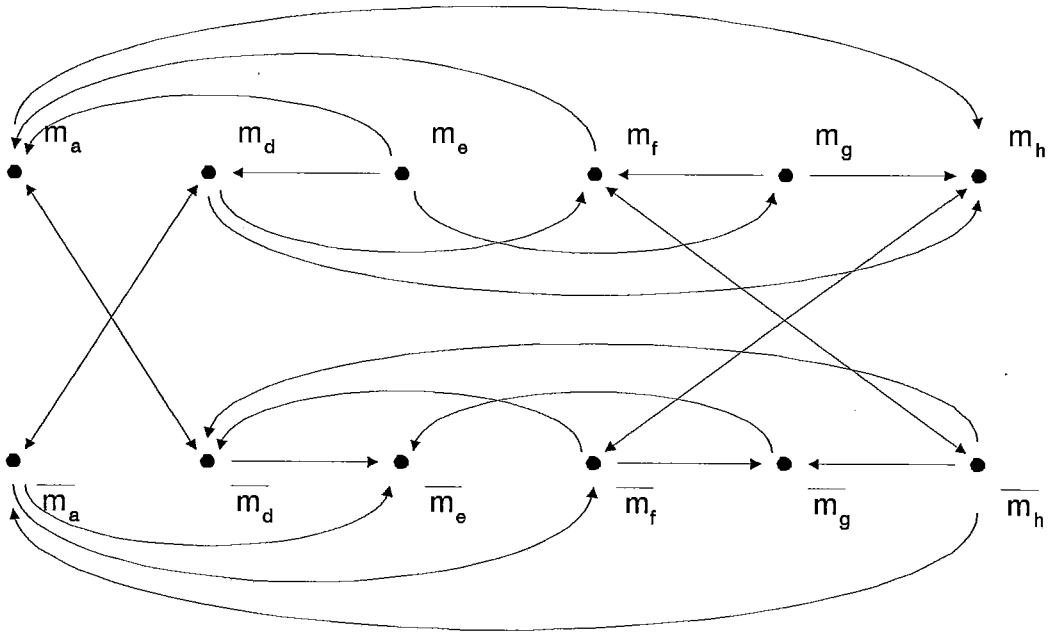


Figura 4.7: *Grafo* $\vec{G}_C$.

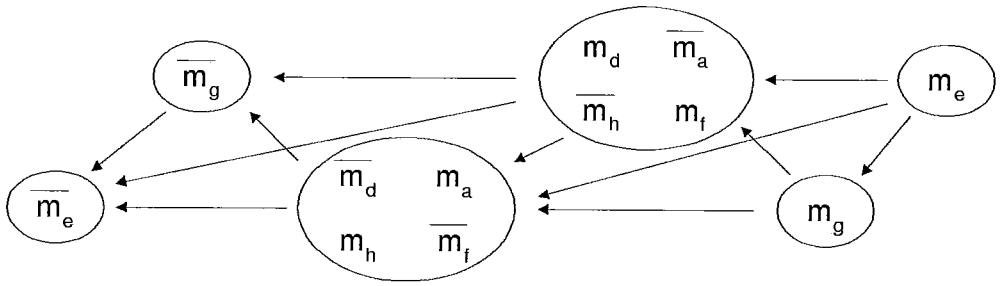


Figura 4.8: *As componentes fortemente conexas de* $\vec{G}_C$ *em ordem topológica reversa.*

Passo 4: retorna: *Sim-G tem uma PCC*:

$A = \{a, b\}$;
$B = \{c, d\}$;
$C = \{e, f\}$;
$D = \{g, h\}$.

Capítulo 5

Decomposição por Cortes

Neste capítulo consideramos alguns problemas de decisão referentes a problemas de reconhecimento de grafos com determinados cortes, tais como corte clique, corte estrela, corte estável, corte k-multipartido completo e outros. Definimos os problemas e exibimos para cada um dos casos de corte estável e corte k-multipartido completo uma transformação polinomial, que estabelece que os problemas são NP-completos. Como conseqüencia obtemos que os problemas de reconhecimento de grafos com corte multipartido completo, assim como o de grafos com corte bipartido completo são NP-completos. Observamos que esses quatro últimos problemas citados estavam em aberto.

5.1 Introdução

Dado um grafo G é uma questão natural perguntar se G contém algum grafo específico H, tal que o conjunto de vértices de H seja um corte de G. Lembramos que um *corte* de um grafo é um subconjunto de vértices desse grafo tal que o grafo obtido após a sua remoção não é conexo.

O interesse por certos tipos de cortes adquire relevância na classe dos grafos perfeitos, por induzirem decomposições, que muitas vezes preservam as características estruturais dessa classe.

Têm sido estudados com atenção tipos especiais de cortes e resultados bastante interessantes têm surgido relacionando grafos minimalmente imperfeitos a esses cortes. Como exemplos podemos citar os cortes cliques e os cortes estrela.

Um *corte clique* em um grafo G é um corte em G que é uma clique. Um resultado bastante conhecido é que um grafo minimalmente imperfeito não pode conter um corte clique. Ou em outras palavras, a identificação de dois grafos perfeitos através de uma clique fornece um grafo perfeito.

Um *corte estrela* em um grafo G é um corte em G que tem um vértice que é adjacente a todos os outros vértices desse corte. Esse tipo de corte foi definido por Chvátal que também mostrou que nenhum grafo minimalmente imperfeito contém um corte estrela (Lema do corte estrela) [5], generalizando o resultado citado anteriormente sobre corte clique. A complexidade do problema do reconhecimento de um grafo com corte clique foi determinada por Whitesides [40] e Tarjan [38], que exibiram algoritmos polinomiais para determinar um corte clique. Já o problema do reconhecimento de um grafo com corte estrela foi resolvido pelo próprio Chvátal [5], que caracterizou grafos com corte estrela e apresentou um algoritmo polinomial decorrente dessa caracterização.

Continuando nessa linha Chvátal definiu um novo tipo de corte que chamou de *corte assimétrico* (skew cutset) [5], que por sua vez generaliza a noção de corte estrela.

Um corte é *assimétrico* quando pode ser particionado em dois conjuntos V_1 e V_2, não vazios, tais que $(x, y) \in E$ se $x \in V_1$ e $y \in V_2$. Chvátal propôs a seguinte conjectura:

Conjectura 5 *Nenhum grafo minimalmente imperfeito contém um corte assimétrico.*

Essa conjectura assume uma importância especial por ser uma propriedade que separa a classe dos grafos particionáveis dos grafos minimalmente imperfeitos (essas duas classes partilham a maior parte das propriedades). Mas essa conjectura tem se mostrado mais difícil do que se esperava. Procura-se então prová-la para casos de cortes mais simples.

O corte multipartido completo é um caso particular do corte assimétrico. Um grafo G é dito *multipartido completo* quando seu conjunto de vértices pode ser particionado em k conjuntos estáveis, de maneira que os vértices de cada uma das partições sejam adjacentes a todos os outros vértices das outras partições. Em particular, quando $k = 2$ temos um grafo *bipartido completo*. Vamos dizer que um corte é *multipartido completo* quando o grafo induzido por ele é um grafo multipartido completo. Analogamente definimos corte bipartido completo.

Recentemente, Cornuejols e Reed [10] mostraram que nenhum grafo minimalmente imperfeito contém um corte multipartido completo.

E quanto ao problema do reconhecimento de um grafo com corte assimétrico? Este problema pode ser esquematizado como segue:

Problema CORTE ASSIMÉTRICO

Instância: Grafo $G = (V, E)$.
Questão: O grafo G admite um corte assimétrico?

Esse também parece ser um problema difícil.

Nesse trabalho, mostramos que o problema de reconhecer um grafo com corte k-multipartido completo é NP-completo. A demonstração desse fato é feita usando o problema CORTE ESTÁVEL, que mencionaremos a seguir. Como corolário temos que o problema de reconhecer um grafo com corte multipartido completo é também NP-completo

5.2 Corte Estável

Um corte é dito *estável* quando o seu conjunto de vértices é um conjunto estável. Veja na figura 5.1 um exemplo de um grafo com corte estável. Tucker [39] mostrou que os únicos grafos minimalmente imperfeitos que contêm cortes estáveis são os ciclos induzidos ímpares de tamanho maior ou igual a cinco. Em outras palavras, um grafo minimalmente imperfeito não admite corte estável, a menos que seja um ciclo induzido ímpar de tamanho maior ou igual a cinco.

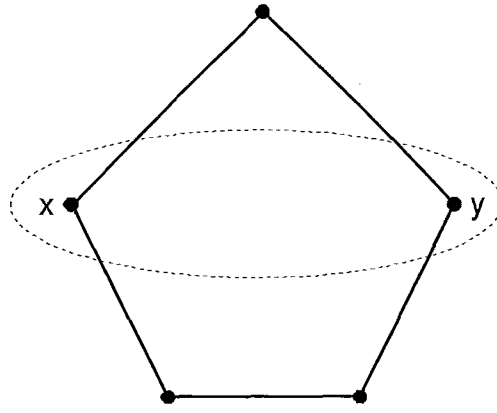


Figura 5.1: *Um grafo com um corte estável* $\{x, y\}$.

E quanto ao problema ao problema do reconhecimento de grafos com cortes estáveis? Este problema pode ser esquematizado como se segue:

Problema CORTE ESTÁVEL
Instância: Grafo $G = (V, E)$.
Questão: O grafo G admite um corte estável?

Corneil e Fonlupt [9] em um artigo recente, ao analisarem operações de composição usando cortes estáveis, colocam essa questão como em aberto tanto para um grafo qualquer, como para um grafo perfeito.

Ao pensar nesse problema é natural pensar no problema "quase" equivalente de achar um corte emparelhamento em um grafo de linha $L(G)$.

Um *corte emparelhamento* em um grafo G é um conjunto F de arestas de G, disjuntas duas a duas, tal que o grafo obtido após a remoção de F não é conexo.

Problema CORTE EMPARELHAMENTO
Instância: Grafo $G = (V, E)$.
Questão: O grafo G admite um corte emparelhamento?

Em 1970 R.L. Graham [21] lançou a seguinte pergunta: o problema do reconhecimento de grafos com cortes emparelhamento é NP-completo? Essa pergunta foi respondida apenas em 1984, por Chvátal [6] que mostrou que reconhecer grafos com cortes emparelhamento para grafos com $\Delta \geq 4$ é NP-completo. Para grafos com $\Delta < 3$ é dado um algoritmo eficiente para reconhecimento de cortes emparelhamento.

Observamos que se G tem vértices com grau 1 então G tem um corte emparelhamento que chamaremos de *trivial*, fácil de achar.

O resultado a seguir é imediato:

Lema 15 *G tem um corte emparelhamento não trivial se e somente se $L(G)$ tem um corte estável.*

E agora, como consequência direta do resultado devido a Chvátal, acima citado, e do Lema 15 segue que:

Lema 16 *O problema* CORTE ESTÁVEL *é NP-completo.*

Prova. A transformação usada é a partir do problema CORTE EMPARELHAMENTO.

Seja G uma instância para o problema CORTE EMPARELHAMENTO, G sem vértices de grau 1. Seja $G' = L(G)$ uma instância para o problema CORTE ESTÁVEL. Essa construção pode ser feita claramente em tempo polinomial.

Basta então mostrar que:

- G tem corte emparelhamento não trivial se e somente se G' tem corte estável.

Mas esse resultado decorre do lema 15. ∎

5.3 Corte Multipartido Completo

Consideremos agora o problema do reconhecimento de grafos com cortes multipartidos completos, que enunciamos a seguir.

Problema CORTE MULTIPARTIDO COMPLETO
Instância: Grafo $G = (V, E)$, k.
Questão: O grafo G admite um corte k-multipartido completo?

Vamos mostrar que o problema CORTE MULTIPARTIDO COMPLETO é NP-completo. Para isso vamos usar um resultado mais forte, i.e., vamos mostrar que o problema é NP-completo mesmo com k fixo, $k \geq 2$.

Consideremos então o seguinte problema:

Problema CORTE k-MULTIPARTIDO COMPLETO, $k \geq 2$
Instância: Grafo $G = (V, E)$.
Questão: O grafo G admite um corte k-multipartido completo?

Usando o resultado da seção anterior provaremos o seguinte resultado:

Teorema 9 *O problema* CORTE k-MULTIPARTIDO COMPLETO *é NP-completo.*

Prova: A transformação usada é a partir do problema CORTE ESTÁVEL.

Seja G uma instância para o problema CORTE ESTÁVEL. Construimos uma instância G' para o problema k-MULTIPARTIDO COMPLETO ao adicionarmos a G os vértices $v_1, v_2, \ldots, v_{k-1}$, tal que cada v_i, $1 \leq i \leq k-1$ é adjacente a todos os vértices de G, e $\{v_i, v_j\} \in E(G')$ para $i \neq j$ (observe que esses $k-1$ vértices constituem uma clique). Essa construção pode ser feita, claramente, em tempo polinomial.

Resta mostrar que :

- G tem um corte estável se e somente se G' tem um corte k-multipartido completo.

Seja S um corte estável em G. Em G', por construção, os vértices $v_1, v_2, \ldots, v_{k-1}$ são adjacentes a todos os vértices de G, em particular a todos os vértices de S. Logo $S \cup \{v_1, v_2, \ldots, v_{k-1}\}$ é um corte de G'. Mas $S \cup \{v_1, v_2, \ldots, v_{k-1}\}$ é um corte k-multipartido completo.

Seja S' um corte k-multipartido completo em G' . S' necessariamente contém os vértices $v_1, v_2, \ldots, v_{k-1}$. Então $S' \setminus \{v_1, v_2, \ldots, v_{k-1}\}$ é um corte estável em G. ∎

Temos então como conseqüencia:

Corolário 1 *O problema* CORTE MULTIPARTIDO COMPLETO *é NP-completo.*

Observamos que o resultado do teorema 9 na verdade mostra uma família de problemas NP-completos indexada em k ($k \geq 2$). Em particular, o primeiro elemento desta família é o problema:

Problema CORTE BIPARTIDO COMPLETO
Instância: Grafo $G = (V, E)$.
Questão: O grafo G admite um corte bipartido completo?

Por outro lado, a construção do grafo G' na prova do teorema 9 mostra uma outra família de problemas NP-completos também indexada em k ($k \geq 1$), definida a seguir.

Problema CORTE k-MULTIPARTIDO COMPLETO ESPECIAL
Instância: Grafo $G = (V, E)$.
Questão: O grafo G admite um corte multipartido completo C onde C tem $k \geq 2$ conjuntos independentes, com $k - 1$ conjuntos contendo um único vértice cada, e um conjunto independente com p vértices ($p \geq 1$)?

Em particular, o primeiro elemento desta família é um tipo especial de corte estrela :

Problema CORTE ESTRELA ESPECIAL
Instância: Grafo $G = (V, E)$.
Questão: O grafo G admite um corte estrela C onde C induz um grafo bipartido completo (denotado por $K_{1,p}$), sendo uma das partes composta de um único vértice e a outra com p vértices ($p > 1$)?

O resultado de que o problema CORTE ESTRELA ESPECIAL é NP-completo adquire um interesse maior quando confrontado com o fato de que o reconhecimento de grafos com corte estrela é polinomial.

Capítulo 6

Conclusões

O encerramento de uma tese não é necessariamente um ponto final nesse trabalho. Discutimos nesse capítulo vários problemas que se colocam a partir dos nossos estudos e resultados aqui obtidos.

Os algoritmos que desenvolvemos para determinar decomposições homogêneas: **Algoritmo Conjunto Homogêneo** e **Algoritmo Par Homogêneo**, assim como o algoritmo para determinar partições em clique-cruz, **Algoritmo PCC**, usam basicamente a mesma técnica. Escolhemos alguns vértices (aleatoriamente ou não, dependendo do caso) e particionamos então o restante do conjunto de vértices do grafo em subconjuntos de acordo com certas relações desses vértices restantes com os escolhidos. A partir daí transformamos o problema em uma instância de 2-SAT (ou quase) considerando todas as cláusulas possíveis induzidas pelas relações de adjacências entre os vértices desses conjuntos. Embora essa técnica tenha funcionado tão bem nos nossos casos, ela não parece ser aplicável a qualquer problema. Tentamos por exemplo, aplicá-la para achar uma decomposição em um grafo induzida por um corte clique. Esse problema é sabidamente polinomial [40, 38]. No entanto só conseguimos transformá-lo em uma instância de 3-SAT, o que claramente nos indica que para esse caso em particular, essa técnica não é apropriada, ou que talvez sejam necessárias certas modificações que não tornam seu uso aconselhável.

Mas, apesar disso, acreditamos que essa técnica pode ser usada com sucesso em outros problemas. Por exemplo, no reconhecimento de grafos split, que são grafos que podem ser decompostos em uma clique e um conjunto estável, sabemos que ela funciona, embora não a tenhamos incluído no nosso trabalho. A operação amálgama , que induz uma decomposição em um grafo,

nos parece ser um caso promissor para a aplicação dessa técnica. Faz parte de nossos futuros planos de trabalho.

Sobre os pares homogêneos colocamos as seguintes questões:

- É possível modificar o **Algoritmo Par Homogêneo** de modo que ele se torne mais rápido? Mais precisamente, será que podemos conseguir um procedimento de inicialização que escolha triplas de vértices apropriadas de forma mais eficiente?

- É possível obter um algoritmo de reconhecimento da classe dos grafos Berge sem touros, usando o **Algoritmo Par Homogêneo**, que seja mais eficiente que o algoritmo existente, desenvolvido por Reed e Sbihi [34]?

- É possível desenvolver algoritmos eficientes para os problemas de otimização combinatória: clique máxima e número cromático para a classe dos grafos Berge sem touros? Ou ainda, será que podemos construir em tempo polinomial uma árvore de decomposição usando pares homogêneos que permita resolver esses problemas de otimização?

Sobre decomposições em cliques, nós mostramos que para o caso particular, que chamamos de Partição em Clique-Cruz, de uma cobertura (H, K), onde H é um grafo sem triângulos, o problema pode ser resolvido em tempo linear. Achamos porém, que o problema generalizado das coberturas por clique, i.e, o problema de determinar se um grafo G admite uma cobertura-(H, K), onde H é um grafo sem triângulos também pode ser resolvido em tempo linear e pretendemos trabalhar nessa direção.

E quanto aos problemas considerados no capítulo 5, gostaríamos de ressaltar que o problema do reconhecimento de grafos com corte assimétrico que está em aberto, adquire relevância especialmente quando considerado juntamente com a conjectura 4 que afirma que nenhum grafo minimalmente imperfeito contém um corte assimétrico. Se essa conjectura for verdadeira, a não existência de corte assimétrico passa a ser uma propriedade que *separa*. Achamos que possivelmente o problema CORTE ASSIMÉTRICO é NP-completo.

Referências Bibliográficas

[1] B.ASPVALL, F.PLASS e R.E.TARJAN, A Linear-Time Algorithm for Testing The Truth of Certain Quantified Boolean Formulas, *Information Processing Letters*, **8(3)** (1979), pp.121-123.

[2] C. BERGE, F'arbung von Gr'aphen deren Samtliche bzw. deren ungerade Kreise starr sind., *wiss. Zeitschr. Martin Luther-Univ., Halle-Wittenberg* **10** (1961), pp.114-115.

[3] R. BLAND, H. HUANG e L. TROTTER, Graphical Properties Related to Minimal Imperfection, *Discrete Mathematics* **27** (1979), pp.11-22.

[4] K.G. CAMERON, "Polyhedral and Algorithmic Ramifications of Antichains", Ph.D. Thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, 1982.

[5] V. CHVÁTAL, Star-Cutsets and Perfect Graphs, *Journal of Combinatorial Theory, Series B* **39** (1985), pp.189-199.

[6] V. CHVÁTAL, Recognizing Decomposable Graphs, *Journal of Graph Theory* **8** (1984), pp.51-53.

[7] V.CHVÁTAL e N.SBIHI, Bull-Free Berge Graphs are Perfect, *Graphs and Combinatorics*, **3** (1987), pp.127-139.

[8] D.G.CORNEIL, Y. PERL e L. STEWART, Cographs: recognition, application and algorithms, *Congre. Num.* **43** (1984), pp. 249-258.

[9] D.G. CORNEIL and J. FONLUPT, Stable Set Bonding in Perfect Graphs and Parity Graphs, *Journal of Combinatorial Theory, Series B* **59** (1992), pp.1-14.

[10] G. CORNUEJOLS and B. REED, Complete Multipartite Cutsets in Minimal Imperfect Graphs, to appear in *Journal of Combinatorial Theory, Series B.*

[11] A. COURNIER, "Sur quelques Algorithmes de Décomposition de Graphes", Thése, Université Montpellier II, 1993

[12] S. EVEN, A. ITAI e A. SHAMIR, On the Complexity of Timetable and Multi-commodity Flow Problems, *SIAM J.Comput.* **5(4)** (1976), pp.691-703.

[13] H. EVERETT, S. KLEIN, B. REED, An Algorithm for Finding Clique-Cross Partitions,

Rapport de Recherche 208, Départment de mathématiques et d'informatique, Université du Québec à Montréal, Montréal, (1993);

VII Congreso Latino-Ibero Americano de Investigation de Operaciones e Ingenieria de Sistemas, Santiago, Chile, 4-8 de julho de 1994.

[14] H. EVERETT, S. KLEIN, B. REED, An Algorithm for Finding Homogeneous Pairs,

Rapport de Recherche 225, Départment de mathématiques et d'informatique, Université du Québec à Montréal, Montréal, (1994);

Seventh SIAM Conference on Discrete Mathematics, Albuquerque, New Mexico, 22-25 de junho de 1994;

submetido a *Discrete Applied Mathematics.*

[15] C.M.H. DE FIGUEIREDO, "Um Estudo de Problemas Combinatórios em Grafos Perfeitos", Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, (1991).

[16] D. FULKERSON, Anti-Blocking Pairs of Polyhedra, *Mathematical Programming*, **Vol. 1** (1971), pp.168-194.

[17] M.R. GAREY and D.S. JOHNSON, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman, San Francisco, 1979.

[18] F. GAVRIL, Algorithms on Clique Separable Graphs, *Ann. Discrete Math.*, **19** (1977), pp.159-165.

[19] M.C. GOLUMBIC, "Algorithmic Graph Theory and Perfect Graphs", Academic press, New York, 1980.

[20] M.C. GOLUMBIC, Comparability Graphs and a New Matroid, *Journal of Combinatorial Theory, Series B*, **22** (1977), pp.68-90.

[21] R.L. GRAHAM, On Primitive Graphs and Optimal Vertex Assignments, *Ann. N.Y. acad. Sci.* **175** (1970), pp.170-186.

[22] M. GRÖTSCHEL, L. LOVÁSZ, A. SCHRIJVER, The Ellipsoid Method and its Consequences in Combinatorial Optimization, *Combinatorica* **1** (1981), pp.169-197.

[23] P. HELL, J. NEŠETŘIL, On the Complexity of H-Colouring, *Journal of Combinatorial Theory, Series B* **48** (1990), pp.92-110.

[24] R.M. KARP, On The Complexity of Combinatorial Problems, *Networks* **5**, (1975), pp.691-703.

[25] K. KILAKOS e S. KLEIN, Sobre a Complexidade do Problema Corte Multipartido Completo, aceito para aparecer em *Pesquisa Operacional.*

[26] L. LOVÁSZ, Normal Hypergraphs and The Perfect Graph Conjecture, *Discrete Mathematics* **2** (1972), pp. 253-267.

[27] L. LOVÁSZ, A Characterization of Perfect Graphs, *Journal of Combinatorial Theory, Series B* **13** (1972), pp.95-98.

[28] L. LOVÁSZ, Perfect Graphs, "Selected Topics in Graph Theory 2", Academic Press, New York, (1983), pp.55-87.

[29] G.MACGILLIVRAY e M.L.YU, Generalized covers of graphs, manuscript, Canada, (1993).

[30] J.H. MULLER e J. SPINRAD, Incremental modular decomposition, *Journal of the Association for Computing Machinery* **1** (1989), pp.1-19.

[31] S. OLARIU, No Anti-Twins in Minimal Imperfect Graphs, *Journal of Combinatorial Theory, Series B* **45** (1988), pp.255-257.

[32] M. PADBERG, Perfect Zero-One Matrices, *Mathematical Programming* **6** (1974), pp.180-196.

[33] B. REED, "A Semi-Strong Perfect Graph Theorem", Ph.D. Thesis, School of Computer Science, McGill University, Montreal, (1986).

[34] B. REED e N. SBIHI, Recognizing Bull-Free Perfect Graphs, preprint, (1990).

[35] J. SPINRAD, P_4-trees and Substitution Decomposition, *Discrete Applied Mathematics* **39** (1992), pp.263-291.

[36] J.L. SZWARCFITER, "Grafos e Algoritmos Computacionais", Editora Campus, Rio de Janeiro, 1983.

[37] R.E. TARJAN, Depth First Search and Linear Graph Algorithms, *SIAM J.Comput.* **1(2)** (1972), pp.146-160.

[38] R.E. TARJAN, Decomposition by Clique Separators, *Discrete Mathematics* **55** (1985), pp.221-232.

[39] A. TUCKER, Colouring Graphs with Stable Sets, *Journal of Combinatorial Theory, Series B* **34** (1983), pp. 258-267.

[40] S. WHITESIDES, An Algorithm for Finding Clique Cutsets, *Information Processing Letters* **12** (1981), pp.31-32.

[41] S. WHITESIDES, A Method for Solving Certain Graph Recognition and Optimization Problems, with Applications to Perfect Graphs, *Ann. Discrete Math.*, **21** (1984), pp.281-297.