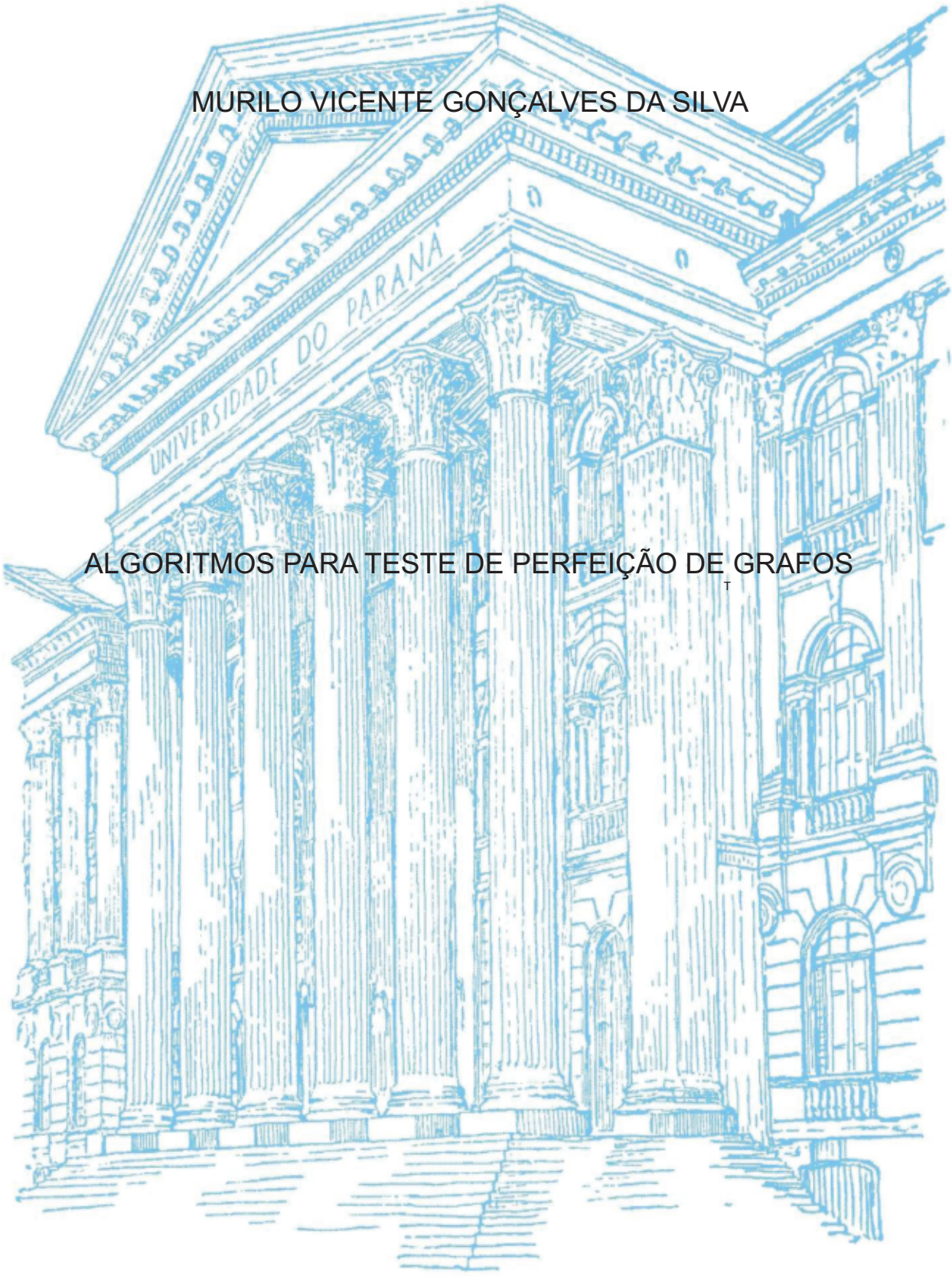
UNIVERSIDADE FEDERAL DO PARANÁ

MURILO VICENTE GONÇALVES DA SILVA



ALGORITMOS PARA TESTE DE PERFEIÇÃO DE GRAFOS

CURITIBA
2004

MURILO VICENTE GONÇALVES DA SILVA

ALGORITMOS PARA TESTE DE PERFEIÇÃO DE GRAFOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. André Luiz Pires Guedes

CURITIBA

2004

SI586a Silva, Murilo Vicente Gonçalves da
Algoritmos para teste de perfeição de grafos / Murilo Vicente Gonçalves da Silva. – Curitiba, 2004.
75 f. : il. color. ; 30 cm.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2004.

Orientador: André Luiz Pires Guedes.

1. Teoria dos grafos. 2. Algoritmos. 3. Matemática. I. Universidade Federal do Paraná. II. Guedes, André Luiz Pires. III. Título.

CDD: 511.5

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno *Murilo Vicente Gonçalves da Silva*, avaliamos o trabalho intitulado, "*Algoritmos para Teste de Perfeição de Grafos*", cuja defesa foi realizada no dia 26 de agosto de 2004, às quatorze horas, no Auditório do Departamento de Informática da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 26 de agosto de 2004.

Prof. Dr. André Luiz Pires Guedes
DINF/UFPR – Orientador

Prof. Dr. Luerbio Faria
FFP/UERJ – Membro Externo

Prof. Dr. Jair Donadelli Jr.
DINF/UFPR – Membro Interno

Resumo

Esta dissertação apresenta e discute os dois recentemente descobertos algoritmos de teste de perfeição de grafos. A parte central dos dois algoritmos é a mesma. Este núcleo que os dois algoritmos compartilham, que certamente é a parte mais complexa dos mesmos, foi discutido detalhadamente e implementado. Até o momento, o autor desta dissertação não tem notícias de outras implementações destes algoritmos.

A apresentação do algoritmo foi dividida em três partes distintas. A primeira parte agrupa vários algoritmos que testam pela presença de subgrafos específicos. A segunda parte estuda em detalhes o núcleo que os dois algoritmos compartilham. A terceira parte apresenta os dois algoritmos de teste de perfeição de grafos propriamente ditos.

Adicionalmente, nesta dissertação foram definidos quatro parâmetros que podem ser associados a um grafo para exprimir seu grau de imperfeição. Estes parâmetros foram denotados ρ_1, ρ_2, ρ_3 e ρ_4. O autor relacionou estes parâmetros com algumas operações que podem ser aplicadas a um grafo imperfeito para torná-lo perfeito. As operações utilizadas para definir estes parâmetros de foram a remoção de arestas do grafo (ρ_1), a insersão de arestas no grafo (ρ_2), a execução de remoção e inserção de arestas no grafo (ρ_3) e, finalmente, a remoção de vértices do grafo (ρ_4).

Mostrou-se que para qualquer grafo temos $\rho_4 \leq \rho_3 \leq \rho_1$ e $\rho_4 \leq \rho_3 \leq \rho_2$. Além disso foram apresentados exemplos de grafos em que cada uma destas desigualdades pode ser estrita. O autor apresentou também alguns limitantes inferiores e superiores para estes parâmetros. Finalmente, utilizando um dos limitantes inferiores para ρ_4, mostrou-se que existem grafos que são "bastante imperfeitos". Mais especificamente, foi demonstrado que existem grafos com n vértices para os quais o número de vértices que deve ser removido para tornás-lo perfeitos é pelo menos $\dfrac{n}{\lg(2n)} - \lg(2n)$.

Palavras-chave: teoria dos grafos, algoritmos, teoria algorítmica dos grafos, grafos perfeitos, otimização combinatória.

Abstract

This dissertation presents and discusses two recently discovered algorithms that test if a graph is perfect. The core shared by the two algorithms is discussed in details and the results of its implementation are presented. It is worthwhile to mention that no other similar implementation is known so far.

The presentation of the algorithms is divided into three parts. The first part presents several algorithms that test some particular subgraphs. The second part reviews the core of the algorithms and the third part presents the two algorithms for perfectness.

Additionally, in this work it is defined four parameters that can measure how imperfect a graph is. These parameters are denoted ρ_1, ρ_2, ρ_3 and ρ_4. The defined parameters are related to some operations that can be applied to a graph to make it perfect. The following operations are considered: edge deletion (ρ_1), edge insertion (ρ_2), both deletion and insertion of edges (ρ_3) and, finally, vertex deletion (ρ_4). It is shown that for any graph it holds that $\rho_4 \leq \rho_3 \leq \rho_1$ and $\rho_4 \leq \rho_3 \leq \rho_2$. It is also shown examples of graphs where such inequalities are strict. Finally, some lower bounds and upper bounds for these paramenters are shown. As a consequence of a lower bound for ρ_4, the author shows that there are "highly" imperfect graphs. More precisely, there are graphs with n vertices where $\rho_4 \geq \dfrac{n}{\lg(2n)} - \lg(2n)$.

Keywords: graph theory, algorithms, algorithmic graph theory, perfect graphs, combinatorial optimization.

Conteúdo

Lista de Figuras

Introdução

A teoria de grafos, um dos maiores ramos da matemática combinatória e uma das áreas teóricas mais importantes em ciência da computação, apresenta-se como um dos mais estimulantes e desafiantes tópicos de pesquisa dentro da computação e da matemática. Vários resultados importantes e bastante sofisticados têm sido obtidos ao longo destes últimos três séculos, porém muitas questões ainda permanecem sem solução. Além disso, do ponto de vista algorítmico, a maioria dos resultados mais elegantes e interessantes foi obtida apenas nas últimas poucas décadas [40].

Um grande número de problemas práticos pode ser visto como problemas em grafos [4]. Além disso, é bastante comum a existência de problemas, que embora aparentemente não possuam relação alguma, compartilhem em sua natureza uma estrutura algorítmica semelhante.

Neste sentido, diferentes problemas podem ser reduzidos a um mesmo problema básico em grafos. Um exemplo marcante é o problema de coloração de grafos, que é utilizado como modelo para a resolução de vários problemas bastante distintos, tais como otimização de código gerado por compiladores [5], alocação de tarefas para pessoas [22] e controle de tráfego aéreo [2].

Infelizmente não se conhece algoritmos eficientes para solucionar a maioria destes problemas básicos em grafos. De fato, com o desenvolvimento da teoria de complexidade computacional, a comunidade científica não espera que existam tais algoritmos.

A teoria de complexidade computacional, uma área virtualmente inexistente até 30 anos atrás [37], se expandiu tremendamente e hoje desponta como uma das áreas mais importantes dentro da ciência da computação. As ferramentas desenvolvidas por esta teoria permitem estudar a complexidade inerente a alguns problemas. Com isso, quando obtém-se resultados demonstrando que um determinado problema muito provavelmente não admite solução eficiente exata, a pesquisa empreendida em torno do mesmo procura outras altenativas.

Uma alternativa bastante comum é busca de soluções aproximadas. Muitas vezes, estas soluções são obtidas por métodos heurísticos ou por meio de abordagens probabilísticas [1]. Uma outra alternativa muito estudada é a busca de classes de instâncias destes problemas que admitam soluções exatas eficientes. Seja como for, muita pesquisa tem sido desenvol-

vida e ainda há de ser desenvolvida neste sentido.

Em geral, quando estamos falando de grafos, instâncias restritas de um problema querem dizer famílias de grafos. E, de fato, diversos problemas NP-completos tornam-se polinomiais em determinadas famílias de grafos.

Diante deste quadro, a procura e o estudo de famílias não triviais de grafos que admitam algoritmos polinomiais têm despertado bastante interesse dos pesquisadores.

Exatamente nesta linha, a família dos grafos perfeitos, grafos em que todo subgrafo induzido tem o número cromático igual ao número de clique (*clique number*), admite algoritmos polinomiais para importantes problemas de otimização combinatória (coloração, clique máxima, conjunto independente máximo), conforme apontou Grötschel [26] em 1981.

Neste contexto, ao longo das últimas duas décadas a pesquisa em teoria de grafos perfeitos articulou-se em torno de dois grandes problemas: a conjectura de Berge e a busca por algoritmos polinomiais para solução de problemas relacionados a grafos perfeitos. Na linha de algoritmos, tanto problemas de otimização quanto algoritmos de reconhecimento de classes de grafos têm sido foco de pesquisas neste contexto. Com relação à conjectura de Berge, os trabalhos publicados pelos pesquisadores buscavam casos especiais em que a conjectura era válida.

Recentemente foram alcançados dois resultados fundamentais na área. O primeiro é a demonstração da Conjectura de Berge, obtida em um trabalho conjunto no final do ano de 2002 por Chudnovsky, Robertson, Seymour e Thomas [9]. O segundo resultado importante é a descoberta de algoritmos polinomiais de reconhecimento de grafos perfeitos. Foram propostos dois algoritmos diferentes que, no entanto, compartilham um núcleo comum. O desenvolvimento deste núcleo é resultado de um trabalho conjunto dos dois grupos de pesquisadores envolvidos na descoberta dos algoritmos em questão. Estes dois algoritmos foram descritos inicialmente em três trabalhos (que são versões "preprint" dos artigos submetidos para publicação): em [8] é apresentado o núcleo dos dois algoritmos, em [11] é apresentado o algoritmo do grupo de Maria Chudnovsky, e finalmente em [18] é apresentado o algoritmo do grupo de Gérard Cornuéjols.

Estes trabalhos foram submetidos à revista *Combinatorica* em 2002. Recentemente, os autores disponibilizaram uma versão condensada em um único artigo dos três trabalhos [7]. Esta versão deve ser a que será publicada no periódico.

O objetivo principal deste trabalho é um estudo destes dois algoritmos e a implementação de parte deles. O trabalho focou-se especialmente no núcleo dos algoritmos, que sem dúvida é a parte mais complexa dos mesmos. Este núcleo foi implementado e alguns resultados são apresentados. Algo importante a ser mencionado é o fato de que até o momento o autor desta dissertação não tem notícias de outras implementações do algoritmo.

Adicionalmente, nesta dissertação são definidos alguns parâmetros que podem ser associados a um grafo para exprimir seu grau de imperfeição. A questão chave é que o recém

descoberto algoritmo de teste de perfeição, que é o foco da dissertação, pode ser utilizado em algoritmos (exponenciais) para calcular estes parâmetros, similarmente ao que acontece, por exemplo, com o algoritmo de teste de planaridade em relação aos problemas de encontrar algumas medidas de planaridade de grafos.

O autor relaciona estes parâmetros de imperfeição com algumas operações que podem ser aplicadas a um grafo imperfeito para torná-lo perfeito. Como contribuição, são obtidos alguns resultados nesta linha.

Voltando ao teste de perfeição, é necessário dizer que o autor dividiu a explicação dos algoritmos em três partes:

A primeira parte, que é descrita no capítulo 2, agrupa vários algoritmos que testam pela presença de subgrafos específicos que não podem aparecer nos grafos de entrada em certas etapas dos algoritmos. Alguns destes testes por subgrafos específicos ocorrem no núcleo dos algoritmos e outros testes ocorrem em um ou nos dois algoritmos de teste de perfeição.

A segunda parte, que aparece no capítulo 3, explica o funcionamento do núcleo do algoritmo. Para que os teoremas pudessem ser mais facilmente compreendidos, alguns deles foram subdivididos em subteoremas e alguns outros sofreram pequenas alterações em suas demonstrações que o autor julgou razoável. O teorema 3.6, por exemplo, era obtido como parte da prova do teorema 3.7 no artigo original [8]. Além desta alteração no teorema 3.7, o autor preferiu enunciar e provar alguns teoremas (3.2, 3.3, 3.4, 3.5), como resultados auxiliares ao invés de utilizar outros dois teoremas um pouco confusos e que não aparecem nesta dissertação (entretanto, é importante ressaltar que estes dois teoremas são usados em demonstrações de alguns teoremas que não foram provadas aqui). Em adição a isso, alguns passos das demonstrações que pareciam um pouco difíceis foram provados separadamente como "fatos".

A terceira parte, que é descrita no capítulo 4, apresenta os dois algoritmos de teste de perfeição de grafos. Como o foco da dissertação é o núcleo dos algoritmos, algumas demonstrações de teoremas foram omitidas neste capítulo.

A outra contribuição deste trabalho, que já foi mencionada, é a implementação do núcleo do algoritmo. O algoritmo é bastante complexo, e a implementação de seu núcleo se apresenta como um desafio formidável. Do ponto de vista técnico, é importante mencionar que o algoritmo foi implementado como um *plugin* do programa *GRAFO*[1], e por este motivo a implementação apresenta-se bastante portável, podendo ser utilizada em futuras implementações dos dois algoritmos de teste de perfeição de grafos. Como este núcleo dos algoritmos foi desenvolvido recentemente e é bastante complexo, esta implementação aparentemente mostra-se como a única do gênero até o momento.

A divisão do trabalho em capítulos é feita da seguinte forma:

[1] O programa GRAFO é um software que tem o objetivo de facilitar a implementação de algoritmos em grafos.

- O capítulo 1 tem duas funções. Inicialmente são apresentados os conceitos e algumas definições básicas de teoria de grafos e de teoria de grafos perfeitos necessários no trabalho. A seguir, uma pequena resenha dos resultados obtidos em teoria de grafos perfeitos que vieram a culminar na obtenção do algoritmo que é o foco principal deste trabalho.

- Nos capítulos 2, 3 e 4 são estudados os algoritmos de reconhecimento de grafos perfeitos propriamente ditos. No capítulo 2 são apresentadas algumas definições mais específicas que são úteis nos algoritmos e também são estudados caso a caso subgrafos e família de subgrafos imperfeitos que não devem aparecer no grafo de entrada em outras etapas mais gerais dos algoritmos. No capítulo 3 é descrito o núcleo que os dois algoritmos compartilham enquanto que no capítulo 4 são apresentados individualmente os dois algoritmos.

- No capítulo 5 são definidos alguns parâmetros de imperfeição e são obtidos alguns resultados nesta linha.

- No capítulo 6 são discutidos alguns resultados obtidos a partir dos testes feitos com os algoritmos que foram implementados.

- O capítulo 7 apresenta a conclusão da dissertação.

Para finalizar, de maneira objetiva as contribuições do autor foram:

- Produção de um texto que apresenta os algoritmos de teste de perfeição de grafos e discute os aspectos relevantes destes algoritmos.

- Implementação de parte dos algortimos estudados.

- Definição de algumas medidas de imperfeição de grafos e obtenção de resultados nesta linha.

Capítulo 1

Fundamentos e Trabalhos Relacionados

Este capítulo tem como objetivo apresentar os conceitos básicos e estabelecer a nomenclatura utilizada no decorrer do trabalho e, além disso, também visa apresentar um breve resumo da evolução da área de teoria de grafos perfeitos apresentando e discutindo os trabalhos mais relevantes da área.

1.1 Definições Básicas

Um grafo G é um conjunto $V(G)$ finito e não vazio de vértices e um conjunto $E(G)$ de arestas, que são subconjuntos de dois elementos de $V(G)$. Dado $u, v \in V(G)$, se $\{u, v\} \in E(G)$, dizemos que existe uma aresta *entre* u e v. Para uma aresta $e = \{u, v\}$, a fim de simplificar a notação, escreveremos uv. Dizemos que uma aresta e é *incidente* (ou simplesmente *incide*) a um vértice v se $v \in e$. Dois vértices u e v são *adjacentes* se $uv \in E(G)$. Neste caso também podemos dizer que u é vizinho de v e vice-versa. Denotamos por $N(v)$ o conjunto dos vértices que são vizinhos de v. De maneira geral, dados $v_1, v_2, ..., v_k \in V(G)$, definimos $N(v_1, v_2, ..., v_k)$ como $N(v_1) \cap N(v_2) ... \cap N(v_k)$. O *grau* de um vértice é o número de arestas incidentes a ele.

Se $|V(G)| = n$ e existe uma aresta entre cada par de vértices de $V(G)$, dizemos que G é um grafo *completo* com n vértices, denotado por K_n. Um grafo G é dito *bipartite* se podemos dividir o conjunto $V(G)$ em dois conjuntos disjuntos A e B de maneira que toda aresta do grafo é da forma $e = \{a, b\}$, onde $a \in A, b \in B$.

Chamaremos de *caminho* uma seqüência $v_0, e_1, v_1, e_2, ..., e_k, v_k$, onde $e_1, e_2, ..., e_k \in E(G)$ e $v_0, v_1, ..., v_k \in V(G)$, e_i é uma aresta entre v_{i-1} e v_i e os vértices v_i, $i = 0, 1, ..., k$ do caminho são distintos. Para simplificar usaremos apenas a seqüência de vértices $v_0, v_1, ..., v_k$ para indicar um caminho. Chamaremos às vezes também este tipo de grafo de *grafo caminho*. O *tamanho* de um caminho P é o número de arestas de P. Se P tem tamanho n dizemos que P é um n-caminho. Indicaremos algumas vezes, como é de cos-

tume na literatura, este caminho pelos vértices que o compõe separados por hífens como em v_1–v_2–...–v_{n-1}–v_n. Dado um caminho $P =$$v_1$–$v_2$–...–$v_{n-1}$–$v_n$, dizemos que v_1 e v_n são as extremidades de P e que $\{v_2, v_3, ..., v_{n-2}, v_{n-1}\}$ são os vértices interiores de P. Denotaremos P^* o conjunto dos vértices interiores de um caminho P. Se tomarmos a definição de caminho e obrigarmos v_0 e v_k coincidirem teremos um *ciclo*. Se o número de vértices de um ciclo é ímpar dizemos que o ciclo é *ímpar*, caso contrário dizemos que o ciclo é *par*. Chamaremos um grafo que não tenha nada além de um ciclo de n vértices de C_n ou de *grafo ciclo*. Se n for ímpar diremos que o grafo é um *grafo ciclo ímpar* e se n for par *grafo ciclo par*. Também chamaremos o C_3 de *triângulo*.

Dado um grafo G, um *subgrafo* H de G é um grafo em que $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Um subgrafo H de G é dito um subgrafo *próprio* de G se $H \neq G$. Um subgrafo H de G *induzido* por um conjunto $A \subseteq V(G)$ é um subgrafo maximal onde $V(H) = A$. Neste caso poderemos denotar $H = G[A]$. O *complemento* de um grafo G, denotado por $\overline{G}$, é um grafo com o conjunto de vértices $V(\overline{G}) = V(G)$ e o conjunto de arestas $E(\overline{G})$, onde $uv \in E(\overline{G})$ se e somente se $uv \notin E(G)$.

Seja um grafo G e um vértice $v \notin V(G)$, usaremos $G + v$ para denotar a inserção do vértice v no grafo G, ou seja, $G+v$ é o grafo G' tal que $E(G') = E(G)$ e $V(G') = V(G) \cup \{v\}$. Agora de maneira semelhante: seja $e = \{v_1, v_2\}$ uma aresta tal que $v_1, v_2 \in V(G)$ e $e \notin E(G)$, U um conjunto de vértices, com $U \cap V(G) = \varnothing$ e X um conjunto de arestas $x_i = \{v_j, v_k\}$, $v_j, v_k \in V(G)$, com $X \cap E(G) = \varnothing$, então $G + e$ é o grafo G' tal que $V(G') = V(G)$ e $E(G') = E(G) \cup \{e\}$, $G + U$ é o grafo G' tal que $E(G') = E(G)$ e $V(G') = V(G) \cup U$ e $G + X$ é o grafo G' tal que $V(G') = V(G)$ e $E(G') = E(G) \cup X$. Definiremos a remoção de um vértice (aresta) ou de um conjunto de vértices (arestas) analogamente: seja $v \in V(G)$, $U \subset V(G)$, $e \in E(G)$ e $X \subseteq E(G)$, então $G - v$ é o subgrafo de G induzido por $V(G) \setminus \{v\}$. $G - U$ é o subgrafo de G induzido por $V(G) \setminus U$. $G - e$ é o grafo G' tal que $V(G') = V(G)$ e $E(G') = E(G) \setminus \{e\}$. $G - X$ é o grafo G' tal que $V(G') = V(G)$ e $E(G') = E(G) \setminus X$[1].

Um grafo G é *conexo* se existe um caminho entre cada par de vértices de G. Se um grafo G não é conexo chamaremos os subgrafos conexos maximais de *componentes conexos* ou simplesmente *componentes* de G. Um *conjunto de corte* S de G é um subconjunto de $V(G)$ tal que $G - S$ tem mais componentes conexos que G.

Dado um grafo G, o *grafo linha* de G é o grafo H construído da seguinte maneira: $V(H) = \{u_1, u_2, ..., u_m\}$, onde $m = |E(G)|$ e cada u_i, para $i = 1, 2, ..., m$, está relacionado a uma aresta e_i de G. Existe uma aresta em $u_k u_l \in E(H)$ se e somente se e_k e e_l são incidentes a um mesmo vértice.

Uma *clique* de um grafo G é um subgrafo completo de G. O *número de clique* de G é igual à cardinalidade da maior clique de G e será denotado por $\omega(G)$. Um *conjunto independente* de um grafo G é um subconjunto I de $V(G)$ tal que para todo par de vértices

[1]Note que há ambigüidade quando escrevemos $G - \{u, v\}$, pois isso pode indicar a remoção da aresta $\{u, v\}$ ou dos dois vértices u e v de G. Entretanto, quando escrevemos $V(G) \setminus \{u, v\}$ e $E(G) \setminus \{u, v\}$ não há ambigüidade alguma.

u e v de I, tem-se $uv \notin E(G)$. Um conjunto independente I de um grafo G é dito *máximo* se nenhum outro conjunto independente de G tem cardinalidade maior do que a cardinalidade de I. Se I é um conjunto independente máximo de G dizemos que $\alpha(G) = |I|$ é o *número de independência* de G.

Uma *coloração* (*apropriada*) de um grafo é um mapeamento $F : V(G) \to \mathbb{N}$ tal que se $uv \in E(G)$ então $F(u) \neq F(v)$. Chamaremos de *cores* os elementos de $\mathbb{N}$ associados aos vértices através de F. Uma k-coloração de um grafo é uma coloração que usa k cores. Um grafo que aceita uma k-coloração é dito k-colorível. O *número cromático* de um grafo G, denotado por $\chi(G)$, é o menor k, tal que exista uma k-coloração de G.

1.2 Grafos Perfeitos

O problema de coloração de grafos tem grande importância prática e teórica [20] e, portanto, a determinação de limitantes inferiores e superiores para o número cromático tem sido um problema de interesse dos pesquisadores em teoria de grafos já há muito tempo.

Um limitante inferior óbvio para o número cromático é $\chi(G) \geq \omega(G)$. Mas este limitante pode ser muito fraco, pois é possível construir um grafo G com $w(G) = 2$ e $\chi(G)$ arbitrariamente alto [4].

A partir disto é natural pensar em grafos em que o número cromático é igual ao número de clique, ou em uma versão mais forte disto em que a propriedade é válida para todo subgrafo induzido do grafo em questão. Esta versão mais forte é justamente a definição de grafos perfeitos:

Definição 1.1 (Grafos Perfeitos) *Um grafo G é perfeito se para todo subgrafo induzido H de G temos $\chi(H) = \omega(H)$.*

Um resultado muito importante na teoria dos grafos perfeitos, que foi obtido independentemente por Fulkerson [24] e por Lovász [31], é o seguinte:

Teorema 1.1 (Teorema dos Grafos Perfeitos) *Um grafo G é perfeito se e somente se $\overline{G}$ é perfeito.*

Utilizando este teorema e mais alguns resultados clássicos de teoria de grafos é possível mostrar que os grafos de um classe de grafos, chamada de classe dos *grafos básicos*, são perfeitos [10]. Veremos na próxima seção que esta classe de grafos tem bastante importância no contexto dos grafos perfeitos. Além disso, um dos algoritmos estudados no capítulo 4 também utiliza estes conceitos. Vamos então aos enunciados:

Definição 1.2 (Grafos Básicos) *Um grafo é básico se ele é:*

- *Um grafo bipartite ou o complemento de um grafo bipartite*

- *Um grafo linha de um grafo bipartite ou o complemento de um grafo linha de um grafo bipartite.*

Teorema 1.2 (Idéia da prova em [10], 2003) *Se um grafo é básico então ele é perfeito.*

Para finalizar esta seção vamos definir o que é um *buraco* em um grafo e com isso enunciar o resultado mais importante da teoria de grafos perfeitos.

Definição 1.3 (Buracos e Antiburacos) *Seja um ciclo C com $n \geq 4$ vértices em G. Se o subgrafo de G induzido pelos vértices de C for um C_n, dizemos que C é um buraco de tamanho n em G. Um antiburaco é o complemento de um buraco. Um buraco (antiburaco) de tamanho ímpar é dito um buraco (antiburaco) ímpar.*

O grafo da figura 1.1 é um exemplo de um grafo contendo um buraco ímpar de tamanho 5, formado pelos vértices $v_1, ..., v_5$. Observe que os ciclos v_2–v_3–u_1–v_2, v_3–v_4–u_3–u_2–v_3 e v_3–v_4–u_3–u_2–u_1–v_3 não são buracos ímpar. O primeiro não é um buraco pois tem tamanho 3, o segundo não tem tamanho ímpar e o terceiro não induz um C_5, pois contém a aresta v_3u_2.

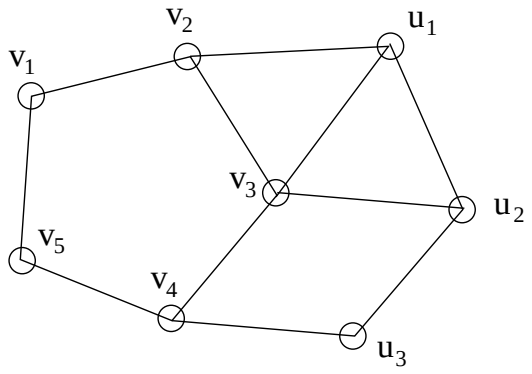


Figura 1.1: Exemplo de grafo contendo um buraco.

Antes do teorema vamos definir o que é um grafo de Berge:

Definição 1.4 (Grafos de Berge) *Dizemos que um grafo G é um grafo de Berge se G e $\overline{G}$ não contêm buracos ímpar.*

Agora o teorema mais importante da teoria de grafos perfeitos:

Teorema 1.3 (Teorema Forte dos Grafos Perfeitos) *Um grafo G é perfeito se, e somente se, G e $\overline{G}$ não contêm buracos ímpar.*

Este último enunciado, que foi conjecturado por Claude Berge em 1961 [3], mantinha-se como um dos grandes problemas em aberto de toda teoria de grafos até recentemente.

Em 2002, um grupo de pesquisadores que há tempos vinha trabalhando na área obtive uma demonstração para esta conjectura [9].

O leitor atento deve ter percebido que um outro enunciado deste resultado seria: "Os grafos de Berge são precisamente os grafos perfeitos". Este resultado é a base do algoritmo que estudaremos neste trabalho.

1.3 As pesquisas em teoria de grafos perfeitos

1.3.1 Teorema Forte dos Grafos Perfeitos

A família dos grafos perfeitos admite algoritmos polinomiais para diversos problemas de otimização combinatória, conforme apontou Grötschel [26], em 1981. Esta é sem dúvida, entre as famílias de grafos, uma das mais interessantes neste sentido, pois diversas classes de grafos que admitem algoritmos polinomiais para alguns problemas difíceis são subconjuntos dos grafos perfeitos [23, 38, 32]. Além disso, não se conhece uma superfamília que contenha os grafos perfeitos e que admita algoritmos polinomiais para estes problemas.

Entretanto, a situação não é tão simples assim. Primeiramente, os algoritmos polinomiais não são de natureza combinatória. Tais algoritmos são baseados no método elipsoidal e, portanto, estão sujeitos à instabilidade numérica. Alguns autores chegam a argumentar que o método é apenas de interesse teórico [32, 28]. Em segundo lugar, justamente pelo fato da família dos grafos perfeitos, quando definidos em termos de $\omega(G)$ e $\chi(G)$, não serem estruturalmente triviais, uma caracterização estrutural dela mostrou-se ser um problema extremamente difícil. Neste sentido, Berge conjecturou em 1961 [3] que um grafo G é perfeito se, e somente se, G e seu complemento não contêm buracos ímpar maiores ou iguais a 5. Esta conjectura impulsionou uma enorme quantidade de pesquisa no assunto desde então, tornando-se um dos grandes problemas em aberto de toda teoria de grafos. Foi apenas recentemente que mostrou-se que a conjectura era de fato um teorema (teorema 1.3 da seção anterior). O resultado foi obtido no final do ano de 2002 por Chudnovsky, Robertson, Seymour e Thomas em um artigo de 148 páginas [9].

Como já mencionamos, o enunciado deste teorema diz que um grafo é perfeito se, e somente se, ele é de Berge. É fácil ver que todo grafo perfeito é um grafo de Berge. Se um grafo perfeito não é um grafo de Berge então ele tem como subgrafo induzido um grafo ciclo ímpar ou o complemento de um grafo ciclo ímpar, mas estes dois grafos não são perfeitos. A dificuldade está em mostrar que todos os grafos de Berge são perfeitos.

Para provar o teorema, os pesquisadores mostraram que ou um grafo de Berge pertence a uma classe de grafos conhecidamente perfeita ou admite uma dentre certas decomposições (partição do conjunto de vértices do grafo respeitando certas restrições) que de maneira geral tem a seguinte característica: se um grafo admite alguma destas decomposições então o grafo não é um contra-exemplo para a conjectura de Berge. São utilizados três tipos possíveis de decomposições: decomposição em *2-join*, decomposição em *M-join* e decomposição em *even skew partitions*. Achamos desnecessário definir aqui formalmente

o que são estas decomposições. Também não definiremos formalmente aqui uma classe de grafos, que aparecerá nos teoremas seguintes, conhecida como classe dos grafos *double split*.

Vamos agora enunciar alguns resultados utilizando estas estruturas que não definimos formalmente.

Teorema 1.4 (*Chudnovsky e outros [9], 2002) Os grafos double split são perfeitos.*

Teorema 1.5 (*Cornuéjols e Cunningham [19], 1985) Se um grafo imperfeito admite uma decomposição 2-join então este grafo contém um buraco ímpar.*

Teorema 1.6 (*Chudnovsky e outros [9], 2002) Se G é o menor grafo de Berge imperfeito então G não admite uma decomposição em even skew partitions.*

Teorema 1.7 (*Chvátal e Sbihi [13], 2002) Se G é o menor grafo de Berge imperfeito então G não admite uma decomposição em M-join.*

A partir dos teoremas 1.2, 1.4, 1.5, 1.6, 1.7 podemos enunciar o resultado obtido em [9] que implica o Teorema Forte dos Grafos Perfeitos.

Teorema 1.8 (*Chudnovsky e outros [9], 2002) Para todo grafo de Berge G temos uma das cinco possibilidades:*

- *G é básico;*
- *G é grafo double-split;*
- *G ou $\overline{G}$ admite um 2-join;*
- *G admite um M-join;*
- *G admite uma partição skew par.*

1.3.2 Grafos Perfeitos e Algoritmos

Praticamente ao mesmo tempo que a Conjectura de Berge[2] foi provada, foram propostos os primeiros algoritmos de reconhecimento para os grafos de Berge (e conseqüentemente para toda a classe dos grafos perfeitos) [18, 8, 11]. Os algoritmos são bastante sofisticados e a descoberta deles também é um tanto surpreendente, visto que a classe dos grafos de Berge existe há mais de 40 anos e até então não se sabia ao menos se o problema de reconhecimento destes grafos pertencia à classe NP [11].

Especialmente nos últimos 20 anos, em função de todo o esforço empreendido na resolução da conjectura a respeito dos grafos de Berge, várias subclasses destes grafos em que tal conjectura era válida foram sendo descobertas. À medida que isso ocorria, diversos

[2]Essa conjectura era conhecida como Conjectura Forte dos Grafos Perfeitos.

resultados importantes sobre estas classes foram sendo obtidos, e entre estes resultados figuram vários algoritmos combinatórios exatos de coloração e de reconhecimento.

Nesta linha, podemos exemplificar os trabalhos de Bourlet e Fonlupt [6] no reconhecimento dos bastante estudados grafos de Meyniel [33], e mais tarde os trabalhos de Figueiredo e Vuskovic [21] no reconhecimento de uma generalização desta classe. Algoritmos de coloração para a classe dos grafos de Meyniel foram apresentados em [27]. Algoritmos de reconhecimento para classes de grafos perfeitos que não contêm certos subgrafos também têm sido propostos ao longo dos últimos anos. Os exemplos mais importantes são os grafos *claw-free* [14], *paw-free* [36] e *dart-free* [12]. Gerber e Hertz propuseram em 2001 [25] um algoritmo polinomial para encontrar o número cromático de grafos em uma superclasse dos grafos *paw-free*. No mesmo ano Fouquet *et al* [23] apresentam algoritmos polinomiais para reconhecer alguns grafos perfeitamente ordenáveis (o caso geral é NP-completo [34]).

Finalmente, no final de 2002, foram descobertos os dois algoritmos polinomiais de reconhecimento de grafos perfeitos [18, 11] que estudaremos neste trabalho. Os dois algoritmos compartilham um núcleo em comum [8], que é chamado de *algoritmo de limpeza de grafos*. Este núcleo é baseado em métodos que alguns dos autores de um dos dois algoritmos já haviam usado em outros contextos, como o reconhecimento de matrizes linearmente balanceadas [15] e o reconhecimento da classe dos grafos livres de buracos pares [16].

Capítulo 2

Definições e Alguns Subgrafos Imperfeitos

Este capítulo tem dois objetivos. Um deles é apresentar várias definições que serão utilizadas no decorrer do trabalho. O outro objetivo é apresentar uma série de algoritmos que testam se um grafo contém certas estruturas, que podem ser certos subgrafos ou certas classes de subgrafos. A presença de cada uma destas estruturas em um grafo implica a existência de um buraco ou de um antiburaco ímpar no mesmo. Estes algoritmos serão úteis nos capítulos subseqüentes do trabalho, pois assume-se que algumas destas estruturas não estão presentes no grafo de entrada do algoritmo de limpeza de grafos (apresentado no capítulo 3) e que algumas outras estruturas também não estão presentes nas entradas dos algoritmos de teste de perfeição em grafos limpos (apresentados no capítulo 4).

2.1 Definições

Seja G um grafo:

Definição 2.1 (Buraco Ímpar Mínimo) *O buraco ímpar C de G é um buraco ímpar mínimo se nenhum outro buraco ímpar de G é menor (tem menos vértices) que C.*

Definição 2.2 (Conjuntos conectados e anticonectados) *O conjunto $X \subseteq V(G)$ é dito um conjunto conectado se $G[X]$ é conexo e é dito um conjunto anticonectado se $\overline{G}[X]$ é conexo.*

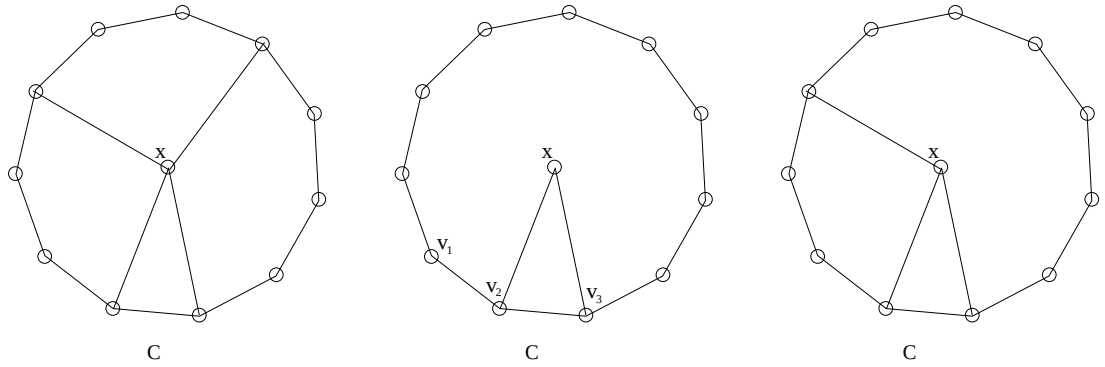
Definição 2.3 (Componentes e Anticomponentes) *Uma componente de $X \subseteq V(G)$ é um subconjunto conectado maximal de X. Uma anticomponente de $X \subseteq V(G)$ é um subconjunto anticonectado maximal de X.*

Definição 2.4 (Vértices X-completos) *Dado $X \subseteq V(G)$, o vértice $v \in V(G) \setminus X$ é X-completo se $uv \in E(G)$ para todo $u \in X$.*

Definição 2.5 (Aresta X-completa) *Dado $X \subseteq V(G)$, a aresta $uv \in E(G)$ é X-completa se u e v são X-completos.*

Definição 2.6 (Vértices C-Major) *Seja C um buraco ímpar de G e v um vértice de G que não esteja em C. Seja X o conjunto de vizinhos de v em C. Dizemos que o vértice v é um C-major se nenhum 2-caminho de C contém X. Quando não causar ambiguidade poderemos dizer simplesmente que v é um vértice major.*

No exemplo da figura 2.1(a) o vértice x é C-major. No exemplo da figura 2.1(b) o vértice x não é C-major, pois os vizinhos de x em C, que são v_2 e v_3, estão contidos no 2-caminho v_1–v_2–v_3.



(a) O vértice x é C-major. (b) O vértice x não é C-major. (c) O vértice x é uma C-raquete.

Figura 2.1: Vértices C-major e C-raquetes.

Desta definição, temos que se v tem 4 ou mais vizinhos em C então v obrigatoriamente é um C-major. Concluímos também que se C é um buraco ímpar mínimo e v tem 1 ou 2 vizinhos, v não pode ser major. No caso em que v tem 1 vizinho é óbvio e caso em que v tem 2 vizinhos em C, v não pode ser major porque C é mínimo. Caso v tenha 3 vizinhos em C, v é major quando dois dos vizinhos de v em C não sejam adjacentes. Este caso é definido agora:

Definição 2.7 (Raquetes) *Dado um buraco C, um vértice C-major v é uma C-raquete se v tem exatamente 3 vizinhos em C. Quando não causar ambiguidade poderemos dizer simplesmente que v é uma raquete.*

No exemplo da figura 2.1(c) o vértice x é C-raquete.

2.2 Entendendo os desenhos de grafos

A partir deste ponto do trabalho, aparecerão em alguns desenhos de grafo com mais elementos do que os usuais vértices e arestas. Nesta seção vamos estabelecer a notação que usaremos nestes desenhos.

- Uma aresta desenhada com linha cheia indica uma aresta que obrigatoriamente deve aparecer no grafo.
- Uma aresta desenhada com linha tracejada indica uma aresta proibida.
- Se não há aresta desenhada entre dois vértices, então a aresta entre este par de vértices é opcional.
- Conjuntos de vértices são envolvidos por uma linha tracejada.
- Uma linha tracejada ligando dois conjuntos de vértices A e B, direcionada de A para B, indica que nenhum vértice de A é B-completo.
- Uma linha cheia espessa ligando dois conjuntos A e B indica que existe todas as arestas possíveis ligando vértices de A a B.
- Uma linha tracejada espessa ligando um vértice a (ou um conjunto de vértices A) a um conjunto de vértices B indica que a vizinhança de a (ou de cada vértice de A) não contém vértices de B.

Na figura 2.2 por exemplo, a aresta v_2v_3 é obrigatória, a aresta v_4v_5 é proibida e a aresta v_4v_6 é opcional. Observe também que os vértices v_1, v_2 e v_3 formam o conjunto A. Na figura é indicado também que nenhum vértice do conjunto A é B-completo, o vértice v_7 não tem nenhum vizinho em B e que há todas as arestas possíveis entre o conjunto B e C, ou seja, no caso o subgrafo induzido pelos conjuntos B e C é um $K_{3,3}$. Além disso não há adjacências entre vértices de B e D.

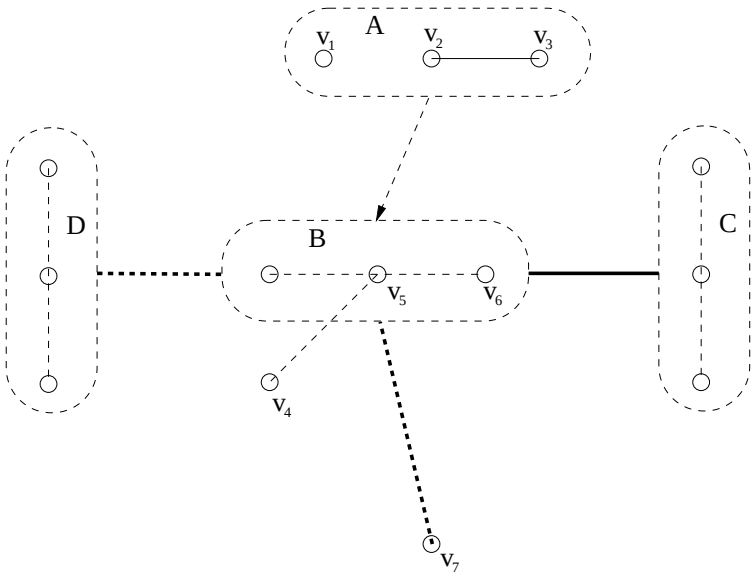


Figura 2.2: Entendendo os desenhos de grafos.

2.3 Algumas estruturas facilmente detectáveis

Nesta seção apresentaremos algumas estruturas que caso estejam presentes em um grafo G então obrigatoriamente G não é um grafo de Berge. Além disso, são apresentados algoritmos polinomiais para testar se um grafo contém alguma destas estruturas.

No capítulo 4 veremos que é possível testar perfeição em um *grafo limpo*, que é um grafo que não possui vértices C-major para certo buraco ímpar C. Um conjunto que contenha estes vértices indesejáveis, é chamado de *cleaner*. No capítulo 3 apresentaremos o *algoritmo de limpeza de grafos*, que é um algoritmo que dado um grafo de entrada, gera como saída vários conjuntos, sendo que um deles é um *cleaner* (na realidade veremos que os conjuntos gerados são "quase" *cleaners*, mas isso não importa agora). A questão importante agora é que tanto o algoritmo do capítulo 3 quanto os algoritmos do capítulo 4 assumem que o grafo de entrada não contém certas estruturas. São justamente estas estruturas que estudaremos agora.

2.3.1 A estrutura T_1

Uma estrutura do tipo T_1 é um buraco de tamanho 5. Para testar se um grafo contém uma estrutura do tipo T_1 basta enumerar todas as permutações de 5 vértices e testar se alguma delas forma um buraco de tamanho 5.

2.3.2 A estrutura T_2

Uma estrutura T_2 em G é uma seqüência $(v_1, ..., v_4, P, X)$ tal que:

- $v_1 - v_2 - v_3 - v_4$ é um caminho induzido em G.
- X é um anticomponente do conjunto de todos os vértices $\{v_1, v_2, v_4\}$-completos.
- P é um caminho em $G \setminus \{v_2, v_3\}$ de extremidades v_1 e v_4.
- nenhum vértice em P^* (lembrando que P^* é o conjunto dos vértices interiores de P) é X-completo ou adjacente a v_2 ou v_3.

A figura 2.3 apresenta um exemplo de um grafo que contém uma estrutura T_2. A seguir um resultado demonstrado em [9]:

Teorema 2.1 (*Chudnovsky e outros [9], 2003) Seja G um grafo de Berge, e X um subconjunto anticonectado de $V(G)$, e P um caminho de tamanho ímpar em G não passando por vértices de X tal que ambas as extremidades são vértices X-completos e nenhum vértice de P^* é X-completo. Então todo vértice X-completo tem um vizinho em P^*.*

Com isso temos:

Teorema 2.2 (*Chudnovsky, Cornuéjols e outros [8], 2003) Se um grafo G contém uma estrutura do tipo T_2, então G não é um grafo de Berge.*

PROVA: Se P é par, temos o buraco ímpar com os vértices v_1–v_2–v_3–v_4–P–v_1. Se P é ímpar, segue do teorema 2.1 que G não é um grafo de Berge, desde que v_2 (poderíamos usar o vértice v_3 também) não tem vizinhos em P^* e é X-completo. □

ALGORITMO 2.1 (*Encontrando estruturas T_2 – Chudnovsky, Cornuéjols e outros [8], 2003)*

- Entrada: grafo G.
- Saída: SIM se G tem uma estrutura T_2 e NÃO caso contrário.

Enumere todos os caminhos v_1–v_2–v_3–v_4. Para cada caminho encontre o conjunto Y de todos vértices $\{v_1, v_2, v_4\}$-completos. No exemplo da figura 2.3 temos $Y = \{w_1, w_2, w_3, w_4\}$. Para cada anticomponente X de Y teste se existe um caminho P entre v_1 e v_4 sem passar por v_2 e v_3, e tal que nenhum vértice de P^* seja adjacente a v_2 ou v_3 e nenhum vértice de P^* seja X-completo. Se encontrar tal caminho P responda SIM e se após enumerar todos os caminhos v_1–v_2–v_3–v_4 não se encontrar um caminho P para nenhum anticomponente X responda NÃO. No exemplo da figura 2.3, para a anticomponente $X = \{w_2, w_3, w_4\}$ encontramos o caminho $P = \{v_1, u_1, u_2, u_3, v_4\}$. □

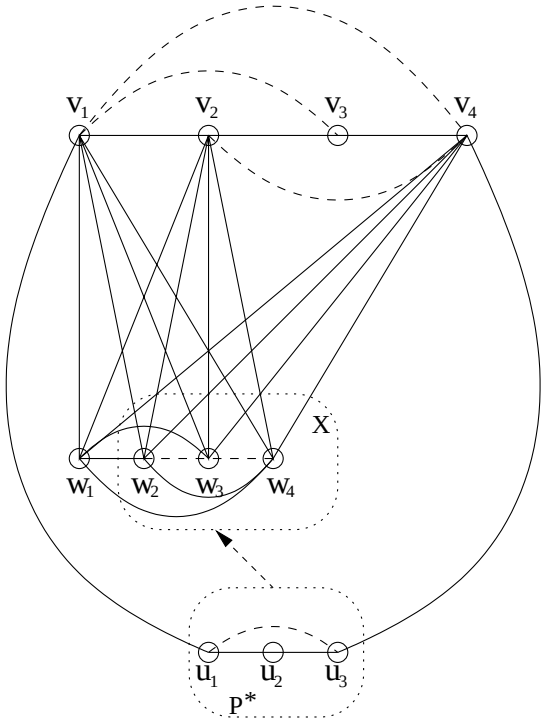


Figura 2.3: Exemplo de um grafo contendo estrutura T_2.

2.3.3 A estrutura T_3

Uma estrutura T_3 em G é uma seqüência $(v_1, ..., v_6, P, X)$ tal que:

- $v_1, ..., v_6$ são vértices distintos de G.
- $v_1v_2, v_1v_4, v_2v_3, v_3v_4, v_3v_5, v_4v_6 \in E(G)$.
- $v_1v_3, v_1v_5, v_1v_6, v_2v_4, v_2v_5, v_2v_6, v_4v_5 \notin E(G)$.
- X é um anticomponente do conjunto de todos os vértices $\{v_1, v_2, v_5\}$-completos.
- v_3 e v_4 tenham não vizinhos em X.
- P é um caminho em $G \setminus (X \cup \{v_1, v_2, v_3, v_4\})$ de extremidades v_5 e v_6.
- nenhum vértice em P^* é X-completo ou adjacente a v_1 ou v_2.
- se $v_5v_6 \in E(G)$ então v_6 não é X-completo.

A figura 2.4 apresenta um exemplo de uma estrutura T_3. A seguir um lema de Roussel-Rubio [39]:

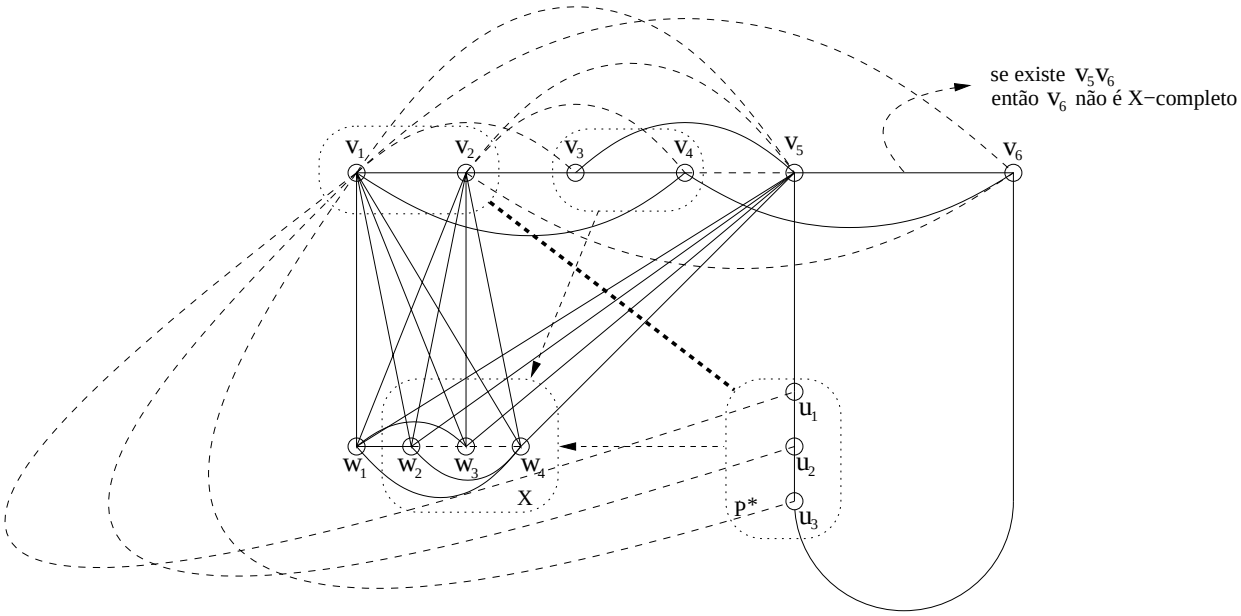


Figura 2.4: Exemplo de estrutura T_3.

LEMA 2.1 (*Roussel e Rubio [39], 2001) Seja G um grafo de Berge, X um subconjunto anticonectado de $V(G)$, P um caminho ímpar p_1–p_2...p_{n-1}–p_n em G sem nenhum vértice em X tendo os vértices das extremidades p_1 e p_2 X-completos e não tendo nenhum vértice de P^* X-completo. Então existe $x, y \in X$, $xy \notin E(G)$ tal que há exatamente duas arestas entre x,y e P^*, que são xp_2 e yp_{n-1}.*

Agora um resultado auxiliar. Para facilitar a leitura sugerimos que o leitor acompanhe o enunciado com a figura 2.5.

TEOREMA 2.3 (*Chudnovsky, Cornuéjols e outros [8], 2003) Seja G um grafo de Berge e X um subconjunto conectado de $V(G)$. Seja v_1, v_2, v_3, v_4, p_1, p_2 vértices distintos e $V(G) \setminus X$ tal que:*

- $v_1v_3 \in E(G)$, $v_2v_4 \in E(G)$ *e* $v_1v_2 \notin E(G)$, $v_1v_4 \notin E(G)$, $v_3v_2 \notin E(G)$, $v_3v_4 \notin E(G)$.
- v_3, v_4 *tem vizinhos em* X *e* v_1, v_2 *não tem.*
- $p_1p_2 \notin E(G)$; v_1, v_2, v_3 *são adjacentes a* p_1 *e* v_4 *não é adjacente a* p_1; v_1, v_2, v_4 *são adjacentes a* p_2 *e* v_3 *não é adjacente a* p_2.
- *Existe um caminho* Q *entre* v_3 *e* v_4 *com* $Q^* \subseteq X$ *e* p_1, p_2 *não tem vizinhos em* Q^*.
- *Nenhum vértice de* X *é* $\{v_3, v_4\}$*-completo.*

Então p_1 *e* p_2 *não tem vizinhos em* X.

PROVA: Suponha que um dos vértices p_1, p_2 tenha como vizinho o vértice $x \in X$. Seja x_1–...–x_k um caminho de vértices de X tal que x_k tenha um vizinho em Q^* com k mínimo. Isso é possível pois X é conexo e Q^* é não vazio.

Para ilustrar a situação, apresentamos na figura 2.5 um exemplo onde $k = 4$. Embora não esteja presente no desenho, em nossa suposição existe uma aresta entre x_1 e um dos vértices p_1, p_2.

Pela minimalidade de k nenhum de x_1,...,x_{k-1} tem vizinhos em Q^* e nenhum de x_2,...,x_k é adjacent a p_1 ou p_2. Sejam A e B caminhos de x_1 a v_3 e x_1 a v_4 respectivamente com interior em $\{x_2, ..., x_k\} \cup Q^*$. Na figura 2.5 temos $A = x_1$–...–x_4–q_1–v_3 e $B = x_1$–...–x_4–q_1–q_2–q_3–v_4. Por simetria podemos assumir que p_1 é adjacente a x_1. Com isso, a partir de B podemos completar um buraco utilizando v_4–v_2–p_1–x_1, e portanto, B tem tamanho ímpar. Como B é ímpar, para que não tenhamos um buraco ímpar completado a partir da união de B e do caminho v_4–p_2–v_1–p_1–x_1 e temos que p_2 tem vizinhos em $B \setminus \{v_4\}$. Como p_2 não tem vizinhos em $Q^* \cup \{x_2, ..., x_k\}$ segue que p_2 também é adjacente a x_1. Se a suposição fosse de que p_2 fosse adjacente a x_1 teríamos como conseqüência que p_1 também teria que ser adjacente a x_1. Isso reafirma a simetria de p_1 e p_2. Como B é ímpar, não podemos completar um buraco com B via v_4–p_2–x_1, e portanto x_1 é adjacente a v_4. Similarmente x_1 é adjacente a v_3, uma contradição com a última hipótese do teorema. □

TEOREMA 2.4 (*Chudnovsky, Cornuéjols e outros [8], 2003) Se um grafo G contém uma estrutura do tipo T_3, então G não é um grafo de Berge.*

PROVA: Seja G um grafo de Berge. Suponha que (v_1,...v_6,P,X) é uma estrutura do tipo T_3 em G. Seja Q um anticaminho entre v_3 e v_4 com interior em X. A figura 2.4 apresenta um exemplo. Neste caso podemos pensar em $Q = v_3$–w_3–w_4–v_4.

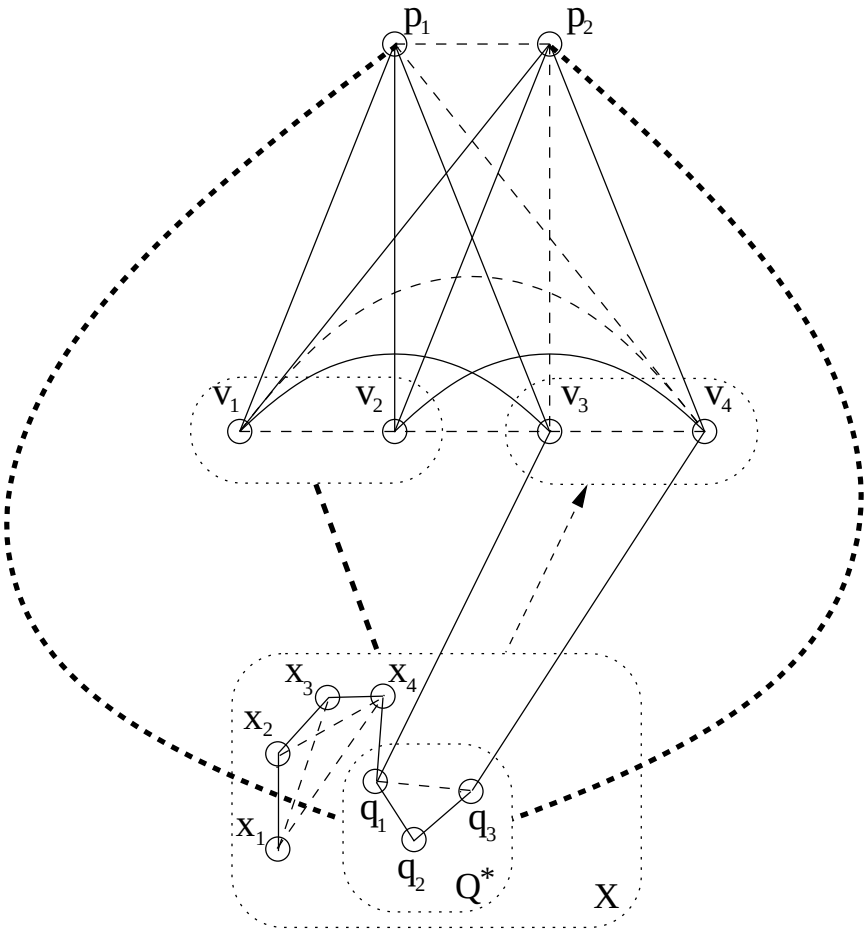


Figura 2.5: Visualização de um exemplo de grafo do teorema 2.3.

Como Q pode ser completado em um antiburaco via v_4–v_5–v_1–v_3, temos que Q é ímpar. Com isso v_1–v_3–Q–v_4–v_2 é um anticaminho ímpar. Utilizando o lema 2.1 sobre o grafo $\overline{G}$, o subconjunto anticonectado $V(P)$ e o caminho v_1–v_3–Q–v_4–v_2, deduzimos que existem $v_1, v_2 \in V(P)$, adjacentes em G, tal que as únicas não arestas entre p_1, p_2 e $V(P)$ são p_1p_4 e p_2p_3. Utilizando o teorema 2.3 sobre o grafo $\overline{G}$ deduzimos que um dos dois casos ocorrem:

- Algum $z \in X$ é não adjacente a ambos v_3, v_4.
- p_1 e p_2 são X-completo.

O primeiro caso é impossível pois z–v_1–v_4–v_3–v_5–z não é um buraco ímpar. O segundo caso é impossível pois nenhum vértice de P^* é X-completo, e se v_5v_6 são adjacentes então por hipótese v_6 não é X-completo. Portanto a suposição de que G tenha uma estrutura do tipo T_3 nos levou a uma contradição. Logo, se G é de Berge, G não possui uma estrutura do tipo T_3. □

ALGORITMO 2.2 (*Encontrando estruturas T_3 – Chudnovsky, Cornuéjols e outros [8], 2003)*

- Entrada: grafo G.
- Saída: SIM se G tem uma estrutura T_3 e NÃO caso contrário.

Enumere todas as tuplas v_1, v_2, v_5 de vértices distintos tal que v_1v_2 é uma aresta e v_5 não é adjacente a ambos v_1 e v_2. Para cada escolha de v_1, v_2 e v_5 encontre o conjunto Y de todos os vértices $\{v_1, v_2, v_5\}$-completos. No exemplo da figura 2.4 temos $Y = \{w_1, w_2, w_3, w_4\}$. Para cada anticomponente X de Y encontre o subconjunto conectado maximal F' contendo v_5 de maneira que v_1 e v_2 não tenham vizinhos em F' e nenhum vértice de F' é X-completo. Para o anticomponente $X = \{w_2, w_3, w_4\}$ da figura 2.4, supondo que $v_5v_6 \in E(G)$, temos $F' = \{v_5, v_6, u_1, u_2, u_3\}$. Seja F a união de F' com todos os vértices X-completos. Em nosso exemplo, $F = F'$. Ainda com a mesma escolha de v_1, v_2,v_5 e X, encontre o conjunto X_4 de todos os vértices candidatos a v_4, que sejam adjacentes a v_1 e não adjacentes a v_2 e v_5 e que tenham um vizinho em F e um não vizinho em X. Em nosso exemplo $X_4 = \{v_4\}$. Agora, para cada vértice de $v_4 \in X_4$ encontre v_3 adjacente a v_2, v_4 e v_5 e não adjacente a v_1 e com um não vizinho em X. Se encontrarmos este vértice então o grafo tem uma estrutura T_4, pois podemos tomar como v_6 um vizinho de v_4 em F, e P um caminho de v_6 a v_5 com interior em F'. Então, se encontramos o vértice v_3 responda SIM, e se após todas as escolhas de v_1, v_2, v_5, X e v_4 não encontramos tal vértice, responda NÃO. □

2.3.4 A Jóia

Uma estrutura Jóia em G é uma seqüência $(v_1, .., v_5, Q)$ tal que:

- $v_1, ..., v_5$ são vértices distintos de G.
- $v_1 - v_2 - v_3 - v_4$ é um caminho induzido em G.
- $v_1v_5 \in E(G)$ e $v_4v_5 \in E(G)$.
- Q é um caminho em G de extremidades v_1 e v_4.
- não existem arestas ligando vértices de $\{v_2, v_3, v_5\}$ a vértices de Q^*.

Um exemplo de uma jóia é apresentado na figura 2.6. Agora um teorema simples enunciando que grafos de Berge não contém jóias. A leitura da demonstração do teorema a seguir fica bastante simples quando feita com a ajuda da figura 2.6.

TEOREMA 2.5 *Se um grafo G contém uma jóia, então G não é um grafo de Berge.*

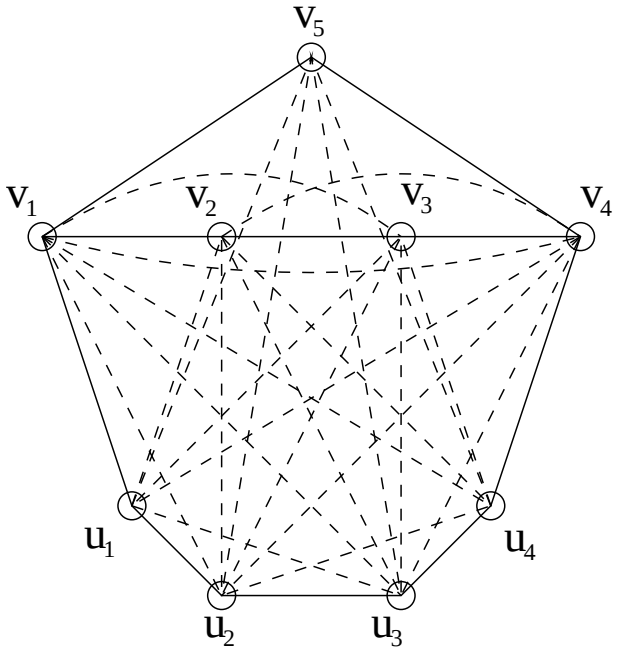


Figura 2.6: Uma jóia.

PROVA: Se o caminho Q de uma jóia tem tamanho ímpar, temos um buraco ímpar começando em v_5, passando por v_1 e usando o caminho Q até v_3 e então retornando a v_5. Se Q é par temos um buraco ímpar começando em v_1 e passando por v_2, v_3, v_4 e usando Q para retornar a v_1. □

ALGORITMO 2.3 (*Encontrando jóias – Chudnovsky e outros [11], 2003)*

- Entrada: grafo G.
- Saída: SIM se G tem uma jóia e NÃO caso contrário.

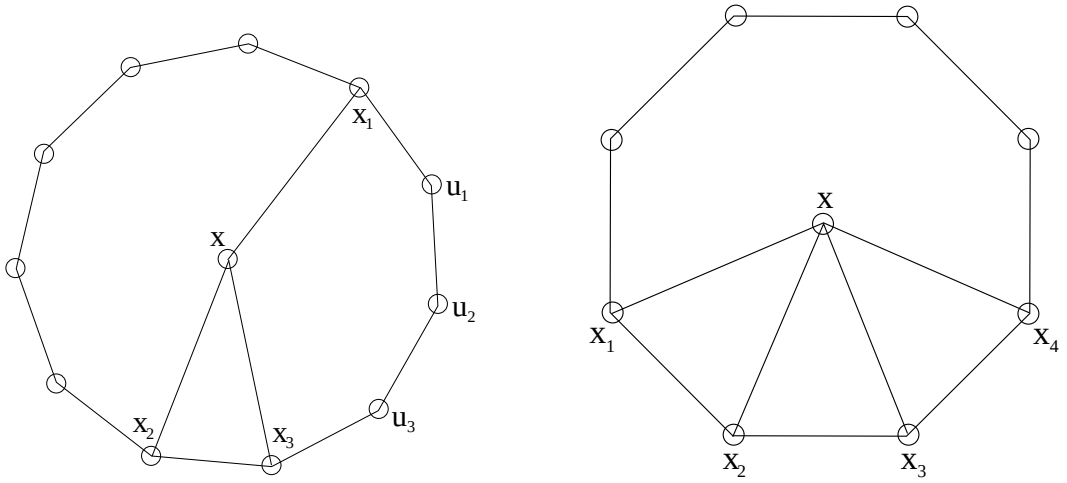
Enumere todas as tuplas v_2, v_3, v_5 de vértices distintos, tal que $v_2v_3 \in E(G)$. Para cada escolha v_2, v_3, v_5, encontre o conjunto F de todos vértices não adjacentes a estes três vértices e encontre todas suas componentes F'. Encontre o conjunto X_1 de vértices candidatos a v_1. O conjunto X_1 consiste dos vértices adjacentes a v_2 e v_5 mas não a v_3. Para cada componente F' de F e cada vértice $v_1 \in X_1$ registre se v_1 tem um adjacente em F'. Construa o conjunto X_4 de vértices candidatos a v_4 de maneira análoga, ou seja, tomando os vértices adjacentes a v_3 e v_5 e não a v_2. Para cada componente F' de F e cada vértice $v_4 \in X_4$ registre se v_4 tem um adjacente em F'. Verifique se existe um par $v_1 \in X_1$ e $v_4 \in X_4$ de vértices não adjacentes, em que ambos possuam um vizinho em uma mesma componente F'. Se existe tal par, responda SIM. Caso após examinar todas as escolhas não encontrarmos tal par de vértices, responda NÃO. □

2.3.5 O C_7 induzido

Para testar se um grafo contém um C_7 induzido (um buraco de tamanho 7) basta enumerar todas as permutações de 7 vértices e testar se alguma delas forma um buraco de tamanho 7.

2.3.6 A Roda

Uma *roda*, denotada por (H, x), é um grafo construído a partir de um grafo ciclo H e um vértice $x \notin V(H)$ e pelo menos três arestas ligando x a vértices de H. O vértice x é o centro da roda. A figura 2.7(a) mostra um exemplo de roda em que os vizinhos do centro x são x_1, x_2 e x_3. Digamos que os vizinhos de x em H sejam os vértices x_1,...,x_k. Dado $i, j \in \{1, ..., k\}$, um subcaminho de H ligando x_i e x_j não contendo nenhum vértice intermediário adjacente a x é um *setor* de H. Um *setor curto* é um setor de tamanho 1 e um *setor longo* é um setor de tamanho maior que 1. Uma roda é ímpar (par) se contém um número ímpar (par) de setores curtos. No exemplo da figura 2.7(a) temos uma roda ímpar. O caminho x_2, x_3 é o único setor curto da roda e o caminho x_1, u_1, u_2, u_3, x_3 é um setor longo.



(a) Uma roda. (b) Uma roda pequena.

Figura 2.7: Rodas.

FATO 2.1 *Se um grafo G contém uma roda ímpar (H, x) então G contém um buraco ímpar.*

PROVA: Se H é um buraco ímpar não há nada a provar. Se H é um buraco par e (H, x) é uma roda ímpar, então algum setor longo P de H é ímpar e portando o subgrafo de G induzido por $V(P) \cup \{x\}$ é um buraco ímpar. □

Uma *roda pequena* é uma roda ímpar (H, x) que contém exatamente um setor longo e x tem 4 vizinhos em H. A figura 2.7(b) mostra uma roda pequena.

Algoritmo 2.4 (Encontrando rodas pequenas)

- Entrada: grafo G.
- Saída: SIM se G contém uma roda pequena e NÃO caso contrário.

Enumere todas as tuplas x, x_1, x_2, x_3. x_4 de $V(G)$ tal que existam as arestas x_1–x_2–x_3–x_4 é um caminho induzido e x é adjacente a todos x_1,x_2, x_3,x_4. Verifique se existe um caminho entre x_1 e x_4 sem passar por x, x_2, x_3. Caso exista tal caminho, o grafo contém uma roda.□

2.3.7 A Pirâmide

Uma pirâmide, denotado por $(X, Y, Z; u)$, é um grafo formado por três caminhos X, Y, Z, onde $X = x, ..., u$, $Y = y, ..., u$ e $Z = z, ..., u$, e X, Y, Z não tem nenhum outro vértice em comum além de u. Além disso obrigatoriamente no máximo um caminho entre X, Y, Z tem tamanho 1 e os vértices x, y, z formam um triângulo. A figura 2.8 mostra uma pirâmide.

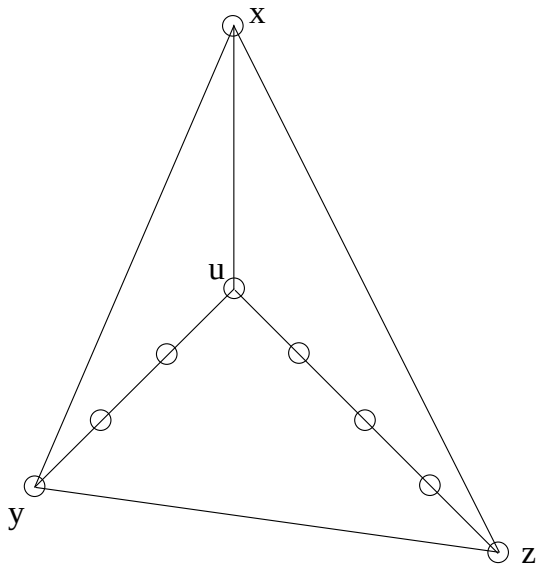


Figura 2.8: Uma pirâmide.

Fato 2.2 *Se um grafo G contém uma pirâmide $(X, Y, Z; u)$ então G contém um buraco ímpar.*

Prova: Entre os três caminhos X, Y, Z, dois deles têm a mesma paridade. Digamos, sem perder generalidade, que X e Y tem a mesma paridade. Então o subgrafo induzido pelos vértices de X mais os vértices de Y formam um buraco ímpar. □

Uma pirâmide $(X, Y, Z; u)$ é uma *pirâmide pequena* se X tem tamanho 1 e Y tem tamanho no máximo 4. A pirâmide da figura 2.8 é pequena.

Algoritmo 2.5 (Encontrando pirâmides pequenas)

- Entrada: grafo G.
- Saída: SIM se G contém uma pirâmide e NÃO caso contrário.

Enumere todas as tuplas u, x, y, z de $V(G)$ tal que x, y e z formem um triângulo e u seja adjacente de x e não seja adjacente a ambos y e z. Para cada tupla, enumere todos os caminhos induzidos P_i de tamanho 2, 3 e 4 entre u e y sem passar por x e z. Para encontrar os caminhos de tamanho 2 basta encotrar os vértices $u_1 \in V(G) \setminus \{u, x, y, z\}$ tal que existam as arestas uu_1 e u_1y. Para os caminhos de tamanho 3 e 4 basta enumerar as tuplas u_1, u_2 e depois u_1, u_2, u_3 e testar as adjacencias apropriadas. Para cada caminho P_i verifique se existe um caminho entre u e z sem passar por x, y e também sem passar pelos vértices interiores de P_i. Caso exista tal caminho responda SIM. Caso todas as tuplas u, x, y, z tenham sido enumeradas sem encontrar tais caminhos responda NÃO. □

Ao contrário do caso específico das pirâmides pequenas, encontrar pirâmides no caso geral é bem mais difícil. O algoritmo não será apresentado aqui. O algoritmo para este caso geral pode ser encontrado em [7]

Capítulo 3

Limpando Grafos

Esta seção estuda o núcleo que os dois algoritmos de teste de perfeição de grafos compartilham. Os dois algoritmos, que serão vistos no próximo capítulo, são na realidade algoritmos de reconhecimento de grafos de Berge. Pelo teorema 1.3 estes algoritmos podem ser utilizados para o teste de perfeição de grafos.

Embora os algoritmos possam responder se um grafo G contém alguma das duas estruturas proibidas para grafos de Berge, que são o buraco e o antiburaco ímpar, eles não podem responder se G contém especificamente uma delas. Na realidade, ninguém sabe ainda se este problema pode ser resolvido em tempo polinomial. O que é mais intrigante é que o problema de testar se um grafo contém um buraco ímpar passando por um vértice específico é NP-completo [11].

O núcleo que os dois algoritmos compartilham é uma etapa conhecida como limpeza de grafos. Os dois algoritmos de teste de perfeição são capazes de descobrir se um grafo contém um buraco ímpar caso este grafo esteja livre de certas estruturas e também não contenha vértices *major* para algum buraco ímpar mínimo. A idéia então é inicialmente eliminar as estruturas proibidas e depois gerar subgrafos induzidos do grafo original, de maneira que algum deles possua um buraco mínimo sem vértices *major*. Para isso, precisamos encontrar uma família de subconjuntos de vértices, tal que algum desses subconjuntos contenha todos os vértices *major* para um buraco mínimo C, mas que não contenha vértices de C. Na realidade, veremos que um conjunto que "quase" possua esta propriedade será suficiente. Na seção seguinte tornaremos mais precisas estas definições e apresentaremos algumas propriedades que alguns conjuntos de vértices *major* possuem.

3.1 Conjuntos de vértices *major*

Definição 3.1 (cleaner) *Um cleaner para um buraco mínimo C de G é um conjunto $X \subseteq V(G) \setminus V(C)$ que contém todos os vértices C-major.*

Definição 3.2 (quasi-cleaner) *Um quasi-cleaner para um buraco mínimo C de G é um conjunto $X \subseteq V(G)$ que contém todos os vértices major para C e que todos os vértices*

de X em C estejam em um 2-caminho de C.

O objetivo do algoritmo de limpeza, como veremos na seção seguinte, é gerar uma família de subconjuntos dos vértices de um grafo de entrada, tal que um destes subconjuntos seja um *quasi-cleaner* para um buraco ímpar mínimo C de G. Para podermos encontrar vértices C-major, vamos estudar nesta seção algumas propriedades que certos conjuntos de vértices major possuem.

Na subseção 3.1.1 veremos que conjuntos independentes de vértices *major* possuem a propriedade de serem *normais* (esta propriedade será definida a seguir). Na subseção 3.1.2 vamos mostrar que se impedirmos que algumas estruturas apareçam no grafo, a propriedade "ser normal" ocorre em um caso mais geral, que são os conjuntos anticonectados (observe que um conjunto independente é um caso particular de conjuntos anticonectados).

3.1.1 Conjuntos Independentes de vértices *major*

Seja C um ciclo e $A, B \subseteq V(C)$ definimos:

Definição 3.3 (A-gap) *Um A-gap de C é um subgrafo de C composto por uma componente X do subgrafo induzido por $V(C) \setminus A$ mais os vértices em A que tenham vizinhos em X e as arestas ligando vértices de A a X. O tamanho de um A-gap é o número de arestas que ele tem.*

No exemplo da figura 3.1(a) para $A = \{v_7, v_1, v_2\}$ temos o A-gap: v_2–v_3–v_4–v_5–v_6–v_7.

Definição 3.4 (A-aresta) *Uma aresta v_1v_2 é uma A-aresta se $v_1, v_2 \in A$.*

Definição 3.5 ((A,B)-gap) *Um (A,B)-gap em C é um caminho P em C, digamos entre a e b, tal que a é o único vértice de A em P e B é o único vértice de B em P. Possivelmente $a = b$ e P tem comprimento 0.*

No exemplo da figura 3.1(b) para $A = \{v_1, v_3, v_4\}$ e $B = \{v_4, v_7, v_8, v_9\}$ temos os (A,B)-gaps: v_9–v_{10}–v_{11}–v_1 e v_4.

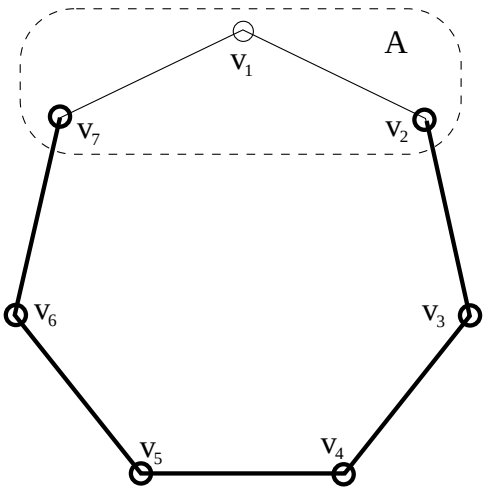
Definição 3.6 (Subconjuntos normais) *Um subconjunto $A \subseteq V(C)$ é normal em C se todo A-gap de C tiver tamanho par.*

Na figura 3.2, o subconjunto $A = \{v_2, v_4, v_5, v_7\}$ é normal.

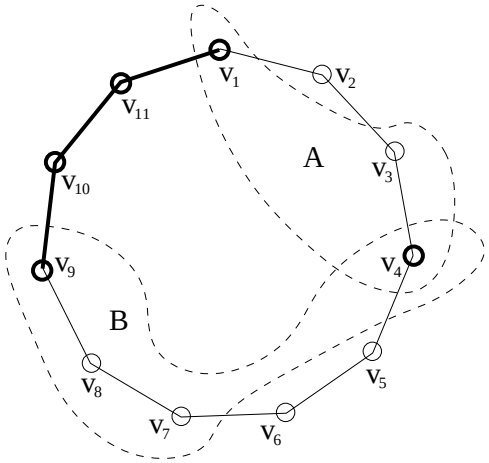
A prova do seguinte teorema que pode ser obtida por indução [8] será omitida aqui.

Teorema 3.1 (Chudnovsky, Cornuéjols e outros [8], 2003) *Seja C um ciclo ímpar. Seja $A_1, ..., A_k \subseteq V(C)$ normais, tal que para $1 \leq i < j \leq k$, cada (A_i, A_j)-gap é par. Então $A_1 \cap ... \cap A_k$ é normal.*

Agora enunciaremos e provaremos alguns teoremas que serão úteis na demonstração dos teoremas 3.6 e 3.7. Salientamos que apesar da idéia da prova dos teoremas 3.6 e 3.7



(a) Um A-gap em um C_7 para $A = \{v_7, v_1, v_2\}$. Os vértices e arestas do A-gap são desenhados com linhas mais fortes.



(b) Os vértices e arestas do (A, B)-gap são desenhados com linhas mais fortes.

Figura 3.1: A-gaps e (A, B)-gaps.

neste trabalho ser a mesma do artigo original (em [8]), na demonstração original os autores utilizavam outros resultados que não enunciamos aqui ao invés dos teoremas a seguir.

TEOREMA 3.2 *Seja C um buraco ímpar mínimo em G e $x \in V(G)$ um C-major. Então segue que:*

1. *O número de vértices adjacentes a x em C é maior ou igual a 3.*
2. *O conjunto dos vértices adjacentes a x em C é normal em C.*

PROVA: 1. Se x tem apenas um vizinho y em C, então y está em um 2-caminho induzido, uma contradição. Suponha que x tem 2 vizinhos y, z em C que não estão em um 2-caminho induzido em C. Então G tem um buraco ímpar menor que C começando em x, usando y e em seguinda um caminho induzido ímpar em C até z e chegando em x novamente. 2. Suponha que o conjunto A de vizinhos de x em C não seja normal. Então existe um A-gap ímpar P em C. A união de P, x e as arestas entre x as extremidades de P é um buraco ímpar menor que C, um absurdo.□

TEOREMA 3.3 *Seja C um ciclo ímpar. Seja $A \subseteq V(C)$ normal em C. Então C tem um número ímpar de A-arestas.*

PROVA: Uma aresta de C ou está em algum A-gap ou é uma A-arestas. Como A é normal, existe um número par de arestas em A-gaps. Portanto, como C é um ciclo ímpar, o número de A-arestas é ímpar. □

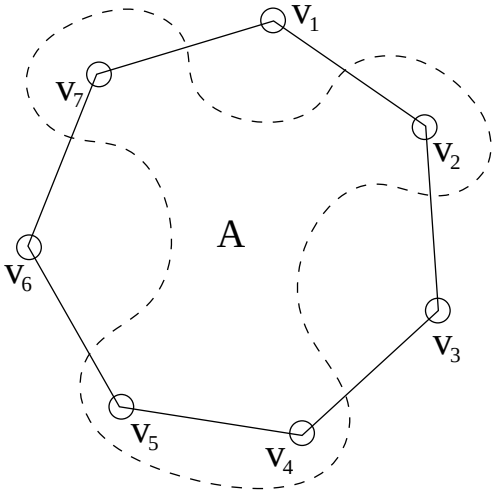


Figura 3.2: O subconjunto de vértices A é normal.

TEOREMA 3.4 *Seja C um ciclo e sejam $A, B \subseteq V(C)$ tal que $A \cap B = \varnothing$. Então C contém um número par de (A, B)-gaps.*

PROVA: Suponha que o número de (A, B)-gaps seja ímpar. Percorrendo o ciclo no sentido horário, sejam $P_1 = v_{1,1}$–...–v_{1,k_1}, $P_2 = v_{2,1}$–v_{2,k_2},..., $P_s = v_{s,1}$–...–v_{s,k_s} os (A, B)-gaps para um s ímpar onde $k_1, k_2, ..., k_s$ é o número de vértices de cada (A, B)-gap. Podemos supor sem perder generalidade que $v_{1,1} \in A$. Conseqüentemente $v_{1,k_1} \in B$. Como $A \cap B = \varnothing$ temos que $v_{2,1} \in B$ e $v_{2,k_2} \in A$ e assim sucessivamente até que último (A, B)-gap P_s tem $v_{s,1} \in A$ e $v_{s,k_s} \in B$. Como $A \cap B = \varnothing$ existe mais um (A, B)-gap com pelo menos uma aresta entre os vértices de B e A antes de P_1 o que é um absurdo, pois P_s é o último (A, B)-gap antes de P_1 percorrendo-se C no sentido horário. □

TEOREMA 3.5 *Seja C um ciclo ímpar. Sejam $A, B \subseteq V(C)$ normais em C, tal que $A \cap B = \varnothing$. Então C contém um (A, B)-gap par.*

PROVA: Uma aresta de C pode ser:

- ou uma aresta de um (A, B)-gap;
- ou uma aresta de um A-gap que não tenha interseção com um (A, B)-gap;
- ou uma aresta de um B-gap que não tenha interseção com um (A, B)-gap;
- ou uma A-aresta;
- ou uma B-arestas.

Primeiramente é importante mencionar que cada classe de aresta descrita acima é disjunta uma da outra, e qualquer aresta do ciclo pertence a uma das classes acima. Vejamos então

o que acontece se supormos que C não contém nenhum (A, B)-gap par. Como o número de (A, B)-gaps é par (teorema 3.4), e C só contém (A, B)-gaps ímpar, o número de arestas em (A, B)-gaps é par. Pelo teorema 3.3, o número de A-arestas mais o número de B-arestas é par. As arestas que restaram em C, são arestas que estão em A-gaps que não tenham interseção com (A, B)-gaps ou em B-gaps que não tenham interseção com (A, B)-gaps. Como C contém um número ímpar de arestas algum destes A-gaps ou B-gaps é ímpar, o que é um contradição, pois A e B são normais. Essa contradição veio da suposição de que C não contém nenhum (A, B)-gap par. □

TEOREMA 3.6 *[Chudnovsky, Cornuéjols e outros [8], 2003] Seja um grafo G e C um buraco ímpar mínimo de G sem raquetes. Seja $x_1, x_2 \in V(G)$ dois vértices major não adjacentes e A_i, para $i = 1, 2$ o conjunto de vértices adjacentes a x_i em C. Assuma que G tem um (A_1, A_2)-gap ímpar P. Então se Q é um (A_1, A_2)-gap par, temos que:*

- $V(P) \cap V(Q) = \varnothing$.
- *Exite uma aresta entre $V(P)$ e $V(Q)$.*

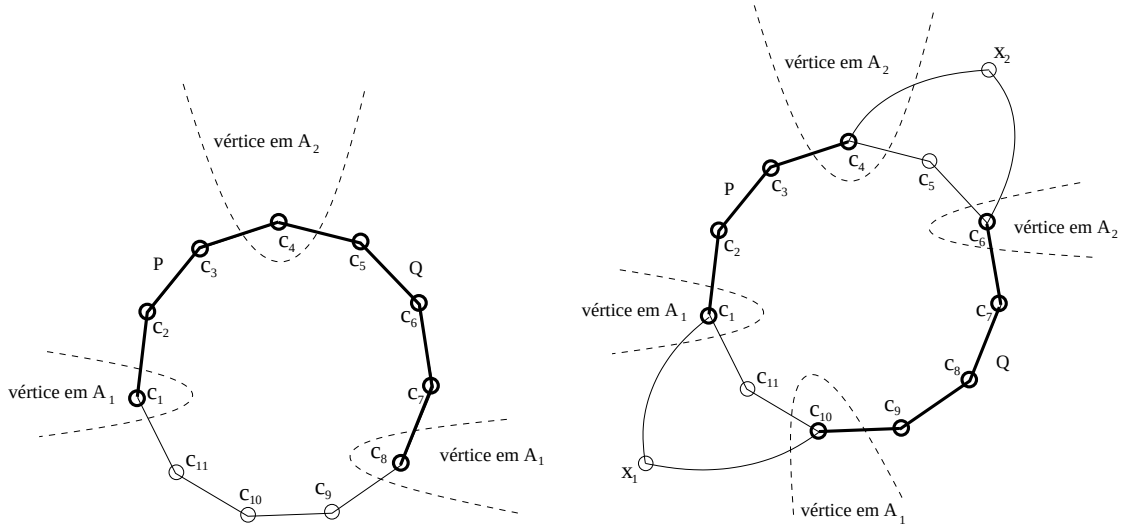
PROVA: Pelo teorema 3.2 para $1 \leq i \leq k$ temos que A_i é normal e que $|A_i| \geq 3$. Como G não tem raquetes, temos que $|A_i| \geq 4$. Seja $c_1, ..., c_{2n+1}$ os vértices de C e $c_1, ..., c_r$ os vértices de P para um r par. Vamos supor que $V(P) \cap V(Q)$ não é vazio, com digamos $c_r \in V(Q)$. Um exemplo onde $r = 4$ é mostrado na figura 3.3(a). Supondo isso temos que a união de P e Q origina ou um A_1-gap ímpar ou um A_2-gap ímpar. No exemplo da figura 3.3(a) vemos que esta união gerando um A_1-gap ímpar.

Agora vamos provar a segunda afirmação. Suponha que não há arestas ligando vértices de P e Q. Um exemplo é apresentado na figura 3.3(b). Então a união de P e Q mais os vértices x_1 e x_2 gera um buraco ímpar. Como C é um buraco ímpar mínimo, então Q só pode ser o caminho c_{r+2}–...–c_{2n}. O exemplo da figura 3.3(b) ilustra o caso de $r = 4$ e $n = 5$. Como $|A_1|, |A_2| \geq 4$ segue que $c_{r+1}, c_{2n+1} \in A_1 \cap A_2$. Mas então c_{r+1}–...c_{2n} é ou um A_1-gap ímpar (caso $c_{r+2} \in A_2$) ou um A_2-gap ímpar (caso $c_{r+2} \in A_1$). Em ambos os casos uma contradição. Para ilustração, a figura 3.3(b) apresenta um exemplo do primeiro caso. □

Com esses resultados podemos demonstrar o seguinte resultado:

TEOREMA 3.7 *[Chudnovsky, Cornuéjols e outros [8], 2003] Seja um grafo G e C um buraco ímpar mínimo de G sem raquetes. Seja X um conjunto independente de vértices major. Então o conjunto de vértices X-completos em C é normal.*

PROVA: Seja $X = \{x_1, ..., x_k\}$ e para $1 \leq i \leq k$ seja A_i o conjunto de vizinhos de x_1 em C. Pelo teorema 3.2 para $1 \leq i \leq k$ temos que A_i é normal e que $|A_i| \geq 3$. Como G não tem raquetes, temos que $|A_i| \geq 4$. Se para $1 \leq i < j \leq k$ ocorre que todo (A_i, A_j)-gap



(a) Suposição de que $V(P) \cup V(Q)$ não é vazio na prova do teorema 3.6.

(b) Suposição de que não há arestas ligando P a Q na prova do teorema 3.6.

Figura 3.3: Desenhos auxiliando a interpretação do teorema 3.6.

é par, então o resultado segue direto do teorema 3.1. Vamos supor então que existe um (A_1, A_2)-gap ímpar P.

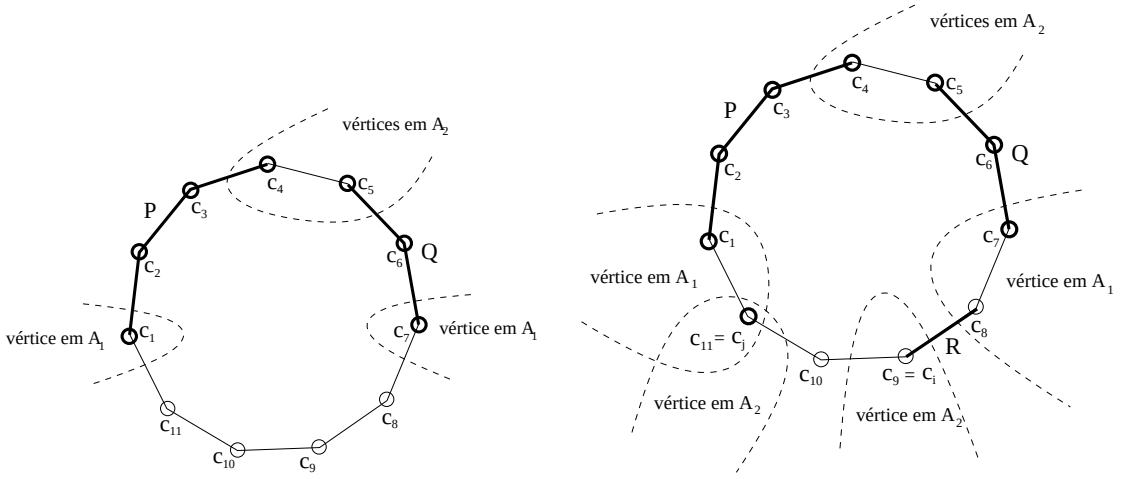
Afirmamos então que existe um (A_1, A_2)-gap par em G. Para provar esta afirmação veremos o que acontece no caso em que $A_1 \cap A_2$ é vazio e no caso em que $A_1 \cap A_2$ não é vazio. Primeiramente vamos considerar o caso em que $A_1 \cap A_2$ não é vazio. Seja o vértice $w \in A_1 \cap A_2$, da definição 3.3 segue que w é um (A_1, A_2)-gap par (de tamanho 0). Se $A_1 \cap A_2$ é vazio então existe um (A_1, A_2)-gap Q par pelo teorema 3.5. Pelo teorema 3.6 podemos assumir que $Q = c_{r+1}, ..., c_s$ para algum s ímpar com $r + 1 \leq s \leq 2n + 1$. Para facilitar a visualização da situação a figura 3.4(a) apresenta um exemplo em que temos o (A, B)-gap ímpar P e o (A, B)-gap par Q. Neste exemplo temos $r = 4$ e $s = 7$.

Como $A_1 \geq 4$, e apenas dois vértices de $P \cup Q$ estão em A_1 (pela definição de gaps) então temos $s \leq 2n - 1$. Agora considere o caminho $R = c_{s+2}$–...–c_{2n}. No exemplo da figura 3.4(a) este caminho seria c_9–c_{10} (para não tornar o desenho confuso, R não está indicado nesta figura). Se vértices de ambos os conjuntos A_1 e A_2 estão em R, então existe um (A_1, A_2)-gap contido em R. Pelo teorema 3.6 este (A_1, A_2)-gap não pode ser par, pois não há arestas entre ele e P e não pode ser ímpar, pois não pode haver arestas entre ele e Q. Portanto, podemos assumir que nenhum dos vértices $c_{s+2}, ..., c_{2n}$ pertence a um dos dois conjuntos A_1 ou A_2. Pela simetria entre A_1 e A_2, podemos assumir sem perder generalidade que nenhum dos vértices $c_{s+2}, ..., c_{2n}$ pertence a A_1. Novamente, como $A_1 \geq 4$, temos que $c_{s+1}, c_{2n+1} \in A_1$. Voltando ao nosso exemplo da figura 3.4(a), temos $c_8, c_{11} \in A_1$ e $c_9, c_{10} \notin A_1$.

Escolha i, j com $s + 1 \leq i \leq j \leq 2n + 1$ mínimo e máximo respectivamente tal que $c_i, c_j \in A_2$. Como $A_2 \geq 4$ temos que i deve ser diferente de j e portanto temos $s + 1 \leq i < j \leq 2n + 1$. Na figura 3.4(b), o atualizamos o exemplo que vínhamos

acompanhado, com as novas inclusões no conjunto A_1 e A_2 e a indicação dos vértices c_i e c_j.

Agora temos que c_{s+1}–...–c_i é um (A_1, A_2)-gap, e como não há arestas entre ele e P, ele é ímpar, ou seja i é ímpar. Similarmente c_j–...–c_{2n+1} é um (A_1, A_2)-gap par e j é ímpar. Mas como $j - i$ é par c_{s+1}–...–c_i e c_j–...–c_{2n+1} são (A_1, A_2)-gaps ímpar e par respectivamente, violando o teorema 3.6. □



(a) Um exemplo em que $r = 4$ e $s = 7$ no buraco ímpar C da demonstração do teorema 3.7.

(b) Atualizando o desenho da figura 3.4(a).

Figura 3.4: Desenhos auxiliando a interpretação do teorema 3.7.

3.1.2 Conjuntos Anticonectados de vértices *major*

O objetivo agora é refinar um pouco o resultado que acabamos de obter. Veremos a seguir (apenas enunciaremos um teorema sem fornecer a demonstração, que é um tanto extensa) que se impedirmos que algumas estruturas apareçam no grafo, o conjunto X do teorema 3.7 não precisa ser independente para que o resultado seja válido, basta que ele seja anticonectado. Vamos então a uma definição:

Definição 3.7 (Conjunto bem-comportado) *Seja C um buraco ímpar em G e X um conjunto anticonectado de vértices major. Dizemos que X é bem comportado se o conjunto de todos os vértices X-completos em C é normal.*

Observe que se um conjunto X de vértices *major* é independente (lembrando que se um conjunto é independente então ele é anticonectado) então X é bem comportado (pelo teorema 3.7). Entretanto nem todo conjunto de vértices *major* são bem comportados.

A questão é que se eliminarmos algumas estruturas de um grafo podemos obter conjuntos anticonectados de vértices *major*. É exatamente isto que o próximo (importante)

teorema, que é a base do algoritmo de limpeza de grafos, expressa. A prova é um tanto extensa e será omitida aqui.

Teorema 3.8 (Chudnovsky, Cornuéjols e outros [7], 2003) *Seja um grafo G não contendo pirâmides e nenhuma das estruturas T_1, T_2, T_3 e tal que ambos G e $\overline{G}$ não contenham Jóias. Então todo conjunto anticonectado de vértices major é bem-comportado.*

Uma versão menos restritiva deste teorema, que foi utilizada em [8], que foi a primeira versão "preprint" dos artigos de perfeição (e que provavelmente não será publicada), é a seguinte:

Teorema 3.9 (Chudnovsky, Cornuéjols e outros [8], 2003) *Seja um grafo G não contendo nenhuma das estruturas T_1, T_2, T_3 e tal que ambos G e $\overline{G}$ não contenham Jóias. Seja C um buraco ímpar mínimo de G sem raquetes. Então todo conjunto anticonectado de vértices major é bem-comportado.*

A questão é que em um dos dois algoritmos (o algoritmo da seção 4.1) de teste de perfeição do próximo capítulo, não é necessário testar a existência de pirâmides. Portanto, esta segunda versão menos restritiva (observe que se um grafo não contém pirâmides então ele não contém raquetes para um buraco mínimo ímpar) do teorema é suficiente.

3.2 O algoritmo de limpeza de grafos

Como já mencionamos, o objetivo do algoritmo de limpeza de grafos é encontrar um conjunto *quasi-cleaner*. Veremos que isso será possível quando o grafo possuir um buraco ímpar mínimo com certas propridades. Definiremos agora como este buraco ímpar mínimo deve ser.

Definição 3.8 (Buraco ímpar mínimo amenable) *Um buraco ímpar mínimo C é amenable se $|V(C)| \geq 7$ e se para todo conjunto anticonectado X de C-majors existe uma aresta X-completa em C.*

Veremos agora que um grafo G não contendo pirâmides e nenhuma das estruturas T_1, T_2, T_3 e tal que ambos G e $\overline{G}$ não contenham jóias, possui um buraco ímpar mínimo *amenable*.

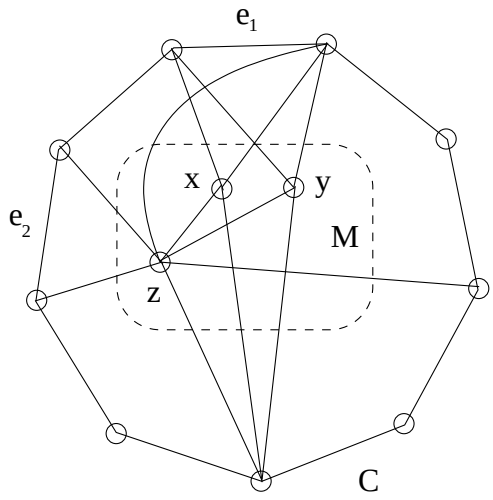
Fato 3.1 *Seja C um ciclo ímpar. Seja $A \subseteq V(C)$ normal em C. Então C contém pelo menos uma A-aresta.*

Prova: Conseqüência direta do teorema 3.3. □

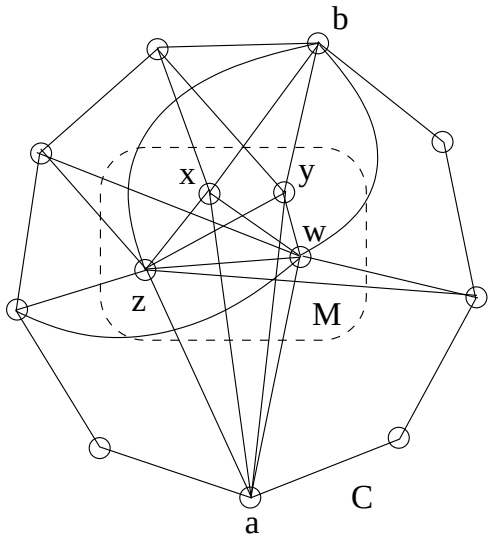
Teorema 3.10 *Seja um grafo G não contendo pirâmides e nenhuma das estruturas T_1, T_2, T_3 e tal que ambos G e $\overline{G}$ não contenham Jóias. Então todo buraco mínimo ímpar é amenable.*

PROVA: Se G não contém buracos ímpar não há o que provar. Seja C um buraco ímpar mínimo. Como G não contém estruturas T_1 o tamanho de C é maior ou igual a 7. Sejam $M_1, M_2, ..., M_k$ os anticomponentes do conjunto dos vértices C-major e $A_1, A_2, ..., A_k$ vértices em C, tal que A_i é M_i-completo. Como M_i é bem comportado (teorema 3.8), A_i é normal (definição de conjunto bem comportado). Como A_i é normal então existe uma A_i-aresta a_ib_i em C (fato 3.1). Como a_i e b_i são M_i-completos, então a aresta a_ib_i é M_i-completa. Portanto C é *amenable*. □

Um exemplo é apresentado na figura 3.5(a). No exemplo da figura, o grafo contém um buraco C amenable de tamanho 9. O conjunto M dos vértices C-major possui dois anticomponentes: um deles contendo os vértices x, y e o outro contendo apenas o vértice z. Observe que a aresta e_1 é $\{x, y\}$-completa e a aresta e_2 é $\{z\}$-completa.



(a) Os anticomponentes $\{x, y\}$ e z e as arestas e_1 e e_2, que são $\{x, y\}$-completa e $\{z\}$-completa respectivamente.



(b) A distância de a e b em C é maior ou igual a 3 e ambos são $\{x, z\}$-completos.

Figura 3.5: Um buraco ímpar amenable C e os anticomponentes de vértices C-major.

O objetivo do algoritmo de limpeza é gerar uma família F de tamanho polinomial de subconjuntos de vértices de $V(G)$ tal que algum destes conjuntos seja um *quasi-cleaner*. A partir do fato 3.10 temos a seguinte situação. Para um buraco amenable C, caso ocorra que o conjunto de vértices C-major M (assumindo M não vazio) seja anticonectado podemos gerar os conjuntos X_i da família F da seguinte maneira: para cada aresta $a_i \in E(G)$, entre, digamos, os vértices a e b, o conjunto X_i é constituído de todos os vértices $\{a, b\}$-completos. Quando a aresta M-completa e_k de C for escolhida (a aresta existe, pois C é *amenable*), o conjunto X_k conterá todos os C-majors.

Entretanto, existem casos em que o conjunto M de C-majors não é anticonectado. O grafo da figura 3.5(a) é um exemplo. Para estes casos veremos que é possível encontrar um *quasi-cleaner*.

Para isso, inicialmente considere o conjunto $X \subseteq M$. Pode ocorrer ou não que existam dois vértices X-completos a, b em C tal que a distância entre a e b em C é maior ou igual a 3. No caso em que $|X| = 1$, como $x \in X$ é C-major, então existem tais vértices. De maneira geral, quando X é pequeno, é mais provável que existam os vértices a e b. Por outro lado, na medida que X cresce é menos provável que existam estes dois vértices.

Para ilustrar a situação, considere o exemplo da figura 3.5(b) (que apresenta o grafo da figura 3.5(a) acrescido do vértice C-major w e de algumas arestas ligando w a outros vértices). Neste caso, observe que no grafo os subconjuntos anticonectados de M são $\{x, y\}$, $\{z\}$ e $\{w\}$. Se tomarmos, por exemplo, $X = \{x, z\}$, teremos os vértices a e b como indicado na figura.

Voltando ao caso geral, tanto no caso onde exista o par de vértices a, b, quanto no caso onde não exista tal par de vértices temos situações favoráveis. No caso em que existe a e b (como no caso da figura 3.5(b)), temos que todos elementos de $N(a, b)$ (lembrando que $N(a, b) = N(a) \cap N(b)$, ou seja, o conjunto dos vértices $\{a, b\}$-completos) são majors, pois C é mínimo, e este conjunto inclui X, pois a e b são X-completos. Com isso, uma maneira de obter um conjunto que contenha X é enumerar todo par a, b e obter $N(a, b)$. Por outro lado, se não existem a e b a intersecção do conjunto dos vértices X-completos A com o conjunto de vértices de C é bastante limitada (estão contidos em um 2-caminho de C), e ainda assim A contém muitos vértices de M (os únicos vértices de M que não estão em A são aqueles que estão em anticomponentes de M que tem intersecção com X). Portanto, se pudermos encontrar X, podemos encontrar muitos vértices de M. Considere um conjunto X de tamanho máximo tal que exista a e b, ou seja a insersão de um vértice C-major m_1 qualquer em X faz com que não existam a e b para $X \cup m_1$. Neste caso podemos obter muitos vértices de M tomando X e obtendo $N(a, b)$ e em seguida fazer a união de $N(a, b)$ com os vértices $(X \cup \{m_1\})$-completos e ainda assim, a intersecção deste conjunto resultante com C estaria contida em um 2-caminho de C.

Na realidade, indo um pouco adiante e não apenas maximizando X, mas maximizando lexicograficamente (como veremos a seguir) X, podemos obter deste processo que acabamos de descrever um *quasi-cleaner*.

Definição 3.9 (tripla relevante) *Uma tripla (a, b, c) é relevante se a e b são distintos e não adjacentes e $c \notin N(a, b)$ (possivelmente $c \in \{a, b\}$.*

Para cada tripla relevante (a, b, c) definimos:

Definição 3.10 ($r(a, b, c)$) *A cardinalidade do maior anticomponente de $N(a, b)$ que contém um vértice não adjacente a c. Se c é $N(a, b)$-completo então $r(a, b, c) = 0$.*

Definição 3.11 ($Y(a, b, c)$) *União dos anticomponente de $N(a, b)$ com cardinalidade estritamente maior que $r(a, b, c)$.*

Definição 3.12 ($W(a, b, c)$) *É o anticomponente de $N(a, b) \cup \{c\}$ que contém c.*

DEFINIÇÃO 3.13 ($Z(a,b,c)$) *É o conjunto dos vértices* $\{Y(a,b,c) \cup W(a,b,c)\}$*-completos.*

DEFINIÇÃO 3.14 ($X(a,b,c)$) *É a união* $Y(a,b,c) \cup Z(a,b,c)$.

Usaremos estas definições no seguinte teorema:

TEOREMA 3.11 (CHUDNOVSKY, CORNUÉJOLS E OUTROS [7], 2003) *Seja* C *um buraco ímpar mínimo em* G *de tamanho maior ou igual a 7. Então existe uma tripla relevante de vértices* (a,b,c) *tal que:*

1. *O conjunto de todos o vértices* C*-major que não estão em* $X(a,b,c)$ *é anticonectado*
2. $X(a,b,c) \cap V(C)$ *está contido em algum 2-caminho de* C

ALGORITMO 3.1 (LIMPANDO GRAFOS – CHUDNOVSKY, CORNUÉJOLS E OUTROS [8], 2003)

- Entrada: grafo G não contendo pirâmides, estruturas do tipo T_1, T_2 ou T_3 e tal que ambos G e $\overline{G}$ não contenham jóias.
- Uma família F com $O(|V(G)|^5)$ subconjuntos de $V(G)$, tal que se C é um buraco ímpar mínimo *amenable* de G, então um dos subconjuntos é um *quasi-cleaner* para C.

Para todo par de vértices adjacentes u, v, encontre $N(u,v)$, e guarde todos estes conjuntos em uma lista. Para cada tripla relevante (a,b,c), compute o conjunto $X(a,b,c)$ e guarde cada um destes conjuntos em uma lista. Agora, para cada conjunto da primeira lista de conjuntos e para cada conjunto da segunda lista, faça a união dos dois conjuntos e insira o conjunto resultante na família F. □

TEOREMA 3.12 *O algoritmo 3.1 é correto.*

PROVA: Suponha que C é um buraco mínimo ímpar *amenable* de G. Pelo teorema 3.11, existe uma tripla relevante (a,b,c) satisfazendo o teorema. Como o conjunto, digamos, T de todos os vértices C-major que não estão em T é anticonectado, e C é *amenable*, existe uma aresta uv de C que é T completa, e portanto $T \subseteq N(u,v)$. Mas $N(u,v) \cup X(a,b,c)$ é um *quasi-cleaner* para C, e é um dos conjuntos da família de saída. □

Capítulo 4

Testando Perfeição

Apresentaremos agora os dois algoritmos que respondem se um grafo contém um buraco ímpar mínimo. Na seção 4.1 é apresentado o algoritmo proposto por Cornuéjols, Liu e Vuskovic. Na seção 4.2 é apresentado o algoritmo proposto por Seymour e Chudnovsky. Os dois algoritmos não são capazes de identificar a presença de um buraco ímpar mínimo em um grafo qualquer, entretanto, utilizando-se os conjuntos *quasi-cleaner* obtidos no capítulo anterior essa identificação de buracos ímpar é possível.

O algoritmo de Seymour e Chudnovsky é capaz de identificar se um grafo G contém buracos ímpar se para algum buraco ímpar mínimo C de G, G não possua C-majors (neste caso diremos que G é "limpo"). Se possuirmos um conjunto *cleaner* para G, basta eliminar os vértices de G que estão neste conjunto e teremos G limpo. O leitor deve recordar, que no final do capítulo passado fomos capazes de encontrar um conjunto *quasi-cleaner*, digamos F, de G (e não um conjunto *cleaner*). Entretanto, é muito fácil obter uma família de tamanho polinomial de subconjuntos de F, tal que um destes subconjuntos seja um *cleaner*. Basta obter todos os subconjuntos de F com cardinalidade $|F| - 3$ [1].

No caso do Algoritmo de Cornuéjols, Liu e Vuskovic temos um caso semelhante, entretanto, para este algoritmo não bastará que o o grafo esteja limpo, mas sim "super-limpo" (o buraco ímpar mínimo sem *majors* ainda deve possuir outras características.). Este algoritmo é o assunto da seção a seguir.

4.1 Algoritmo de Cornuéjols, Liu e Vuskovic

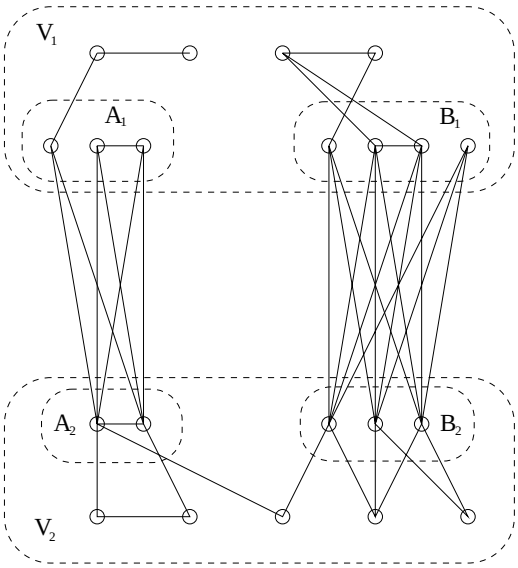
O algoritmo que apresentaremos nesta seção é baseado em um teorema de decomposição para grafos sem buracos ímpar que os criadores do algoritmo provaram em [17]. Antes de apresentar o teorema vamos a algumas definições.

Definição 4.1 (2-join) *Um grafo G possui um 2-join $V_1|V_2$ com os conjuntos especiais (A_1, A_2, B_1, B_2) se $V(G)$ pode ser particionado nos conjuntos V_1 e V_2 tal que para $i = 1, 2$, V_i contém conjuntos A_i e B_i disjuntos e não vazios, e tal que todo vértice de A_1 é adjacente*

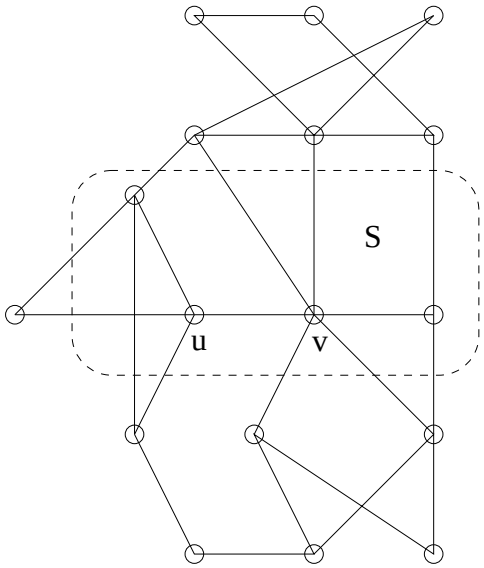
[1] Desprezando aqui detalhes, como o caso onde $|F| \leq 3$.

a todo vértice de A_2 e todo vértice de B_1 é adjacente a todo vértice de B_2 e não há outras adjacencias entre V_1 e V_2. Além disso, para $i = 1, 2$, $|V_i| > 2$ e o grafo induzido por V_i não é um grafo caminho.

Um exemplo de um grafo contendo um 2-join é apresentado na figura 4.1(a).



(a) Um grafo contendo um 2-join $V_1|V_2$ com os conjuntos especiais A_1, A_2, B_1 e B_2.



(b) A estrela dupla S centrada em uv é um conjunto de corte

Figura 4.1: Decomposição de grafos que não contém buracos ímpar.

Definição 4.2 (Estrela Dupla) *Um conjunto S de vértices é uma estrela dupla se S contém dois vértices adjacentes u e v tal que $S \subseteq N(u) \cup N(v)$. Dizemos que S é centrado em uv.*

Estaremos interessados nos casos em que uma estrela dupla é um conjunto de corte. Um exemplo de um grafo contendo uma estrela dupla que é um conjunto de corte é apresentado na figura 4.1(b).

Agora o teorema de decomposição:

Teorema 4.1 *[Cornuéjols e outros [18], 2003] Se um grafo G não contém buraco ímpar então G se enquadra em um dos três casos:*

i. G é um grafo básico.

ii. G contém um estrela-dupla como conjunto de corte.

iii. G contém um 2-join.

A idéia geral do algoritmo que veremos nessa seção baseia-se no teorema acima da seguinte maneira. Quando um grafo G possui um 2-join, podemos utilizar este 2-join para

obter dois subgrafos induzidos G_1, G_2 de G de tal maneira que G_1 e G_2 não contêm buracos ímpar se, e somente se, G não contém buracos ímpar (veremos isso mais adiante). Se tivessemos a situação de que quando um grafo G contivesse uma dupla estrela como conjunto de corte S pudessemos utilizar S para decompor G em subgrafos induzidos de maneira G não contém buraco ímpar se, e somente se, os subgrafos obtidos não contêm buracos ímpar (infelizmente não se conhece tal decomposição), poderíamos utilizar o seguinte algoritmo para testar se o grafo possui um buraco ímpar.

Dado um grafo de entrada G, teste se G contêm um 2-join ou uma estrela dupla. Se G possui alguma destas estruturas, utilize uma delas para decompor G da maneira que dissemos acima. Aplique recursivamente este processo aos subgrafos gerados até que nenhum subgrafo possui 2-joins ou estrelas duplas como conjunto de corte. Pelo teorema 4.1, G não tem buracos ímpar se, e somente se, todos estes subgrafos são básicos. O que é mais importante é que é possível encontrar estrelas duplas e 2-joins em tempo polinomial.

Mas como já mencionamos, não se conhece uma decomposição utilizado estrelas duplas como sugerimos acima. Entretanto, veremos que para grafos super-limpos (que serão definidos adiante), é possível utilizar a decomposição utilizando estrelas duplas. No algoritmo do capítulo 3, mencionamos que não é necessário eliminar as pirâmides, e para isso, assume-se que o grafo de entrada não possui raquetes. Veremos na seção que segue como lidar com isso. Na seqüência, na seção 4.1.2 apresentaremos as decomposições que podem ser aplicadas aos grafos limpos.

4.1.1 Construindo Grafos Super-limpos

Seja G um grafo e H um buraco ímpar em G. Seguem agora algumas definições:

Definição 4.3 (Vértice Limpo) *Se um vértice $v \in V(G) \setminus V(H)$ não é H-major dizemos que v é limpo para H.*

Definição 4.4 (Tipos de Vértices Limpos) *Dado um vértice limpo v, dizemos que v é do tipo c_1 se v tem um vizinho em H. Dizemos v é do tipo c_2 se v tem dois vizinhos v_1 e v_2 em H e $v_1v_2 \in E(G)$. Dizemos que v é do tipo c_3 em dois casos:*

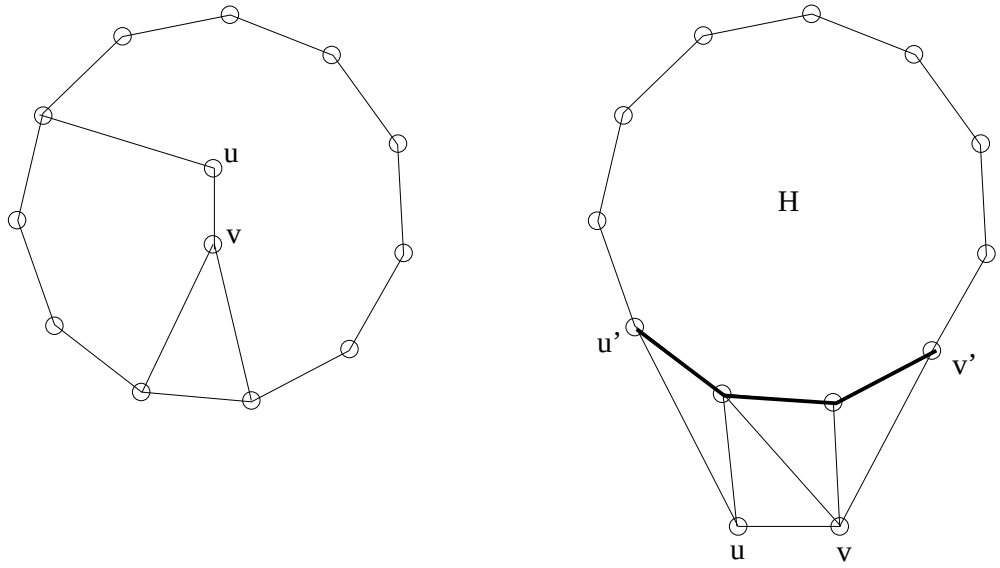
i. Se v tem dois vizinhos v_1 e v_2 em H e a distância entre v_1 e v_2 em H é 2.

ii. Se v tem três vizinhos contidos em um 2-caminho de H.

Definição 4.5 (Raquete Longa) *Uma raquete longa para H é uma aresta $uv \in E(G)$, tal que u é um vértice limpo do tipo c_2 para H, v é um vértice limpo do tipo c_1 para H e cada subcaminho de H entre os vizinhos de u e v tem tamanho pelo menos 3.*

Um exemplo de um grafo contendo uma raquete longa é apresentado na figura 4.2(a).

Definição 4.6 (Tenda) *Uma tenda para H é uma aresta uv tal que ambos u e v são vértices limpos para H, e para algum subcaminho P de H com tamanho 3 e extremidades*



(a) Raquete Longa (b) Tenda

Figura 4.2: (a) Uma raquete longa uv. (b) Uma tenda uv.

u' e v', temos $(N(u) \cup N(v)) \cap V(H) \subseteq V(P)$, e u é adjacente a u' e v é adjacente a v'. Note que u não é adjacente a v' e v não é adjacente u'

Um exemplo de um grafo contendo uma tenda é apresentado na figura 4.2(b).

Definição 4.7 (Substituição por vértice) *Dado um vértice limpo do tipo c_3 com os vizinhos contidos em um subcaminho v_1–v_2–v_3 de H. Dizemos que o buraco H' induzido por $(V(H) \setminus \{v_2\}) \cup \{u\}$ é obtido a partir de H por substituição por vértice.*

Um exemplo da operação de substituição por vértice em um buraco H é apresentado na figura 4.3.

Definição 4.8 (Substituição por tenda) *Considerando a tenda uv da definição 4.6, dizemos que o subgrafo H' induzido por $(V(H) \setminus V(P)) \cup \{u, v, u', v'\}$ é obtido a partir de H por substituição por tenda.*

Um exemplo da operação de substituição por tenda em um buraco H é apresentado na figura 4.3.

Definição 4.9 ($C_G(H)$) *Definimos por $\mathcal{C}_G(H)$ a família de buracos de G obtidos de H por meio de sequências de substituições por vértices e substituições por arestas.*

Definição 4.10 (Buracos Limpos e Super-limpos) *Um buraco H em um grafo G é limpo se não existem vértices $v \in V(G) \setminus V(H)$, tal que v é H-major. Dizemos que H é super-limpo se todos os buracos de $\mathcal{C}_G(H)$ são limpos.*

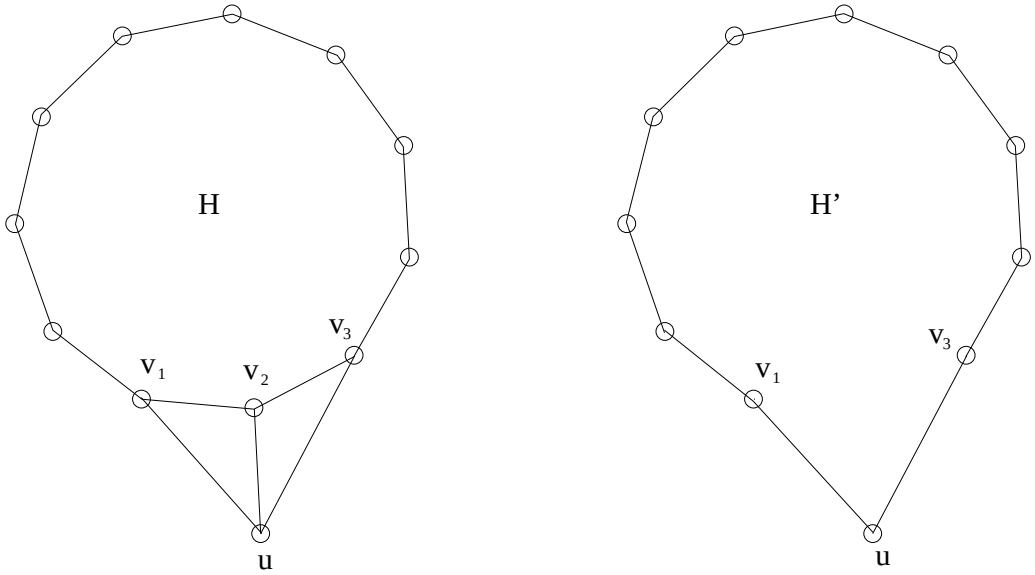


Figura 4.3: Substituição por vértice.

Definição 4.11 (Grafos Limpos) *Um grafo G é limpo se G não contém buracos ímpar ou se G contém um buraco ímpar mínimo limpo.*

Definição 4.12 (Grafos Super-Limpos) *Um grafo G é limpo se G não contém buracos ímpar ou se G contém um buraco ímpar mínimo super-limpo.*

Definição 4.13 (Grafos Filtrados) *Um grafo G é filtrado se G não contém nenhuma estrutura do tipo T_1, T_2 e T_3, e também não contém jóias, rodas pequenas, pirâmides pequenas e C_7 induzidos. Além disso $\overline{G}$ não contém jóias.*

Definição 4.14 ($\mathcal{R}(G, H, u)$) *Seja u uma raquete para um buraco mínimo H de G. Seja $P_1 = x_0$–x_1–x_2–x_3 e $P_2 = y_0$–y_1–y_2–y_3–y_4 dois subcaminhos de H tal que u é adjacente a x_1, x_2 e y_2. Definimos o grafo $\mathcal{R}(G, H, u)$ como o subgrafo de G induzido por $V(G) \setminus [\bigcup_{i=1}^{2} N(x_i) \cup \bigcup_{i=1}^{3} N(y_i) \cup N(u)) \setminus V(P_1) \cup V(P_2))]$*

Um exemplo da aplicação da operação $\mathcal{R}$ a um grafo é apresentado na figura 4.5. Observe que neste exemplo o vértice major m foi removido com a aplicação da operação $\mathcal{R}$.

Definição 4.15 ($\mathcal{R}_l(G, H, u)$) *Seja u uma raquete longa para um buraco mínimo H de G. Seja $P_1 = x_0$–x_1–x_2–x_3 e $P_2 = y_0$–y_1–y_2–y_3–y_4 dois subcaminhos de H tal que u é adjacente a x_1, x_2 e v é adjacente a y_2. Definimos o grafo $\mathcal{R}(G, H, u)$ como o subgrafo de G induzido por $V(G) \setminus [\bigcup_{i=1}^{2} N(x_i) \cup \bigcup_{i=1}^{3} N(y_i) \cup N(u)) \setminus V(P_1) \cup V(P_2))]$*

Um exemplo da aplicação da operação $\mathcal{R}_l$ a um grafo é apresentado na figura 4.6. Observe que neste exemplo o vértice major m também foi removido com a aplicação da operação $\mathcal{R}_l$.

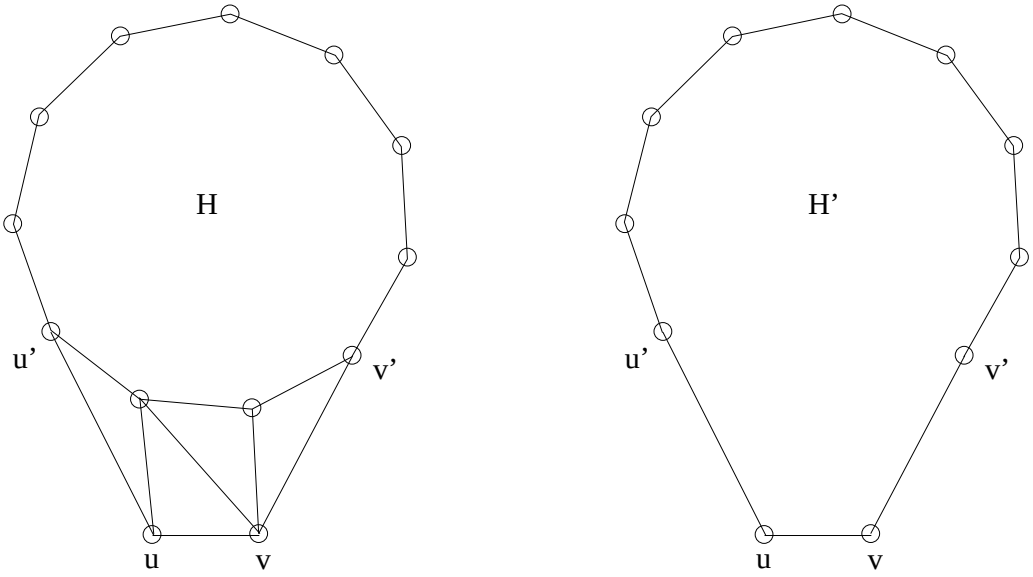


Figura 4.4: Substituição por tenda.

TEOREMA 4.2 *[Cornuéjols e outros [18], 2003] Suponha que um grafo filtrado G contém um buraco ímpar mínimo H e uma raquete u para H. Seja $G' = \mathcal{R}(G, H, u)$. Então:*

i. G' contém H e H é super-limpo em G'.

ii. Todo buraco de $\mathcal{C}_{G'}(H)$ tem um par de vértices u, v, tal que a distância entre u e v é pelo menos 4 em G'.

TEOREMA 4.3 *[Cornuéjols e outros [18], 2003] Suponha que um grafo filtrado G contém um buraco ímpar mínimo H. Se H é limpo e $\mathcal{C}_G(H)$ não tem raquetes, então H é super-limpo.*

TEOREMA 4.4 *[Cornuéjols e outros [18], 2003] Suponha que um grafo filtrado G contém um buraco ímpar mínimo H sem raquetes. Suponha que exista uma raquete longa uv para H. Seja $G' = \mathcal{R}_l(G, H, u)$. Então:*

i. G' contém H e H é super-limpo em G'.

ii. Todo buraco de $\mathcal{C}_{G'}(H)$ tem um par de vértices u, v, tal que a distância entre u e v é pelo menos 4 em G'.

TEOREMA 4.5 *[Cornuéjols e outros [18], 2003] Suponha que um grafo filtrado G contém um buraco ímpar mínimo H tal que $\mathcal{C}_G(H)$ não tem raquetes. Sejam u, v, dois vértices adjacentes e limpos para H. Então um dois dois casos ocorre:*

i. uv é uma tenda ou uma raquete longa para H

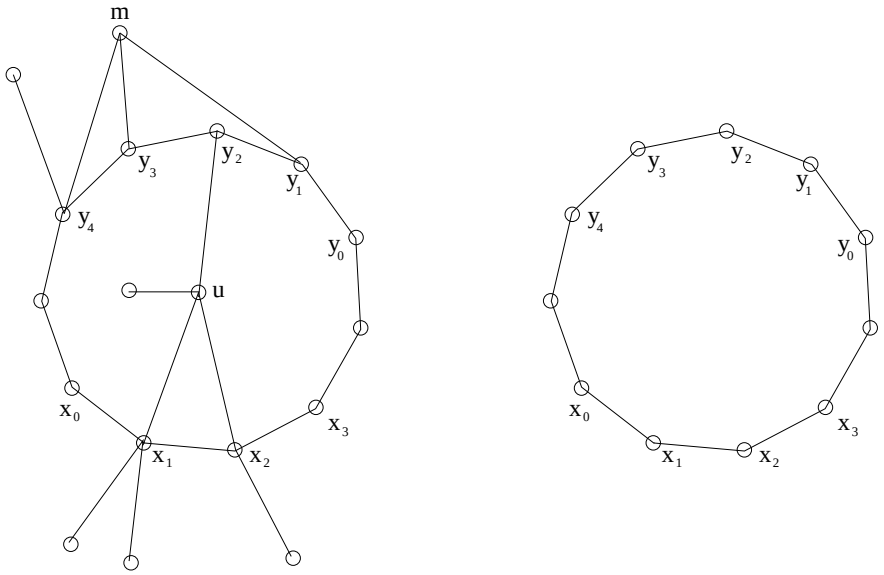


Figura 4.5: À esquerda um grafo G contendo um buraco H com uma raquete u. Os caminhos P_1 e P_2 são destacados com arestas espessas. À direita o grafo $G' = R(G, H, u)$. Observe que o vértice major m foi removido com a aplicação desta operação.

ii. Os vértices de $(N(u) \cup N(v)) \cap V(H)$ *são contidos em um subcaminho* P *de* H *de tamanho pelo menos 2, e se* P *tem tamanho 2, então* u *ou* v *são vértices limpos do tipo* c_3 *para* H*.*

Algoritmo 4.1 (Algoritmo de construção de grafo super-limpos)

- Entrada: grafo G filtrado.
- Saída: Uma família $\mathcal{F}$ com $O(|V(G)|^{10})$ subgrafos induzidos de G tal que, G não tem buracos ímpar se e somente se todos os grafos de F não contém buracos ímpar. Além disso, se G contém um buraco ímpar mínimo H, então para algum grafo $F \in \mathcal{F}$, F contém H e H é super-limpo em F, e todo buraco ímpar de $\mathcal{C}_F(H)$ tem um par de vértices a distância pelo menos 4 em F.

Passo 1: Faça $\mathcal{F} = G$.

Passo 2: Para cada P_1, P_2, u, onde $P_1 = x_0$–x_1–x_2–x_3 e $P_2 = y_0$–y_1–y_2–y_3–y_4 são dois caminhos induzidos, e $u \in N(x_1) \cap N(x_2)$, adicione a $\mathcal{F}$ o subgrafo de G induzido por $V(G) \setminus [\bigcup_{i=1}^{2} N(x_i) \cup \bigcup_{i=1}^{3} N(y_i) \cup N(u)) \setminus V(P_1) \cup V(P_2))]$. Observe que esta operação faz o papel das operações $\mathcal{R}$ e $\mathcal{R}_l$

Passo 3: Aplique o algoritmo 3.1 a G. Seja $\mathcal{X}$ denotar a família de saída do algoritmo. Para cada $X \in \mathcal{X}$ e cada $Y \subseteq X$ obtido pela remoção de pelo menos 3 vértices de X contidos em um 2-caminho, adicione o grafo induzido por $V(G) \setminus V(Y)$ em $\mathcal{F}$. □

Teorema 4.6 *[Cornuéjols e outros [18], 2003] O algoritmo 4.1 é correto.*

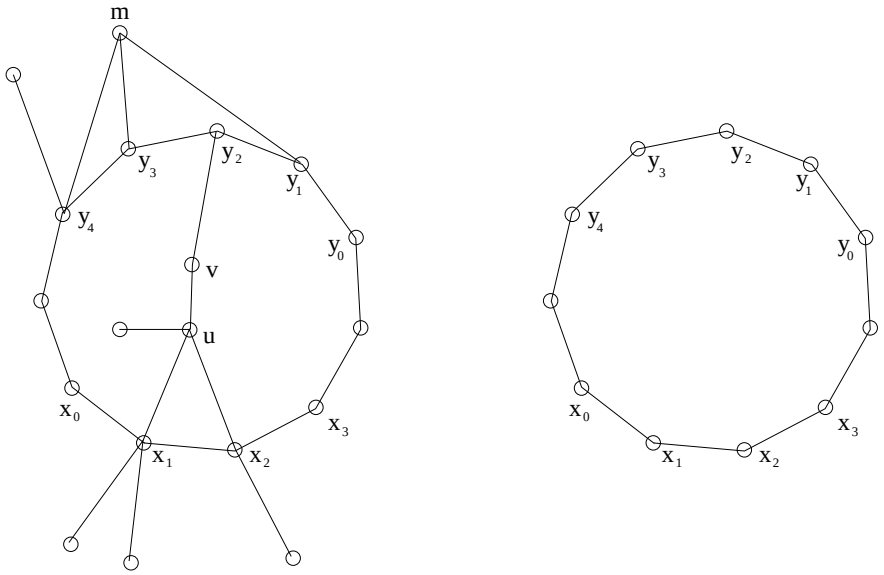


Figura 4.6: À esquerda um grafo G contendo um buraco H com uma raquete longa uv. Os caminhos P_1 e P_2 são destacados com arestas espessas. À direita o grafo $G' = R(G, H, u)$. Observe que o vértice major m foi removido com a aplicação desta operação.

Prova: Suponha que G contenha um buraco ímpar mímimo H. Observe que $\mathcal{C}_G(H)$ tem tamanho maior ou igual a 7. Se $\mathcal{C}_G(H)$ não é livre de raquete o algoritmo é correto pelo teorema 4.2 e pelo passo 2 do algoritmo. Se $\mathcal{C}_G(H)$ é livre de raquetes mas não é livre de raquetes longas, então o algoritmo é correto pelos teoremas 4.4 e 4.6 e pelo passo 2 do algoritmo. Finalmente, suponha que $\mathcal{C}_G(H)$ não é livre de raquetes e raquetes longas. O teorema 4.6 e o passo 3 implicam que o algoritmo é correto. Para ver isso, note que algum grafo F da família $\mathcal{F}$ de saída do algoritmo de limpeza (algoritmo 3.1) contém H e H é super-limpo. Seja I algum buraco de $\mathcal{C}_F(H)$. Como I tem tamanho estritamente maior do que 7, então I contém dois vértices x e y distantes pelo menos 4 em I. Suponha que exista um caminho P ligando x a y em F de tamanho menor que 4. Como I é limpo P não pode ter tamanho 2. Então P tem tamanho 3. Mas pelo teorema 4.5 há uma raquete longa em I, contradizendo a suposição que $\mathcal{C}_G(H)$ não tem raquetes longas. Portanto x e y estão a distância pelo menos 4 em F. □

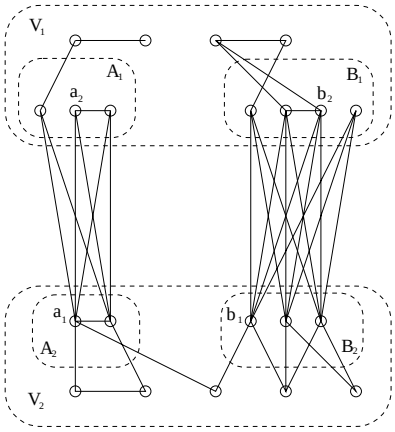
Agora que já sabemos como tornar um grafo super-limpo, vamos mostrar como decompor este grafo super-limpo utilizando a partir de 2-joins ou estrelas dupla como conjunto de corte que o grafo contenha.

4.1.2 Decompondo Grafos Super-limpos

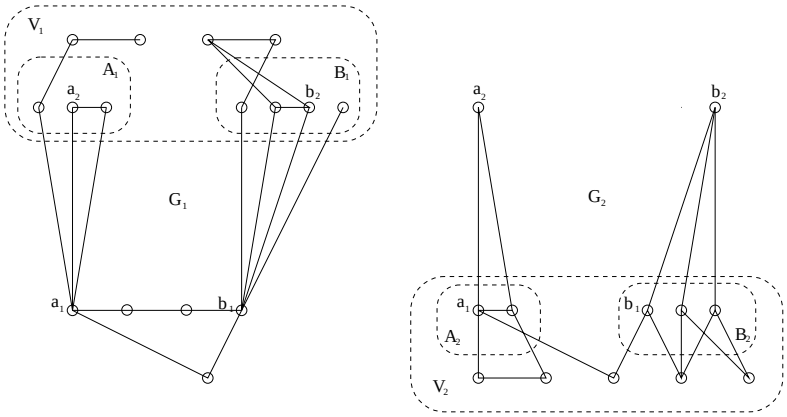
Definição 4.16 (Blocos de Decomposição por 2-join) *Seja $V_1|V_2$ um 2-join de um grafo G com os conjuntos especiais (A_1, A_2, B_1, B_2). Se não existe um caminho de um vértice de A_2 até um vértice de B_2 em $G[V_2]$, então definimos o bloco G_1 como o subgrafo de G induzido por $V_1 \cup \{a_2, b_2\}$, onde $a_2 \in A_2$ e $b_2 \in B_2$. Caso contrário, seja Q_2 um*

caminho mínimo ligando A_2 até B_2 em $G[V_2]$. Neste caso definimos o bloco G_1 como o grafo obtido de $G[V_1 \cup V(Q_2)]$ substituindo-se o caminho Q_2 por um grafo caminho M_2 de tamanho 4 se Q_2 tem tamanho par, e de tamanho 5 se Q_2 tem tamanho ímpar.

Um exemplo de decomposição por 2-join é apresentado na figura 4.7(a). Na figura 4.7(a) é apresentado um grafo G contendo um 2-join $V_1|V_2$. Na figura 4.7(b) são apresentados os grafos G_1 e G_2, que são os blocos de decomposição de G por 2-join.



(a) O grafo G



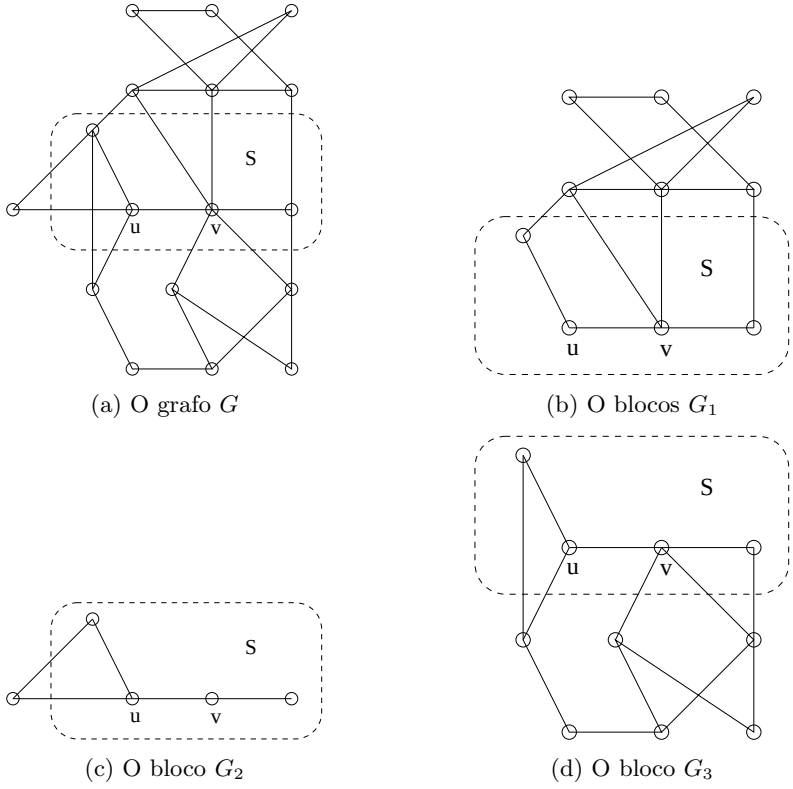
(b) O grafo G é decomposto nos blocos G_1 e G_2

Figura 4.7: (a) Um grafo G. (b) Os grafos G_1 e G_2 que são os blocos de decomposição de G por 2-join.

Teorema 4.7 *[Cornuéjols e outros [18], 2003] Sejam G_1 e G_2 dois blocos de uma decomposição por 2-join de G. Então G não tem buracos ímpar se, e somente se, G_1 e G_2 não contém buracos ímpar. Além disso, se G contém um buraco ímpar limpo maior que 5, então G_1 ou G_2 contém um buraco ímpar maior que 5.*

Definição 4.17 (Blocos de Decomposição por Estrela Dupla) *Seja uma estrela dupla S que é um conjunto de corte para G e C_1, C_2,..., C_n os componentes conexos do grafo induzido por $V(G) \setminus S$. Definimos os blocos de decomposição como os grafos G_1, G_2, ..., G_n, onde $G_i = G[V(C_i) \cup S]$*

Um exemplo de decomposição por estrela dupla é apresentado na figura 4.8.



(a) O grafo G (b) O blocos G_1

(c) O bloco G_2 (d) O bloco G_3

Figura 4.8: Decomposiçao do grafo G pela estrela dupla S.

Decomposição por Estrela Dupla

Seja uv uma raquete longa para um buraco ímpar H e $P_1 = x_0$–x_1–x_2–x_3 e $P_2 = y_1$–y_2–y_3 dois subcaminhos de H tal que u é adjacente a x_1, x_2 e v é adjacente a y_2 e y_1 está no subcaminho entre x_1 e y_2 de H que não contém x_2. Uma estrela dupla S' centrada em uv que seja um conjunto de corte pode destruir H. Para superar isso podemos detectar quando isso ocorre. Observe que quando S' destrói H então $N(u) \cup N(v)$ também é um conjunto de corte que é uma estrela dupla que destrói H. Seja Q_1 o subcaminho de H ligando x_0 a y_1 que não contém x_1 e Q_2 o subcaminho de H ligando x_3 a y_3 que não contém x_2. Com isso vamos definir agora um tipo de raquete longa que é possível ser detectada:

Definição 4.18 (Raquetes Longas Detectáveis) *Uma raquetes longa uv é detectável para H se $S = N(u) \cup N(v)$ é um conjunto de corte que é uma estrela dupla centrada em uv, tal que Q_1 e Q_2 (como definido acima) de $G - S$.*

Os autores do algoritmo dizem em [18] como detectar raquetes longas detectáveis, e obviamente se é encontrada tal estrutura o grafo não é perfeito.

TEOREMA 4.8 *[Cornuéjols e outros [18], 2003] Seja um grafo filtrado G que contém um buraco ímpar mínimo H. Se G é decomposto por uma estrela dupla S centrada em uv, então um dos dois casos ocorre:*

i. uv é uma raquete longa detectável para algum buraco ímpar.

ii. Algum buraco ímpar $H' \in \mathcal{C}_G(H)$ está contido inteiramente em um dos blocos de decomposição.

ALGORITMO 4.2 (*Decomposição por Estrelas Duplas – Cornuéjols e outros [18], 2003)*

- Entrada: um grafo filtrado G, tal que se G contém um buraco ímpar, então para um buraco ímpar mínimo H, todo buraco de $\mathcal{C}_G(H)$ contém um par de vértices u, v, tal que a distância entre u e v é 4.
- Saída: um buraco ímpar de G ou uma família $\mathcal{L}$ de subgrafos induzidos de G satisfazendo as seguintes propriedades:
  - i. G não tem buracos ímpar se, e somente se, todos os grafos de $\mathcal{L}$ não têm buracos ímpar.
  - ii. Os grafos de $\mathcal{L}$ não contém estrelas duplas como conjunto de corte.
  - iii. O número de grafos em $\mathcal{L}$ é $O(|V(G)|^2)$.

Passo 1: Faça $\mathcal{L} = \varnothing$ e $\mathcal{L}' = G$

Passo 2: Se $\mathcal{L}' = \varnothing$ pare a execução do algoritmo. Caso contrário, remova um grafo F de $\mathcal{L}'$. Se a distância de todo par de vértices de F for menor que 4 vá para o passo 2. Se F não tem estrelas duplas como conjunto de corte, então adicione F em $\mathcal{L}$ e vá para o passo 2. Caso contrário, seja um conjunto de corte que é uma estrela dupla centrada em uv em F. Verifique se uv é uma raquete longa detectável. Caso seja pare a execução do algoritmo. Caso contrário obtenha os blocos de decomposição e os adicione a $\mathcal{L}'$ e vá para o passo 2. □

TEOREMA 4.9 *[Cornuéjols e outros [18], 2003] O algoritmo 4.2 é correto.*

PROVA: Vamos verificar as três possíveis saídas do algoritmo ((i), (ii) e (iii)). A saída (ii) está correta pela construção do algoritmo. Vamos verificar que a saída (i) está correta. Como os subgrafos em $\mathcal{L}$ são subgrafos induzidos de G, se G não contém um buraco ímpar então nenhum grafo de $\mathcal{L}$ possui um buraco ímpar. Suponha que G contenha um buraco ímpar mínimo super-limpo H. Sabemos que todo buraco em $\mathcal{C}_G(H)$ contém dois vértices distantes pelo menos 4 em G. Portanto, pelo teorema 4.8, a saída é um buraco ímpar ou algum grafo em $\mathcal{L}$ contém um buraco ímpar de $\mathcal{C}_G(H)$. O saída (iii) é correta por que o número de grafos em $\mathcal{L}$ é limitado pelo número de pares de vértices cuja distâcia entre eles é pelo menos 4 em G. Para ver isso, seja S um conjunto de corte que é uma estrela dupla de um grafo F, e $F_1, ..., F_m$ os blocos de decomposição. Seja u e v dois vértices de

F que estão a uma distância maior ou igual a 4 em G (e portanto em F). Os vértices u e v não podem estar em blocos diferentes, pois neste caso ambos estariam em S e como S é uma estrela dupla, a distância de qualquer par de vértices em S é menor ou igual a 3. Portanto nenhum par de vértices separados por uma distância maior ou igual a 4 podem estar em diferentes grafos de $\mathcal{L}$. □

Decomposição por 2-join

TEOREMA 4.10 *[Cornuéjols e outros [18], 2003] Se um grafo G contém um 2-join $V_1|V_2$ com os conjuntos especiais (A_1, A_2, B_1, B_2) tal que $V_1 \setminus (A_1 \cup B_1) = \varnothing$ e $V_2 \setminus (A_2 \cup B_2) = \varnothing$ então G não contém um buraco limpo de tamanho maior que 5.*

TEOREMA 4.11 *[Cornuéjols e outros [18], 2003] Suponha que G contém um 2-join $V_1|V_2$ com os conjuntos especiais (A_1, A_2, B_1, B_2) tal que pelo menos um de $V_1 \setminus (A_1 \cup B_1)$ e $V_2 \setminus (A_2 \cup B_2)$ não é vazio. Sejam G_1 e G_2 os blocos de decomposição por este 2-join. Se G não contém uma estrela dupla como conjunto de corte então ocorre:*

i. Ambos $V_1 \setminus (A_1 \cup B_1)$ e $V_2 \setminus (A_2 \cup B_2)$ são não vazios.

ii. Para $i = 1, 2$, G_i não tem uma estrela dupla como conjunto de corte.

iii. Para $i = 1, 2$, $|V_i| \geq 6$.

ALGORITMO 4.3 (*Decomposição por 2-join – Cornuéjols e outros [18], 2003)*

- Entrada: um grafo super-limpo G não contendo estrelas duplas como conjunto de corte e não contendo buracos ímpar de tamanho 5.
- Saída: um buraco ímpar de G ou uma família $\mathcal{L}$ de subgrafos induzidos de G satisfazendo as três seguintes propriedades:
  - i. G não tem buracos ímpar se, e somente se, todos os grafos de $\mathcal{L}$ não tem buracos ímpar.
  - ii. Todo grafo $F \in \mathcal{L}$ não contém uma estrela dupla como conjunto de corte não contém um 2-join.
  - iii. O número de grafos em $\mathcal{L}$ é $O(|V(G)|)$.

Passo 1: Faça $\mathcal{L} = \varnothing$ e $\mathcal{L}' = G$

Passo 2: Se $\mathcal{L}' = \varnothing$ pare a execução do algoritmo. Caso contrário, remova um grafo F de $\mathcal{L}'$. Se F não contém um 2-join então adicione F em $\mathcal{L}$ e vá para o passo 2. Caso contrário, seja $V_1|V_2$ um 2-join de F com os conjuntos especiais (A_1, A_2, B_1, B_2). Se $V_1 \setminus (A_1 \cup B_1) = \varnothing$ e $V_2 \setminus (A_2 \cup B_2) = \varnothing$, desconsidere F e vá para o passo 2. Caso contrário, construa os blocos de decomposição F_1 e F_2 de F. Para $i = 1$ ou $i = 2$, se $|V_i| \leq 7$, verifique diretamente se V_i possui um buraco ímpar. Se F_i possui um buraco

ímpar retorne o buraco ímpar, caso contrário, desconsidere F_i. Se $|V_i| > 7$ adicione F_i a $\mathcal{L}'$ e vá para o passo 2. □

TEOREMA 4.12 *[Cornuéjols e outros [18], 2003] O algoritmo 4.4 é correto .*

PARTE DA PROVA: O ítem (i) da saída está correto pelos teoremas 4.7 e 4.10. O ítem (ii) está correto pela construção do algoritmo e pelo teorema 4.11 □

4.1.3 Teste de Perfeição de Cornuéjols, Liu e Vuskovic

ALGORITMO 4.4 (*Reconhecendo Grafos Super-limpos sem Buraco Ímpar – [18], 2003)*

- Entrada: Um grafo G filtrado super-limpo.
- Saída: SIM, se G não possui um buraco ímpar e NÃO, caso contrário.

Passo 1: Aplique o algoritmo de decomposição por dupla estrela. Se a saída do algoritmo é um buraco ímpar retorne NÃO. Caso contrário guarde a família de saída deste algoritmo em $\mathcal{L}_1$.

Passo 2: Faça $\mathcal{L}_2 = \varnothing$. Para cada grafo de $\mathcal{L}_1$ aplique o algoritmo de decomposição por 2-join. Se a saída do algoritmo é um buraco ímpar retorne NÃO. Caso contrário guarde a família de saída deste algoritmo em $\mathcal{L}_2$.

Passo3: Verifique se todo grafo de $\mathcal{L}_2$ é básico. Em caso afirmativo retorne SIM. Caso contrário retorne NÃO. □

Para testar a perfeição de um grafo G basta aplicar o algoritmo de limpeza e o algoritmo acima em G e em $\overline{G}$.

4.2 Algoritmo de Chudnovsky e Seymour

TEOREMA 4.13 *Seja G um grafo não contendo pirâmides ou jóias, e seja C um buraco mínimo ímpar limpo em G. Sejam u, v dois vértices distintos não adjacentes em C. Sejam L_1, L_2 dois subcaminhos de C ligando u e v, tal que $|E(L_1)| \leq |E(L_2)|$. Seja P qualquer caminho mínimo em G entre u e v. Então*

i. P, L_1 têm o mesmo tamanho.

ii. $P*$ é disjunto de L_2*, e não há nenhuma aresta ligando vértices de $P*$ a vértices de L_2*, e portanto u–P–v–L_2–u é um buraco, digamos, C'.

iii. C' é um buraco ímpar mínimo em G e é limpo.

ALGORITMO 4.5 (ALGORITMO PARA PROCURAR BURACOS LIMPOS MÍNIMOS)

- Entrada: grafo G sem pirâmides e sem jóias.
- Saída (duas possibilidades):
  i. G contém um buraco ímpar.
  ii. G não contém um buraco ímpar mínimo limpo.

Para cada par de vértices u, v, encontre o menor caminho $P(u, v)$ entre u e v. Para cada 3-tupla u, v, w encontre os caminhos mínimos $P(u, v)$, $P(v, w)$ e $P(w, u)$ teste se a união destes caminhos forma um buraco ímpar H. Se existir o buraco ímpar H retorne que G contém um buraco ímpar. Se após examinar todas 3-tuplas não se encontrar um buraco ímpar formado por estes caminhos, retorne que G não contém um buraco ímpar mínimo limpo. □

TEOREMA 4.14 *[Chudnovsky e Seymour [11], 2003] O algoritmo 4.5 é correto.*

PROVA: Pela contrução do algoritmo, quando a saída do algoritmo é (i), a resposta é obvimente correta. Vamos mostrar que quando a saída é (ii) a saída é sempre correta também, ou seja a saída do algoritmo não é (ii) quando o grafo contém um buraco ímpar mínimo ímpar.

Para mostrarmos isso, assuma que o grafo de entrada G contém um buraco ímpar mínimo limpo C. Escolha os vértices u, v, w em C de maneira razoavelmente espalhada em C. Mais precisamente, escolha u, v, w tal que cada componente de $C \setminus \{u, v, w\}$ contenha no máximo $n-1$ vértices, sendo que C tem tamanho $2n+1$. O algoritmo vai encontrar o menor caminho $P(u, v)$ entre u e v. Agora seja L_1 o caminho em C ligando u a v sem passar por w e L_2 o caminho em C ligando u a v passando por w. Então L_1 tem tamanho menor ou igual a n pela escolha de u, v, w e portanto, pelo teorema 4.13, temos que L_1 e $P(u, v)$ têm o mesmo tamanho, e que o o buraco ímpar mínimo C_1, obtido substituindo-se L_1 por P_1 (observe que P_1 e L_1 podem ser o mesmo caminho, mas isso não é problema), pelo teorema 4.13 também é limpo. Repetindo o processo para para os outros pares de $\{u, v, w\}$, a partir do buraco C_1 obtemos um buraco ímpar mínimo limpo construído pelos caminhos $P(u, v)$, $P(v, w)$ e $P(w, u)$. □

ALGORITMO 4.6 (*Teste de Perfeição de Chudnovsky e Seymour [11], 2003)*

- Entrada: Um grafo G sem pirâmides e sem jóias.
- Saída: SIM se G é perfeito e NÃO se G não é perfeito.

Aplique o algoritmo de limpeza 3.1 sobre o grafo de entrada. Seja $\mathcal{X}$ a família de saída do algoritmo. Para cada $X \in \mathcal{X}$, enumere todos os subconjuntos $Y \subseteq X$ com $|Y| \leq 3$, e aplique o algoritmo 4.5 ao subgrafo induzido por $V(G) \setminus (X \setminus Y)$. Se G tem um buraco

ímpar, então um destes grafos contém um buraco ímpar mínimo limpo que será detectado. Se não for encontrado um buraco ímpar em nenhum destes subgrafos induzidos, rode todo o algoritmo novamente no complemento de G. Se ambos G e $\overline{G}$ não tiverem um buraco ímpar, então retorne SIM. Caso contrário retorne NÃO. □

Os autores deste algoritmo, mostraram em [11] e em [7], que é possível realizar este mesmo teste em tempo $O(|V(G)|^8)$ (a complexidade de tempo do teste acima é $O(|V(G)|^{12})$) com algumas modifições no algoritmo que acabamos de apresentar, mas não apresentaremos esta versão aqui.

Capítulo 5

Subgrafos e Supergrafos Perfeitos

Neste capítulo vamos estudar algumas operações que podem ser aplicadas a um grafo para torná-lo perfeito. A idéia geral é associar a um grafo um parâmetro que exprima o grau de imperfeição do grafo. Se o número de operações que devem ser realizadas em um grafo para torná-lo perfeito é grande, de certa maneira podemos dizer que o grafo é bastante imperfeito. Por outro lado, se o número de operações a ser aplicada é pequeno, podemos dizer que o grafo é pouco imperfeito. No caso extremo, quando nenhuma operação é necessária, o grafo é perfeito.

Deve-se mencionar que os algoritmos de teste de perfeição podem ser utilizados para o cálculo destes parâmetros, assim como, por exemplo, os algoritmos de teste de planaridade [30] são a base de algoritmos que procuram por subgrafos perfeitos. Alguns pesquisadores já haviam apresentado resultados de complexidade computacional na linha de subgrafos perfeitos [35, 29], entretanto, a definição de parâmetros de perfeição associados a essas operações não foi encontrada na literatura. Além disso todos os resultados, exceto quando é feita a devida referência, deste capítulo foram obtidos pelo autor.

5.1 Operações para tornar um grafo perfeito

Vamos definir agora quatro parâmetros (ρ_1, ρ_2, ρ_3 e ρ_4), onde cada um deles está relacionado a uma operação que pode ser aplicada ao grafo para torná-lo perfeito.

A primeira operação considerada é a remoção de arestas.

Definição 5.1 (Subgrafo Perfeito Máximo) *Dado um grafo G, o tamanho do menor conjunto $A \subseteq E(G)$, tal que $G - A$ é um grafo perfeito, é denotado por $\rho_1(G)$.*

O objetivo agora é considerar a inserção de arestas.

Definição 5.2 (Supergrafo Perfeito Mínimo) *Dado um grafo G, o tamanho do menor conjunto $B = \{b_1, b_2, ..., b_k\}$, onde $b_i \notin E(G)$ para $(1 \leq i \leq k)$, tal que $G + B$ é um grafo perfeito, é denotado por $\rho_2(G)$.*

Na próxima definição, permitimos tanto remoção quanto inserção de arestas.

DEFINIÇÃO 5.3 (SUBGRAFO PERFEITO MAIS PRÓXIMO) *Dado um grafo G, denotamos de $\rho_3(G)$ o menor número $|A|+|B|$ tal que $A \subseteq E(G)$ e $B = \{b_1, b_2, ..., b_k\}$, onde $b_i \notin E(G)$ para $(1 \leq i \leq k)$, e $(G - A) + B$ é um grafo perfeito.*

Para finalizar, vamos considerar a operação mais drástica de remoção de vértices.

DEFINIÇÃO 5.4 (SUBGRAFO INDUZIDO PERFEITO MÁXIMO) *Dado um grafo G, o tamanho do menor conjunto $X \subseteq V(G)$, tal que o subgrafo de G induzido por $G - X$ é perfeito, é denotado por $\rho_4(G)$.*

Neste capítulo, em várias situações onde não há ambigüidade, escreveremos simplesmente ρ_1 ao invés de $\rho_1(G)$ e $\overline{\rho_1}$ ao invés de $\rho_1(\overline{G})$. Idem para ρ_2, ρ_3, ρ_4, ω, χ e α (lembrando que ω é o tamanho da maior clique, χ é o número cromático e α o número de independência de um grafo).

Vale a pena mencionar que nas operações relacionadas ao parâmetro ρ_1 e ρ_4 obtém-se um subgrafo do grafo original. Na operação relacionada ao parâmetro ρ_2, obtém-se um supergrafo do grafo original. No caso do parâmetro ρ_3, nenhum dos dois casos acima é aplicável.

FATO 5.1 *Para qualquer grafo temos:*

i. $\rho_4 = \overline{\rho_4}$.

ii. $\rho_1 = \overline{\rho_2}$ *e* $\rho_2 = \overline{\rho_1}$.

iii. $\rho_3 = \overline{\rho_3}$.

PROVA: *(i)* Conseqüência direta do teorema 1.1. *(ii)* Seja G um grafo com $\rho_1(G) = k$ e seja $A \subseteq E(G)$ um subconjunto de k arestas tal que $G - A$ é perfeito. Segue do teorema 1.1 que $\overline{G} + A$ é perfeito e que obviamente A é minimo. Para ver que $\rho_2 = \overline{\rho_1}$ observe que se $\rho_1 = \overline{\rho_2}$ então $\overline{\rho_1} = \overline{\overline{\rho_2}} = \rho_2$. *(iii)* Seja $A \subseteq E(G)$ e $B \cap E(G) = \varnothing$ tal que $\rho_3(G) = |A| + |B|$ e $(G - A) + B$ é perfeito. A partir do teorema 1.1 (novamente) temos que $(\overline{G} - B) + A$ é perfeito e que $|A|+|B|$ é mínimo para tornar $\overline{G}$ um grafo perfeito. □

TEOREMA 5.1 *Para qualquer grafo temos $\rho_4 \leq \rho_3 \leq \rho_1, \rho_2$.*

PROVA: A desigualdade $\rho_3 \leq \rho_1, \rho_2$ é conseqüência direta da desigualdade $\rho_3 \leq min(\rho_1, \rho_2)$. Agora vamos mostrar que $\rho_4 \leq \rho_3$.

Seja G um grafo com $\rho_3(G) = k$. Vamos mostrar que $\rho_4 \leq \rho_3$ exibindo um conjunto $X \subseteq V(G)$, com $|X| \leq k$, tal que $G - X$ é perfeito.

Seja $A = \{a_1, a_2, ..., a_{k_1}\} \subseteq E(G)$ e $B = \{b_1, b_2, ..., b_{k_2}\}$, onde $b_i \notin E(G)$ para $(1 \leq i \leq k_2)$, tal que $H = (G - A) + B$ é perfeito e $|A| + |B| = k$. Seja $a_i = \{v^1_{a_i}, v^2_{a_i}\}$ para

$1 \leq a_i \leq k_1$ e $b_i = \{v_{b_i}^1, v_{b_i}^2\}$ para $1 \leq b_i \leq k_2$. Tome $X = X_1 \cup X_2$ tal que os conjuntos de vértices $X_1 = \{x_1^1, x_1^2, ..., x_1^{k_1}\}$ e $X_2 = \{x_2^1, x_2^2, ..., x_2^{k_1}\}$ são construidos da seguinte maneira:

Para $1 \leq i \leq k_1$, faça cada $x_1^i = v_{a_i}^j$, para $j = 1$ ou $j = 2$. Para $1 \leq i \leq k_2$, faça cada $x_2^i = v_{b_i}^j$, para $j = 1$ ou $j = 2$.

Em outras palavras, os vértices removidos de G para torná-lo perfeito são os vértices que são adjacentes às $\rho_3(G)$ arestas que devem ser removidas e inseridas em G para torná-lo perfeito.

Agora, vamos mostrar que para X construido desta maneira temos que $G - X$ é perfeito. Suponha que $G - X$ é imperfeito. Então existe um buraco ímpar ou em $G - X$ ou em $\overline{G} - X$, com, digamos, os vértices $u_1, u_2, ..., u_l$, para algum l ímpar, $5 \leq l \leq |V(G)| - |X|$. Se este buraco ímpar aparece em $G - X$ ele deve aparecer em H. Se este buraco ímpar aparece em $\overline{G} - X$ ele deve aparecer também em $\overline{H}$. Mas H (e $\overline{H}$), são perfeitos, o que é uma contradição. □

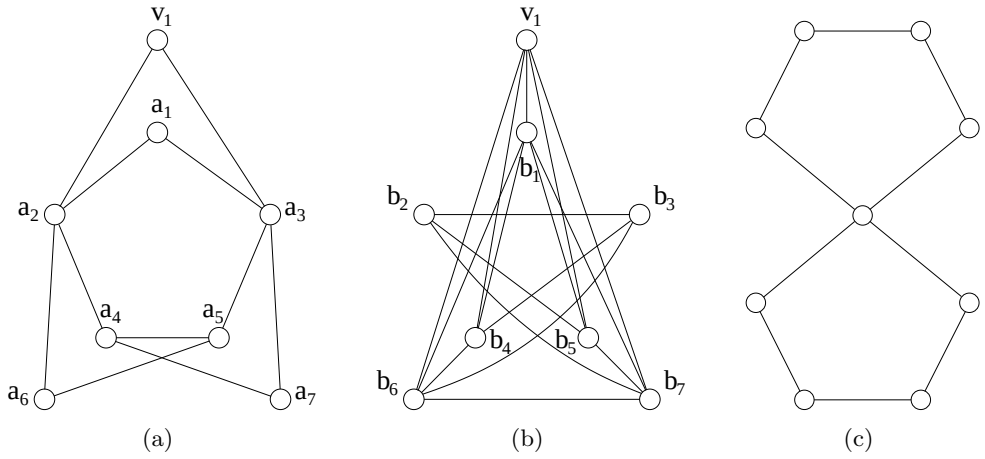


Figura 5.1: (a) O grafo G. (b) O grafo H (isomórfico ao grafo $\overline{G}$). (c) Um grafo com $\rho_4 < \rho_3$.

Observe que o parâmetro ρ_3 não carrega muita informação se não existem grafos em que ocorre a desigualdade estrita $\rho_3 < \rho_1, \rho_2$. Podemos construir tais grafos a partir dos grafos G e H apresentados nas figuras 5.1(a) e 5.1(b) respectivamente. O grafo construido é apresentado na figura 5.1. Agora veremos esta construção em detalhes.

Primeiramente, note que $\rho_1(G) = 2$ e $\rho_2(G) = 1$ (com a inserção da aresta a_2a_3, por exemplo, o grafo torna-se perfeito). Conseqüentemente temos $\rho_3(G) = 1$. Para o grafo H temos $\rho_1(H) = 2$ e $\rho_2(H) = \rho_3(H) = 1$ (com a remoção da aresta b_2b_3, por exemplo, o grafo torna-se perfeito), pois H é isomorfico à $\overline{G}$.

Seja G' um grafo com o conjunto de vértices $V(G') = V(G) \cup V(H)$ e conjunto de arestas $E(G') = E(G) \cup E(H)$ (veja a figura 5.1). Removendo a aresta b_2b_3 de $E(G')$ e inserindo a_2a_3 em $E(G')$, temos um grafo perfeito. Pela construção de G' é fácil ver que $G'' = (G' + \{a_2a_3\}) - \{b_2b_3\}$ não tem buracos ímpar.

Vamos olhar agora para $\overline{G''}$. Pela construção sabemos que o subgrafo de $\overline{G'}$ induzido por $\{v_1, a_1, a_2, ..., a_7\}$ não tem buracos ímpar. O mesmo ocorre para o subgrafo induzido por $\{v_1, b_1, b_2, ..., b_7\}$. Seja $A = \{a_1, ..., a_7\}$ e $B = \{b_1, ..., b_7\}$.

A última coisa a verificar é se existe um buraco ímpar usando vértices de ambos A e B. Mas tais buracos não existem em $\overline{G''}$ existem todas as arestas ligando vértices de A a B em $\overline{G''}$.

Também não é difícil encontrar um caso onde ocorre a desigualdade estrita $\rho_4 < \rho_3$. Um exemplo é mostrado na figura 5.1(c).

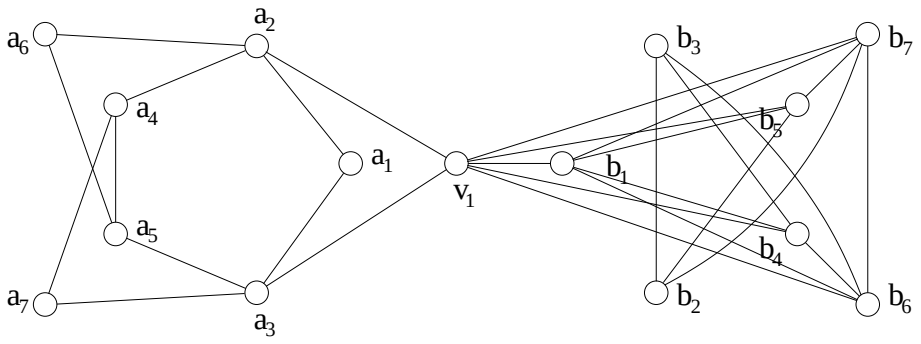


Figura 5.2: O grafo G'.

5.2 Limitantes para os Parâmetros de Perfeição

5.2.1 Limitantes Superiores

Seguem dois limitantes superiores óbvios para ρ_1 e ρ_2 (e conseqüentemente para ρ_3 e ρ_4).

FATO 5.2 *Para todos os grafos com n vértices e m arestas tem-se*

i. $\rho_1 \leq min\left(m - (n-1), \frac{m}{2}\right)$

ii. $\rho_2 \leq min\left(\left[\binom{n}{2} - m\right] - (n-1), \frac{\binom{n}{2} - m}{2}\right)$

PROVA: *(i)* Conseqüência direta do fato que árvores e grafos bipartites e do fato de que para todo grafo G, existe $X \subseteq E(G)$, $|X| \leq m/2$, tal que $G - X$ é bipartite (veja [1]). *(ii)* Segue do fato 5.1(ii). □

5.2.2 Limitantes Inferiores e Grafos Bastante Imperfeitos

Começaremos esta seção com alguns limitantes inferiores para ρ_4. Em seguida mostraremos que existem grafos, que de certa forma podemos dizer que são bastante imperfeitos.

LEMA 5.1 *Para todos os grafos com n vértices temos*

i. $\chi - \omega \leq \rho_4$

ii. $\overline{\chi} - \alpha \leq \rho_4$

iii. $n/\alpha - \omega \leq \rho_4$

PROVA: *(i)* Esta desigualdade é conseqüência de que para todo $v \in V(G)$ temos $\chi(G-v) \geq \chi(G) - 1$ (em outras palavras, o número cromático de um grafo não pode cair em mais de uma unidade com uma remoção de vértice). *(ii)* Conseqüência direta de (i) e do fato 5.1(i). *(iii)* Esta desigualdade vem de (i) e da bastante conhecida desigualdade $\alpha\chi \geq n$. □

Do teorema 5.1, temos que os limitantes do lema 5.1 são válidos para ρ_1, ρ_2 e ρ_3.

Como existem grafos com $\omega = 2$ e χ arbitrariamente alto [4], podemos concluir a partir do lema 5.1(i) que existem grafos com ρ_4 arbitrariamente alto. Entretanto, deste resultado não podemos saber se ρ_4 é alto quando comparado com o tamanho do conjunto de vértices do grafo. Para obter esta informação vamos usar um pouco da teoria dos números de Ramsey.

O número $r(k, l)$ é o tamanho máximo que o conjunto de vértices de um grafo G possa ter, de maneira que $\alpha(G) < k$ e $\omega(G) < l$. Ramsey provou (veja [4]) que o número $r(k, l)$ é bem definido para todos inteiros $k, l > 0$. A seguir enunciaremos uma resultado bastante conhecido na teoria dos números de Ramsey é o seguinte (a prova pode ser encontrada em [4]).

TEOREMA 5.2 *Seja* $m = min(k, l)$*, temos* $r(k, l) \geq 2^m$.

Agora uma definição que será útil adiante:

DEFINIÇÃO 5.5 *O grafo G é um grafo* (k, k)*-ramsey se G é um grafo com* $r(k, k) - 1$ *vértices, com* $\alpha(G) < k$ *e* $\omega(G) < k$.

Agora vamos mostrar que existem grafos com ρ_4 alto mesmo quando comparado com o número de vértices de um grafo. Para obter esse resultado, vamos combinar o lema 5.1(iii) com o teorema 5.5. Em certo sentido, podemos dizer que estes grafos são bastante imperfeitos. Vamos enunciar o resultado da seguinte maneira:

TEOREMA 5.3 *Existem grafos com n vértices e*

$$\rho_4 \geq \frac{n}{\lg(2n)} - \lg(2n).$$

PROVA: Do teorema de Ramsey sabemos que $r(k, k)$ é bem definido para todo inteiro $k > 0$. Do teorema 5.5 temos que $r(k, k) \geq 2^{k/2}$. Seja G um grafo (k, k)-ramsey. Do lema 5.1(iii) temos que

$$\rho_4 \geq \frac{r(k,k) - 1}{(k-1)} - (k-1) \geq \frac{n}{\lg(2n)} - \lg(2n)$$

o que prova o teorema. □

5.3 Complexidade Computacional

Natanzon, Shamir e Sharan [35] mostraram em 1999 que o problema de decisão relacionado aos problemas de encontrar o subgrafo perfeito máximo, o supergrafo perfeito mínimo e o problema do grafo perfeito mais próximo[1] são NP-completos. Estes três problemas podem ser visto com o problema de encontrar os parâmetros ρ_1, ρ_2 e ρ_3.

Uma conseqüência do teorema de Lewis e Yannakakis [29] abaixo é que o problema de encontrar ρ_4 é um problema NP-completo.

Teorema 5.4 (Lewis e Yannakakis) *Se* Π *é uma propriedade de grafos satisfazendo as seguintes condições:*

i. Existem infinitos grafos satisfazendo a propriedade Π *(grafos que pertencem a* Π*).*

ii. Existem infinitos grafos não satisfazendo Π*.*

iii. Se um grafo G *tem a propriedade* Π *então* Π *ocorre em todo subgrafo induzido de* G*.*

Então o seguinte problema é NP-completo: Dado um grafo G *e um inteiro positivo* $k \leq |V(G)|$*, existe um subconjunto* $V' \subseteq V(G)$ *com* $|V'| \geq k$ *tal que* Π *ocorre no subgrafo de* G *induzido por* V'*?*

[1] Natanzon chama este problema de "Minimum Perfect Edition".

Capítulo 6

Implementação

O objetivo deste capítulo é apresentar e discutir alguns resultados obtidos da implementação do algoritmo de limpeza de grafos. Os algoritmos foram implementados como *plugins* que são carregados por um sofware chamado GRAFO. Desta maneira, em diferentes implementações dos algoritmos de teste de perfeição que possam vir a ser realizadas utilizando o programa GRAFO como plataforma, o código do autor pode ser facilmente utilizado.

6.1 O programa GRAFO

O programa GRAFO é um software que visa auxiliar a implementação de algoritmos em grafos. O software, que foi desenvolvido pelo autor desta dissertação em parceria com Oliver M. van Kaick, permite que um usuário desenhe grafos (também é possível carregar grafos do disco e modificá-los, além da opção de se gerar grafos aleatórios) e execute algoritmos sobre estes grafos fazendo acesso a ítens de menu. A saída dos algoritmos é apresentada na tela do programa GRAFO. A partir do grafo de entrada, o programa GRAFO pode gerar como saída um outro grafo, uma mensagen de texto ou ainda evidenciar conjuntos de vértices ou de arestas no grafo de entrada. O exemplo da figura 6.1(a) mostra o programa GRAFO evidenciando um conjunto de vértices e um conjunto de arestas em um dado grafo de entrada.

A principal utilidade do programa grafo é auxiliar a implementacao de algoritmos em grafos. O software foi concebido de tal forma que o usuário possa utilizar a biblioteca para representação e manipulação dos grafos do GRAFO e compilar seu programa como "biblioteca dinâmica". A idéia é que o programa GRAFO carregue a biblioteca construída como um *plugin* e o usuário possa executá-lo sobre um grafo (gerado pelo usuário, por exemplo) usando os menus do programa GRAFO. A figura 6.1(b) mostra um menu do programa GRAFO com uma série de *plugins* carregados.

6.2 Implementando o algortimo de limpeza de grafos

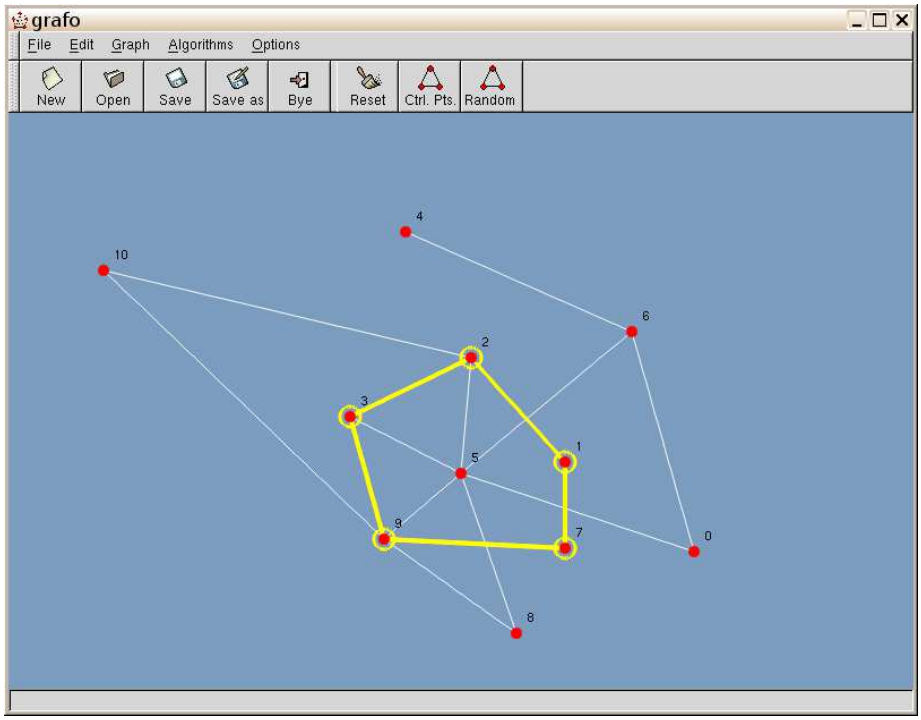
6.2.1 Detectando Jóias e estruturas T_1, T_2, T_3

Para detectar Jóias e estruturas T_1, T_2 e T_3 em grafos foram criados quatro *plugins* que podem ser carregados independentemente. Na figura 6.2(a) e 6.2(b) temos exemplos da execução do programa GRAFO carregando os *plugins* que implementam algoritmos que testam a presença de estruturas T_1 e T_2 respectivamente. Na figura 6.3(a) um exemplo da execução do programa GRAFO carregando os *plugins* que testam a presença de estruturas T_3. Um exemplo de execução do algoritmo que procura jóias em grafos é apresentado na figura 6.3(b).

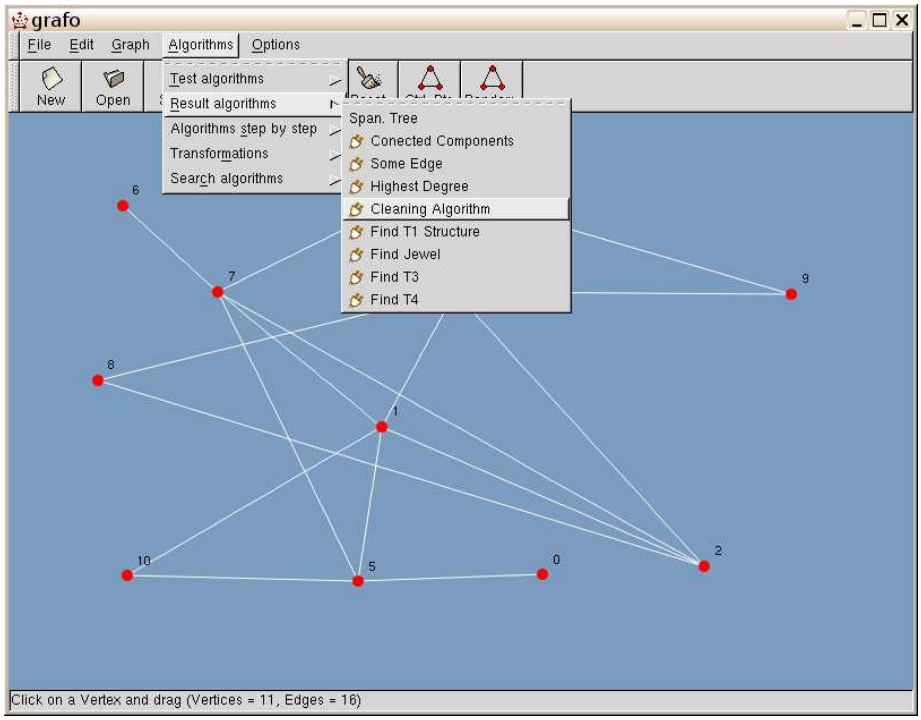
6.2.2 Gerando Candidatos a *quasi-cleaner*

A implementação do algoritmo de limpeza proprimente dito é um *plugin* que inicialmente executa os testes por estrutura T_1, T_2 e T_3 em G e depois executa o teste por jóias em G e em $\overline{G}$. Para executar isso o *plugin* faz uso de funcionalidades do programa GRAFO que permite que um *plugin* execute outros *plugins* que estejam carregados.

Na execução do algoritmo de limpeza, caso encontre-se alguma das estruturas proibidas, o algoritmo interrompe a execução e a estrutura encontrada é evidenciada no grafo de entrada. Por exemplo, se foi encontrada uma estrutura T_2, é apresentada a saída do *plugin* que encontrou esta estrutura. A figura 6.4(a) mostra um exemplo da execução do algoritmo de limpeza em que foi encontrada jóia. Caso não seja encontrada nenhuma destas estruturas o plugin gera todos os conjuntos, os guarda em memória. Por fim é apresentada uma mensagen indicando que o algoritmo de limpeza foi executado na barra inferior do programa. Um exemplo é apresentado na figura 6.4(b). Nas figuras 6.5(a) e 6.5(b) são apresentados mais dois exemplos de execução do algoritmo de limpeza.

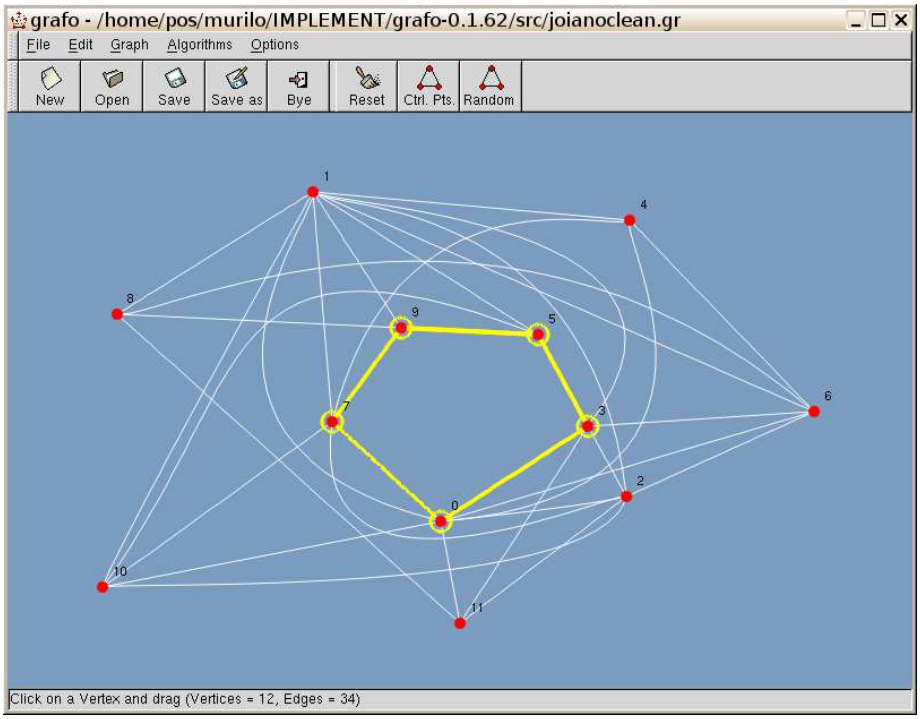


(a) Neste exemplo a saída do programa GRAFO é um conjunto de vértices e de arestas evidenciados.

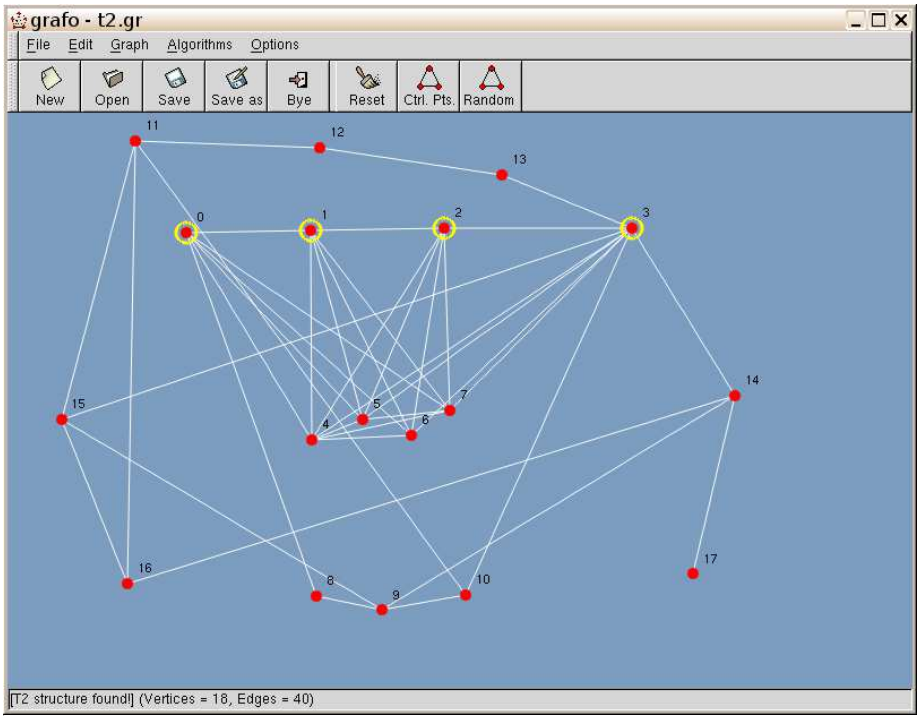


(b) Exemplo do *plugin* chamado *Cleaning Algorithm* carregado pelo programa GRAFO e sendo selecionado para execução pelo usuário.

Figura 6.1: O programa GRAFO.

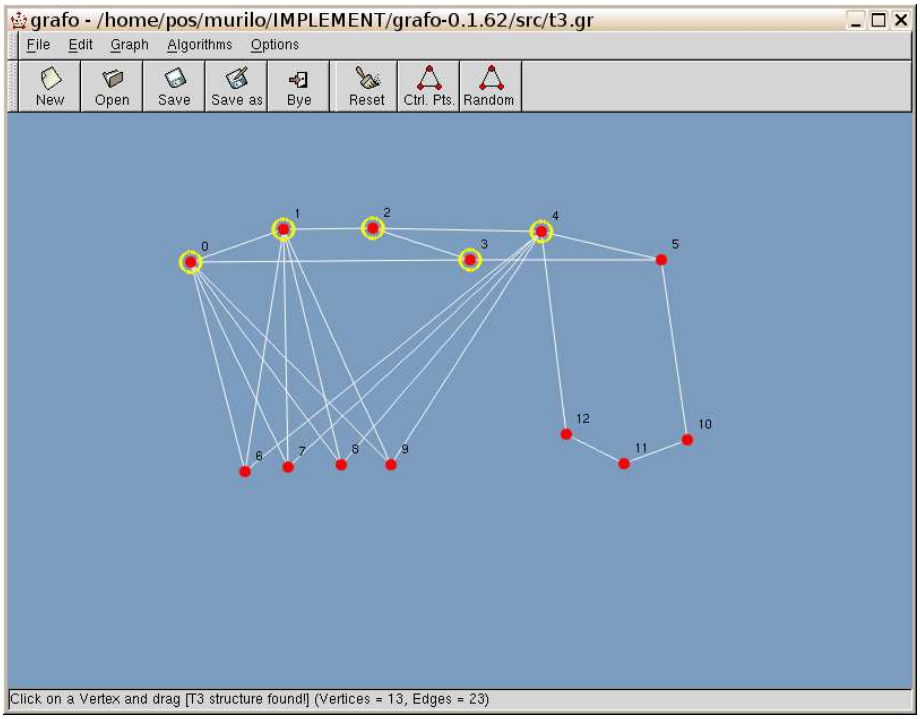


(a) Uma estrutura T_1 encontrada em um grafo de tamanho 12, com cada aresta aparecendo com probabilidade 1/2. Os vértices e as arestas que formam o buraco de tamanho 5 foram evidenciados pelo programa GRAFO.

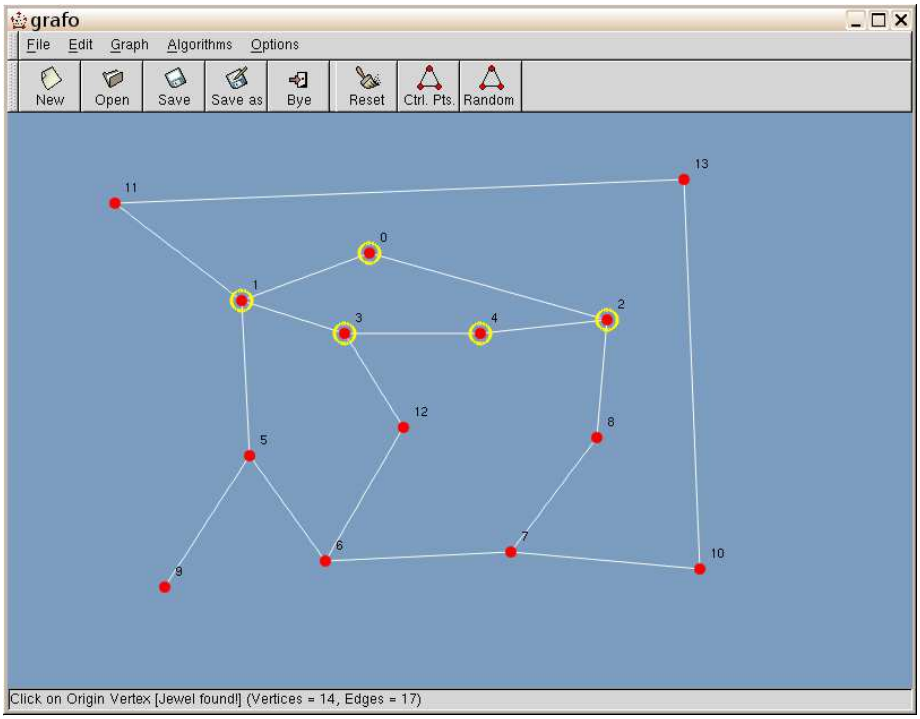


(b) Uma estrutura T_2 encontrada em um grafo construído pelo usuário. Os vértices 0, 1, 2 e 3 foram evidenciados pelo programa GRAFO. Isto indica que estes vértices são os vértices v_1, v_2, v_3 e v_4 da definição de estruturas T_2, na seção 2.3.2.

Figura 6.2: Encontrando estruturas T_1 e T_2.

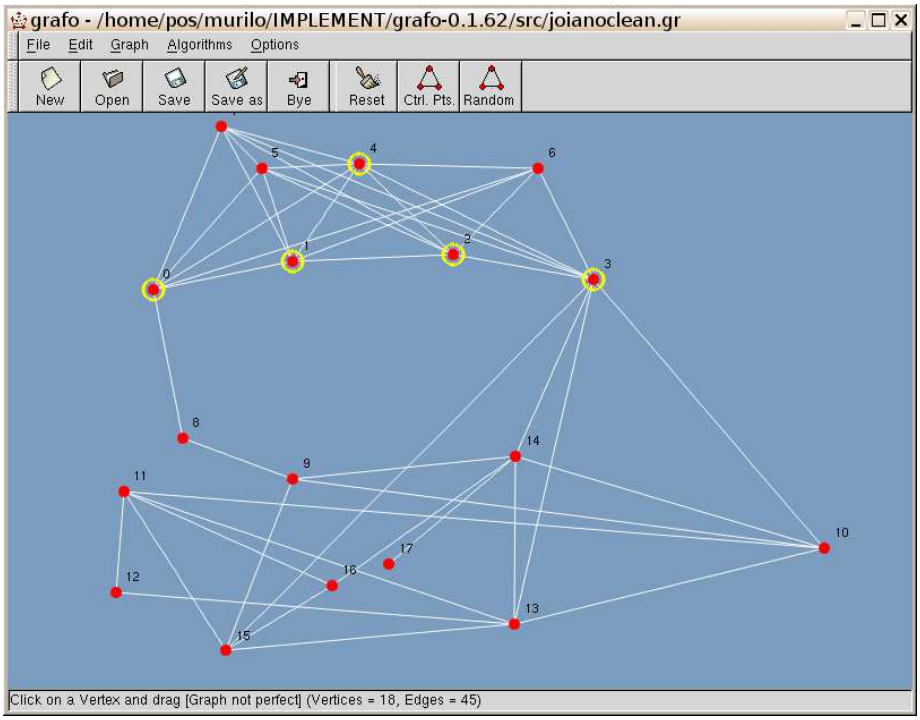


(a) Uma estrutura T_3 encontrada em um grafo construído pelo usuário. Os vértices 0, 1, 2, 3 e 4 foram evidenciados pelo programa GRAFO. Isto indica que estes vértices são os vértices v_1, v_2, v_3, v_4 e v_5 da definição de estruturas T_3, na seção 2.3.3.

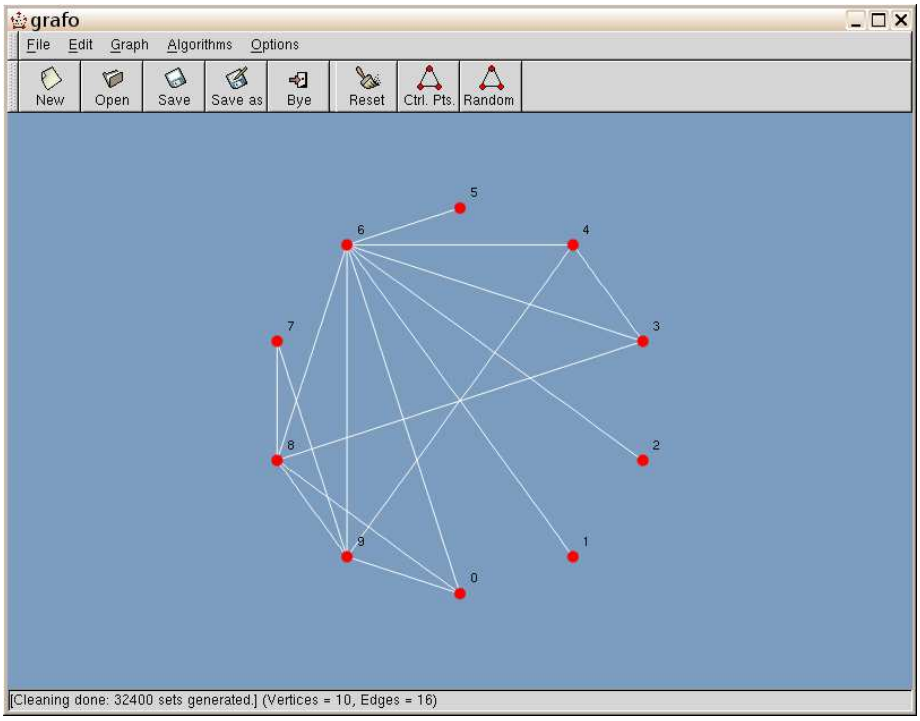


(b) Uma Jóia encontrada em um grafo construído pelo usuário. Os vértices 0, 1, 3, 4 e 2 foram evidenciados pelo programa GRAFO. Isto indica que estes vértices são respectivamente os vértices v_5, v_1, v_2, v_3 e v_4 da definição de jóias, na seção 2.3.4.

Figura 6.3: Encontrando estruturas T_3 e Jóias.

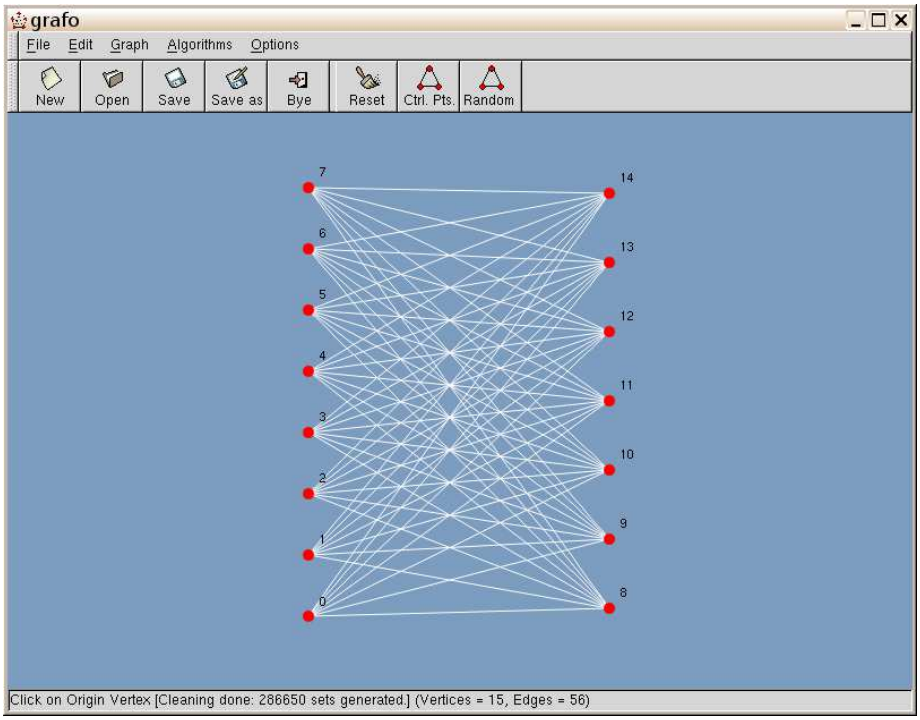


(a) Durante a execução do algoritmo de limpeza foi identificada uma jóia e portanto a execução foi interrompida e a jóia foi evidenciada.

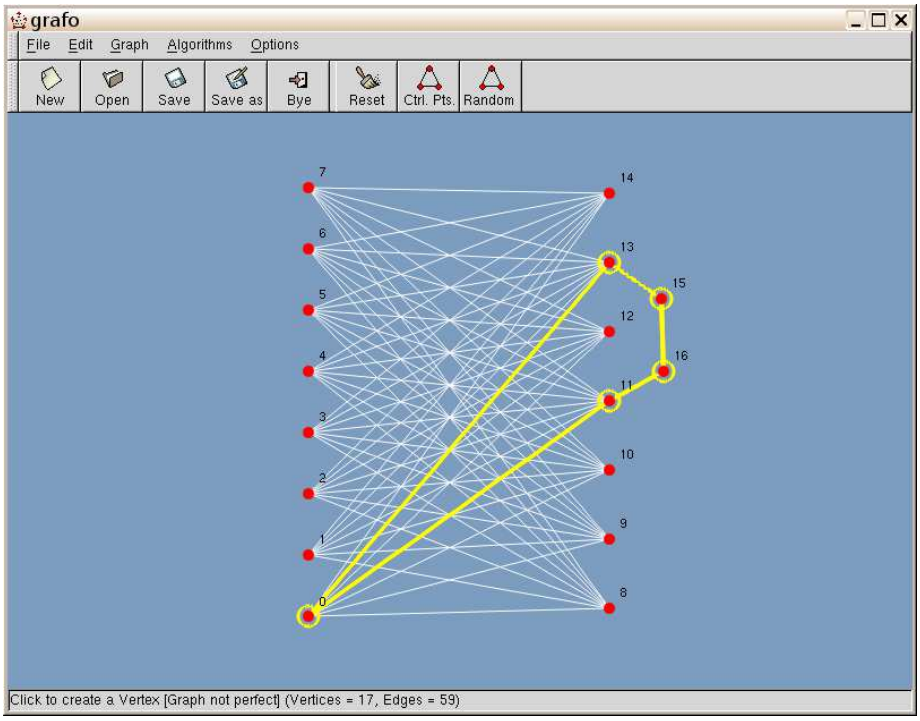


(b) Execução do algoritmo de limpeza sobre um grafo de 10 vértices e com arestas aparecendo com probabilidade 3/10. Nenhuma estrutura proibida foi encontrada e 32400 conjuntos candidatos a *quasi-cleaner* foram gerados.

Figura 6.4: Executando o algoritmo de limpeza.



(a) Neste exemplo, o algoritmo de limpeza foi executado sobre um grafo bipartite. Nenhuma estrutura proibida foi encontrada.



(b) Inserindo-se os vértices 15 e 16 e as arestas $\{13, 15\}$, $\{15, 16\}$ e $\{16, 11\}$ no grafo bipartite anterior, quando executado o algoritmo de limpeza, a saída é a estrutura proibida T_1 que foi econtrada.

Figura 6.5: Mais algumas execuções do algoritmo de limpeza.

Capítulo 7

Conclusões e Trabalhos Futuros

O objetivo principal desta dissertação foi um estudo dos dois algoritmos de teste de perfeição de grafos. O núcleo dos dois algoritmos, que sem dúvida é a parte mais complexa dos mesmos, foi estudado mais a fundo e implementado. Algo importante a ser mencionado é o fato de que até o momento, o autor desta dissertação não tem notícias de outras implementações destes algoritmos.

Adicionalmente, nesta dissertação foram definidos os parâmetros ρ_1, ρ_2, ρ_3 e ρ_4, que podem ser associados a um grafo para exprimir seu grau de imperfeição. O autor relacionou estes parâmetros com algumas operações que podem ser aplicadas a um grafo imperfeito para torná-lo perfeito e mostrou que $\rho_4 \leq \rho_3 \leq \rho_1$ e $\rho_4 \leq \rho_3 \leq \rho_2$. Foram apresentados também exemplos de grafos em que cada uma destas desigualdades pode ser estrita. Além disso foram apresentados limitantes superiores e inferiores para estes parâmetros, e como conseqüência foi demonstrado que existem grafos com n vértices para os quais o número de vértices que deve ser removido para tornás-lo perfeitos é pelo menos $\frac{n}{\lg(2n)} - \lg(2n)$.

Como trabalho futuro o autor pretende implementar os dois algoritmos de teste de perfeição utilizando em ambos o núcleo já implementado. Além disso o autor pretende obter mais resultados envolvendo os parâmetros ρ_1, ρ_2, ρ_3 e ρ_4. Uma possibilidade seria obter limitantes mais "justos" para subfamílias dos grafos perfeitos.

Bibliografia

[1] Alon, N. e Spencer, J. *The probabilistic method*. John Wiley and Sons Inc., 1992.

[2] Barnier, N. e Brisset, P. Graph coloring for air traffic flow management. In *CPAIOR'02: Fourth Int. Workshop on Integr. of AI and OR Techniques in Constraint Prog. for Comb. Optimisation Problems* (Le Croisic, France, Março 2002), pp. 133–147.

[3] Berge, C. Färbung von Graphen, deren sämtliche bzw. deren ungeraden Kreise starr sind (Zusammenfassung). *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe 10* (1961), 114.

[4] Bondy, J. A. e Murty, U. S. R. *Graph Theory with Applications*. North-Holland, New York, 1976.

[5] Briggs, P. *Register allocation via graph coloring*. PhD thesis, 1992.

[6] Burlet, M. e Fonlupt, J. Polynomial algorithm to recognize meyniel graph. *Ann of Discrete Math 21* (1984), 85–92.

[7] Chudnovsky, M., Cornuejols, G., Liu, X., Seymour, P. e Vuskovic, K. Recognizing Berge Graphs. *submited to Combinatorics* (2004).

[8] Chudnovsky, M., Cornuejols, G., Seymour, P., Liu, X. e Vuskovic, K. Cleaning for bergeness, 2003. *preprint*.

[9] Chudnovsky, M., Robertson, N., Seymour, P. e Thomas, R. The strong perfect graph theorem. `http://www.math.gatech.edu/~thomas/spgc.html`.

[10] Chudnovsky, M., Robertson, N., Seymour, P. D. e Thomas, R. Progress on perfect graphs. *Mathematical Programming Series B*, 97 (2003), 405–422.

[11] Chudnovsky, M. e Seymour, P. Recognizing berge graphs, 2003. *preprint*.

[12] Chvátal, V., Fonlupt, J., Sun, L. e Zemirline, A. Recognizing dart-free perfect graphs. *SIAM Journal of Computing 31* (2002), 1315–1338.

[13] Chvátal, V. e Sbihi, N. Bull-free Berge graphs are perfect. *Graphs and Combin. 3* (1987), 127–139.

[14] Chvátal, V. e Sbihi, N. Recognizing claw-free perfect graphs. *Journal of Combinatorial Theory B 44* (1988), 174–176.

[15] Conforti, M., Cornuejols, G., Kapoor, A. e Vuskovic, K. Balanced 0,+1,-1 matrices, Part II: Recognition Algorithm. *Journal of Combinatorial Theory B 81* (2001), 275–306.

[16] Conforti, M., Cornuejols, G., Kapoor, A. e Vuskovic, K. Even-hole-free graphs, Part II: Recognition Algorithm. *Journal of Graph Theory 40* (2002), 238–266.

[17] Conforti, M., Cornuejols, G. e Vuskovic, K. Decomposition of odd-hole-free graphs by double star cutsets and 2-joins. *Discrete Applied Mathematics 141* (2004), 41–91.

[18] Cornuejols, G., Liu, X. e K. Vuskovic, . A polynomial algorithm for recognizing perfect graphs. *preprint* disponível em `http://integer.gsia.cmu.edu/`.

[19] Cornuéjols, G. e Cunningham, W. H. Compositions for perfect graphs. *Discrete Math. 55* (1985), 245–254.

[20] da Silva, M. Coloração de Grafos. Tech. rep., 2003. `http://www.inf.ufpr.br/~murilo/color.ps.gz`.

[21] de Figueiredo, C. M. H. e Vuskovic, K. Recognition of quasi-meyniel graphs. *Discrete Applied Mathematics 113* (2001), 225–260.

[22] de Werra, D. An introduction to timetabling. *European Journal of Operations Research*, 19 (apr 1985), 151–162.

[23] Fouquet, J. L., Roussel, F., Rubio, P. e Thuillier, H. New classes of perfectly orderable graphs. *Discrete Mathematics 236* (2001), 95–109.

[24] Fulkerson, D. Blocking and anti–blocking pairs of polyhedra. *Math. Programming*, 1 (1971), 168–194.

[25] Gerber, M. U. e Hertz, A. A transformation which preserves the clique number. *Journal of Combinatorial Theory B 83* (2001), 320–330.

[26] Grötschel, M., Lovász, L. e Schrijver, A. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1*, 2 (1981), 169–197.

[27] Hertz, A. A fast algorithm for colouring meyniel graphs. *Journal of Combinatorial Theory B 50* (1990), 231–240.

[28] Hougardy, S. Inclusions between classes of perfect graphs. `http://www.informatik.hu-berlin.de/~hougardy/paper/classes.html`.

[29] Lewis, J. e Yannakakis, M. The Node–deletion problem for hereditary properties is NP–complete. *Journal of Computer and System Sciences 20* (1980), 219–230.

[30] Liebers, A. Planarizing graphs — a survey and annotated bibliography. *J. Graph Algorithms and Applications 5*, 1 (2001), 1–74.

[31] Lovász, L. Normal hypergraphs and the perfect graph conjecture. *Discrete Math*, 2 (1972), 253–267.

[32] Maffray, F. e Preissmann, M. Sequential colorings and perfect graphs. *Discrete Applied Mathematics 94* (1999).

[33] Meyniel, H. The graphs whose odd cycles have at least two chords. *Ann of Discrete Math 21* (1984), 115–120.

[34] Middendorf, M. e Pfeiffer, F. On the complexity of recognizing perfectly orderable graphs. *Discrete Math 80* (1990), 327–333.

[35] Natanzon, A., Shamir, R. e Sharan, R. Complexity classification of some edge modification problems. In *Workshop on Graph-Theoretic Concepts in Computer Science* (1999), pp. 65–77.

[36] Olariu, S. Paw-free graphs. *Inform. Process. Lett.* (1988), 53–54.

[37] Papadimitriou, C. H. *Computational Complexity*. Addison-Wesley, 1995.

[38] Roussel, F. e Rusu, I. An $O(n^2)$ algorithm to color Meyniel graphs. *Discrete Mathematics 236* (2001), 95–109.

[39] Russel, F. e Rubio, P. About skew partitions in minimal imperfect graphs. *Journal of Combinatorial Theory B 83* (2001), 171–190.

[40] Sedgewick, R. *Algorithms in C, part 5*. Addison-Wesley, 2002.