Edmundo Inácio Júnior

Planarização de Grafos por Divisão de Vértices

Curitiba – Paraná

Agosto de 2003

Edmundo Inácio Júnior

Planarização de Grafos por Divisão de Vértices

Dissertação submetida ao Curso de Pós-Graduação em Informática, Setor de Ciências Exatas da *Universidade Federal do Paraná* em cumprimento parcial às exigências à obtenção do grau de *Mestre em Informática*.

Orientador:

Cândido Ferreira Xavier de Mendonça Neto

Co-orientador:

Ademir Aparecido Constantino

Curitiba – Paraná

Agosto de 2003

UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTMENTO DE INFORMÁTICA

Os abaixo-assinados certificam por meio desta que leram e recomendaram a Universidade Federal do Paraná a *aprovação* da dissertação intitulada "**Planarização de Grafos por Divisão de Vértices**" de **Edmundo Inácio Júnior** defendida em ___ *Agosto de 2003*, *Curitiba – Paraná* , em cumprimento parcial às exigências à obtenção do grau de **Mestre em Informática**.

Banca Examinadora:

Orientador:
Cândido Ferreira Xavier de Mendonça Neto
Universidade Estadual de Maringá
Departamento de Informática

Co-orientador:
Ademir Aparecido Constantino
Universidade Estadual de Maringá
Departamento de Informática

Examinador Externo:
Luérbio Faria
Universidade do Estado do Rio de Janeiro
Faculdade de Formação de Professores

Examinador Interno:
André Luiz Pires Guedes
Universidade Federal do Paraná
Departamento de Informática

Dedico este trabalho

à Deus,

que me é fonte constante de inspiracão,

aos meus Pais e Irmã,

pelo apoio a minha formação acadêmica.

Agradecimentos

Não há dúvidas de que ao se elaborar um trabalho de pesquisa, são requeridas horas e mais horas de esforço. Este desafio, sem sombra de dúvida, não poderia se concretizar sem que houvesse o envolvimento de várias pessoas, que com diferentes maneiras, contribuíram de forma expressiva à presente dissertação.

Primeiramente, agradeço à orientação dos Profs. Xavier e Ademir, que de forma tão hábil auxiliaram-me na elaboração de cada parte do projeto bem como da dissertação, estabelecendo uma rota segura e precisa à conclusão do mesmo. A paciência, o entusiasmo, o bom humor e a disponibilidade deles foram importantes nos momentos de dificuldades.

Agradeço ao Profs. André e Consularo. Todas as suas observações quanto à forma e conteúdo do trabalho foram pertinentes e valiosas e com certeza serviram para a melhoria do mesmo. Agradeço também aos professores e amigos do Programa de Pós-Graduação em Informática, onde passei a maior parte do tempo realizando os trabalhos de suas disciplinas bem como da dissertação.

Agradeço aos amigos e também acadêmicos Letícia, Newton, Emerson, aos casais César e Sandra e Arnaldo e Andréia pelo apoio e auxílio, sendo estes minha família em Maringá, sempre presentes para "o que der e vier". Agradeço ao Sensei Roberto Nagahama pelas horas de descontração e prática de judô, que contribuíram para a minha saúde física e mental.

Por fim, mas não por último, agradeço a minha família pelo incentivo e apoio a este novo desafio, a minha irmã Inara e meu cunhado Salvador e especialmente aos meus pais Edmundo e Cecília por sempre apoiarem minhas decisões e permanecerem fortes e constantes no apoio a minha formação acadêmica.

A todos, meu MUITO OBRIGADO...

O Homem deve criar a oportunidade e

não somente encontrá-la.

Francis Bacon (1561 – 1626).

Resumo

Essa dissertação tem como **tema** o estudo de um sub-campo da Teoria dos Grafos conhecida como *planarização de grafos*. Um grafo *planar* é aquele que pode ser desenhado em um plano de maneira que não haja cruzamentos de arestas. "Planarizar" um grafo significa transformar um grafo não-planar, através de um número mínimo de operações, em um grafo planar. O **objetivo** da dissertação é apresentar o método *planarização de grafos por divisão de vértices*, método esse, que pode ser considerado novo no leque dos conhecidos e publicados métodos de planarização: remoção de vértices ou arestas, particionamento planar ou introdução de vértices *dummy*. As **justificativas** para o estudo desse assunto e método podem ser sintetizadas em três pontos: na importância que esse campo de estudo tem para as diversas áreas que se utilizam da representação gráfica para disporem suas informações, tais como projetos de banco de dados e de circuitos integrados; no caráter menos destrutivo desse método de planarização em relação a estrutura original do grafo; e no caráter inédito de sua implementação, até então inexistente. Para atingir-se tal propósito a **metodologia** abrangeu dois aspectos: a revisão dos conceitos básicos sobre a teoria de grafos e a específica relacionada aos métodos de planarização e a estrutura de dados chamada de *árvore-PQ* por eles utilizada; e a revisão dos aspectos conceituais do trabalho iniciado na Tese de doutoramento de Mendonça [17], dando um passo a mais, levando a cabo a implementação do algoritmo aqui intitulado de Planariza por Divisão – PPD. Os **resultados** apontaram para um algoritmo com complexidade de tempo $O(n^2)$ e de espaço $O(n + m)$. Duas modificações importantes foram realizadas no algoritmo original melhorando sua eficiência em termos de *número de vértices divididos* e condições limitantes da estrutura de dados utilizada bem como do algoritmo PPD, em si, foram evidenciadas.

Palavras-chave: planarização de grafos, divisão de vértices, árvores-PQ, número de vértices divididos.

Abstract

The **statement problem** that this work addressed was the study of *graph planarization*, a subfield of the Graph Theory. A graph is *planar* if it can be drawn in a plane without edge crossings between non incident edges or overlappings. "Planarize" a graph means transform it with a small number of operations into a planar graph. The **research objetive** is to present a new method intituled *Graph Planarization by Vertex Splitting* regardless the known and published method as: edge or vertex deletion, planar partitioning, and dummy vertices. The **reasons** of studying both theme and method can be synthesized into tree important pillars: in the increasing rule that graphical techniques, especially visualization, played in the actual world, such as data bank projects, which apply the popular notation Entity- Relationship – ER, and VLSI projects; in the less destructive features that our planarization method enables in concern with the original graph structure; and in the unpublished character of the implementation, until then inexistent. To reach such purpose the **methodology** included two aspects: the revision of the basic concepts on the graph theory and the specific related to the planarization methods as well as the data structure call *PQ-Tree* used by them; and the revision of the conceptual aspects of the work start by Mendonça [17] during his Doctor Degree, giving a step further, carrying out the implementation of the algorithm entitled for us of the *Split-Planarize*. The **results** pointed out that Split-Planarize algorithm performs its task in time complexity $O(n^2)$ and uses space $O(n + m)$. Two important modifications were accomplished in the original algorithm improving its efficiency, measured in terms of *vertex splitting* and constraints that narrow the performance of the Split-Planarize algorithm such as the data structure used as well as the algorithm, itself, were evidenced.

Key-words: graph planarization, vertex splitting, PQ-Trees, splitting number.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Capítulo 1
Apresentação

Além da escrita, que é considerada a principal forma de se difundir a informação, a representação gráfica vem sendo cada vez mais usada, tornando-se uma poderosa forma de auxiliar, ou em diversos casos, de se tornar a única maneira possível pela qual a informação pode ser entendida e interpretada. Provas empíricas disso, podem ser encontradas quando da revisão de trabalhos (técnicos, científicos etc.) nas mais diversas áreas (Exatas, Humanas, Biológicas etc.), onde a representação gráfica se faz presente (diagramas, fluxogramas ER, *layout* de circuitos VLSI etc.).

Dentro deste grande campo, muitas situações da vida real podem ser representadas de forma conveniente por meio de diagramas denominados de *grafos*. Intuitivamente, um grafo pode ser definido como sendo um conjunto de pontos (vértices) no espaço que são inter-conectados por um conjunto de linhas (arestas). Como exemplo, os pontos poderiam ser representações de pessoas e as linhas do relacionamento "a amizade" entre elas; os pontos poderiam ser cidades e a linhas as estradas que as unem.

O estudo desse tipo de representação está compreendido dentro do campo da Teoria dos Grafos, mais especificamente no de desenhos de grafos. Esta área, entre outros interesses, está preocupada com a forma pela qual os grafos podem ser desenhados, objetivando obter formas simples e intelígiveis para que o leitor possa entendê-los da melhor maneira possível. Contudo, via de regra, isto quase sempre não é uma tarefa fácil de se obter. Em geral, considera-se que o desenho de um grafo é inteligível quando ele atende a determinados critérios estéticos como de *simetria, distribuição uniforme de vértices, comprimento uniforme das arestas e se possível em linha reta e número mínimo de cruzamentos.*

Dentre os critérios citados acima, o de cruzamento de arestas tem recebido especial atenção, e seu estudo está compreendido no campo de estudo da *planarização de grafos*. O principal interesse deste campo é determinar se um grafo pode ser desenhado em uma superfície plana de modo que não haja cruzamentos de arestas. O problema de planarização de grafos, à parte de seu interesse teórico, tem grande aplicações práticas. Por exemplo, no *layout* de circuitos eletrônicos VLSI existe uma representação planar de um dado circuito? Se não, qual é o número mínimo de arestas ou vértices cuja remoção torna a representação do circuito planar?

Dado um grafo, o problema é verificar se o mesmo pode ser desenhado em um plano de maneira que não haja cruzamentos de suas arestas. Diversos algoritmos baseados em remoção de vértices ou arestas foram desenvolvidos nos últimos anos baseados na utilização da estrutura de dados conhecida como árvore-PQ existindo um considerável número de publicações a respeito, tais como, por exemplo as encontradas nos trabalhos de Booth e Lueker [10], Vollen [73], Mehlhorn e Mutzel [61], Lempel, Even e Cederbaum [54] e Jayakumar, Thulasiraman e Swamy [44].

1.1 Objetivos

O Objetivo da dissertação é explorar a estrutura de dados da Árvore-PQ de Booth e Lueker [10] para implementar um eficiente algoritmo de planarização pelo método de divisão de vértices. Tem-se como objetivo específico que a implementação do algoritmo Planariza por Divisão apresente resultados iguais ou melhores com relação:

- ao tempo de execução e
- a alteração da estrutura original do grafo

comparados aos algoritmos dos métodos existentes, principalmente ao de remoção de arestas.

1.2 Justificativas

A operação de divisão de vértice já vem sendo discutida a algum tempo em trabalhos como os de Hartsfield, Jackson e Ringel [36, 43, 37] e de Schaffer [67]. Entretanto, o método de planarização por divisão de vértices, compreende um nova abordagem, uma vez que, em nosso conhecimento, não há evidências de publicações a respeito dos resultados de sua implementação, a não ser em alguns trabalhos que o descreveram em termos conceituais, tais como os de Mendonça e seus colegas [17, 20, 21, 79, 27].

1.3 Estrutura da proposta

A proposta está dividida em sete capítulos. O capítulo 1 busca situar o leitor no contexto em que se insere o trabalho, destacando o problema, os objetivos e a estrutura da proposta. Os capítulos 2, 3 e 4 compreendem a revisão bibliográfica.

O capítulo 2 apresenta as principais definições sobre a Teoria dos Grafos necessárias para o entendimento da dissertação. O capítulo 3, traz uma breve descrição do estado da arte dos trabalhos relacionados com planarização de grafos, descrevendo as principais abordagens e medidas de não-planaridade. Esta seção descreve também os principais algoritmos desenvolvidos até o momento sobre a planarização de grafos.

Já o capítulo 4 descreve os principais conceitos relacionados a estrutura de dados conhecida como *árvore-PQ*, desenvolvida por Booth e Lueker [10] que compreende a base do nosso algoritmo. Ainda neste capítulo há menção à implementação do algoritmo de teste de planaridade implementado por Young [81], o qual será aqui utilizado.

O capítulo 5 propõe uma nova abordagem ao problema da planarização de grafos, utilizando a operação de *divisão de vértices*. Ele apresenta os conceitos sobre a operação de divisão de vértices e sobre a medida de não-planaridade a ela associada e traz, em detalhes, a implementação do algoritmo PPD bem como a análise de sua complexidade. Algoritmos adicionais à planarização como o de busca em profundidade (*DFS*) e *numeração-st* também serão discutidos, ainda que sucintamente.

O capítulo 6 apresenta os resultados obtidos – em termos de número de vértices divididos e tempo de execução – do algoritmo PPD sobre quatro diferentes classes de grafos: completos, completos bipartidos, completos convertidos nos Y_k e cartesianos. Um exemplo de execução do algoritmo sobre um grafo também é fornecido. O último, o capítulo 7 delineia as principais resultados e idéias que surgiram com o desenvolvimento desse trabalho. Limitações à planarização encontradas durante a realização do trabalho também são evidenciadas.

A quem possa interessar, toda a dissertação foi desenvolvida utilizando-se os programas MiKTeX[1] e WinEdt[2], que provê os software de edição de texto TeX e LaTeX para *Windows*. Os ilustrações foram criadas utilizando-se os programas *Tgif*[3] e *AGD-Demo*[4].

[1] http://www.miktex.org
[2] http://www.winedt.com
[3] http://bourbon.cs.umd.edu:8001/tgif
[4] http://www.mpi-sb.mpg.de/AGD/index.html

Capítulo 2
Terminologias

Este capítulo introduz o vocabulário básico acerca da Teoria dos Grafos que será útil ao longo de toda a dissertação. Dos diversos livros disponíveis sobre assunto, baseamos nossas definições nos de Even [23], Gibbons [30], Nishizeki e Chiba [65], mas principalmente no de Bondy e Murty [9]. O capítulo está subdividido em duas seções. A primeira aborda as principais definições sobre grafos. A segunda seção, de escopo mais restrito, aborda a definições acerca da planaridade de grafos.

2.1 Definições básica sobre grafos

Um grafo pode ser visualizado por uma *representação geométrica*, na qual o conjunto de pontos distintos (vértices) distribuídos arbitrariamente no plano, são unidos por um conjunto de linhas (arestas). Para um dado grafo G definimos como V seu conjunto de vértices e E seu conjunto de arestas, escrevendo $G = (V, E)$. Uma definição mais formal, também encontrada nos livros sobre Teoria do Grafos, é dada como segue.

Definição 1 (Grafo). Seja G um grafo representado por uma tripla ordenada (V, E, ψ_G) onde V é um conjunto não vazio de vértices, E um conjunto de arestas e ψ_G uma função de incidência que associa a cada aresta e de G um par não ordenado de vértices (não necessariamente distintos) de G denominados de u e v. Dado um grafo G, $|V|$ é denominado por n e $|E|$ por m.

Se e é uma aresta e u e v são vértices tais que $\psi_G(e) = \{u, v\}$, então e é dita unir u e v e os dois vértices são chamados de extremos de e. Neste caso dizemos que o vértice u é vizinho de v ou que u e v são adjacentes. Um grafo tem *múltiplas* arestas ou arestas *paralelas* se existir mais de uma aresta entre o mesmo par de vértices $\{u, v\}$. Uma aresta $\psi_G(e_i) = \{u, u\}$ é chamada de laço. Um grafo com laços e múltiplas arestas é chamado de *multigrafo*.

Um grafo é *finito* se ambos os conjuntos de vértices e arestas são finitos. Um grafo é dito ser *simples* se não possuir múltiplas arestas e nem laços. Um grafo é dito ser *orientado* se suas arestas são pares ordenados (u, v) onde a direção é de u para v, caso contrário o grafo é dito como não-orientado. Uma aresta orientada é representada com uma seta tendo u

como “cauda” e v como “cabeça”. Este trabalho considera somente grafos simples, finitos e não-orientados, portanto omitiremos estas palavras. Omitiremos também a função de incidência escrevendo somente que um grafo G é um par não ordenado $G=(V, E)$. A Figura 2.1 demonstra a maioria dos conceitos até aqui descritos.

(a) multigrafo (b) simples

Figura 2.1: Exemplo de grafo orientado e não orientado

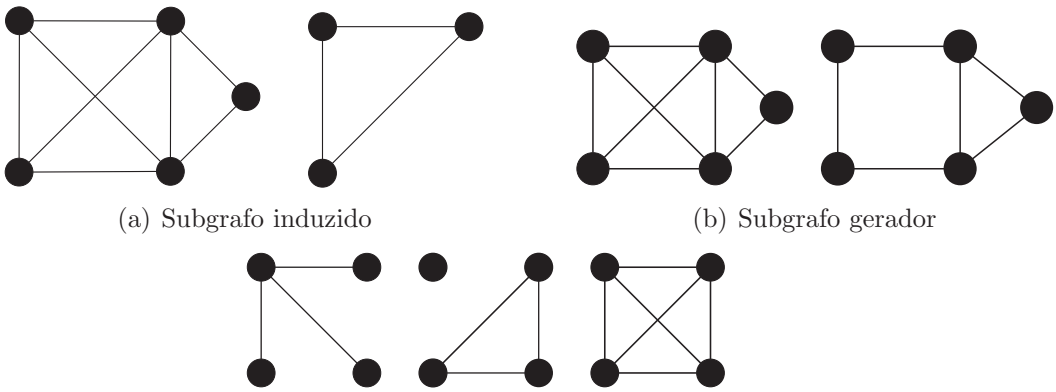
O *grau* de um vértice u, denotado por $d(u)$, é o número de arestas adjacentes a u. Em se tratando de grafos orientados, o vértice u tem grau de entrada e grau de saída correspondente as arestas *incidentes* (que entram - *incoming*) e *dissidentes* (que saem - *outgoing*) em u, respectivamente. O grau mínimo (máximo) de um grafo G é o grau mínimo (máximo) dentre todos os vértices de G.

Dado um grafo $G=(V, E)$ não-orientado, uma seqüência $\{u,\ e_1,\ u_1,\ e_2, \ldots,\ u_{k-1},\ e_k,\ v\}$ de vértices e arestas é chamada de *caminho* de um vértice u para v em G. Se todos os vértices forem distintos a seqüência recebe o nome de *caminho elementar*. Se as arestas forem distintas, a seqüência recebe o nome de *trajeto*. Um caminho que contenha cada vértice do grafo exatamente uma vez é chamado *hamiltoniano*. Se $u = v$, temos então um *ciclo*. Por outro lado, algum caminho ou ciclo que contenha cada aresta do grafo, também exatamente uma única vez, é denominado *euleriano*. Se um grafo G não contém ciclos, ele é chamado de acíclico. Um grafo é *conexo* se existe um caminho para todo par de vértices distintos de G, caso contrário é dito como *desconexo*.

Dado um grafo $G=(V, E)$, um grafo $H = (V', E')$ é chamado de *subgrafo* de G se $V' \subseteq V$ e $E' \subseteq E$. Trata-se, portanto, de um grafo obtido pela retirada de um conjunto $V - V' \subseteq V$ de vértices do grafo original, acompanhado de suas arestas incidentes. Se, além disso, H possuir toda aresta $e = \{u, v\}$ de G tal que ambos os vértices u e v estejam em E', então H é denominado o *subgrafo induzido pelo conjunto de vértices* V' (*induced subgraph*).

Dado um grafo $G=(V, E)$, um grafo $H = (V', E')$ é chamado de *subgrafo parcial* de G se $V' = V$ e $E' \subseteq E$. Trata-se, portanto, de um grafo obtido pela supressão de ligações do grafo original G. Alguns autores chamam H de *subgrafo gerador* (*spanning subgraph*).

Se $G_1 = (V_1,\ E_1)$ e $G_2 = (V_2,\ E_2)$ são dois (não necessariamente distintos) subgrafos de um grafo G =*(V, E)*, então o subgrafo $G' = (V_1 \cup V_2,\ E_1 \cup E_2)$ de G é chamado de união de G_1 e G_2. A Figura 2.2 traz um exemplo dos conceitos acima descritos.

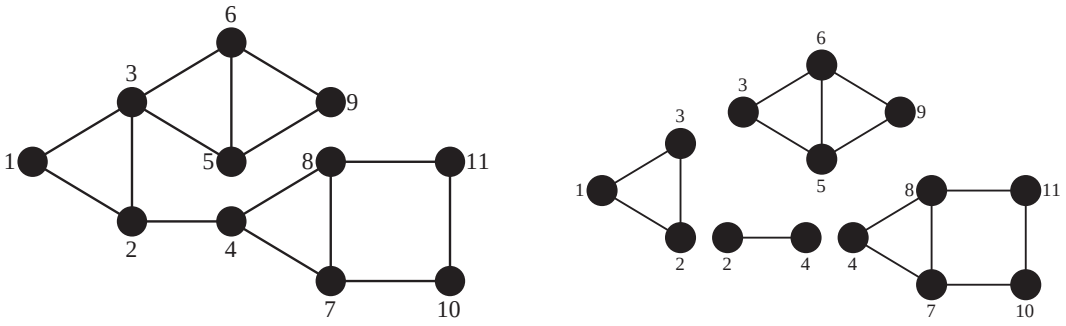


(a) Subgrafo induzido (b) Subgrafo gerador

(c) União de G

Figura 2.2: Exemplos de subgrafos

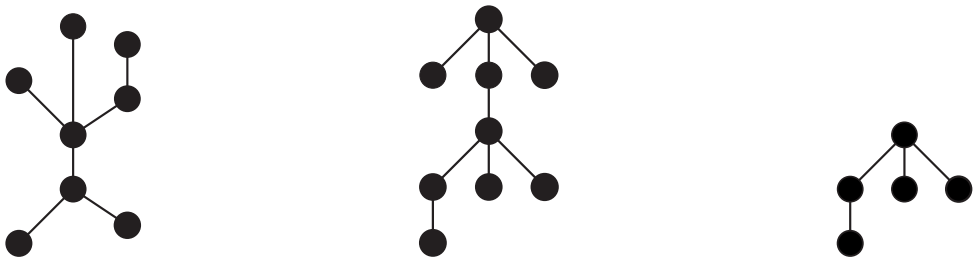
Um grafo G é *k-conexo* se pelo menos k vértices tiverem de ser removidos de G antes que o grafo resultante seja desconectado ou antes que o grafo resultante consista de somente um único vértice. Um vértice v é denominado *articulação* ou *vértice de corte* quando sua remoção de G o desconecta. Analogamente, uma aresta e é denominada de *ponte* quando a sua remoção de G o desconecta. Sendo assim, um grafo é biconexo (2-conexo) em vértices (arestas) se e somente se não possuir articulações (pontes). Denominam-se componentes biconexos do grafo G qualquer subgrafo de G que seja biconexo em vértices. Cada componente biconexo é também chamado de *bloco* do grafo. No grafo da Figura 2.3 (a), os vértices 2, 3 e 4 são articulações e a aresta (2,4) é uma ponte. Seus componentes biconexos estão indicados na Figura 2.3 (b).



(a) grafo com articulações (b) blocos

Figura 2.3: Exemplos de articulações, ponte e blocos

Um grafo conexo é uma *árvore* se cada vértice de grau maior que 1 for um *vértice de corte*, como mostra a Figura 2.4 (a). Os vértices de uma árvore são chamados de *nós*. Uma árvore "enraizada" (*rooted tree*) é uma árvore que possui uma orientação (implícita) partindo da *raiz*. A raiz é o único nó na árvore que não possui arestas incidentes, que aparece no topo da árvore, conforme Figura 2.4 (b). Nós que não possuem arestas dissidentes são chamados de *folhas*, e nós tanto com arestas incidentes como dissidentes são chamados *internos*.



(a) Livre (b) Enraizada (c) Subárvore

Figura 2.4: Exemplos de árvores

O nó raiz é o *ancestral* (*ancestor*) de todos os nós abaixo dele, e estes são os *descendentes* (*descendants*) do nó raiz. As folhas são suas próprias descendentes. Um nó não raiz, e todos seus descendentes definem uma *subárvore*, com este nó sendo a raiz da subárvore, conforme Figura 2.4 (c). Os ancestrais e descendentes imediatos de um certo nó são referenciados como *pai* e *filhos*, respectivamente. Nós que são filhos de um mesmo pai são chamados *irmãos* (*siblings*).

(a) K_5 (b) $K_{3,3}$

Figura 2.5: Exemplos de grafo completo e completo bipartido

Se todos os vértices de um grafo G são dois a dois adjacentes, então G é dito *grafo completo*. Um grafo completo com n vértices é denotado pelo símbolo K_n. Se for possível dividir os vértices de um grafo G em dois subconjuntos disjuntos, V_1 e V_2, de maneira que toda aresta de G conecta um vértice de V_1 com V_2, então G é dito como *bipartido*. Se todos os vértices de V_1 são conectados com todos os vértices de V_2 então G é dito ser um *grafo completo bipartido* denotado por $K_{n1,n2}$, onde $n_1 = |V_1|$ e $n_2 = |V_2|$. A Figura 2.5 mostra um exemplo de grafo completo e completo bipartido.

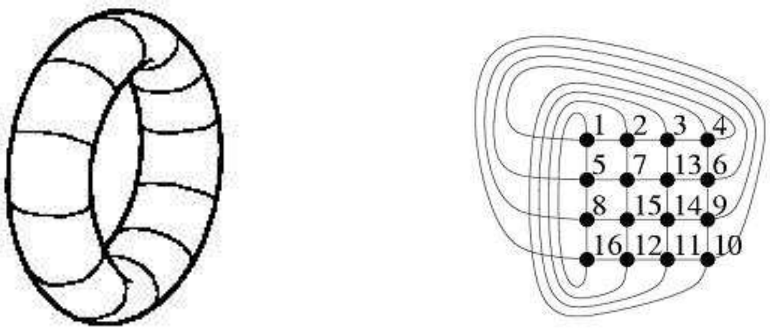
Dois grafos G_1 e G_2 são *isomorfos* se há uma correspondência 1-para-1 entre os vértices de G_1 e os vértices de G_2 tal que o número de arestas que unem qualquer dois vértices em G_1 é igual ao número de arestas que unem os correspondentes dois vértices em G_2. Por exemplo, a Figura 2.6 mostra dois grafos isomorfos, cada um sendo uma representação do $K_{3,3}$.

(a) (b)

Figura 2.6: Exemplos de grafos isomorfos

Um *ciclo* ou *circuito* C_n é um grafo com n vértices $\{v_0, v_1, \ldots, v_{n-1}\}$ tal que todo vértice v_i é adjacente exatamente a dois vértices $v_{(i-1) \ mod \ n}$ e $v_{(i+1) \ mod \ n}$ em C_n. O produto cartesiano C_{n1} x C_{n2} dos ciclos C_{n1} e C_{n2} é o grafo contendo $n_1 n_2$ vértices $\{v_{i,j}\}$ e $2n_1 n_2$ arestas $\{v_{i,j}, \ v_{i,(i+1) \ mod \ n1, \ j}\}$ e $\{v_{i,j}, \ v_{i,(j+1) \ mod \ n2}\}$, para $0 \leq i < n_1$ e $0 \leq j < n_2$.

Considerando que cada vértice do C_{n1} x C_{n2} é representado por um ponto no plano com coordenadas (i, j), esse grafo é chamado de *grafo cartesiano*[1]. Eles também são conhecidos como *grafos toroidais*, pois eles podem ser desenhados no toro sem que as arestas se cruzem e sem que haja sobreposição de arestas e vértices. O toro é uma superfície topologicamente igual a uma esfera com uma alça, ou igual a forma de um "pneu", como mostra a Figura 2.7 (a). A Figura 2.7 (b) traz um exemplo de grafo cartesiano.



(a) Toro (b) Grafo Cartesiano

Figura 2.7: Exemplo de toro e de grafo cartesiano C_3 x C_3

[1]Como esse trabalho se concentra nos grafos não planares, os grafos C_{n1} x C_{n2} aqui considerados serão somente os com $n_1, n_2 \geq 3$.

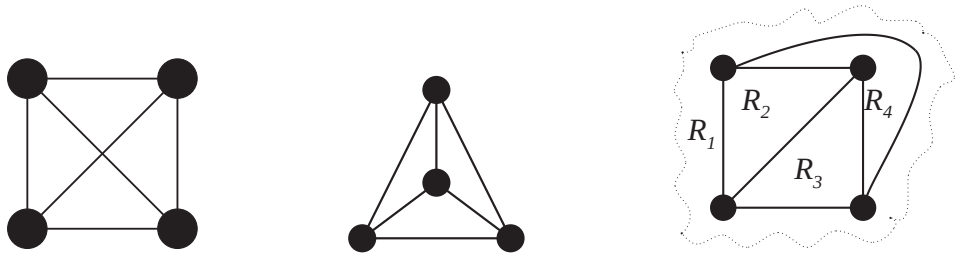
2.2 Planaridade e grafos planares

Na seção anterior foram descritos alguns conceitos gerais sobre grafos. Nesta seção, se discute as definições e propriedades dos grafos planares. Retornando à noção de representação geométrica dos grafos, um grafo planar pode ser definido como se segue:

Definição 2 (Planaridade). Seja G um grafo e R uma representação geométrica de G em um plano. A representação R é chamada *plana* quando não houver cruzamento de linhas (arestas) em R, a não ser em seus vértices, naturalmente. Um grafo G é dito *planar* quando admitir alguma representação plana.

Este conceito pode ser estendido para outras superfícies, como por exemplo a esfera. Assim, de um modo geral diz-se que o grafo G é *imersível* (*embedding*) em uma superfície S se existir uma representação geométrica R de G, desenhada sobre S, tal que duas linhas (arestas) de R não se cruzem, a não ser em seus vértices. É comum ver a denominação acima descrita de um desenho de grafo planar ser denominada de *imersão planar* (*planar embedding*) ou simplesmente uma *imersão* (*embedding*) de G.

Seja G um grafo planar e R uma *imersão* de G, em um plano P. Então as linhas de R dividem P em regiões, as quais são denominadas de *faces* de R. Todas as faces, exceto uma, são delimitadas no plano pelos vértices e arestas e elas são chamadas de *faces internas*. A face não delimitada é chamada de *face externa*. A Figura 2.8 mostra um grafo planar. O desenho do grafo em (a) não é uma imersão, mas os em (b) e (c) são, sendo que em (c) são mostradas as três faces internas $\{R_2,\ R_3$ e $R_4\}$ e uma externa $\{R_1\}$.



(a) (b) (c)

Figura 2.8: Exemplo de grafo planar com sua *imersão*

Segue-se que duas representações planas de um mesmo grafo possuem sempre o mesmo número de faces, sendo o número de faces denotado por f. Euler percebeu que o número de vértices, arestas e faces de um grafo não são independentes, e os relacionou da seguinte maneira em seu teorema:

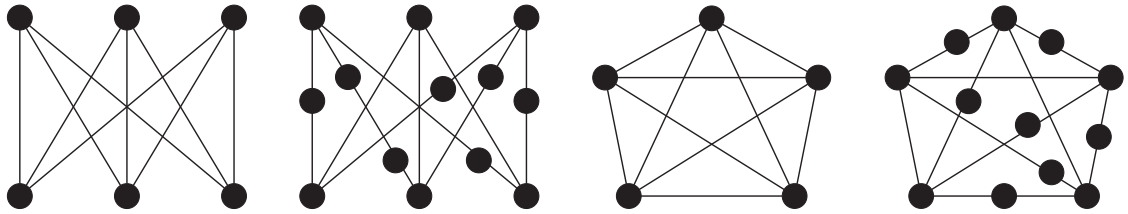
Teorema 1 (Teorema de Euler). *Seja* G *um grafo planar com n vértices, m arestas e f faces, então* $n - m + f = 2$.

Em uma representação geométrica planar de um grafo tendo o máximo possível de arestas, cada face deve ser um triângulo. O número de arestas em tal grafo é então $m = 3n - 6$, e qualquer grafo de n vértices com $m > 3n - 6$ arestas não é planar. Para um grafo planar completo bipartido, todas as faces possuem 4 arestas em sua volta, sendo que o número máximo de arestas é de $2(n_1 + n_2) - 4$. Provas destes Teoremas podem ser encontradas, por exemplo, em Nishizeki e Chiba [65] e Gibbons [30].

Uma outra maneira importante de se caracterizar um grafo planar foi publicada em 1930 por Kuratowski [51], e é normalmente conhecida como Teorema de Kuratowski. Contudo, antes de se apresentar o Teorema de Kuratowski é necessário definir somente mais uma propriedade de grafo, conhecida como *homeomorfismo*.

Denomina-se *subdivisão* de uma aresta $e = \{u,\ v\}$ é a inserção de um novo vértice w em e dividindo e em duas partes $e_1 = \{u,\ w\}$ e $e_2 = \{w,\ v\}$. Em outras palavras, se uma aresta $e = \{u,\ v\}$ de um grafo *G = (V, E)* é substituída por um caminho $u\ e'\ v_e\ e''\ v$ introduzindo um novo vértice $v_e \notin V$, então nós dizemos que um grafo G' é uma subdivisão de um grafo G, quando G' puder ser obtido de G, através de uma seqüência de subdivisões de arestas de G'. Apesar da não unicidade no conceito, alguns autores[2] definem que um grafo G' é *homeomorfo* a um grafo G se o primeiro pode ser obtido do segundo por uma seqüência de divisões de arestas. A Figura 2.9 traz exemplos de grafos homeomorfos ao $K_{3,3}$ e ao K_5, respectivamente.

Teorema 2 (Teorema de Kuratowski). *Um grafo G é planar se e somente se ele não contém como subgrafo grafos homeomorfos ao* $K_{3,3}$ *e ao* K_5.



(a) $K_{3,3}$ (b) Homeomorfo (c) K_5 (d) Homeomorfo

Figura 2.9: Exemplos de grafos homeomorfos

[2]O conceito aqui utilizado pode não ser o mais usualmente encontrado, porém é o apropriado para a definição de grafos planares, utilizada, por exemplo em Bondy e Murty [9] e Even [23].

Capítulo 3
Métodos de Planarização

Este capítulo apresenta uma breve descrição dos trabalhos relacionados à área de planarização de grafos, com destaque as principais abordagens e as medidas de não-planaridade associadas a elas. Para uma maior explanação sobre o tópico sugerimos ao leitor o artigo de Liebers [56].

Quando um grafo G não pode ser desenhado no plano sem cruzamentos de arestas, pode-se efetuar operações de *planarização* de maneira que o grafo resultante *G'* seja planar. Há diversas operações que *planarizam* um grafo, isto é, que alteram a forma do grafo original para que o grafo resultante seja planar. Toda técnica de planarização altera alguma parte da estrutura do grafo original. Dentre as operações conhecidas podem ser citadas as de *remoção* de vértices; *remoção* de arestas; *particionamento planar* e introdução de vértices *dummy*.

A cada uma dessas operações está associado uma medida de não-planaridade, também conhecida pelo termo *medida* de planaridade, que quantifica o quanto um grafo está distante de ser planar. Nas linhas a seguir, irá-se descrever algumas das principais abordagens à planarização e suas respectivas medidas de não-planaridade.

Antes de iniciar-se as descrições das operações um importante conceito deve ser explicado, pois ele permeia todos as operações abaixo e suas medidas de não-planaridade. Seja S um conjunto e $S' \subseteq S$. Diz-se que S' é *maximal* em relação a uma certa propriedade P, quando S' satisfaz a propriedade P e não existe subconjunto $S'' \supset S'$, que também satisfaz P. Observe que a definição acima não implica necessariamente que S' seja o maior subconjunto de S satisfazendo P. Quando isso ocorre, diz-se que S' é *máximo*. De maneira análoga, define-se também conjunto *minimal* em relação a uma certa propriedade. Sendo que o menor de todos é denominado de *mínimo*.

3.1 Remoção de vértices

Nesta operação, um vértice é removido com seu conjunto de arestas incidentes com o objetivo de tornar um dado grafo *G = (V, E)* planar. Esta operação é repetida o menor número de vezes possível para que o grafo *G'* resultante seja um subgrafo induzido máximo. Este

método tem diversas aplicações na Teoria dos Grafos, mas não tem aplicação para desenhos automáticos de grafos, principalmente por que a operação destrói a informação da estrutura original do grafo, conforme destaca Liebers [56]. Uma definição formal deste conceito pode ser:

Definição 3 (Subgrafo Planar Induzido Máximo). Seja $G' = (V',\ E')$ um subgrafo planar induzido de um grafo $G = (V,\ E)$ tal que não exista um subgrafo planar $G'' = (V'',\ E'')$ de G com $|V''| > |V'|$, então G' é dito ser um subgrafo planar induzido máximo de G.

O problema de remover a menor quantidade possível de vértices de um grafo para que o grafo resultante seja planar significa encontrar, para um dado grafo G, o máximo subgrafo induzido de G. A esse problema o *número de vértices removidos* (*vertex deletion number*) de um grafo G, denotado por $vd(G)$, é o número mínimo de vértices que precisam ser removidos de G para produzir um grafo planar. Tal problema foi apresentado da seguinte forma por Garey e Johnson [28]:

Problema 3 (Subgrafo Planar Induzido Máximo).

> **Instância:** Um grafo $G = (V, E)$ e um positivo inteiro $K \leq |V|$.
> **Pergunta:** Existe um subconjunto $V' \subseteq V$ com $|V'| \geq K$ tal que subgrafo G induzido por V' seja planar?

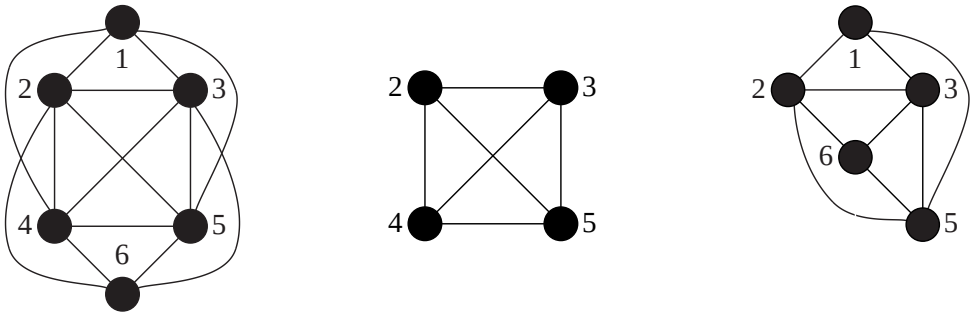
Como muitos problemas em combinatória e otimização, o problema de encontrar o subgrafo planar induzido máximo também é reportado como sendo NP-Completo, como por exemplo nos trabalhos de Garey e Johnson [28, 29] e de Lewis e Yannakakis [55]. Por esse motivo, os pesquisadores concentram suas esforços em um problema mais "tratável" que é encontrar o subgrafo planar induzido maximal, que Liebers [56] definiu da seguinte forma:

Definição 4 (Subgrafo Planar Induzido Maximal). Seja $G' = (V', E')$ um subgrafo planar induzido de um grafo $G = (V, E)$ tal que todo subgrafo induzido de G pelo conjunto de vértices $V'' = V' \cup \{v\}$ com $v \in V \backslash V'$ é não planar, então G' é dito ser um subgrafo planar induzido maximal de G.

Desse modo, dado um grafo G o objetivo é encontrar um subgrafo planar induzido maximal. Note que todo subgrafo planar induzido máximo é também um subgrafo planar induzido maximal, mas o contrário não é verdade. Isto porque o subgrafo planar induzido maximal é maximal com respeito a inclusão de vértices em seu conjunto de vértices,

enquanto que um subgrafo planar induzido máximo é máximo com respeito a cardinalidade de seu conjunto de vértices. Definição análoga será realizada na próxima seção com relação a remoção de vértices.

Na Figura 3.1 (a) o Grafo G não é planar, pois contém um K_5 como subgrafo. Em 3.1 (b) G_1 é um subgrafo planar induzido maximal de G. Já em (c), G_2 além de ser um subgrafo planar induzido maximal de G, também é um subgrafo planar induzido máximo de G.



(a) G (b) G_1 (c) G_2

Figura 3.1: Exemplo de subgrafo planar induzido maximal e máximo

3.2 Remoção de arestas

Se um grafo $G = (V, E)$ com sua aresta $e \in E$ é transformado em um grafo $G' = (V, E \setminus \{e\})$, então dizemos que G' foi obtido de G pelo método de *remoção de aresta*. Aplicando-se esta operação repetidamente para um dado grafo G não planar, o mesmo pode ser transformado em um grafo planar G'. O objetivo é encontrar um subgrafo planar com o maior número de arestas possíveis, isto é, o objetivo é planarizar um grafo G pela remoção do menor número de arestas possíveis. Liebers [56], colocou da seguinte maneira esse conceito:

Definição 5 (Subgrafo Planar Máximo). Se um grafo $G' = (V,\ E')$ é um subgrafo planar de um grafo $G = (V,\ E)$ tal que não exista um subgrafo planar $G'' = (V,\ E'')$ de G com $\mid E'' \mid > \mid E' \mid$, então G' é dito ser um subgrafo planar máximo de G.

O conjunto de arestas removidas de G, $|E| - |E'|$, é chamado de *número de arestas removidas* (*skewness*) de G e é denotado por $sk(G)$. O $sk(G)$ é 0 se e somente se G for planar. Esta abordagem tem sido estudada intensivamente na maioria dos algoritmos de planarização encontrados na literatura, tais como os de Chiba, Nishioka e Shirakawa [15], Hopcroft e Tarjan [39], Ozawa e Takahashi [66], Lempel, Even e Cederbaum [54] e Jayakumar, Thulasiraman e Swamy [44]. É grande sua aplicação também em desenhos de

grafos planares, como, por exemplo, afirma Di Battista et al. [5]. Garey e Johnson [28] definiram o problema de encontrar o subgrafo planar máximo como segue:

Problema 4 (Subgrafo Planar Máximo).

> **Instância:** Um grafo $G = (V, E)$ e um positivo inteiro $K \leq |E|$.
> **Pergunta:** Existe um subconjunto $E' \subseteq E$ com $|E'| \geq K$ tal que o subgrafo $G' = (V,\ E')$ seja planar?

Como na seção anterior, o problema de encontrar o menor número de arestas que removendo-as, resulta em um subgrafo planar máximo foi provado ser um problema NP-Completo por diversos autores, entre eles Liu e Geldmacher [57], Yannakakis [80] e Watanabe et al. [74]. Como encontrar o subgrafo planar máximo é um problema NP-Completo, os pesquisadores concentram seus esforços em encontrar o subgrafo planar maximal que foi definido por Libers [56] como segue:

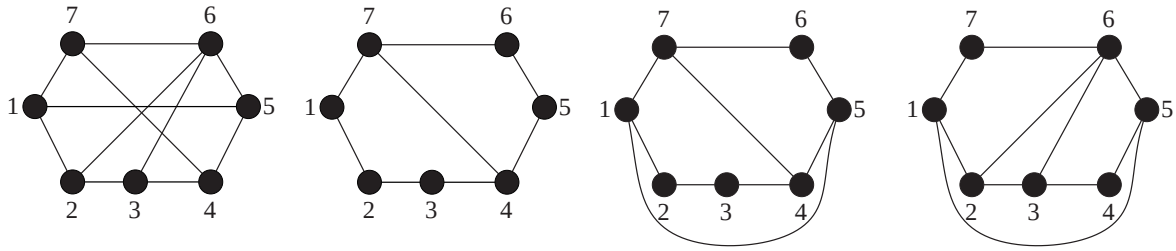
Definição 6 (Subgrafo Planar Maximal). Seja $G' = (V, E')$ um subgrafo planar de um grafo $G = (V, E)$ tal que todo subgrafo $G'' \in \{(V,\ E' \cup \{e\}) \mid e\ \in E \setminus E'\}$ é não planar, então G' é dito ser um subgrafo planar maximal de G.

Em outras palavras, um subgrafo planar maximal é maximal em relação a inclusão de arestas em seu conjunto de arestas, enquanto que um subgrafo planar máximo é máximo com relação a cardinalidade de seu conjunto de arestas. Aqui, também cabe a mesma observação feita na seção anterior, de que todo subgrafo planar máximo é também maximal, mas o contrário não. Estas definições são análogas às definições 3 e 4 que se referem ao conjunto de vértices.

A Figura 3.2 traz exemplos de subgrafos planares máximos e maximais. Em (a) G é um grafo não planar, pois contém um homeomorfo ao $K_{3,3}$ como subgrafo. Em (b) G_1 é um subgrafo planar, mas não é maximal: a aresta $\{1, 5\}$ pode ser adicionada a G_1 sem destruir a planaridade. O resultado está em G_2 em (c). Outro subgrafo planar maximal de G é G_3 em (d). Note que G_3 é também um subgrafo planar máximo.

3.3 Particionamento planar

Nas seções anteriores foram vistas as operações de remoção de vértices e arestas de um grafo G com o objetivo de obter um subgrafo planar G'. Agora o objetivo desta operação é dividir o conjunto de arestas do grafo em um número mínimo de partições de maneira



(a) G (b) G_1 (c) G_2 (d) G_3

Figura 3.2: Exemplos de subgrafos máximos e maximais

que cada uma delas seja um subgrafo planar de G. Posto de outra forma, o objetivo é obter um conjunto de subgrafos planares cuja união seja G. Dessa forma a definição pode ser dada como segue:

Definição 7 (Número de particionamentos). O *número de particionamentos* (*thickness*) de um grafo G, denotado por $tk(G)$, é o número mínimo se subgrafos planares cuja união seja G.

Obviamente o grafo será planar se e somente se seu $tk(G) = 1$. Como exemplo, considere os dois subgrafos planares do $K_{3,3}$ cuja união é o próprio $K_{3,3}$ na Figura 3.3, e os três subgrafos do K_9 cuja união é o K_9 na Figura 3.4.

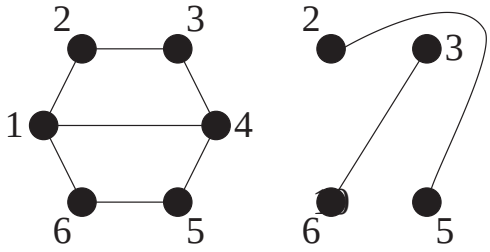
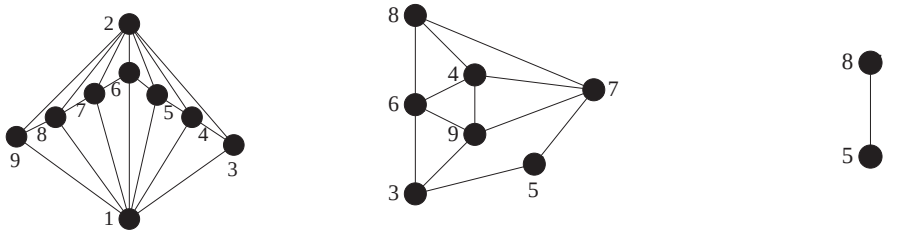


Figura 3.3: Dois subgrafos planares do $K_{3,3}$ cuja união é o $K_{3,3}$

Como o $K_{3,3}$ é o menor grafo bipartido completo não planar, sua decomposição em dois subgrafos planares mostra que $tk(K_{3,3}) = 2$. O *número de particionamentos* do K_9 já não é tão fácil de se determinar. A prova de que o $tk(K_9) = 3$ pode ser encontrada em Battle, Harary e Kodama [6] como também, posteriormente, em Tutte [72].

É interessante notar que se os grafos na Figura 3.3 tiverem seus desenhos sobrepostos de maneira que cada vértice identificado por seu número i no primeiro subgrafo caia exatamente sobre o vértice identificado como i do segundo subgrafo, então não se tem somente dois subgrafos cuja união é o $K_{3,3}$, mas sim duas imersões planares de dois grafos cuja união fornece o desenho de um $K_{3,3}$. Note que os três subgrafos do K_9 na Figura 3.4

são desenhados de maneira que sua união não resulte em uma representação geométrica planar do K_9.



(a) G_1 (b) G_2 (c) G_3

Figura 3.4: Três subgrafos planares cuja união é o K_9

O problema de encontrar a menor quantidade possível de subgrafos cuja união resulte em um grafo planar também foi provado por Mansfield [59] como sendo um problema NP-Completo. O autor postulou da seguinte forma:

Problema 5 (Número de Particionamentos).

> **Instância:** Um grafo $G = (V, E)$ e um positivo inteiro K.
> **Pergunta:** Existe um subconjunto $G' \subseteq G$ com $|G'| \geq K$ tal que a união dos subgrafo de $G' = G$?

Este método vem sendo amplamente estudado, destacando-se os trabalhos de Mutzel, Odenthal e Scharbrodt [64] e de Hobbs [38], uma vez que sua vantagem é preservar o número de arestas de um grafo dado. Sua aplicação prática está concentrada no projeto de circuitos impressos para a qual cada partição torna-se uma camada em uma placa com múltiplas camadas. Entretanto, como destaca Liebers [56], os algoritmos que se dispõe a encontrar o *número de particionamento* de um grafo são poucos na literatura.

3.4 Operações e medidas adicionais

Além destas operações e medidas de planaridade apresentadas, outras também tem sido estudadas, como por exemplo a operação de *dummy* vértices e a medida associada a ela, o *número de cruzamentos* (*crossing number*) evidenciando, segundo Liebers [56], os trabalhos de Guy [34] e Turán [71]. Em desenhos de grafos como também em outras áreas de aplicação tais como em projeto de circuitos VLSI, o interesse é obter o desenho de um dado grafo com a menor quantidade possível de cruzamentos de arestas, sendo a definição, dada por Liebers [56], como se segue:

Definição 8 (número de cruzamentos). O número de cruzamentos de um grafo G, denotado por $cr(G)$, é o menor número K tal que G possa ser desenhado no plano com no máximo K cruzamentos de arestas.

Obviamente, o $cr(G)$ é 0 se e somente se o grafo é planar. Dado um grafo G com n vértices, m arestas e sua representação geométrica no plano com um $cr(G) > 0$, o mesmo pode ser transformado em grafo planar pela introdução $cr(G)$ vértices novos no lugar dos cruzamentos de arestas. O grafo novo G' tem $n + cr(G)$ vértices e $m + 2cr(G)$ arestas. A essa operação é dado o nome de *dummy* vértices. A Figura 3.5 mostra essa operação. Em (a) tem-se o K_5 desenho com 5 cruzamentos de arestas. Em (b) sua imersão possuí somente um cruzamento, pois o $cr(K_5) = 1$ e por fim, em (c) O vértice v foi introduzido exatamente entre as arestas que se cruzam para se planarizar o K_5.

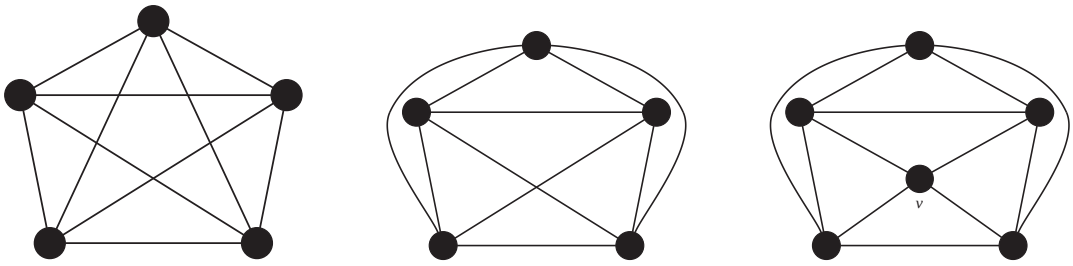


Figura 3.5: Exemplo de operação de *dummy* vértice

Essa operação é repetida até que o grafo se torne planar e sua aplicação em desenhos de grafos tem recebido considerável atenção, como por exemplo em [5] que descreve os seguintes passos: introduzir vértices *dummy* para planarizar o grafo; então aplica-se um algoritmo de desenho de grafo no grafo planar; depois elimina-se os vértices *dummy* e se re-introduz os cruzamentos de arestas no desenho. Essa operação também é utilização na ferramenta *AGD - Algorithm for Drawing Graph* de [32]. O problema de encontrar o *número de cruzamentos* de um grafo qualquer também é conhecido como sendo NP-Completo devido a Garey e Johnson [29] e pode assim ser descrito:

Problema 6 (Número de Cruzamentos).

> **Instância:** Um grafo $G = (V, E)$ e um positivo inteiro K.
> **Pergunta:** Existe uma imersão de G tal que $cr(G) \leq K$?

Além dessas, existe também a operação de divisão de vértices e sua respectiva medida de não-planaridade que é o *número de vértices divididos* (*splitting number*). Porém, esta operação será detalhada no capítulo 5 por ser o principal objetivo desta dissertação.

3.5 Medidas de não-planaridade

Como mencionado anteriormente, todos os problemas relacionados às medidas de não-planaridade são NP-Completos. Contudo para certas classes específicas de grafos, em geral, grafos simétricos ou com características regulares tais como K_n, $K_{n1,n2}$ e o C_{n1} x C_{n2}, ou para valores fixos de K esses problemas se tornam polinomiais, como veremos a seguir. A Tabela 3.1 a seguir apresenta alguns dos valores de medidas de não-planaridade conhecidas para as classes de grafos K_n e $K_{n1,n2}$ e C_{n1} x C_{n2}[1]. Existem outras classes de grafos e outras medidas conhecidas, porém para o escopo da dissertação estas classes acima são suficientes. Maiores detalhes podem ser encontrados nos trabalhos de Liebers [56], Xavier [79] e Even [23].

Classe	*Medida de não-planaridade*	*Referências*
Número de vértices removidos		
K_n	$vd(K_n) = n - 4$, para $n > 4$	[79]
$K_{n1,n2}$	$vd(K_{n1,n2}) = min\{n_1, n_2\} - 2$, para $n_1, n_2 \geq 3$	[79]
Número de arestas removidas		
K_n	$sk(K_n) = \frac{(n-3)(n-4)}{2}$,para $n \geq 4$	[56]
$K_{n1,n2}$	$sk(K_{n1,n2}) = (n_1 - 2) - (n_2 - 2)$, para $n_1, n_2 \geq 2$	[73]
C_{n1} x C_{n2}	$min\{n_1, n_2\} - \delta_{3,n1}\delta_{3,n2} - \delta_{3,n1}\delta_{4,n2} - \delta_{4,n1}\delta_{3,n2}$	[18]
Número de particionamentos		
K_n	$tk(K_n) = \lfloor \frac{n+7}{6} \rfloor$, para $n \geq 1, n \neq 9, n \neq 10$	[76, 6, 72, 75]
$K_{n1,n2}$	$tk(K_{n1,n2}) = \lceil \frac{n1n2}{2(n1+n2-2)} \rceil$	[8]
	quando $n1, n2$ são par e $n1 \leq n2$, então $n2 = \lfloor \frac{2k(n1-2)}{n1-2k} \rfloor$	[7]
Número de cruzamentos		
K_n	$cr(K_n) \leq \frac{1}{4} \lfloor \frac{n}{2} \rfloor \lfloor \frac{n-1}{2} \rfloor \lfloor \frac{n-2}{2} \rfloor \lfloor \frac{n-3}{2} \rfloor$,para $n \leq 10$	[33, 34, 76]
	$cr(K_n) \geq \frac{1}{120} n(n-1)(n-2)(n-3)$, para $n \geq 5$	[52]
$K_{n1,n2}$	$cr(K_{n1,n2}) \leq \lfloor \frac{n1}{2} \rfloor \lfloor \frac{n1-1}{2} \rfloor \lfloor \frac{n2}{2} \rfloor \lfloor \frac{n2-2}{2} \rfloor$,para $min(n_1, n_2) \leq 6$	[50]
	ou $K_{7,n2}$ para $n2 \leq 10$	[78]

Tabela 3.1: Medidas de não-planaridade para classes especiais de grafos

[1] δ é o símbolo Kronecher, e.g., $\delta_{i,j}$ é 1 se $i = j$ e 0 se $i \neq j$.

Capítulo 4
Árvore-PQ

Este capítulo está dividido em três seções. A primeira apresentada uma breve descrição da idéia geral da estrutura de dados árvore-PQ, compreendendo o algoritmo de redução, e a relação de combinações permitidas. A segunda comenta sobre o uso dessa estrutura de dados para teste de planaridade, fazendo um rápido retrospecto das primeiras publicações a respeito. Essa seção também comenta sobre o algoritmo de *numeração-st*, pois o grafo necessitar ter uma numeração especial para que seja utilizado nessa estrutura de dados. Por fim, a terceira seção, descreve alguns aspectos importantes da implementação da árvore-PQ de Young [81] bem como a forma de apresentação dos resultados da presente dissertação.

Sugerimos ao leitor dirigir-se ao artigo de Booth e Lueker [10], aos livros de Even [23], Nishizeki e Chiba [65], Thulasiraman e Swamy [70] ou o relatório técnico de Leipert [53] para uma visão mais aprofundada e detalhada do assunto.

4.1 A estrutura de dados Árvores-PQ

A estrutura de dados intitulada árvore-PQ, ou segundo Carmo [12] *Priority Queue Tree*, utilizada para teste de planaridade em grafos, é o ponto básico pelo qual o algoritmo Planarização por Divisão partirá. A estrutura de dados árvore-PQ foi desenvolvida por Booth e Lueker [10] para ordenar combinações[1] permitidas de um conjunto de elementos, onde alguns subconjuntos tem que ser consecutivos. As outras duas aplicações ainda descritas neste artigo referem-se a testes de *propriedade de 1's consecutivos* (*consecutive ones property*) e *grafos intervalos* (*interval graphs*).

Atualmente, a estrutura de dados árvore-PQ é empregada amplamente na teoria de grafos. Entre os vários trabalhos, pode-se citar, por exemplo: Jayakumar, Thulasiraman e Swamy [44], Jünger, Leipert e Mutzel [45, 46, 47] e Vollen [73] para planarização de grafos; Chiba et al. [13], Chiba, Onoguchi e Nishizeki [14], Kant [48, 49] para desenhos de grafos; Meidanis, Porto e Tellesna [63] e Annexstein e Swaminathan [1] para manipulação

[1] Ao longo de todo o texto será empregado o termo *combinações* ao invés de *permutações* pelo primeiro ser mais comum a língua portuguesa, apesar de no original constar *permutation*.

de matrizes; Di Battista e Nardelli [4] para grafos planares hierárquicos; Eades et al. [22] para dominância de desenhos. Além desses, sua aplicação tem sido estendida para outras áreas como, por exemplo, de biologia e química, onde existem certos tipos de combinações que exigem uma restrição com relação a ordem de seus subconjuntos.

Uma árvore-PQ T é uma árvore enraizada (*rooted tree*, ver seção 2.1, p. 20), cuja a estrutura de dados representa o conjunto de combinações sobre o conjunto U. A árvore-PQ é projetada para restringir as combinações dos elementos de U, encontrando as combinações permitidas em relação a S, uma família de subconjuntos de U. Combinações permitidas do conjunto U em relação a $S_i \in S$, são as combinações onde todos os elementos de S_i aparecem consecutivamente. A árvore-PQ assegura isto pela imposição de restrições de como seus nós serão combinados. Os nós de T, mostrados na Figura 4.1 são de três tipos:

- Folhas: que representam os elementos de U;
- Nós-P: que convencionamos representar pelo desenho de um círculo e
- Nós-Q: que convencionamos representar pelo desenho de um retângulo.

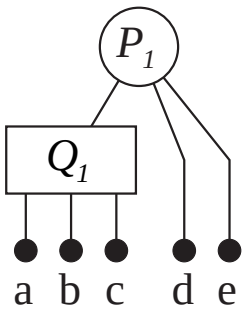
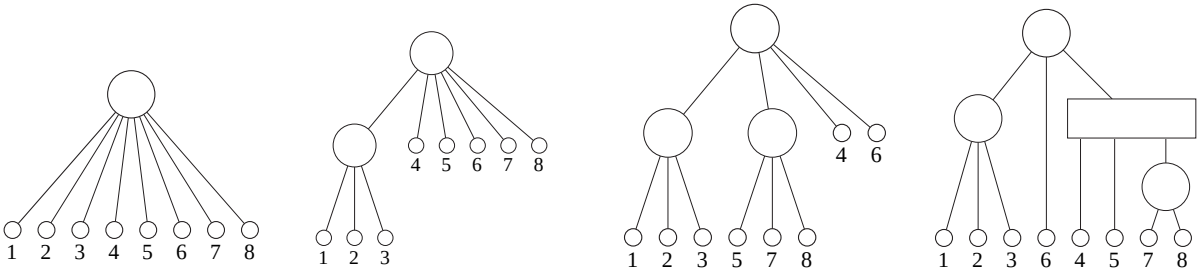


Figura 4.1: Árvore-PQ

A árvore-PQ é construída passo-a-passo. A cada passo uma nova restrição, dada por subconjunto $S \in U$, é introduzida, e a árvore é alterada de modo a representar esta restrição. Inicialmente não há restrições e a Figura 4.2 (a) mostra esta árvore inicial, no qual a raiz da árvore esta representada por um nó-P, não tendo nenhuma imposição de ordem para seus filhos.

Quando um subconjunto é introduzido, a árvore deve assegurar que os elementos do subconjunto possam combinar somente entre eles. Para representar isso, os novos elementos são agrupados sob um novo nó-P, de maneira que os elementos fora deles não se misturem, como mostra a Figura 4.2 (b). Conforme novos subconjuntos são adicionados, eles não podem ser separados ou contidos no subconjuntos já reduzidos, sendo então formada uma

ordem restrita dos elementos de U. Na árvore, isto é solucionado introduzindo um novo tipo de nó chamado nó-Q, no qual seus filhos devem manter uma ordem específica. Por isso os nós-Q são desenhados como retângulos, para representam que eles são parentes mais restritos do que os nós-P. A Figura 4.2 (c) e (d), ilustram estes novos casos.



(a) $T(U, U)$ (b) $T(U, S_1), S_1 = 1, 2, 3$ (c) $T(U, S_2), S_2 = 5, 7, 8$ (d) $T(U, S_3), S_3 = 4, 5$

Figura 4.2: Exemplo de Restrições sobre o conjunto $U = \{1, 2, 3, 4, 5, 6, 7, 8\}$

A árvore é orientada de maneira que a ordem de seus filhos é importante. A *fronteira* de T são todas as permutações permitidas de todas as folhas de T lidas da esquerda para a direita. O conjunto de combinações representados por T é gerado pelo rearranjo dos filhos de cada nó-P e Q de acordo com as regras que definem a construção de uma árvore-PQ. A Tabela 4.1 fornece os dados que consistem nas definições gerais de uma árvore-PQ e o item 6 dessa Tabela explicita as regras de combinações para cada tipo de nó.

O conjunto de combinações de S representadas por T é o conjunto das possíveis fronteiras das árvores T' obtidas a partir de T pelo rearranjo de seus filhos segundo as duas regras do item 6 da Tabela 4.1. Por exemplo, o conjunto de combinações representadas pela árvore-PQ da Figura 4.1 correspondem as doze árvores-PQ da Figura 4.3.

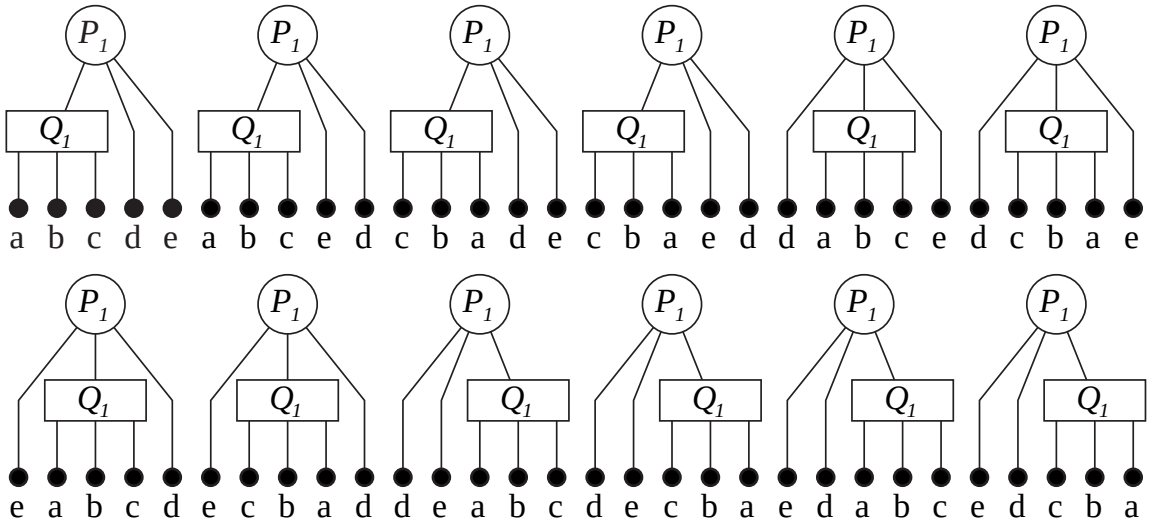


Figura 4.3: Permutações permitidas para árvore-PQ da Figura 4.1

1. A *Árvore Universal*, da Figura 4.2 (a), tem todos os elementos de U como folhas e filhos da raiz, que é um único nó-P.
2. Cada elemento $a \in U$ é uma árvore, consistindo de uma folha, tendo como raiz ele mesmo.
3. Cada elemento de U aparece exatamente uma vez, como folha, na árvore-PQ que representa U.
4. Nós-P tem pelo menos dois filhos.
5. Nós-Q tem pelo menos três filhos.
6. Duas árvores-PQ são idênticas se e somente se uma pode ser obtida da outra por zero ou mais combinações equivalentes. Há somente dois tipos de combinações permitidas:
   - Os filhos de um nó-P podem ser combinados livremente.
   - Os filhos de um nó-Q só podem ser invertidos, isto é, revertida sua ordem.
7. Quando uma árvore-PQ T é restrita com relação a S_i, a árvore é dita ser *reduzida com relação a* S_i, e a nova árvore é denominada $T(U, S_i)$.

Tabela 4.1: Definições da árvore-PQ

Há somente uma operação importante na árvore-PQ: *redução* em relação a S, sendo S um subconjunto de U. Esta operação foi provada por Booth e Lueker [10] no seguinte teorema postulado por Vollen [73]:

Teorema 7 (Teorema Fundamental da árvore-PQ). *Seja $S \subseteq U$ um subconjunto de U, e seja T uma árvore-PQ com exatamente os elementos de U em sua fronteira. Seja $T(U, S)$ a redução de T em relação a S. As combinações de U permitidas por $T(U, S)$ são então exatamente aquelas combinações permitidas por T na qual os elementos de S aparecem consecutivamente.*

Se este conjunto de combinações é vazio, então T é chamada de *irredutível* em relação a S, e $T(U, S)$ irá retornar uma árvore vazia (*null tree*). Normalmente, um conjunto U e um conjunto de subconjuntos de U, S, são dados, e a tarefa consiste em produzir uma árvore-PQ que represente todas as restrições impostas por estes subconjuntos. Para obter tal árvore uma classificação mais refinada é feita a fim de facilitar a manipulação da árvore-PQ.

Dado uma árvore-PQ T e um subconjunto S_i, todas folhas pertencentes ao subconjunto S_i são chamadas de *pertinentes*, caso contrário, elas são chamadas de folhas *não pertinentes*.

Da mesma forma, diz-se que um nó não folha X é um *nó pertinente* se qualquer uma das folhas descendentes de X na árvore-PQ T for pertinente. Se todas as folhas descendentes do nó X na árvore-PQ T forem pertinentes, diz-se que o nó X é *cheio*. De forma inversa, se nenhuma das folhas descendentes do nó X na árvore-PQ T são pertinentes, diz-se que o nó X é *vazio*. Um nó interno que não é nem cheio e nem vazio é um nó *parcial* em relação à S_i.

O conjunto de folhas descendentes de um nó lidas da esquerda para a direita é chamado de sua *fronteira*. A menor subárvore de T cuja fronteira contenha todos os elementos S_i (folhas pertinentes) é uma *subárvore pertinente* de T em relação à S_i. A raiz dessa subárvore é a *raiz pertinente*. Exemplificando, na Figura 4.4 o nó Q_1 é um nó vazio; o nó P_2 é uma raiz pertinente; o nó Q_2 é um nó cheio; as folhas pertinentes estão numeradas como o número 8; e as folhas não pertinentes estão numeradas com os números 9, 10, 11 e 12.

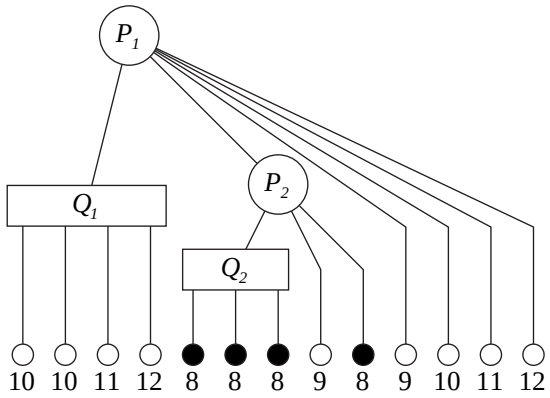


Figura 4.4: Árvore-PQ classificada em relação aos seus nós

Uma vez que a raiz pertinente tenha sido identificada o algoritmo pode, então, usar a série de *padrões de trocas - template matchings* - descritas em [10] para construir uma nova árvore em que todas as folhas pertinentes apareçam consecutivamente, se tal árvore existir. Para obter tal árvore, o algoritmo é iniciado com a árvore-PQ $T(U, U)$, e a cada $S_i \in S$, o procedimento REDUCTION(T, S_i) é chamado. O resultado final é uma árvore-PQ que representa as combinações permitidas. Neste caso, todas as folhas pertinentes aparecerão como filhas de um único nó. Além disso, a operação de redução pode ser realizada em dois passos:

- *Bolha*: neste passo é identificada a subárvore pertinente na qual a operação de redução será processada.
- *Redução*: neste passo, uma série de operações de padrões de trocas são realizadas sobre os nós-P e Q de maneira que todas a folhas pertinentes sejam colocadas consecutivamente.

Um padrão de troca - *template* - consiste um modelo, que descreve uma certa configuração da árvore-PQ, e um troca a ser realizada para esse modelo se tal configuração é encontrada na árvore. Padrões de troca são simples para as folhas, pois existem somente dois modelos. Ou uma folha é membro de S, isto é, é uma folha pertinente ou ela não é membro de S e portanto é uma folha não-pertinente. Neste caso não há nada a ser feito na árvore a não ser nomear as folhas de cheias ou vazias.

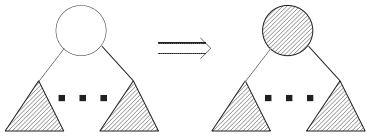
Já para qualquer outro nó interno os padrões de troca são mais complicados. O objetivo é assegurar que após a troca a fronteira da árvore enraizada até a raiz pertinente tenha todas as suas folhas pertinentes aparecendo como uma subseqüência consecutiva da fronteira. Desse modo, existem nove padrões de troca possíveis[2], todos apresentados na Figura 4.5, página 38. Para se interpretar os desenhos dos padrões de troca seguem-se algumas convenções:

- nós que são nomeados como vazios são desenhados sem preenchimento;
- nós que são nomeados cheios estão sombreados/hachurados;
- somente os nomes (cheios ou vazios) dos filhos importam. Os tipos de filhos (nós-p, nós-q ou folhas) não importam durante a operação do reconhecimento do padrão. Assim, todos os filhos estão desenhados como triângulos, desconsiderando seus tipos;
- os padrões de troca consistem de duas partes. A primeira – desenhada a esquerda – é o modelo que se procurar identificar na árvore. A segunda – desenhada a direita – é o padrão de troca que deverá ser feito, quando tal padrão se aplica.

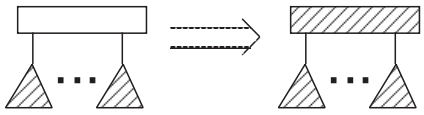
As figuras 4.5 (a) e (b) representam os dois padrões de troca mais simples, ou seja, aqueles cujo todos os filhos são cheios tanto para o nó-P quanto para o nó-Q. Nestes, basta marcar o pai também como pertinente. Já as figuras em (c) e (d) por sua vez tratam dos nós-P que possuem tanto filhos vazios quanto cheios, sendo em (c) o nó ancestral a raiz pertinente e em (d) não. Os filhos cheios devem ser colocados juntos de maneira a garantir que todas as folhas pertinentes sejam consecutivas. Isto é o que ocorre em (c), todos os filhos cheios são agrupados sob um novo nó-P que se torna, então, a raiz pertinente. Note que antes da troca ser feita todos os filhos do nó-P eram livres para combinarem-se em qualquer ordem, seja ela qual fosse. Após a troca isto não é mais possível, filhos vazios e cheios não poderão mais serem mesclados.

Muitos nós-P não são a raiz da subárvore pertinente. Além disso, sempre deve existir somente uma raiz para que a árvore seja redutível. Nestes casos, os tanto os filhos cheios como os vazios serão agrupados sob um novo nó-Q com uma nova denominação *parcial*

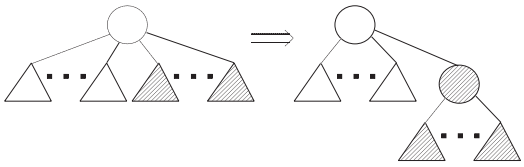
[2]P0 e Q0 de [10] não estão sendo contados, pois não são utilizados.



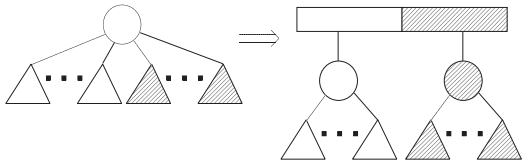
(a) nó-P com filhos cheios



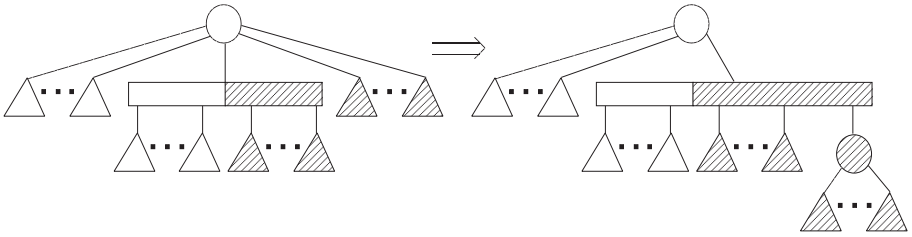
(b) nó-Q com filhos cheios



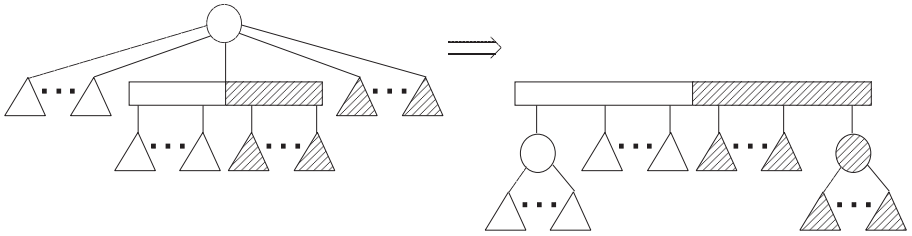
(c) nó-P (raiz) com filhos cheios e vazios



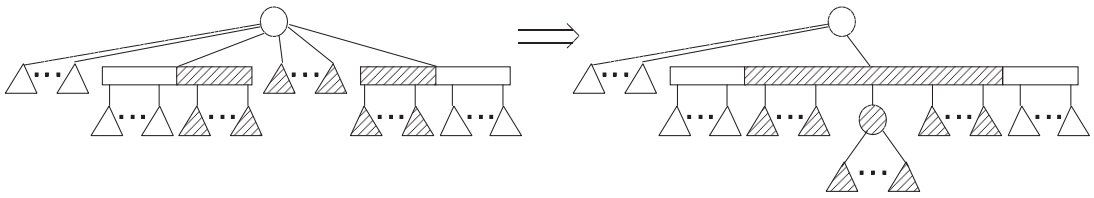
(d) nó-P (não raiz) com filhos cheios e vazios



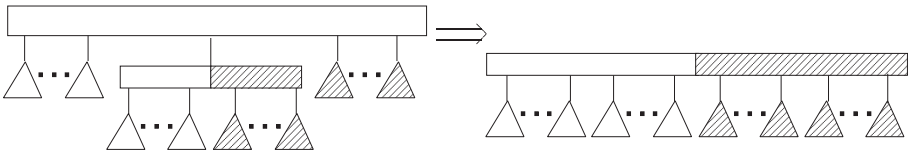
(e) nó-P (raiz pertinente) com um filho parcial



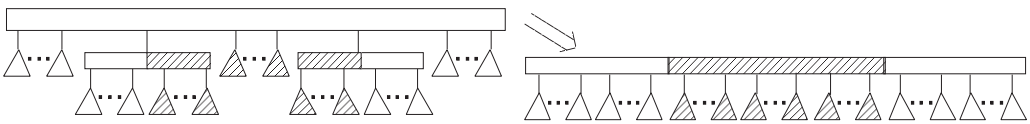
(f) nó-P (não raiz pertinente) com um filho parcial



(g) nó-P com dois filhos parciais



(h) nó-Q com um filho parcial



(i) nó-Q com dois filhos parciais

Figura 4.5: Padrões de Troca

único (*singly partial*). Ele é um novo nó que só pode aparecer sob um nó interno. Como mostrado em (d) eles são parcialmente hachurados para indicar onde estão os filhos cheios. A partir deste os demais casos podem ser derivados. As figuras em (e) e (f) tratam com nós-P que possuem um filho parcial único, sendo em (e) o nó-P a raiz pertinente e em (f) não. Caso semelhante é mostrado em (h), sendo o nó raiz um nó-Q.

Por fim, as Figuras (g) e (i) correspondem aos casos finais que lidam como nós-P e nó-Q que possuem *duplos filhos parciais únicos* (*doubly partial*). Em ambos os casos, o nó ancestral deverá ser obrigatoriamente a raiz pertinente. Isto é fácil de perceber se atentarmos para a figuras em questão. Cada filho parcial único tem um de seus extremos com filhos cheios e o outro extremo com filhos vazios. A única possibilidade de uni-los em um novo nó é o que esta apresentado nas figuras em (g) e (i). Se houver qualquer outro nó parcial em qualquer outra posição é impossível fazê-lo consecutivo com o duplo filho parcial único formado. Esta claro também que qualquer outra folha pertinente que não tenha como ancestral este mesmo nó, fica também impossibilitada de ser posta consecutivamente.

4.2 Árvores-PQ para teste de planaridade

Hoje em dia, diversos algoritmos que testam a planaridade de grafos são conhecidos, baseados em uma das duas técnicas globais, chamadas de método de *adição de aresta* e método de *adição de vértice*. Estes termos se referem ao princípio dos algoritmos. O algoritmo de adição de aresta para determinar se um dado grafo é planar ou não foi desenvolvido pela primeira vez por Auslander e Parter [2] e Goldstein [31]. Hopcroft e Tarjan [39] melhoraram o algoritmo para rodar em tempo linear. Após esta primeira abordagem, diversos versões simplificadas e relacionadas apareceram. Dentre elas podem ser citadas o trabalho de Williamson [77] e o algoritmo de Fraysseix e Rosenstiehl [16].

O algoritmo de Hopcroft e Tarjan [39] testa a planaridade de um dado grafo para cada um de seus componentes biconexos usando a seguinte idéia recursivamente: Seja $G = (V, E)$ um grafo biconexo e seja $T = (V, E')$ uma árvore de profundidade (*depth first search tree*)[3] de G com seu vértice raiz v, e deixe C ser um ciclo contendo v e consistindo de arestas de E' mais uma aresta vinda de $E \backslash E'$. Para cada aresta e de G que não é parte de C mas que tem pelo menos um dos vértices finais em C, considere um grafo G_e de G e teste (recursivamente) se ele pode ser desenhado no plano com certas arestas ao redor a face

[3]Para uma descrição do algoritmo de busca em profundidade recorrer, por exemplo, aos livros de Bondy e Murty [9], Even [23], Gibbons [30] e Manber [58].

externa. Após isto ter sido feita para cada aresta e que emana de G, teste se desenhos de diferentes subgrafos de G_e podem ser juntados ao desenho de G no plano.

O algoritmo feito por Hopcroft e Tarjan testa se um dado grafo é planar, mas não é tão claro como extrair o desenho do grafo do algoritmo, caso o grafo seja planar. Mutzel et al. [62, 61] modificaram o algoritmo de teste de planaridade para então, também, possibilitar um desenho combinatório do grafo em tempo linear.

Outro algoritmo de teste de planaridade linear foi desenvolvido por Lempel, Even e Cederbaum [54]. Os autores definiram uma *st-numeração* (*st-numbering*) como segue: Seja $G = (V, E)$ um grafo biconexo, e seja $\{s, t\} \in E$ uma aresta de G. Um grafo *st-numerado* é uma função bijetora $f : V \rightarrow \{1, 2, \ldots, |V|\}$ tal que $f(s) = 1$, $f(t) = |V|$, e para todo vértice $v \in V \setminus \{s, t\}$ existem vértices u e w em V com $\{u, v\} \in E, \{v, w\} \in E$, e $f(u) < f(v) < f(w)$.

Os autores mostraram que um grafo *st-numerado* sempre existe e pode ser executado em tempo $O(n)$. A idéia do algoritmo de teste de planaridade é a seguinte: para um grafo biconexo G, compute a *st-numeração*, e então, tente formar um grafo planar iniciando com o vértice *st-numerado* de 1 e pela adição dos vértices de G com suas arestas incidentes um por um de acordo com sua *st-numeração* ascendente. Even e Tarjan [24, 25] mostraram que uma *st-numeração* pode ser computada em tempo linear usando busca em profundidade. Usando este resultado e introduzindo a estrutura de dados da árvore-PQ, Booth e Lueker [10] aperfeiçoaram o algoritmo de planaridade de Lempel, Even e Cederbaum, executado-o em tempo linear.

O algoritmo foi modificado para também permitir um desenho combinatório para o grafo se ele for planar por Chiba et al. [13]. Even [23], por exemplo descreveu o algoritmo original de Lempel, Even e Cederbaum, e Kant [49] e Vollen [73], por exemplo, descreveram a implementação de Booth e Lueker [10] usando árvores-PQ. Recentemente, dois diferentes algoritmos de teste e desenho de grafos planares foram propostos por Hsu [68, 40, 41] e Boyer e Myrvold [11], principalmente como opção a não implementação de árvores-PQ por sua complexidade.

4.3 Aspectos da implementação

Apesar da complexidade na implementação dos algoritmos baseados na estrutura de dados da árvore-PQ, o mesmo tem sido largamente utilizado em algoritmos de teste de planaridade. Essa estrutura de dados vem sendo adaptada, para também lidar com algoritmos de planarização baseados em uma das abordagens descritas no capítulo 3. Na

literatura, a implementação de Leipert [53] e a mais conhecida. As principais razões são as idéias implícitas nela: uma implementação que prove características necessárias para sua extensão a outros propósitos, e a exaustiva documentação oferecida pelo mesmo [53]. Leipert implementou a estrutura geral da árvore-PQ de Booth e Lueker [10] como uma classe modelo (*template*) em C^{++}.

Contudo, tal implementação por seu grau de sofisticação traz consigo uma característica: a dificuldade de uma rápida assimilação e entendimento. Por tal motivo, adotou-se nessa dissertação a implementação de Young [81]. Sua implementação[4] permitiu o melhoramento de alguns aspectos da implementação original proposta por Booth e Lueker [10]. Sua implementação foi escrita em Pascal, o que simplifica seu entendimento, e as principais mudanças e melhoramentos mencionados por Young [81], p. 2, são:

1. uma abordagem *top down* usando recursão ao invés de iteração;
2. a árvore é analisada e então o trabalho requerido é feito ao invés de uma abordagem de tentativa e erro;
3. Vários melhoramentos na estrutura de dados foram adicionados para eliminar pesquisas e
4. maior atenção foi dada ao problemas de "zerar- *reseting* - a árvore para o próximo passo.

De maneira a testar o algoritmo Planariza por Divisão, foram escritos procedimentos adequados a geração dos grafos de entrada. Todos eles são biconexos e st-numerados. Cada arquivo contém um conjunto de grafos, e cada arquivo representa uma classe específica de grafos. As classes escolhidas para a análise são aquelas que possuem as medidas de não-planaridade *número de arestas removidas - skewness* e *número de vértices divididos - splitting number* conhecidos, pois assim pode ter um parâmetro de comparação da eficiência do algoritmo proposto.

As classes compreendem os grafos completos K_n, os grafos completos bipartidos $K_{n1,n2}$, os grafos completos K_n convertidos em Y_k[5] e os grafos cartesianos C_{n1} x C_{n2}. Como descrito no capítulo 5, a medida de não-planaridade *número de arestas removidas*, trata com as fórmulas utilizadas por Jayakumar, Thulasiraman e Swamy [44] para se calcular o número mínimo de arestas que necessitam ser removidas para que o grafo se torne planar evidenciando, que apesar de parecerem semelhantes as fórmulas aqui propostas, elas guardam

[4]Suas idéias são em co-autoria com os professores Richard E. Ladner e Michael Fischer.

[5]Na seção 5.1 do capítulo 5 será apresentada com detalhes como se obter essa classe de grafos e que importância ela tem para nosso prblema.

uma grande diferença implícita, por partirem de princípios totalmente diferentes para a otimização do problema, isto é, arestas a serem removidas ou vértices a serem divididos.

Cada arquivo possuí um conjunto de grafos com a seguinte forma de disposição: a primeira linha do arquivo se destina a armazenar o número total de vértices do grafo. A partir daí, para cada vértice do grafo, com exceção do n (último) é destinada uma linha do arquivo, sendo a ordem de entrada o a da st-numeração, isto é, iniciando-se do vértice rotulado de 1 até $n - 1$. Para cada vértice do grafo, delimitada entre parênteses, está sua lista de adjacência.

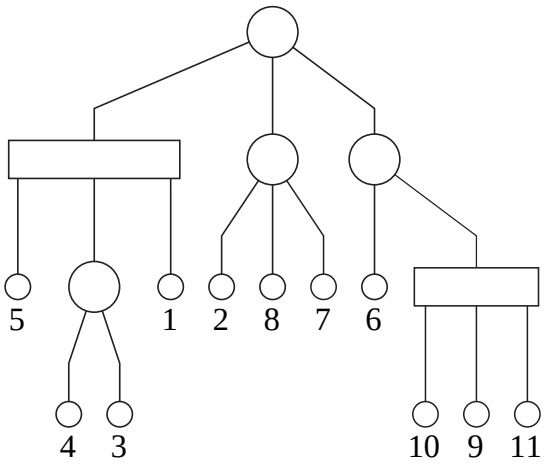
Cabe ressaltar que a lista de adjacência é um pouco diferente da normalmente apresentada nos livros, pois ela só lista os vértices que saem daquele vértice, isto é, os vértice rotulados com números maiores que aquele vértices, o que chamamos de vértices dissidentes (*outgoing*). O símbolo ";" é utilizado para marcar o final de cada conjunto de números que representa um grafo, e o final do arquivo é delimitado pelo símbolo ".". A Tabela 4.2 a seguir representam uma exemplo do arquivo `Kn.in` que traz os dois primeiros grafos completos não-planares.

```
4
1(2 3 4)
2(3 4)
3(4);
5
1(2 3 4 5)
2(3 4 5)
3(4 5)
4(5);
```

Tabela 4.2: Exemplo de arquivo de entrada

Como saída, os resultados são expressos de duas formas diferentes. Cada passo da redução da árvore-PQ está representado de forma gráfica e por uma expressão. Na expressão os filhos de um nó-P são delimitados pelos símbolos "<" e ">", enquanto que os filhos de um nó-Q são delimitados pelos símbolos "[" e "]" e as folhas que representam a fronteira da árvore são rotuladas com seus respectivos número advindos da st-numeração. Já a forma gráfica é bem intuitiva e utiliza as formas geométricas descritas na seção anterior, isto é,

círculos hachurados para os nós-P, retângulos para nós-Q e círculos sem preenchimento e menores para folhas. A Figura 4.6 a seguir, exemplifica essas duas formas.



(a) Representação gráfica

⟨ [5 ⟨4 3⟩ 1] ⟨ 2 8 7 ⟩ ⟨ 6 [10 9 11] ⟩ ⟩

(b) Representação algébrica

Figura 4.6: Formas de apresentação dos resultados

Capítulo 5
Planarização por divisão de vértices

Muitas heurísticas que buscam remover um número mínimo de arestas para que o grafo resultante seja planar têm sido reportadas [66, 15]. A maioria desses algoritmos não são adequados para adaptação para o Grafo Planar Dividido (*Split Planar Graph*). Uma das melhores tentativas é o algoritmo *Planarize* de Jayakumar, Thulasiraman e Swamy [44]. O algoritmo usa complexidade de tempo de $O(n^2)$ e de espaço $O(m + n)$ para um grafo com n vértices e m arestas.

O algoritmo PPD proposto utiliza idéias similares as do algoritmo de Jayakumar, Thulasiraman e Swamy para realizar operações de divisão de vértices ao invés de remoção de arestas. Como esse, o algoritmo PPD também está baseado no algoritmo de teste de planaridade de Lempel, Even e Cederbaum [54], e sua implementação utiliza a estrutura de dados da árvore-PQ, de Booth e Lueker [10].

Esta seção está divida em duas subseções. A subseção 5.1 aborda os principais conceitos relacionados ao método de divisão de vértices e a medida de não-planaridade a ele associada: o *número de vértices divididos* (*splitting number*). Mostra também que, as operações de divisão de vértices e remoção de arestas são problemas distintos.

Já a subseção 5.2 trata com o detalhamento do algoritmo PPD em si, fornecendo uma visão geral do algoritmo. Faz isso explicitando como os conceitos da estrutura de dados da árvore-PQ, da representação geométrica do grafo chamada de *forma arbusto* e da st-numeração se relacionam para nosso problema. A partir desse ponto, é detalhado como o algoritmo executa sua tarefa.

Ainda dentro da subseção 5.2 são abordados os detalhes dos dois principais passos do algoritmo Planariza por divisão. A subseção 5.2.1 explicita como calcular o número de grupos pertinentes que necessitam ser renomeados para que o grafo seja planar. A subseção 5.2.2 de posse desses números, irá decidir como combinar esses grupos, de maneira obter esse grupos pertinentes. A análise da complexidade do algoritmo é tratada na seção 5.3. Essa, traz também, na forma de um pseudo-código em pascal, o algoritmo 1 Planarização por Divisão.

5.1 Divisão de vértices

Intuitivamente, um vértice v pode ser dividido fazendo-se duas cópias v_1 e v_2 e ligando-se as arestas incidentes em v com v_1 ou v_2. Esta operação esta ilustrada na Figura 5.1. Uma definição formal desta operação segue:

Definição 9 (Divisão de Vértice). A *operação de divisão de vértice*, ou simplesmente, divisão, de um vértice $v \in V(G)$ particiona (divide) o conjunto de vizinhos (os vértices adjacentes) de v em dois subconjuntos não vazios P_1 e P_2 e adiciona a $G \setminus v$ dois novos e não adjacentes vértices v_1 e v_2, tal que P_1 é o conjunto de vizinhos de v_1 e P_2 é o conjunto de vizinhos de v_2

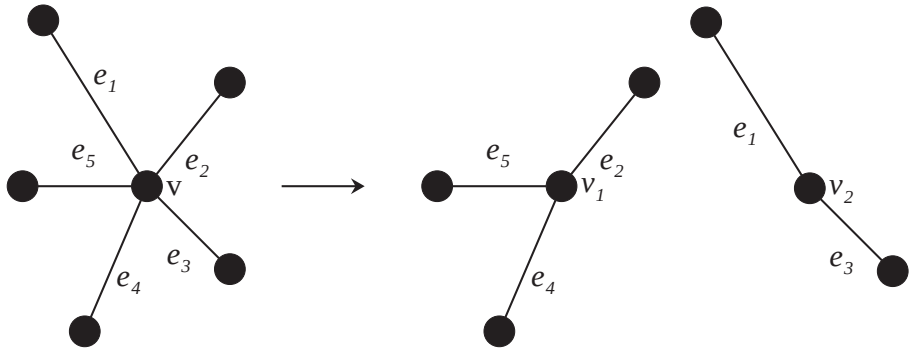


Figura 5.1: Exemplo de operação de divisão de vértice

Se um grafo G' é obtido de um grafo G por uma seqüência de k operações de divisão de vértices, ele é chamado de *grafo dividido de* G ou também de *grafo resultante de* G, conforme convencionaram Faria, Figueiredo e Mendonça [27]. A definição de divisão de vértices usada para nosso problema não se restringe a dividir um vértice v em exatamente dois vértices. Como será demonstrado mais adinate, existirá situações em que um determinado vértice poderá ser dividido em mais dois vértices.

Esta operação, como indica Liebers [56] tem aparecido em vários diferentes contextos, entre eles, pode ser citado como exemplo, a decomposição de grafos em seus blocos biconexos que significa realizar a divisão de vértice para cada vértice de corte. Obviamente a maior contribuição desta operação está na área de desenhos de grafos planares, que é o foco da atenção desta proposta: dado um grafo G, deseja-se conhecer o menor número k, tal que G possa ser planarizado por k operações de divisão de vértices. Em outras palavras:

Definição 10 (Número de vértices divididos). Seja G um grafo, o *número de vértices divididos* de G, denotado por $sp(G)$, é o menor numero inteiro positivo k, tal que um grafo planar possa ser obtido de G por k operações de divisão de vértices.

Obviamente, $sp(G) = 0$ se e somente se o grafo for planar. Investigações sobre o *Número de vértices divididos* parecem ter sido iniciadas por volta da década de 80, com os trabalhos de Jackson e Rangel [43] tratando com grafos completos, de Hartsfield, Rangel e Jackson [37] com grafos completos bipartidos e Schaffer [67] com grafos cartesianos C_{n1} x C_{n2}. Mais recentemente destacam-se os trabalhos de Eades, Faria, Figueiredo e Mendonça [20, 17, 26, 27], em estabelecer a NP-completude do problema do número de vértices divididos dado e de Mendonça e Xavier para classes específicas de grafos, como o Q_4 e o C_{n1} x C_{n2} [79].

Encontrar o número de vértices divididos de um grafo G somente foi provado ser NP-Completo recentemente por Faria, Figueiredo e Mendonça [27], sendo conhecido como o problema do *Grafo Planar Divido* (*Split Planar Graph*). Esse estudo iniciou-se com tese de doutorado de Mendonça [17] a respeito da definição de dois problemas e da prova de que o primeiro deles era NP-completo, como mostrado a seguir:

Problema 8 (Conjunto Elegível do Grafo Planar Dividido).

> **Instância:** Um grafo $G = (V, E)$, um subconjunto de vértices elegíveis de S e um positivo inteiro $K \leq |S|$.
> **Pergunta:** Existe um conjunto menor ou igual a K operações de divisão de vértices sobre S que resulte em um grafo planar?

Teorema 9. *O Conjunto Elegível do Grafo Planar Dividido é NP-Completo.*

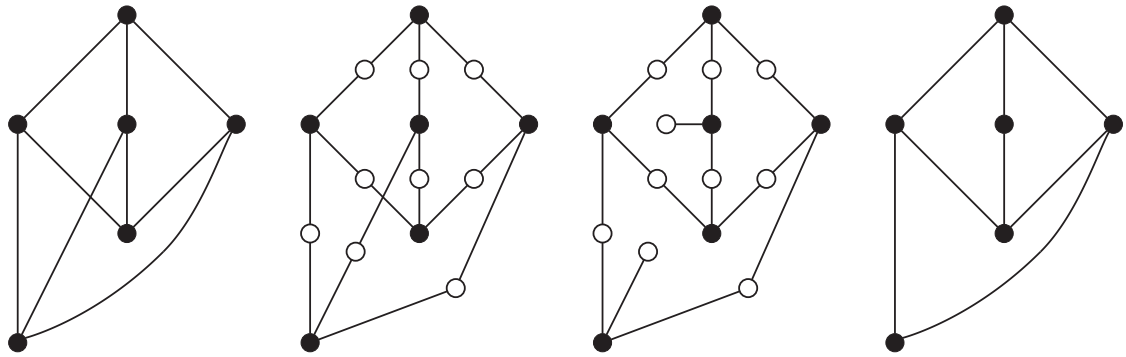
Prova: Mendonça [17] provou o Teorema 9, acima, pela transformação do problema do *Subgrafo Planar Máximo* (problema 4, página 27) no problema 8, acima.

Problema do Subgrafo Planar Máximo [28]

> **Instância:** Um grafo $G = (V, E)$, e um positivo inteiro $K \leq |E|$.
> **Pergunta:** Existe um subconjunto $E' \subseteq E$ com $|E'| \geq K$ tal que $G' = (V,\ E')$ seja planar?

Deixe o grafo $G = (V, E)$ e o inteiro positivo $K \leq |E|$ ser uma instância arbitrária do problema do subgrafo planar máximo. Construa uma instância do conjunto elegível do grafo planar dividido como segue: Substitua, cada aresta $e = \{uv\} \in E$ por um caminho $ue'v_e e''v$, isto é, o mesmo que subdividir cada aresta e uma vez. Chame o grafo resultante de H. Deixe $K' = |E| - K$ (note que $K' \leq |E| = |S|$), e deixe $S = \{v_e | e \in E\}$ ser o conjunto de vértices criados através das subdivisões. Logo, H, S e K' definem uma instância do problema do conjunto elegível do grafo planar dividido. O grafo G tem um subgrafo

planar com K ou mais arestas se e somente se H pode ser planarizado por K' operações de divisão de vértices. É fácil de se ver que esta instância pode ser construída em tempo polinomial. Além disso, para planaridade, remover uma aresta e em G é equivalente a dividir o vértice v_e em G'.



(a) G (b) H (c) H' (d) G'

Figura 5.2: Redução para o conjunto elegível do grafo planar dividido

A Figura 5.2 mostra os passos da redução para o $K_{3,3}$. Em (a) tem-se um tradicional grafo G bipartido completo não planar $K_{3,3}$. Em (b), cada aresta é subdividida, inserindo-se um vértice (em branco). Em (c), um dos vértices criados precisou ser dividido para que o grafo H em (b) fosse planarizado, criando-se o grafo H'. De forma alternativa, a remoção da aresta que foi subdividida pela operação de divisão de vértice no grafo H' em (c), culminou em um subgrafo planar G' em (d). □

Problema 10 (Grafo Planar Dividido).

> **Instância:** Um grafo $G = (V, E)$, e um positivo inteiro $K < |S|$.
> **Pergunta:** Existe um conjunto menor ou igual a K operações de divisão de vértices sobre V que resulte em um grafo planar?

Apesar de Mendonça [17], naquela época, não ter encontrado solução para o segundo problema 10 acima, ele evidenciou que para as classes de grafos com $d(G) \leq 3$, os problemas do Grafo Planar Dividido e do Subgrafo Planar Máximo são equivalentes. Finalmente, em 2001, Faria, Figueiredo e Mendonça [27] estabeleceram a complexidade do problema do Grafo Planar Dividido e a prova usa uma redução um tanto complexa do problema 3-Satisfabilidade, que sugerimos recorrer ao seu artigo para melhor compreensão. Como postulando no Teorema 11, a prova de que o Planar Dividido é NP-Completo, mesmo

quando restritos a grafos cúbicos implicou em provar que isso também é valido para problema do Subgrafo Planar Máximo.

Teorema 11. *Grafo Planar Dividido é NP-Completo, mesmo quando restrito a grafos cúbicos.*

Mendonça [17] também enfatizou a diferença entre as operações de divisão de vértices e remoção de arestas através de um exemplo interessante. Para apresentar-se o exemplo, é necessário recorrer-se a algumas definições:

- O *número de vértices divididos* (*splitting number*) de um grafo G, denotado por $\phi(G)$, é o tamanho mínimo do conjunto de operações de divisão de vértices de G que resulta em um grafo planar.
- O *Número de arestas removidas* (*edge deletion number*) de um grafo G, denotado por $\rho(G)$, é o tamanho mínimo do conjunto de arestas do grafo G cuja remoção resulta em um subgrafo planar.
- O problema do Grafo Planar Dividido está relacionado com $\phi(G)$ e o problema do Subgrafo Planar Máximo está relacionado com $\rho(G)$. A proposição a seguir utiliza $\phi(G)$ e $\rho(G)$ para avaliar a relação entre os dois problemas acima, respectivamente.

Proposição 12 (Mendonça [17]).

i) Para qualquer grafo G, $\phi(G) \leq \rho(G)$.

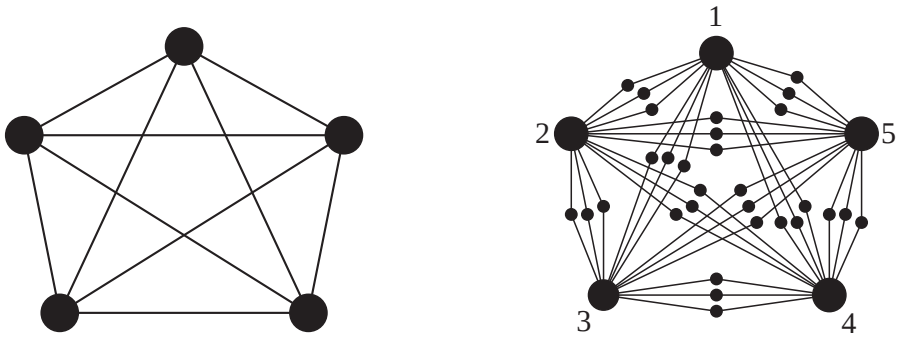
ii) Para todo número positivo inteiro n existe um grafo G com n vértices tal que $\phi(G) = \rho(G)$.

iii) Para cada número positivo inteiro k existe um grafo G com $10k + 15$ vértices tal que $\frac{\phi(G)}{\rho(G)} = k$.

Prova: Da parte (i), segue-se do fato que para cada aresta $e = \{u, v\}$ removida de um grafo G pode-se encontrar uma operação de divisão de vértice equivalente pelo divisão de um dos vértices da aresta e.

Da parte (ii), a classe dos grafos com grau de vértice $d(u) \leq 3$ é suficiente.

Da parte (iii), seja k um positivo inteiro, e seja Y_k (uma grafo K_5 engrossado) um grafo construído como se segue: Substitua cada aresta do K_5 por um grafo com k caminhos de comprimento 2. Como exemplo a Figura 5.3 mostra o grafo completo K_5 convertido no Y_3.



(a) K_5 (b) Y_3

Figura 5.3: Conversão do K_5 no Y_3

Primeiramente, Mendonça [17] mostrou que

$$\rho(Y_k) = k \tag{5.1.1}$$

Se menos do que k arestas são removidas, então para cada par $\{u,\ v\}$ de vértices do grafo K_5 original, existirá pelo menos um caminho de comprimento 2 de u para v. Dessa maneira, o grafo resultante após a remoção de $e \leq k$ arestas conterá uma subdivisão do K_5, e pelo Teorema de Kuratowski [51] 2 da página 23, o grafo não é planar. Desse modo $\rho(Y_k) \geq k$. A desigualdade na outra direção, $\rho(Y_k) \leq k$, segue do fato de que a remoção de uma aresta de cada um dos k caminhos de comprimento 2 entre o par de vértices (u, v) do grafo original K_5 resulta em um grafo planar.

Depois, Mendonça [17] mostrou que

$$\phi(Y_k) = 1 \tag{5.1.2}$$

A primeira desigualdade $\phi(Y_k) \geq 1$ é evidente, uma vez que o Y_k é uma subdivisão do K_5, e como tal, é não planar. Para provar que $\phi(Y_k) \leq 1$, basta mostrar que uma única operação de divisão de vértice produz um grafo planar. Escolha um vértice v a partir do grafo original K_5. Suponha que v tenha como vizinhos os vértices a, b, c e d do grafo original K_5. Agora, divida v em Y_k em duas cópias v_1 e v_2 tal que todo caminho de comprimento 2 de v para a e b seja incidente a v_1 e todo caminho de comprimento 2 de v para c e d seja incidente v_2. Isto resulta em um grafo planar, conforme a Figura 5.4 mostra, onde o vértice rotulado de 4 foi dividido no $v_{4.1}$ e $v_{4.2}$.

Segue então, de 5.1.1 e 5.1.2, que $\frac{\phi(G)}{\rho(G)} = k$ □

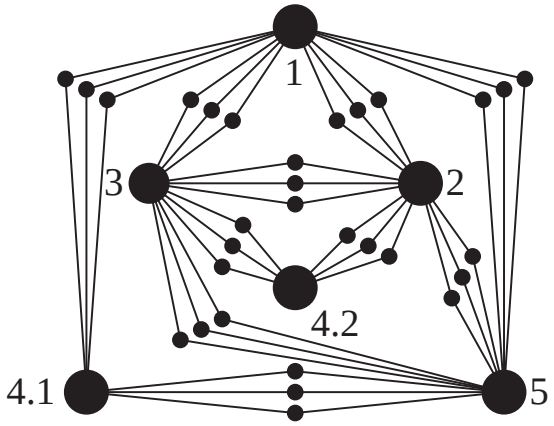


Figura 5.4: Grafo Planar Dividido do Y_3

Como pode ser visto, remover arestas de um grafo não é a mesma coisa que dividir vértices. A principal vantagem da operação de divisão de vértices reside no fato de que ela não destrói a principal informação do grafo dado, ou seja, o "relacionamento existentes entre os vértices". Tome como exemplo, uma abstração da representação de um banco de dados por intermédio de grafos. Se uma aresta for removida, a principal informação, que é o relacionamento entre as entidades será comprometido, para não falar perdido.

Já no método de divisão de vértices, essa informação é preservada, e por conseguinte, uma cópia dessa entidade é feita, sendo que o problema de descobrir quais vértices são cópia pode ser feito em tempo constante. A Tabela 5.1 encerra essa seção mostrando os valores conhecidos da medida de não-planaridade *número de vértices divididos* para as classes de grafos completos, completos bipartidos e cartesianos (toroidais).

Classe	*Número de vértices divididos*	*Referências*
K_n	$sp(K_n) = \lceil \frac{(n-3)(n-4)}{6} \rceil$, para $n \geq 3$ e $n \neq 6, 7$ e 9 $sp(K_n) = \lceil \frac{(n-3)(n-4)}{6} \rceil + 1$, para $n = 6, 7$ e 9	[37]
$K_{n1,n2}$	$sp(K_{n1,n2}) = \lceil \frac{(n1-2)(n2-2)}{2} \rceil$, para $n1, n2 \geq 2$	[43]
C_{n1} x C_{n2}	$min\{n_1, n_2\} - 2\delta_{3,n1}\delta_{3,n2} - \delta_{3,n1}\delta_{4,n2} - \delta_{4,n1}\delta_{3,n2}$	[18]

Tabela 5.1: *Número de vértices divididos* do K_n, $K_{n1,n2}$ e C_{n1} x C_{n2}

5.2 O algoritmo

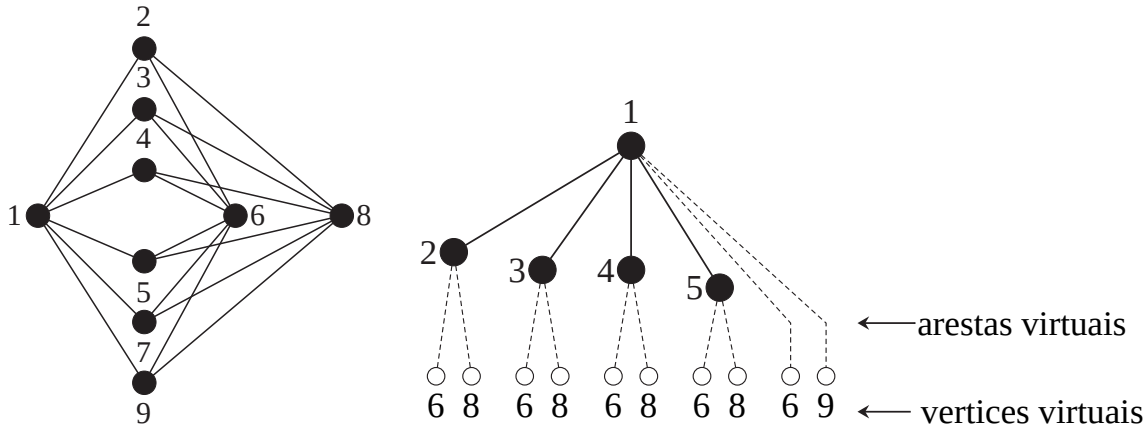
Apesar de o algoritmo de Lempel, Even e Cederbaum [54], no qual o algoritmo proposto é baseado, tratar somente com grafos biconexos, Vollen [73] comenta que planaridade não é uma propriedade que depende do grafo ser conexo ou biconexo, orientado ou não-orientado, simples ou com múltiplas arestas.

Um grafo desconexo é planar se e somente se seus componentes conexos forem planares e um grafo conexo é planar se e somente se seus componentes biconexos forem planares. Múltiplas arestas podem ser desenhadas em paralelo. Então, qualquer grafo pode ser testado com relação à sua planaridade, testando o grafo simples e não-orientado implícito nele, um componente biconexo por vez. Sendo assim, trataremos somente com grafos finitos, simples, não-orientados e biconexos, para simplificar nossa implementação.

Para se testar a planaridade de um grafo utilizando-se a estrutura de dados da árvore-PQ é necessário que os vértices do grafo tenham numeração especial. Even e Tarjan [24, 25, 69], desenvolveram um algoritmo chamado de *st-numeração* (*st-numbering*), que realiza tal essa tarefa em tempo linear. Um grafo G é chamado de *grafo-st* quando seus vértices são numerados por inteiros $1, 2, \ldots, n$, tal que, o vértice numerado como 1 é adjacente ao numerado como n e cada outro vértice j é adjacente a pelo menos um par de vértices i e k satisfazendo $i < j < k$. O vértice numerado como 1 (n) é chamado de *fonte* (*sumidouro*) e denotado por s (t). Todo grafo biconexo tem uma st-numeração, conforme provaram Lempel, Even e Cederbaum [54].

Seja G_k, $1 \leq k \leq n$, um subgrafo induzido de G pelo conjunto de vértices $V_k = \{1, 2, \ldots, k\}$. Seja B_k um grafo consistindo do subgrafo G_k e todas as arestas de G as quais emanam dos vértices de V_k e entram nos vértices de $V - V_K$ em G. Estas arestas são chamadas de *arestas virtuais* e os vértices em que elas incidem em $V - V_k$ são chamados de *vértices virtuais*. Os vértices virtuais são nomeados da mesma forma que sua correspondentes em G, mas são mantidos separados. Além disso, em B_k pode haver diversos vértices com a mesma denominação, cada um com exatamente uma arestas incidente. Um desenho de B_k é chamado de *Forma Arbusto* (*Bush Form*) de B_k se no desenho os vértices com maiores numeração aparecerem nos níveis mais altos e todos os vértices virtuais aparecerem em um mesmo nível.

A Figura 5.5 (a) mostra um $K_{6,3}$ com uma *st-numeração*, sendo que em (b) mostra-se o desenho da Forma Arbusto B_5. As linhas pontilhadas são arestas virtuais e os vértices sem preenchimento desenhados todos no mesmo nível inferior da Forma Arbusto são os vértices virtuais.



(a) Grafo-st do $K_{6,3}$ (b) Forma Arbusto B_5

Figura 5.5: Exemplo de grafo-st e sua Forma Arbusto

Já a Tabela 5.2 mostra que o grafo-st, da Figura 5.5, torna-se "orientado", podendo-se estabelecer um conjunto de arestas incidentes (*incoming*), que saem dos vértices v_i e para o vértice v_k, com $i < k$ e um conjunto de arestas dissidentes (*outgoing*) que partem do vértice v_k para os vértices v_j, com $K < J$.

Vértice	S_i (*incidentes*)	$\Rightarrow$	S_d (*dissidentes*)
1	( )	$\Rightarrow$	(1-2 1-3 1-4 1-5 1-7 1-9)
2	(1-2)	$\Rightarrow$	(2-6 2-8)
3	(1-3)	$\Rightarrow$	(3-6 3-8)
4	(1-4)	$\Rightarrow$	(4-6 4-8)
5	(1-5)	$\Rightarrow$	(5-6 5-8)
6	(2-6 3-6 4-6 5-6)	$\Rightarrow$	(6-7 6-9)
7	(1-7 6-7)	$\Rightarrow$	(7-8)
8	(2-8 3-8 4-8 5-8 7-8)	$\Rightarrow$	(8-9)

Tabela 5.2: Conjunto de arestas incidentes e dissidentes usado pelo algoritmo PPD

Conforme foi provado por Even [23] e Lempel, Even e Cederbaum [54] um *grafo-st* é planar se e somente se para cada B_k, $2 \leq k \leq n-2$, existir um desenho planar B'_k isomorfo a B_k, tal que, em B'_k todas os vértices virtuais nomeados de $k+1$ aparecerem consecutivamente. Para o problema de planarização, a árvore-PQ T_k é usada para representar a Forma Arbusto B_k no algoritmo. Os nós de T_k correspondem como se segue:

- Folhas: os vértice virtuais de B_k;
- Nós-P: vértices de corte de B_k e
- Nós-Q: os componentes biconexos máximos de B_k.

Sejam G um grafo-st biconexo e $T_1, T_2, \ldots, T_{n-1}$ uma árvore-PQ correspondente a Forma Arbusto $B_1, B_2, \ldots, B_{n-1}$ de G. Para o problema de divisão de vértices, uma reclassificação dos nós é necessária, de acordo com sua fronteira, como segue:

(i) Um nó X é do *tipo A* se a subárvore enraizada até X pode ser rearranjada tal que todas as folhas pertinentes descendentes de X aparecem consecutivamente no meio da fronteira com pelo menos uma folha não pertinente de cada ponta (final) da fronteira. Por exemplo, o nó P_1 na Figura 5.6 é do tipo A.

(ii) Um nó X é do *tipo B* se a fronteira da subárvore enraizada até X consistir somente de folhas pertinentes. Em outras palavras, X é um nó cheio. Por exemplo, o nó Q_2 na Figura 5.6 ;é do tipo B.

(iii) Um nó X é do *tipo H* se a subárvore enraizada até X pode ser rearranjada tal que todas as folhas pertinentes descendentes de X apareçam consecutivamente em uma das extremidades (finais) da fronteira. Os nós P_1 e P_2 na Figura 5.6 são do tipo H. Por exemplo, para se conseguir tal configuração para o nó P_1, o nó P_2 deve ser movido para a esquerda e os filhos de P_2 devem ser rearranjados de maneira que os vértices 8 apareçam consecutivamente à esquerda.

(iv) Um nó X é do *tipo V* se a subárvore enraizada até X pode ser rearranjada tal que todas as folhas não pertinentes descendentes de X apareçam consecutivamente no meio da fronteira com pelo menos uma folha pertinente aparecendo em cada extremidade (final) da fronteira. Por exemplo, o nó P_2 na Figura 5.6 é do tipo V.

(v) Um nó X é dito ser do *tipo W* se a fronteira da subárvore enraizada até X consistir somente de folhas não pertinentes. Em outras palavras, X é um nó vazio. Por exemplo, o nó Q_1 na Figura 5.6 é do tipo W.

Note que nem toda árvore-PQ é de um dos tipos *A, B, H, V* ou *W*. Entretanto, como mais tarde será visto, é necessário transformar (essencialmente pela divisão) as árvores em um destes tipos. Porém, antes disso, é necessário expor o conceito central do algoritmo de planarização de Ozawa e Takahashi [66]:

Teorema 13. *Um grafo G com* n-vértices *é planar se e somente se as raízes pertinentes em todas as árvores-PQ $T_2, T_3, \ldots, T_{n-1}$ de G são do tipo* B, H ou A.

O teorema acima significa dizer que uma árvore-PQ é *redutível* se sua raíz pertinente é do tipo *B, H ou A*, caso contrário ela é *irredutível*. As duas árvores T_1 e T_{n-1} são

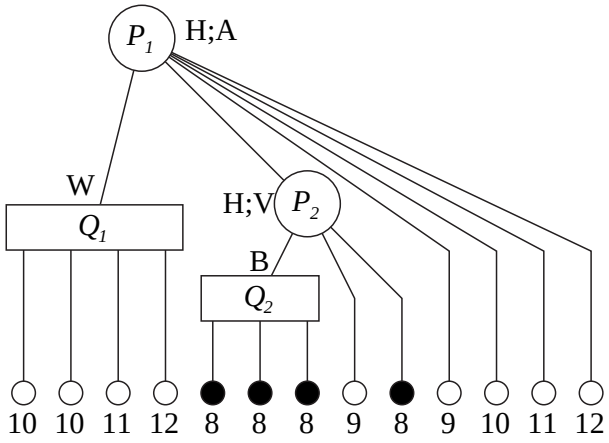


Figura 5.6: Árvore-PQ classificada com relação aos seus nós

redutíveis. A primeira árvore porque tem somente uma folha pertinente correspondente a aresta $\{1, 2\}$, e a última porque tem somente um tipo de folha pertinente, que são os nós correspondentes ao vértice virtual n.

Considere uma árvore-PQ T_i irredutível de um grafo não planar. Isto é, é impossível rearranjar T_i de maneira que as folhas pertinentes sejam consecutivas. Dessa maneira, o objetivo é fazer uma árvore cuja fronteira tenha tão poucos grupos de folhas pertinentes consecutivos quanto possível. Um grupo de folhas pertinentes consecutivas será aqui denominado de *grupo pertinente*. Suponha agora uma árvore cuja fronteira possui mais do que um grupo pertinente. O número de grupos pertinentes em uma árvore pode ser então reduzido através das operações de combinação da árvore-PQ definidas previamente. Porém, quando a árvore é irredutível, é necessário fazer com que grupos pertinentes se tornem não pertinentes renomeando suas folhas como não pertinentes. Por exemplo, a árvore da Figura 5.7 pode se tornar redutível renomeando o grupo 1 ou 2 (grupos de folhas pertinentes em negrito) para não pertinente. Com efeito, esta operação divide os vértices representados pelas folhas pertinentes.

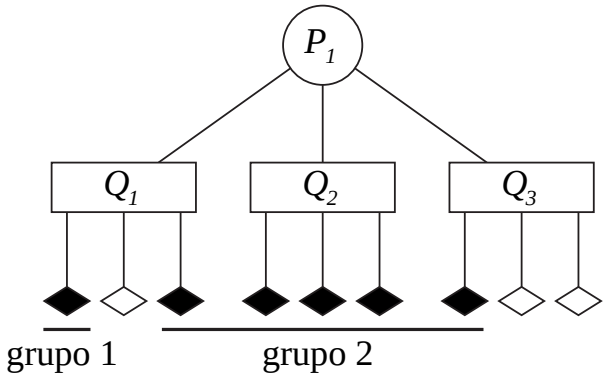


Figura 5.7: Redução do número de grupos pertinentes

Adicionalmente, as renomeações exigem um pouco mais de complexidade. Algumas vezes, além de se renomear um grupo, será necessário fazer a árvore ser de um dos tipos *W, B, A, H* ou V. Para um nó X chame de *w, b, a, h* e v o número mínimo de tais renomeações necessárias para transformar um árvore enraizada até X do tipo *W, B, A, H* ou V, respectivamente. Por exemplo, o nó P_1 da árvore na Figura 5.8 (a) tem $w = 2$. Usando as operações de combinação da árvore-PQ, pode-se rearranjar sua fronteira de maneira a obter dois grupos pertinentes como mostrado em (b), então renomeia-se ambos os grupos para não pertinentes, obtendo-se a árvore do tipo W mostrada em (c).

Além disso, P_1 tem $b = \infty$ porque nenhuma renomeação dos grupos fará com que todas as folhas se tornem pertinentes. Seguindo, P_1 tem $a = 1$, sendo que para obter (d) basta renomear a folha mais à esquerda da árvore em (b); $h = 1$, renomeando o grupo formado pelo filho mais à direita de Q_1, a folha pertinente de P_1 e o filho mais a esquerda de P_2 da árvore em (b), obtém-se a árvore em (e); e por fim $v = 1$, obtendo-se (f) à partir das operações de combinação da árvore-PQ inverte-se de lugar as duas folhas de P_1, renomeia-se o grupo formado pelos filhos mais à direita de Q_1 e o mais à esquerda de P_2 da árvore em (b).

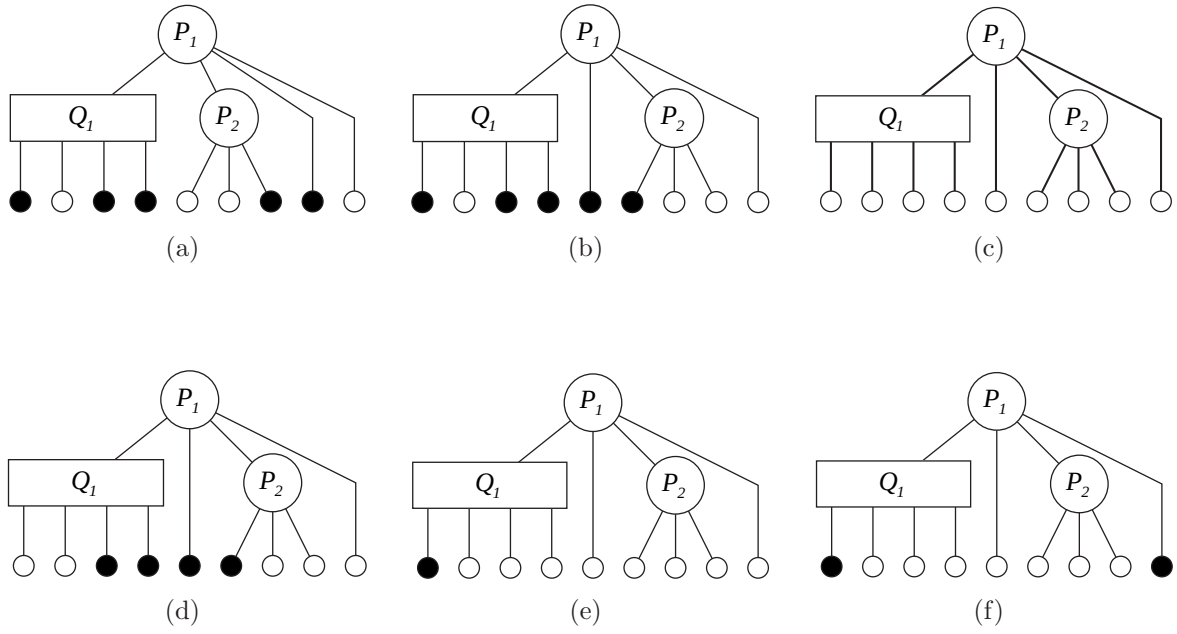


Figura 5.8: Valores de *b, w, h, a* e v para a raiz

Cada iteração do algoritmo desenha um vértice por vez, consistindo de uma das modificações do processo de redução acima descrito para as árvores-PQ. Este novo processo consiste de três passos:

1. *Bolha*: neste passo é identificada a subárvore pertinente (na qual é realizada a redução e a renomeação dos grupos pertinentes). Também é calculado o conjunto de valores *w, b, a, h* e *v* para cada nó da subárvore pertinente. Este processo é descrito em maiores detalhes a seguir.

2. *Definição da Partição da fronteira em grupos pertinentes e redução para cada um dos grupos*: neste passo é definido o particionamento das folhas pertinentes em grupos pertinentes usando os valores de *w, b, a, h* e *v*. Então são renomeados todos os grupos, com exceção de um, como não pertinentes, e aplicado o procedimento de redução original como descrito anteriormente. O nó pertinente é trocado por um novo *nó-P* na árvore-PQ; na prática, isto desenha uma cópia do vértice $i+1$. Este procedimento é repetido até que todos os grupos sejam trocados por nós-P.

3. *Ajuste das partições adjacentes*: neste passo cada nó-P criado pelo passo anterior é percorrido em direção à raiz para encontrar um novo nó-P que não tem como ancestral um nó-Q ou que tenha o mais alto nó-Q parcial ancestral. Então, todos os vértices virtuais adjacentes ao grupo pertinente original serão ligados com este nó. Este passo pode ser opcional.

A cada iteração do algoritmo é desenhado um novo vértice, sendo que o número total de vértices do grafo aumenta. Neste caso, toda partição é desenhada consecutivamente no passo 2 sendo que cada uma corresponde a um grupo pertinente. Somente um dos grupos pertinentes (escolhida no passo 3) preserva as arestas vinda $i+1$ para o conjunto de vértices ainda não desenhados. A política de escolher um grupo pertinente sem um nó-Q ancestral ou o nó-Q ancestral mais alto é usada para permitir uma número maior de possibilidades de combinações para os descendentes de $i+1$. Nos parágrafos a seguir serão descritos em maiores detalhes os dois passos.

5.2.1 Passo 1: Bolha

Seja $Par(X)$ e $F(X)$ o conjunto parcial e cheio dos filhos do nó X, respectivamente. Seja $P(X)$ o conjunto de filhos pertinentes do nó X. Note que $P(X) \subseteq F(X)$ e que quando um nó não pode ser transformado nos tipos *W, H, A, V* ou *B*, seu valor é computado como $m =\mid E \mid$, representando o número infinito. A vírgula "," utilizada nas expressões do tipo (se $h_i \leq (a_i + \frac{1}{2}, v_i + \frac{1}{2})$ denota o operado lógico "*e*". Há diversas possibilidades nas quais podem ser calculados os valores *b, w, a, a* e *v*, sendo elas investigadas a seguir.

(i) O nó X é uma folha pertinente. Neste caso:

- $w = 1$,
- $b = 0$,
- $h = m$, $a = m$ e $v = m$, uma folha pertinente não pode ser feita um desses tipos.

(ii) O nó X é um nó cheio. Neste caso:

- $w = 1$,
- $b = 0$,
- $h = 1$, $a = 2$ e $v = 1$, estas três possibilidades não são boas porque elas não diminuem o número de grupos. Na prática, todas podem ser feitas m.

(iii) O nó X é um nó não pertinente ou vazio. Neste caso:

- $w = 0$ porque todas as folhas são não pertinentes,
- $h = m$, $a = m$, $v = m$ e $b = m$, nenhuma partição pode mudar a fronteira deste nó.

(iv) O nó X é um nó-P parcial. Agora alguns subcasos devem ser considerados, como se segue:
Seja θ_b, θ_h, θ_a, θ_v, θ_1 e θ_2 as funções a seguir.

$$\theta_b(X) = \begin{cases} \frac{1}{2} & \text{se } b_i < m \text{ para qualquer filho i de X} \\ 0 & \text{caso contrário} \end{cases}$$

$$\theta_h(X) = \begin{cases} \frac{1}{2} & \text{se } h_i \leq (a_i + \frac{1}{2}, v_i + \frac{1}{2}) \text{ para qualquer filho parcial } i \text{ de } X \\ 0 & \text{caso contrário} \end{cases}$$

$$\theta_a(X) = \begin{cases} \frac{1}{2} & \text{se } v_i \geq a_i < h_i - \frac{1}{2} \text{ para qualquer filho parcial } i \text{ de } X \\ 0 & \text{caso contrário} \end{cases}$$

$$\theta_v(X) = \begin{cases} \frac{1}{2} & \text{se } v_i < (h_i - \frac{1}{2}, a_i) \text{ para qualquer filho parcial } i \text{ de } X \\ 0 & \text{caso contrário} \end{cases}$$

$$\theta_1(X) = \begin{cases} \frac{1}{2} & \text{se } h_i \leq (a_i + \frac{1}{2}, v_i + \frac{1}{2}) \text{ ou } a_i \geq v_i < h_i - \frac{1}{2} \\ & \text{para qualquer filho parcial } i \text{ de } X \\ 0 & \text{caso contrário} \end{cases}$$

$$\theta_2(X) = \begin{cases} \frac{1}{2} & \text{se } h_i \leq (a_i + \frac{1}{2}, v_i + \frac{1}{2}) \text{ ou } a_i \geq v_i < h_i - \frac{1}{2} \\ & \text{para pelo menos dois filhos parciais } i \text{ de } X \\ 0 & \text{caso contrário} \end{cases}$$

- O nó X não pode ser feito um nó tipo B ao menos que ele seja um nó cheio, e portanto $b = m$.

- Para fazer o nó X um nó tipo W: todos seus filhos pertinentes devem ser feitos tipo W. Portanto, o número w pode ser calculado como segue,

$$\alpha = \max\{\theta_v - \theta_h,\ 0\} + \sum_{i\ \in\ Par(X)} \min\{h_i + \frac{1}{2},\ a_i + 1,\ v_i + 1\}$$

$$w = \left\lceil \alpha + \min\left\{ \max\{\theta_b - \theta_h,\ 0\},\ \max\{\theta_b - \theta_v,\ 0\} \right\} \right\rceil$$

- Para fazer o nó X um nó tipo H: Há duas possibilidades. Na primeira, todos seus filhos cheios devem ser deixados como tipo B, um filho parcial deve ser feito tipo H (se algum existir) e todos os outros filhos parciais devem ser feito tipo W. De forma alternativa, se nenhum filho pode ser feito tipo H, então todos os filhos cheios são deixados como tipo B (se eles existem) e todos os filhos parciais são feitos tipo W. Desse modo, o valor de h pode ser calculado como segue,

$$h = \begin{cases} \lceil \alpha - \max\{\theta_h,\ \theta_v\} \rceil & \text{se } \theta_h\ +\ \theta_v > 0 \\ \lceil \alpha \rceil & \text{se } \theta_h\ +\ \theta_v = 0 \text{ e } \theta_b\ >\ 0 \\ m & \text{caso contrário} \end{cases}$$

- Para fazer o nó X um nó tipo A: Há duas possibilidades. Na primeira, um filho parcial deve ser feito tipo A e todos os outros feito tipo W. De forma alternativa, todos os filhos cheios devem ser mantidos como tipo B, dois filhos parciais feitos tipo H e todos os outros filhos parciais feitos tipo W. Note que o tipo V não é considerado uma vez que, um nó tipo V pode ser feito tipo H. Além disso, há outras possibilidades tais como fazer todos os nós parciais tipo W e deixando todos os nós cheios como tipo B. Entretanto, esta não é uma boa combinação, pois pode-se fazer o nó X um nó tipo H com um número menor de partições. Assim, o valor de a pode ser calculado como segue,

$$a = \begin{cases} \lceil \alpha - \theta_1 - \theta_2 \rceil & \text{se } \theta_1 \ + \ \theta_2 < 1 \text{ e } \theta_a \ = \ \frac{1}{2} \\ \lceil \alpha - 1 \rceil & \text{se } \theta_1 \ + \ \theta_2 = 1 \\ m & \text{caso contrário} \end{cases}$$

- Para fazer o nó X um nó tipo V: Há duas possibilidades. Na primeira, dois filhos parciais devem ser feitos tipo H, todos os filhos cheios devem ser mantidos como tipo B, e todos os outros filhos parciais devem ser feitos tipo W. De forma alternativa, um filho deve ser feito tipo V (se $|Par(X)| = 1$) e todos os filhos cheios devem ser mantidos como tipo B. Note que há outras possibilidades para combinar os filhos de X. Por exemplo, um filho pode ser feito tipo H e pelo menos um nó cheio deve ser mantido como tipo B. Entretanto, esta também não é uma boa opção, uma vez que este nó pode ser feito tipo H com um número menor de partições. Então, o valor de v pode ser calculado como segue,

$$v = \begin{cases} \lceil \alpha - 1 \rceil & \text{se } |Par(X)| = 1 \text{ e } \theta_v = \frac{1}{2} \\ \lceil \alpha - \theta_1 - \theta_2 \rceil & \text{se } \theta_1 \ + \ \theta_2 = 1 \\ m & \text{caso contrário} \end{cases}$$

(v) O nó X é um nó-Q parcial.

Como os filhos de um nó-Q podem somente ser invertidos, é necessário atravessar seus filhos da esquerda para a direita combinando-os de maneira a fazer o menor número de partições para construir a árvore do tipo requerido.

Seja H_r e H_l os nós tipo H com suas folhas pertinentes mais à direita e mais à esquerda, respectivamente.

- Note que o nó X não pode ser feito um nó tipo B ao menos que ele seja um nó cheio, e portanto $b = m$.

- Para fazer o nó X um nó tipo W: é necessário atravessar seus filhos i procurando por um valor mínimo de $\mu = \min\{h_i + \frac{1}{2},\ a_i + 1,\ v_i + 1\}$ para um nó parcial. Se μ for $h_i + \frac{1}{2}$, $a_i + 1$ ou $v_i + 1$, então i é feito do tipo H, A ou V, respectivamente. Para cada nó i do tipo H, i é feito tipo H_l ou H_r como se segue: caso o nó i seja o filho mais a esquerda X, então o nó i é feito tipo H_r se seu irmão da esquerda é um nó tipo B, V, ou H_l; caso contrário, o nó i é feito tipo H_l; senão o nó i é feito tipo H_r se seu irmão da esquerda não é um nó do tipo B, V, nem H_r; caso contrário, o nó i é feito tipo H_l. Filhos de X que são nós cheios são mantidos como tipo B.

Se i tem $\mu = h_i + \frac{1}{2} = v_i + 1 \leq\ a_i + 1$ ou $\mu = h_i + \frac{1}{2} = a_i + 1 \leq\ v_i + 1$, então i é feito tipo H_l ou H_r como se segue: caso o nó i seja o filho mais a esquerda X, então o nó i é feito tipo H_r se seu irmão da esquerda é um nó tipo B, V, ou H_l; caso contrário, o nó i é feito tipo H_l; senão o nó i é feito tipo H_r se seu irmão da esquerda não é um nó do tipo B, V, nem H_r; caso contrário, o nó i é feito tipo H_l. Se i tem $\mu = a_i + 1 = v_i + 1 <\ h_i + \frac{1}{2}$, então i é feito tipo A ou V como se segue: o nó i é feito tipo A se ele é o filho mais à esquerda de X ou se o irmão da esquerda de i for do tipo W, A, ou um nó tipo H_l; caso contrário, o nó i é feito tipo V.

Agora que já se sabe o tipo de cada filho de X, pode-se contar o número de partições dada pelos $b_i,\ w_i,\ h_i,\ v_i$ e a_i dos nós feitos tipo *B, W, H, V* e A, respectivamente, mais o número de partições que seguem os padrões de troca a seguir.

- Para fazer o nó X um nó tipo A: escolhe-se qualquer partição a partir das operações de padrões de troca realizadas para encontrar o número w com as folhas não pertinentes mais à esquerda e mais à direita. Assim, $a = w - 1$, caso contrário $a = m$.

- Para fazer o nó X um nó tipo H: o filho mais à esquerda ou o filho mais à direita devem ser nós pertinentes que podem ser feitos tipos H, V ou B; caso contrário $h = m$. Sem perda de generalidade, suponha que o filho mais à esquerda de X seja pertinente. Então os filhos de X são atravessados da esquerda para a direita para encontrar a seqüência máxima $P_l(X)$ do filho pertinente de X onde somente o filho mais à direita é parcial. Seguindo agora esse padrão de troca, todos os outros nós são feitos tipo W. Neste caso $h = w - 1$.

- Para fazer o nó X um nó tipo V: os filhos mais à esquerda e mais à direita devem ser nós pertinentes que podem ser feitos tipos H, V ou B; caso contrário $v = m$. Além disso, são encontradas as duas seqüências $P_l(X)$ e $P_r(X)$. Todos os outros filhos são feitos tipo W. Neste caso, $v = w - 2$.

Isto conclui o primeiro passo do algoritmo. A seguir é apresentado o segundo passo.

5.2.2 Passo 2: Partição e Redução

A partir deste ponto já é possível determinar se uma árvore é redutível ou não observando os valores de b, h e a. Se um desses valores for 0 para a raiz pertinente, então a árvore T_i será redutível, caso contrário ela será irredutível. No caso de ela ser irredutível, deve-se

então determinar o conjunto de grupos pertinentes a serem renomeados. Em primeiro lugar, a subárvore pertinente é feita tipo B, H ou A dependendo de qual valor entre b, h ou a é o menor. Note que a raiz pertinente não pode ser feita tipo W ou V. No primeiro caso, implicaria em T_i se tornar vazia e no último caso T_i ainda continuaria irredutível. Além disso, se a raiz pertinente pode ser feita tipo B, então ela é um nó cheio e não há nada a ser feito.

O algoritmo neste passo atravessa a árvore-PQ no sentido raiz-folha (*top down*) fazendo seus nós tipo *B, W, H, A* ou *V* baseado no tipo do pai e nos valores de b, w, h, a, ou v. O tipo de cada filho já está previamente definido pelo passo anterior. Quando o nó pai é cheio ou é um nó-Q parcial o conjunto de grupos pertinentes também já foi definido pelo passo anterior. Entretanto, é necessário definir como combinar os filhos de um nó-P parcial para formar o conjunto de grupos pertinentes. Deixe X ser um nó-P parcial para o qual deseja-se determinar seu conjunto de grupos pertinentes. Todos os filhos de X que são folhas pertinentes serão tratados como nós cheios do tipo B. A maneira de se combinar os filhos de X depende do tipo de nó que X será feito, como segue:

- Se o nó X é para ser feito tipo W ou A: utilizando-se as operações de combinação da árvore-PQ, os filhos de X são combinados na seguinte ordem da esquerda para a direita: todos filhos tipo A, todos filhos tipo W, um filho tipo H que foi feito tipo H_r, todos filhos tipo B, todos filhos tipo V, e todos os demais filhos tipo H fazendo-os se alternarem entre H_l e H_r, iniciando com H_l. Esta combinação fornece o menor número de grupos pertinentes na maioria dos casos. Nos demais casos, o nó X pode ser feito tipo H com um número menor de grupos pertinentes.

- Se X é para ser feito tipo H: então os filhos de X são combinados na seguinte ordem: todos filhos tipo A, todos filhos tipo W, um filho tipo H que foi feito H_r, todos filhos tipo V, e todos os demais filhos tipo H que tinham sido feitos H_l e H_r, iniciando com H_l e todos os filhos tipo B.

- Se X é para ser feito tipo V: então os filhos de X são combinados na seguinte ordem: todos filhos tipo H com excessão de um, alternativamente àqueles feitos H_l e H_r, iniciando com H_l, todos filhos tipo W, todos os filhos do tipo A, os demais filhos tipo H que foram feitos H_r, todos filhos tipo B e todos filhos tipo V. Novamente, há alguns poucos casos no qual esta combinação não resulta no menor número de grupos pertinentes. Entretanto, nestes casos pode-se fazer o nó X tipo H com um menor número de grupos pertinentes.

Em todas as combinações acima quando alguns tipos são perdidos eles não são considerados. Por exemplo, suponha que o nó X tenha filhos que podem somente serem feitos tipo A ou B. Neste caso, se X é para ser feito tipo H, então os nós tipo A aparecerão consecutivamente como os filhos mais à esquerda de X e os nós tipo B aparecerão consecutivamente como os filhos mais à direita de X.

Isto completa a descrição do passo 2. Na próxima seção é discutida a complexidade do algoritmo proposto.

5.3 Análise da complexidade do algoritmo

Nesta seção é mostrado que o algoritmo de Planarização por Divisão realiza sua tarefa em complexidade de tempo de $O(n^2)$ e requer de espaço $O(n + m)$. De forma genérica, a cada iteração i, o algoritmo realiza duas principais operações em uma árvore-PQ T_i:

(a) Fazer T_i redutível. Isto pode ser alcançado através do passo 1 e parte do passo 2 onde são definidos os grupos de folhas pertinentes que representam as partições dos vértices virtuais, e todas, menos uma partição, são renomeadas uma por vez.

(b) Redução de T_i para T_i^*. Aplica-se as operações de redução de Booth e Lueker [10] para cada grupo pertinente. Cada grupo é inserido na árvore-PQ pela criação de um novo nó-P. Na forma arbusto B_i a partição correspondente é desenhada.

Para (b) considera-se o algoritmo de redução da árvore-PQ de Booth e Lueker [10] que reduz conforme determinado pelos autores processa a operação de redução em tempo $O(n + m + s)$, onde:

$n = |V|$: sendo V o conjunto correspondente ao conjunto de todos os elementos que aparecem como folhas da árvore-PQ.

$m = |E|$: sendo E o número de arestas totais do grafo dado. O número de reduções é proporcional ao número de arestas do grafo inicial devido às operações de divisão.

$s = m$: sendo s a soma do número de folhas pertinentes de todas as árvores-PQ T_i.

Portanto, o algoritmo de redução de Booth e Lueker processa a redução de todas as árvores-PQ T_i em tempo $O(n + m)$ para o algoritmo de Planariza por Divisão.

Já para (a) o objetivo é fazer com que esse passo seja executado com a mesma eficiência da do algoritmo de Booth e Lueker. Conforme evidenciam os autores, isso é possível porque os ponteiros para os pais são mantidos para todos os filhos dos nós-P e para o filho mais à esquerda e mais à direita dos nós-Q. Contudo, JTS modificou este algoritmo de redução fazendo com que estes ponteiros estejam sempre disponíveis em sua versão, o algoritmo *Planarize*. Para o algoritmo proposto, o PLANARIZA POR DIVISÃO, uma pequena modificação é necessária. Elas são descritas abaixo. A Figura 5.2 mostra o procedimento PLANARIZA POR DIVISÃO em um pseudo-código em Pascal.

- CALCULA(T_i). Este procedimento calcula os valores de w, h, e a. Ele é uma versão modificada do procedimento *Compute* de JTS que, além de calcular os valores acima, calcula também o valor de v usada na fórmula proposta da seção 5.2.1.

- AJUSTA-NÓS(T_i). Este procedimento substituiu o *Delete-Nodes* de JTS que removia as folhas marcadas que correspondiam aos vértices virtuais. Esta nova versão encontras o conjunto de grupos cuja renomeação faz T_i redutível. Este procedimento foi discutido na seção 5.2.2.

- BOLHA(T_i). Os dois procedimentos anteriores são eficientes quando se processa somente filhos pertinentes de qualquer nó-P na árvore-PQ T_i. Portanto, este procedimento divide o conjunto de filhos de um nó-P em dois grupos, um consistindo somente de filhos pertinentes e outro consistindo de filhos não pertinentes. Isto é útil porque evita que seja atravessado todos os filhos de um nó-P para encontrar seus filhos pertinentes que podem aparecer em qualquer ordem. Além disso, este procedimento também calcula o número de filhos pertinentes e o número de folhas pertinentes descendentes de um nó-P. Ademais, ele encontra o ponteiros pai para todos os nós pertinentes. Este procedimento é usado sem modificação na proposta. Note que todos os ponteiros que apontam para o pai encontrados por este procedimento são válidos para cada grupo, e com isso, o algoritmo de Booth e Lueker torna-se possível.

- ATUALIZA-DESCENDENTES(T_i). Este procedimento determina o número de folhas descendentes de cada nó pertinente. Este procedimento também foi usado sem alterações. Cada nó X contendo folhas pertinentes que correspondem a todos os grupos para os quais a árvore-PQ T_i foi reduzida é atualizado pelo decremento do número de suas folhas pertinentes. Esta operação também deve ser realizada para todos os ancestrais de X. As folhas pertinentes correspondentes a um grupo são

então removidas e um novo nó-P é adicionado a árvore. Este procedimento também incrementa os ancestrais do novo nó-P pelo incremento do número de folhas do novo nó-P para cada um dos seus ancestrais. O procedimento AJUSTA-GRUPOS(T_i) também é responsável pelo decremento do número de nós pertinentes da árvore-PQ para todos os grupos.

Para se demonstrar a complexidade do algoritmo PLANARIZA POR DIVISÃO é necessário recorrer a dois Lemas.

Lema 14. *Operações de divisão não aumentam o número de filhos de um nó-Q ou o número de nós-P pertinentes.*

Prova: Uma vez que somente uma das cópias do vértice dividido $i+1$ mantém os vizinhos maiores que $i+1$, somente um dos nós-P criados a cada iteração i pode ser o filho de um nó-Q ou um nó-P pertinente. □

Lema 15. *Há no máximo $n + d(i+1)$ nós-Q pertinentes na árvore-PQ.*

Prova: Note que operações de divisão podem aumentar o número de nós-Q pertinentes. Entretanto, um nó-Q representa um bloco em B_i o que implica que o nó deve ter pelo menos 3 vértices. Para ser um nó-Q pertinente um de seus vértices j do correspondente bloco, exceto aquele com o menor valor de j, deve ter um vizinho com valor maior do que j, ou então o nó-Q não teria filho algum. Portanto, o número de nós-Q na árvore-PQ não excede $n + d(i+1)$. □

Feito essas considerações, a complexidade dos procedimentos acima usados pelo algoritmo PLANARIZA POR DIVISÃO seguem o seguinte Lema:

Lema 16. *Os procedimentos* CALCULA, AJUSTA-GRUPOS, BOLHA *e* ATUALIZA-DESCENDENTES *podem processar todas as árvores-PQ em tempo $O(n^2)$.*

Prova: Todos os procedimentos acima atravessam os filhos de todos os nós-Q pertinentes. A quantidade total de processamento para um nó-Q é proporcional a quantidade de filhos que possui. Um filho de um nó-Q, que está contido em um bloco no grafo desenhado B_i, representa um vértice que contém pelo menos uma folha descendente. O número destes vértices não excede a i. Além disso, um nó pode aparecer somente uma vez como filho de

somente um nó-Q na árvore-PQ. Isto provém do argumento acima e do Lema 5.3 que o número total de filhos de um nó-Q na árvore-PQ T_i não excede a $O(n + d(i))$. Portanto, a quantidade de processamento realizada para todos os procedimentos acima para um nó-P pertinente é proporcional ao número de seus filhos pertinentes. O filho de um nó-P é uma folha, um nó-P ou um nó-Q. O número de nós-P na árvore-PQ não excede a i. O número de folhas pertinentes na árvore-PQ não excede $d(i + 1)$. Partindo-se, novamente, do Lema 5.3 e do argumento acima existe no máximo $O(n + d(i + 1))$ nós pertinentes em uma árvore-PQ T_i. Logo, a quantidade de processamento realizada ambos os procedimentos para todos os nós-P pertinentes não é maior do que $O(n + d(i + 1))$. Somando-se a quantidade de processamento de ambos os procedimentos para todos os nós-P e nós-Q para todas as árvores-PQ T_i obtém-se a quantidade de processamento realizada para ambos os procedimentos que é $O(m + n^2) = O(n^2)$, desde que $m \leq n^2$. □

Foi mostrada que a complexidade de tempo total requerida pelos procedimentos CALCULA, AJUSTA-GRUPOS, BOLHA e ATUALIZA-DESCENDENTES é $O(n^2)$. Essencialmente, a complexidade vem do fato que a quantidade de processamento realizada por estes procedimentos é proporcional ao número de nós pertinentes de cada árvore-PQ T_i. Estes procedimentos podem fazer uma árvore se tornar redutível para que depois possa ser aplicado o algoritmo de redução de Booth e Lueker. O Teorema a seguir finaliza esta seção, como segue:

Teorema 17. *O procedimento* PLANARIZA POR DIVISÃO *determina e realiza um conjunto de operações de divisão em um grafo não planar dado G produzindo um grafo planar em tempo* $O(n^2)$ *e espaço* $O(m + n)$.

Require: grafo finito, não orientado e biconexo G
Ensure: grafo planar dividido G'

Comment o procedimento PLANARIZA POR DIVISÃO determina o conjunto de grupos (partições de um vértice) $P_3 = v_3,\ P_4, \ldots,\ P_{n-1}$ que representam as partições dos vértices $v_3,\ v_4, \ldots,\ v_{n-1}$. Estas partições determinam o conjunto de operações de divisão em G que produzem um GRAFO PLANAR DIVIDO G'

1: **Begin**
2: construa a *Árvore-PQ* inicial $T_1 = T_1^*$;
{`Folhas-Descendentes(X)` é o número de folhas descendentes do nó X}
3: `Folhas-Descendentes`$(1) \leftarrow d(1)$;
4: $P_2 \leftarrow v_2$;
5: $Z \leftarrow P_2$;
6: **For** cada folha X correspondente a um vértice em P_2
7: `Folhas-Descendentes`$(X) \leftarrow 1$;
8: **For** $i = 2$ to $n - 2$
9: **Begin**
10: faça P_{i+1} ser v_{i+1};
11: construa a *Árvore-PQ* inicial T_i a partir de T_i^*;
12: ATUALIZA-DESCENDENTES(T_i);
{onde Z é o grupo que mantém as folhas adjacentes}
13: **For** o nó-P Z correspondente ao vértice i
14: `Folhas-Descendentes`$(Z) \leftarrow |\{j \in Adj(i)|\ j > i\}|$;
15: **For** cada folha X correspondente a uma folha de Z
16: `Folhas-Descendentes`$(X) \leftarrow 1$;
17: BOLHA (T_i);
18: CALCULA (T_i);
19: AJUSTA-GRUPOS (T_i);
20: **For** cada grupo correspondente a uma das partições P_1, $P_2, \ldots,\ P_k$ de P_{i+1}
{Uma *árvore-PQ* $T_{i,j}$ é uma *árvore-PQ* T_i no qual o conjunto de folhas pertinentes são restritos aos grupos correspondentes as partições P_j de P_{i+1}}
21: REDUZA $T_{i,j}$ para $T_{i,j}^*$;
22: Escolha um dos grupos Z correspondente a uma partição P_{i+1} que mantém os vértice adjacentes;
23: Ajuste $T_i^* \leftarrow T_{i,k}$;
24: **end**
25: **End** PLANARIZA POR DIVISÃO

Algoritmo 1: Planariza por Divisão (G)

Capítulo 6
Resultados

Os resultados estão divididos em duas seções. A seção 6.1 traz um exemplo da execução do algoritmo sobre o grafo completo bipartido $K_{6,3}$. A seqüencia de grafos apresentada é exatamente a que o algoritmo executa, porém por motivos didáticos, a representação pelas figuras torna possível alguns comentários não vistos na versão implementada.

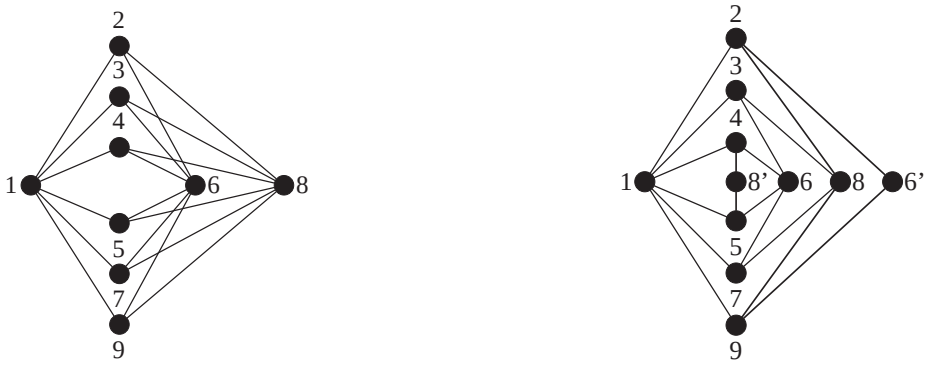
A segunda seção apresenta os resultados do número de vértices divididos e do tempo de execução do algoritmo Planariza por Divisão - PPD para as classes de grafos completos, completos bipartidos, os $Y_{k,kn}$ e os cartesianos. Essas classes foram escolhidas por possuírem a medida de não-planaridade número de vértices divididos (*splitting number*) conhecida.

Além disso, essas mesmas classes também possuem a medida de não-planaridade número de arestas removidas (*skewness*) também conhecida, o que possibilitou comparações entre essas duas operações de planarização, bem como evidenciar, empiricamente, que as fórmulas criadas por Jayakumar, Thulasiraman e Swamy [44], são totalmente diferentes das aqui utilizadas.

Assim, subseção 6.2.1 mostra os resultados para a classe de grafos completos K_n, a subseção 6.2.2 para a classe de grafos completos bipartidos $K_{n1,n2}$, a subseção 6.2.3 para os grafos completos K_5 ao K_8 convertidos no Y_2 e Y_3, e por fim a subseção 6.2.4 para a classe de grafos cartesianos (toroidais) C_n x C_m.

6.1 Executando o algoritmo sobre um grafo

O algoritmo descrito na seção anterior aparenta ser excessivamente complexo. Entretanto, apesar dos muitos detalhes, sua estrutura geral é simples. Com o intuito de fornecer uma noção sobre como o algoritmo opera, o grafo da Figura 5.5 (a) servirá como exemplo, sendo utilizada também a mesma *numeração-st* daquele grafo, reproduzido aqui, na Figura 6.1 (a). Em (b) está o Grafo Planar Divido ótimo, isto é, seu sp $(K_{6,3}) = 2$. Como veremos a seguir, o algoritmo nem sempre encontra a solução ótima, uma vez que, Faria, Figueiredo e Mendonça [27] demonstraram que o Problema do Subrafo Planar Dividido é NP-Completo.



(a) Grafo-st $K_{6,3}$ (b) Grafo Planar Divido Mínimo

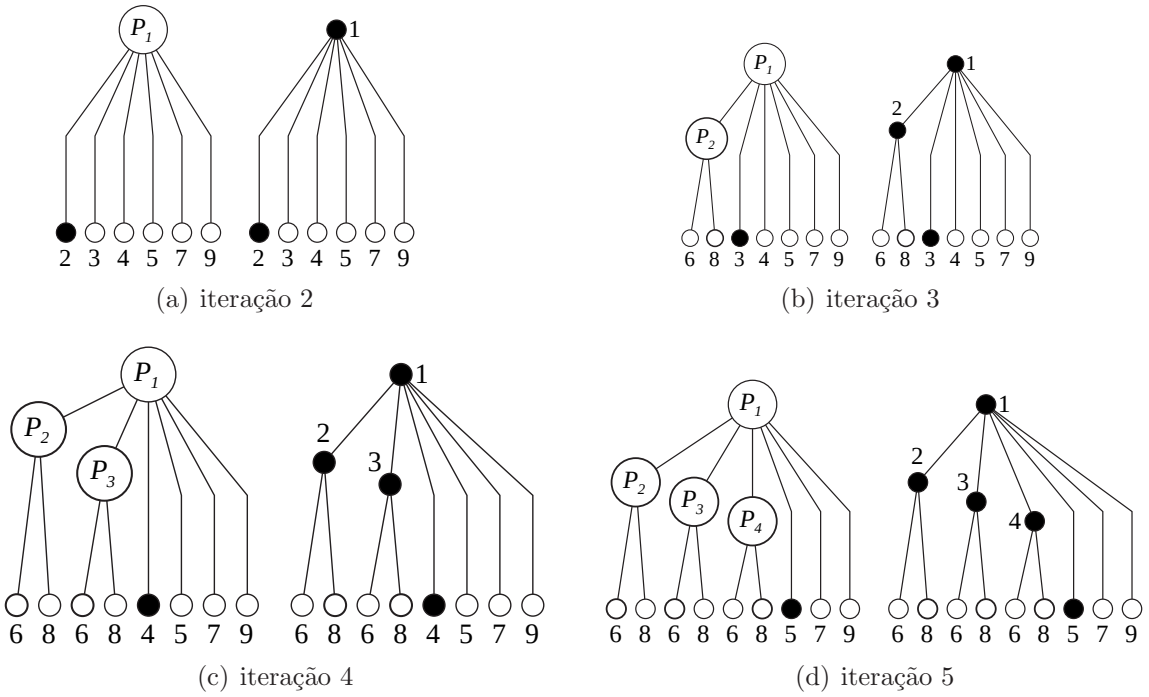
Figura 6.1: Grafo-st $K_{6,3}$ com sua divisão ótima

Em cada passo do exemplo é apresentada a árvore-PQ T_i e sua respectiva forma arbusto B_k. Para cada nó pertinente é mostrado os valores de b, w, h, a e v em um vetor nesta ordem, respectivamente. Desse modo, um vetor como esse (0,1,2,3,18) para um nó X especifica o número de grupos que devem ser renomeados para fazer X do tipo B, W, H, A e V, respectivamente.

O algoritmo inicia desenhando o vértice rotulado de 1. A árvore-PQ B_1 correspondente tem o nó-P P_1 com os vértices virtuais rotulados de 2, 3, 4, 5, 7 e 9. A folha rotulada de 2 tem valores (0,1,18,18,18) (note que o número total de arestas é 18, de maneira que tal número no vetor representa o infinito). As folhas não pertinentes, rotuladas de 3, 4, 5, 7 e 9, tem valores (18,0,18,18,18). Finalmente, o nó P_1 tem valores (18,1,0,0,18). Logo, a árvore enraizada até P_1 é redutível. Então T_1 é feita tipo H, vide Figura 6.2 (a).

Na próxima iteração o vértice rotulado de 2 é desenhado, o qual torna-se o nó P_2 na árvore-PQ correspondente. A folha rotulada de 3 torna-se pertinente. As folhas não pertinentes estão rotuladas de 4, 5, 6, 7, 8 e 9. A folha pertinente tem valores (0,1,18,18,18). As folhas não pertinentes têm valores (18,1,18,18,18). Finalmente, o nó P_1 tem valores (18,1,0,0,18). Novamente, a árvore enraizada até o nó P_1 é redutível e então é feita tipo H, conforme Figura 6.2 (b). As próximas duas iterações seguem o mesmo procedimento da do passo anterior, e elas estão ilustradas na Figura 6.2 (c) e (d).

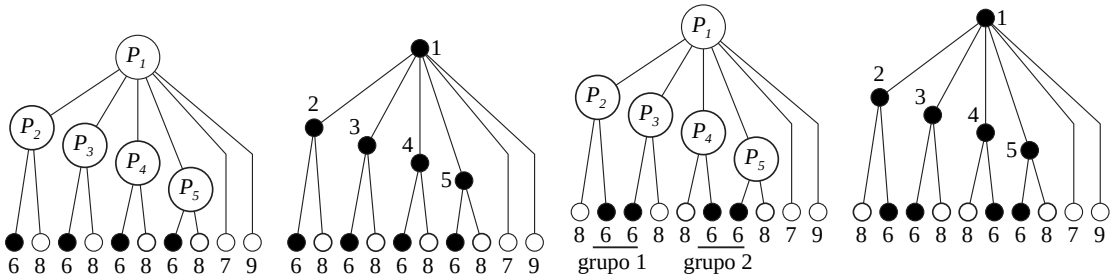
Ao final da quinta iteração obtém-se a árvore-PQ T_5 e sua forma arbusto B_5 mostradas na Figura 6.3 (a). A seguir, o algoritmo entra na iteração 6 e processa as folhas pertinentes rotuladas de 6. As folhas pertinentes têm valores (0,1,18,18,18). As folhas não pertinentes têm valores (18,0,18,18,18). Os nós P_2, P_3, P_4 e P_5 têm valores (18,1,0,18,18). Finalmente, o nó P_1 tem valores (18,2,2,1,1). Logo, o nó P_1 é feito do tipo A com um grupo renomeado de não pertinente. Os nós P_2 e P_4 são feitos H_r e os nós P_3 e P_5 são feitos H_l, como mostra



(a) iteração 2 (b) iteração 3

(c) iteração 4 (d) iteração 5

Figura 6.2: Exemplo de execução do algoritmo Planariza por Divisão



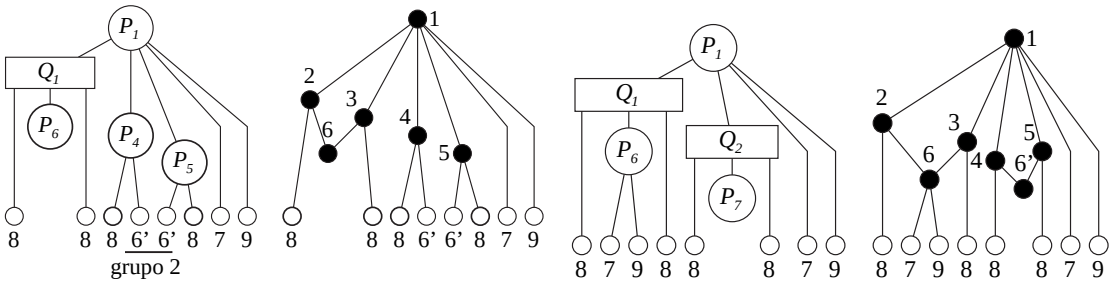
(a) iteração 6, bolha (b) iteração 6, padrão de troca

Figura 6.3: Exemplo de execução do algoritmo Planariza por Divisão

a Figura 6.3 (b).

A seguir, o algoritmo renomeia os filhos pertinentes do grupo 2 (filhos pertinentes dos nós P_4 e P_5) como não pertinentes e desenha o grupo pertinente 1 (filhos pertinentes dos nós P_2 e P_3). Este grupo torna-se o nó-P P_6 na Figura 6.4 (a). Finalmente, renomeia-se os filhos pertinentes do grupo 1 como não pertinentes e desenha o grupo pertinente 2. Este grupo torna-se o nó-P P_7, na Figura 6.4 (b). Os ancestrais mais altos do nó-Q são pai para ambos os nós P_6 e P_7. Logo, escolhe-se P_6 para ser adjacente com todos os vértices virtuais adjacentes ao vértice original rotulado de 6. Uma vez que o nó P_7 não está sendo usado, ele é incorporado ao nó-Q pai formando o nó Q_3, conforme Figura 6.5 (a). Isto concluí a iteração 6.

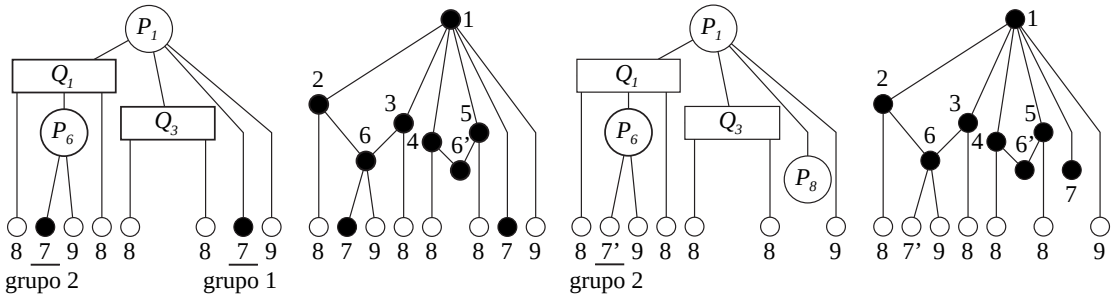
Para a iteração 7, os vértices virtuais rotulados de 7 são pertinentes, conforme mostra



(a) iteração 6, desenhando grupo 1 (b) iteração 6, desenhando grupo 2

Figura 6.4: Exemplo de execução do algoritmo Planariza por Divisão

a Figura 6.5 (a). Os valores para as folhas são os mesmos da iteração anterior. Os valores para o nó P_6 são (18,1,0,18,18); para o nó Q_1 são (18,1,18,0,18); para o nó Q_3 são (18,0,18,18,18); e finalmente para o nó P_1 são (18,2,1,1,18). Logo, P_1 é feito tipo H com um grupo renomeado para não pertinente, como mostra a Figura 6.5 (b) e o grupo 1 é desenhado.



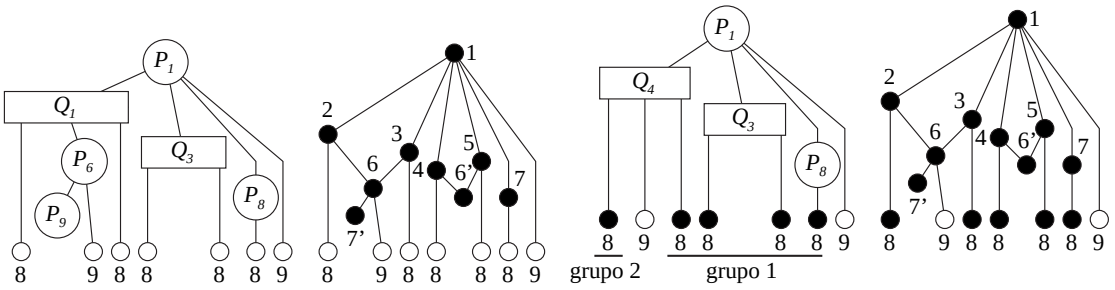
(a) iteração 7, bolha (b) iteração 7, desenhando grupo 1

Figura 6.5: Exemplo de execução do algoritmo Planariza por Divisão

Continuando, o grupo 2 é renomeado e escolhe-se (aleatoriamente) que o grupo 1 irá preservar os vértices virtuais adjacentes, como mostrado na Figura 6.6 (a). Isto concluí a iteração 7. Para a iteração 8 os nós P_6 e P_8 são incorporados ao nó-Q Q_4. conforme Figura 6.6 (b). Os valores para o nó Q_4 são (18,2,1,18,0); para o nó Q_3 são (0,1,1,18,18); para o nó P_8 são (0,1,18,18,18); finalmente para o nó P_1 são (18,2,1,1,1). Logo, P_1 é feito tipo H com um grupo pertinente renomeado para não pertinente; Q_4 é feito tipo H_r; Q_3 tipo B e P_8 tipo B.

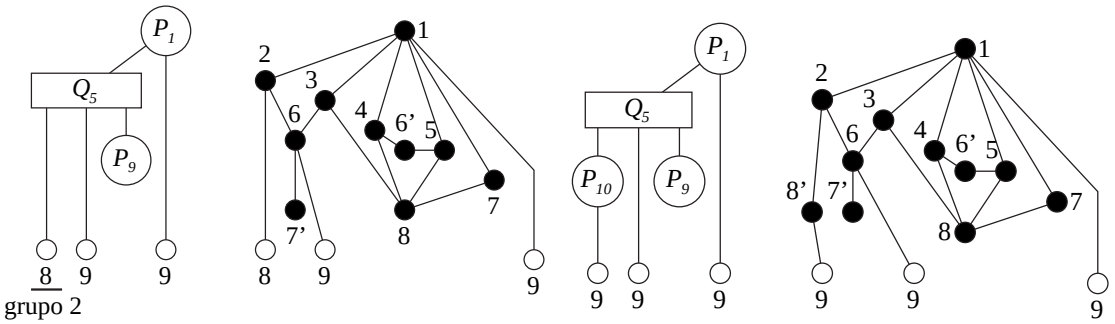
A seguir, o grupo 2 é renomeado para não pertinente e o grupo 1 é desenhado, como mostra na Figura 6.7 e a seguir, o grupo 2 também é desenhado e escolhe-se o grupo 2 (aleatoriamente) para receber os vértices virtuais, ficando o grafo como o apresentado na Figura 6.7. Isto concluí a iteração 8.

Uma vez que o nó P_9 não tem vértices virtuais adjacentes e o nó P_10 possui somente um filho, eles são incorporados ao nó-Q, formando o nó Q_6 da Figura 6.8 (a). Para a iteração



(a) iteração 7, desenhando grupo 2 (b) iteração 8, bolha

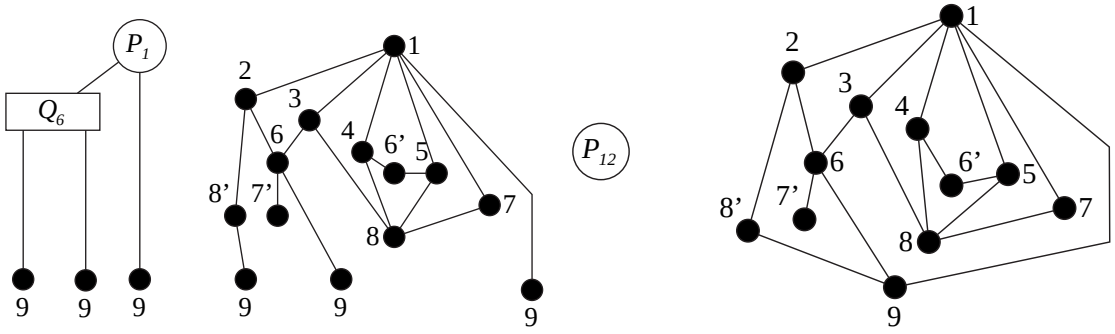
Figura 6.6: Exemplo de execução do algoritmo Planariza por Divisão



(a) iteração 8, desenhando grupo 1 (b) iteração 8, desenhando grupo 2

Figura 6.7: Exemplo de execução do algoritmo Planariza por Divisão

9, as folhas pertinentes são as rotuladas de 9. Todos os nós na árvore pertinente são cheios, e portanto, a árvore enraizada até P_1 é redutível. Assim, todos os nós pertinentes são feitos tipo B e o desenho final do grafo é mostrado na Figura 6.8 (b). Para este grafo, foram divididos 3 vértices, sendo que 4 arestas seriam necessárias serem removidas para que ele se torna-se planar pelo método de remoção de arestas.



(a) iteração 9, bolha (b) iteração 9, desenho final

Figura 6.8: Exemplo de execução do algoritmo Planariza por Divisão

6.2 Comparação entre várias classes de grafos

Os resultados para as subseções 6.2.1 a 6.2.4 são apresentados por gráficos, ao invés de tabelas, pois o primeira forma mostra de forma bem clara a eficiência do algoritmo PPD, tanto em relação ao número de vértices divididos, como para seu tempo de execução. Todos os casos de teste foram realizados em um computador com processador *AMD K-6 III de 400 Mhz* utilizando o software *Borland Pascal for Windows, versão 7.0*.

Paras os gráficos referentes ao número de vértices divididos, a comparação pode ser feita de três formas. A primeira é com relação ao número de vértices divididos ótimos, ou seja, aquele encontrado pela fórmula de Hartsfield, Jackson e Ringel [37]. A comparação entre o número de vértices divididos ótimo e o obtido pelo algoritmo PPD fornece uma medida no sentido de 'o quanto o algoritmo está longe da situação ótima'.

A segunda, ao contrário, torna possível verificar que, na verdade, ele está mais longe ainda de ser péssimo, se comparado ao pior caso. O pior caso seria aquele que, a cada iteração, não haveriam grupos pertinentes sendo formados, e quantas fossem as folhas pertinentes seriam as divisões de vértices.

A terceira forma, seria comparar a eficiência do algoritmo PPD, caso ele fosse implementado com as mesmas fórmulas usadas no algoritmo Planarize de Jayakumar, Thulasiraman e Swamy [44], que como comentado no seção 5.2.1, partem de abordagens diferentes. Enquanto as fórmulas aqui apresentadas buscam minimizar o número de grupos pertinentes as usadas em [44] buscam maximizar o número de folhas pertinentes consecutivas.

6.2.1 Resultados para os grafos completos

Dois conjuntos de grafos de entrada foram utilizados para os testes dessa classe. No primeiro, figuram os grafos com número de vértices entre 5 e 50 ($5 \leq n \leq 50$). No segundo conjunto, além de todos os grafos no intervalo de $5 \leq n \leq 50$, também foram testados os grafos K_{100}, K_{150} e o K_{200}, que podem ser considerados problemas de entrada grandes.

O algoritmo PPD foi executado para esses dois conjuntos de grafos de entrada, gerando como resultado, o número de vértices divididos e o tempo de execução, em segundos, conforme figuras 6.9 e 6.11 para o primeiro caso de teste, e as figuras 6.10 e 6.12, para o segundo caso de teste. Passaremos a discutir cada um deles agora.

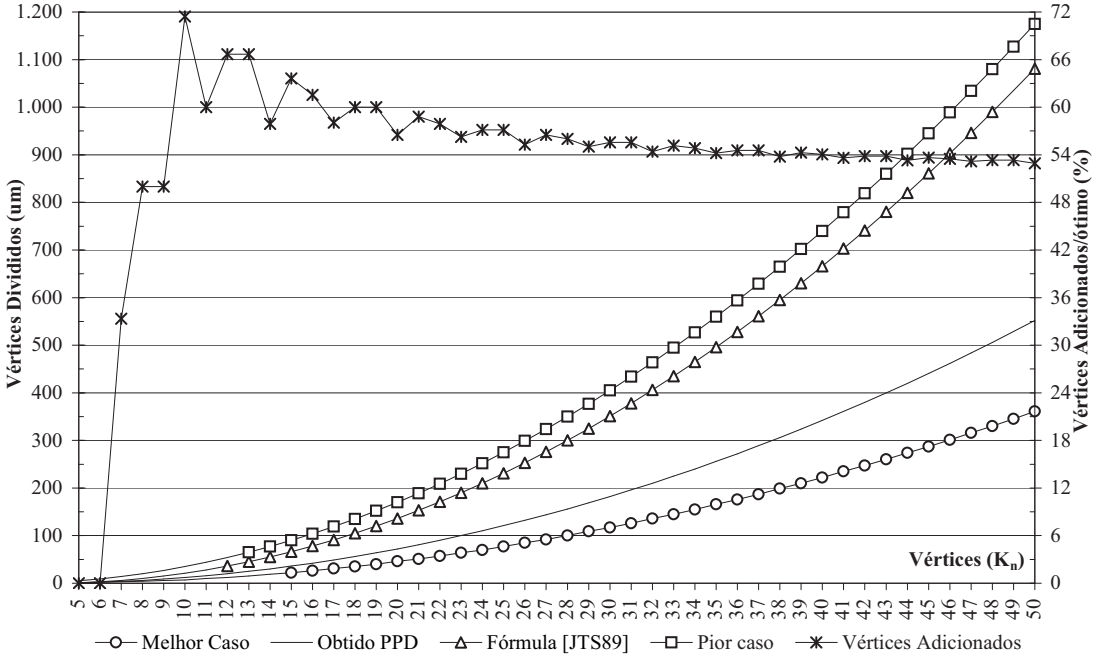


Gráfico 6.9: Número de Vértices Divididos do K_5 ao K_{50}

Como pode ser observado no Gráfico 6.9 a curva rotulada de 'Obtido PPD' apresenta resultados muito mais próximos do melhor caso do que do pior caso. Nota-se que as fórmulas utilizadas em [44] não são adequadas ao problema de divisão de vértices, pois não minimizam o número de grupos pertinentes, ficando assim, com resultados bem próximos ao do pior caso.

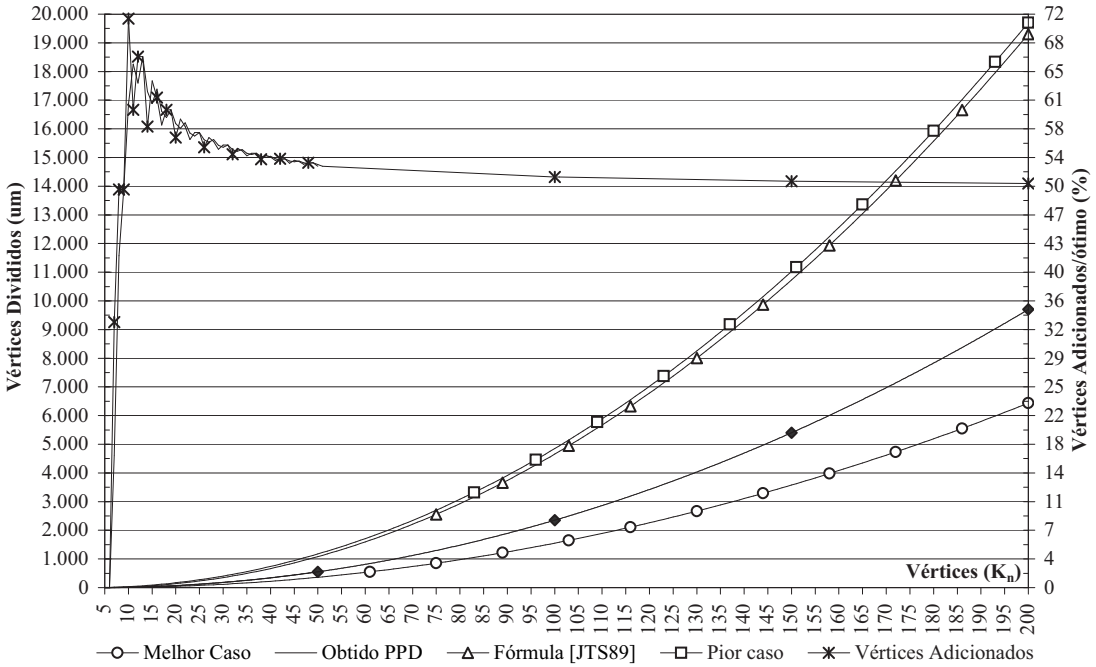


Gráfico 6.10: Número de Vértices Divididos do K_5 ao K_{50} + K_{100}, K_{150} e K_{200}

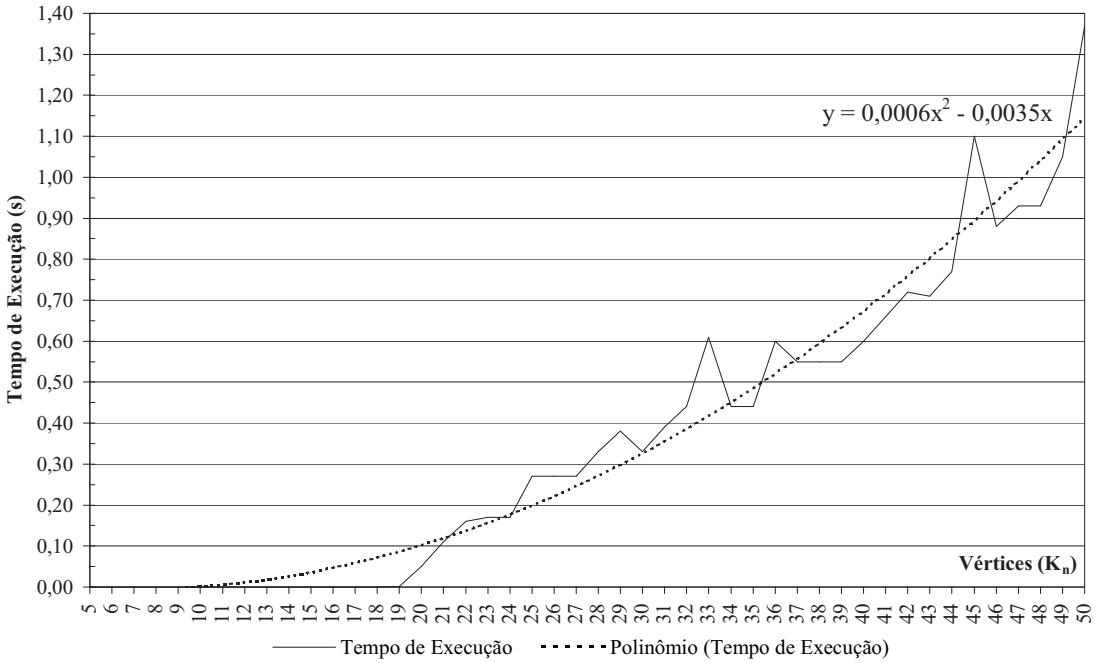


Gráfico 6.11: Tempo de execução do K_5 ao K_{50}

Apesar de ter-se a impressão que com o aumento do problema de entrada (número de vértices maiores do grafo) a algoritmo vai perdendo sua eficiência (maior distância entre as curvas 'melhor caso' e 'Obtido PPD'), na verdade o que acontece é exatamente o inverso, como pode ser verificado pela linha rotulada de 'vértices adicionados' do Gráfico 6.9.

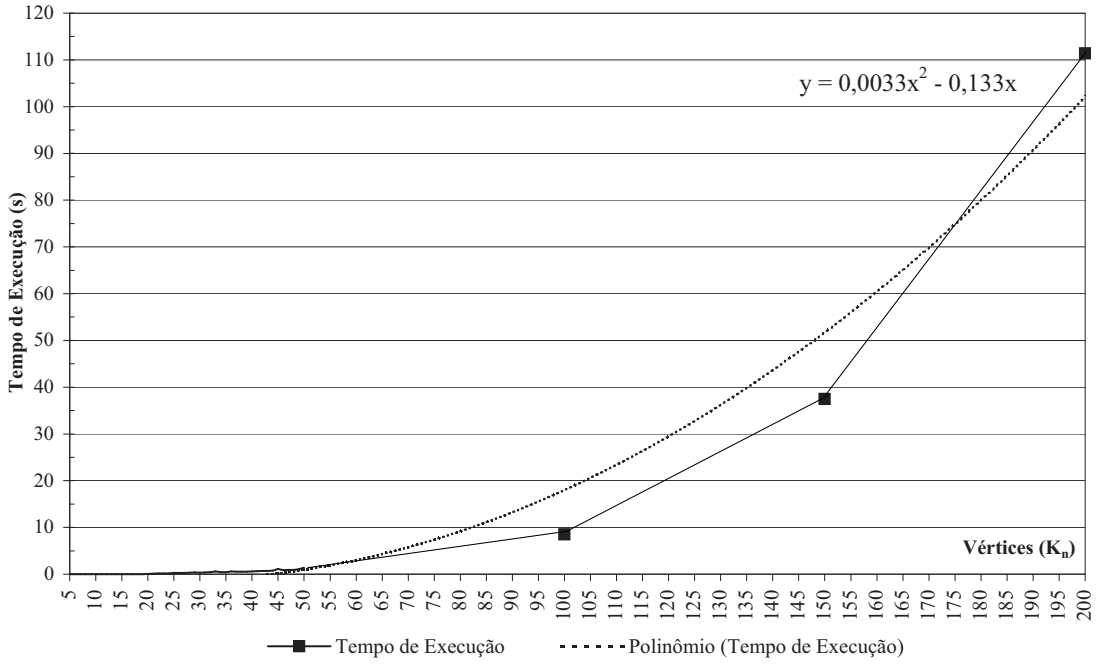


Gráfico 6.12: Tempo de execução do K_5 ao K_{50} + K_{100}, K_{150} e K_{200}

Essa linha, representa a porcentagem da quantidade de vértices adicionados ao grafo, em relação a quantidade original n. Percebe-se que a partir do K_{10}, que acrescenta 71% de vértices, esse valor vai decrescendo até 53% no K_{50}. Isso leva a hipótese que para valores de n ainda maiores, sua eficiência melhorasse mais ainda. Isso é o que procuramos explorar através do Gráfico 6.10.

Nesse gráfico percebe-se a tendência para a melhora da eficiência do algoritmo PPD prossegue, ainda que lentamente. De um montante de 53% de vértices adicionados ao K_{50} passou a 51% para o K_{200}. Infelizmente na versão implementada não foi possível teste com n ainda maiores, pois como se observa, já para esses grafos a quantidade de vértices divididos é extremamente alta.

Com relação ao tempo de execução, mais importante de do analisar a valor nominal em si, é verificar o comportamento da curva. Como provado na seção 5.3, a análise da complexidade do tempo de execução revelou ser $O(n^2)$. Assim, os dados aqui, somente corroboram demonstrando tal comportamento.Tanto o Gráfico 6.11 como o 6.12 mostram que o comportamento quadrático do algoritmo PPD se mantém.

6.2.2 Resultados para os grafos completos bipartidos

Três conjuntos de grafos de entrada foram utilizados para essa classe. Os dois primeiros seguem a mesma lógica da subseção anterior, isto é, um conjunto com grafos completos bipartidos entre o $K_{3,3}$ ao $K_{35,35}$ com número de vértices entre 6 ao 70 ($3 \leq n_1 = n_2 \leq 35$). O segundo conjunto, além de todos os grafos do primeiro conjunto, também contém os grafos $K_{70,70}$, $K_{105,105}$ e o $K_{140,140}$, que podem ser considerados problemas de entrada grandes.

O terceiro conjunto de grafos de entrada foi necessário para testar o comportamento do algoritmo quando $n_1 \neq n_2$. Dessa forma este conjunto possui 561 grafos completos bipartidos, indo do $K_{3,3}$ ao $K_{35,35}$, tendo como intermediários, nesse intervalo, os Grafos $K_{n1,n2}$ ao $K_{n1,35}$, com $n_1 = n_2$ e $3 \leq n_1 \leq 35$.[1]

O algoritmo PPD foi executado para esses três conjuntos de grafos de entrada, gerando como resultado, o número de vértices divididos e o tempo de execução, em segundos, conforme Gráficos 6.13 e 6.16 para o primeiro caso de teste, Gráficos 6.14 e 6.17, para o segundo caso de teste e os Gráficos 6.15 e 6.18, para o terceiro caso.

O desempenho do algoritmo PPD para os grafos completos bipartidos foi melhor do que para os grafos completos. Isso era esperado uma vez que, essa classe de grafos possui

[1]Por exemplo: $K_{3,3}$, $K_{3,4}$, $K_{3,5}$, ..., $K_{3,35}$, $K_{4,4}$, $K_{4,5}$, ..., $K_{4,35}$, e assim por diante até $K_{35,35}$.

aproximadamente a metade de arestas para uma mesma quantidade de vértices, que por conseguinte, significa uma menor quantidade de possíveis cruzamentos de arestas.

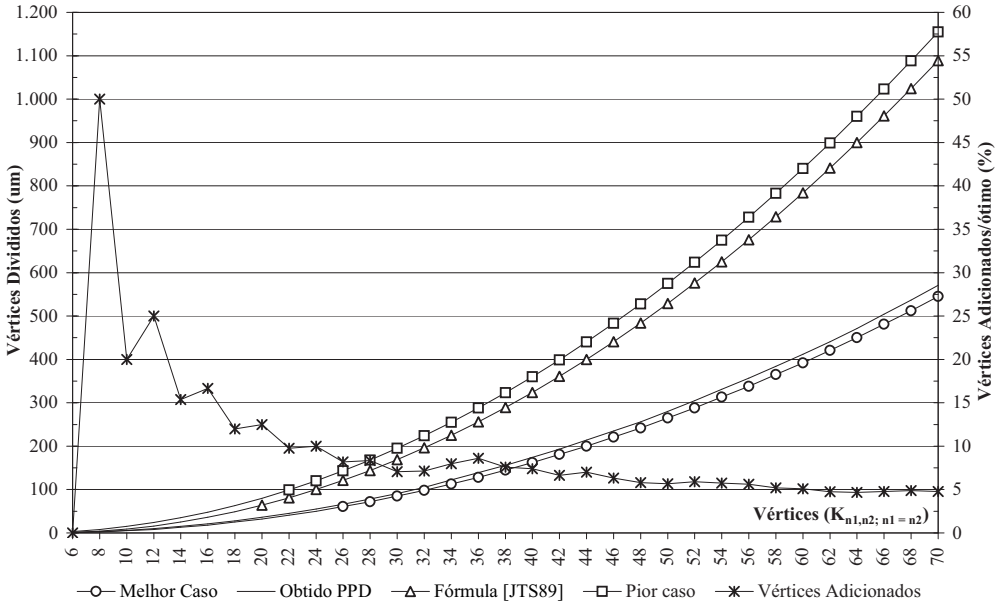


Gráfico 6.13: Número de Vértices Divididos do $K_{3,3}$ ao $K_{35,35}$, sendo $n_1 = n_2$

Como pode ser observado no Gráfico 6.14 a curva rotulada de 'Obtido PPD' apresenta resultados bem próximos a curva do melhor caso. Esse resultado foi atingido, em parte, pela revisão e alteração da regra de formação dos grupos pertinentes para os nós-Q, da fórmula original de Mendonça [17]. Essa modificação foi detalhadamente descrita no item (v) da seção 5.2.1.

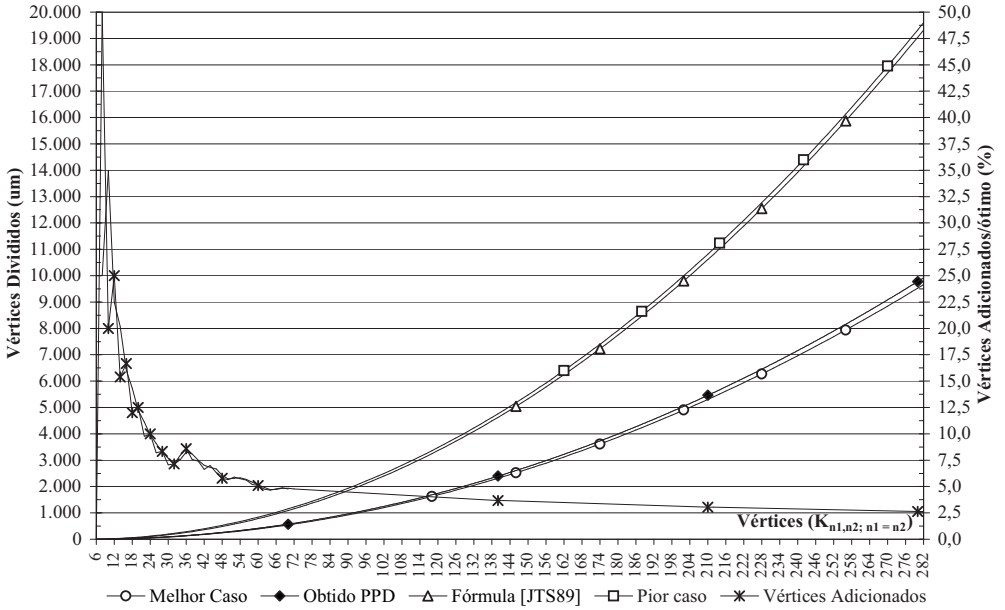


Gráfico 6.14: Número de Vértices Divididos do $K_{3,3}$ ao $K_{35,35}$ + $K_{70,70}$, $K_{105,105}$ e $K_{140,140}$

Também para essa classe de grafos, a eficiência do algoritmo PPD – medida pela diferença entre o número de vértices divididos acrescentados ao grafos original pelo algoritmo em relação ao melhor caso – mostrou-se aumentar com o aumento do problema. Partindo-se de uma porcentagem de 50% de vértices adicionados além do necessário para o $K_{4,4}$, chega-se a somente 5% para o $K_{35,35}$.

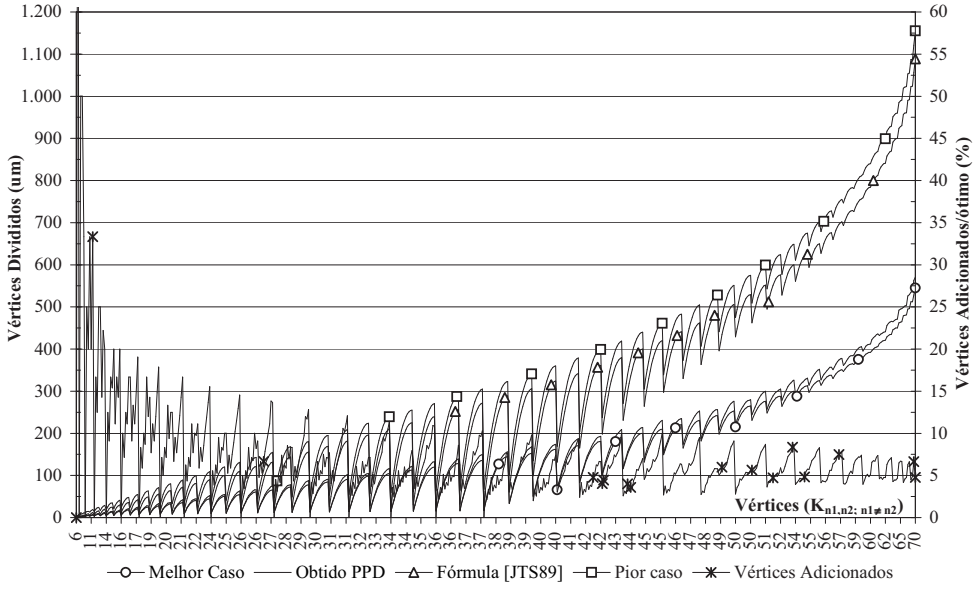


Gráfico 6.15: Número de Vértices Divididos do $K_{3,3}$ ao $K_{35,35}$, sendo $n_1 \neq n_2$

Para o segundo caso de teste, conforme o Gráfico 6.14, pode-se verificar que a eficiência do algoritmo PPD se manteve, isto é, seus resultados continuaram próximos aos do melhor caso. A tendência em dividir vértices se aproximando na situação ótima também se manteve. Como pode ser observado, a curva 'Vértices Adicionados', que se situava a 5% no $K_{35,35}$, caiu para 2,5%, para o último grafo testado $K_{140,140}$ de 280 vértices, resultando em 9.772 vértices divididos dos 9.522 realmente necessários.

Os grafos completos bipartidos com $n_1 \neq n_2$, Gráfico 6.15, também apresentaram comportamento semelhante aos testes anteriores. Além disso, evidenciaram outra situação interessante. Para grafos com a mesma quantidade de total de vértices, quanto mais próximos forem os valores de $n1$ e $n2$, maior será a quantidade de arestas existentes e com isso maior será a quantidade de vértices divididos, diminuindo a eficiência do algoritmo PPD. Por esse motivo é que as curvas do Gráfico 6.15 forma um padrão de zigue-zague.

Com relação ao tempo de execução, os Gráficos 6.16, 6.17 e 6.18 mostraram que a função quadrática foi a que melhor representou seus resultados, corroborando com a demonstração teórica.

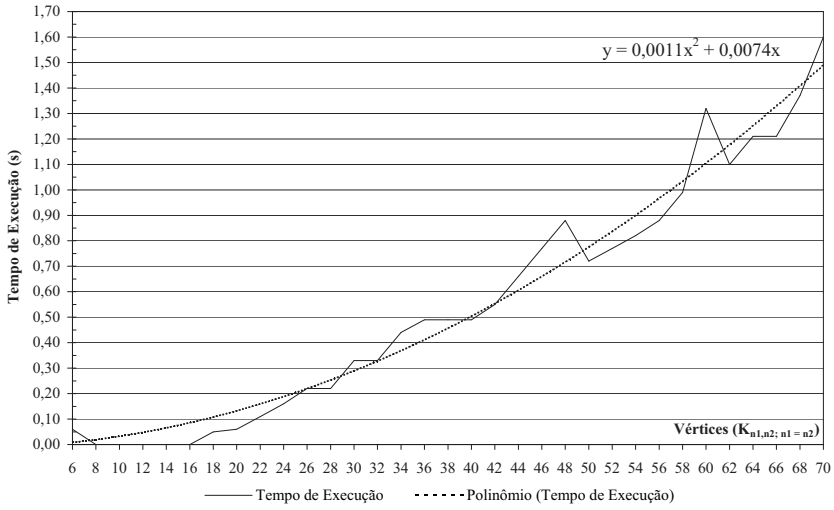


Gráfico 6.16: Tempo de execução do $K_{3,3}$ ao $K_{35,35}$, sendo $n_1 = n_2$

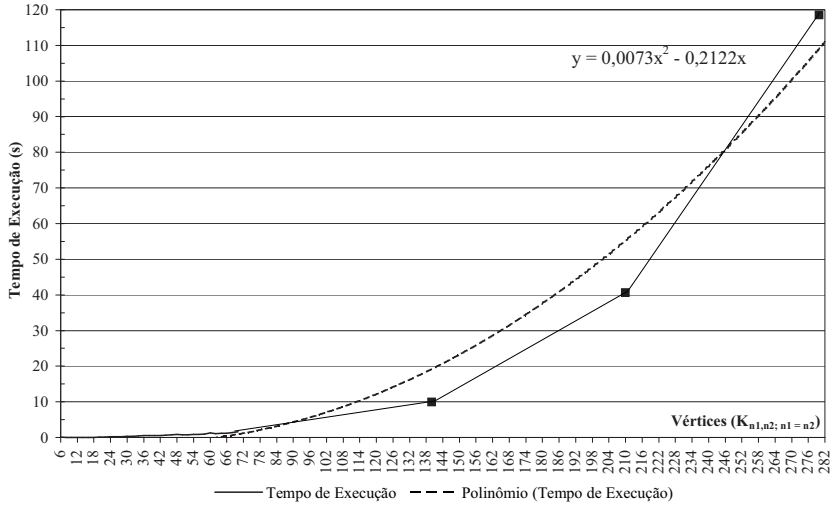


Gráfico 6.17: Tempo de execução do $K_{3,3}$ ao $K_{35,35} + K_{70,70}$, $K_{105,105}$ e $K_{140,140}$

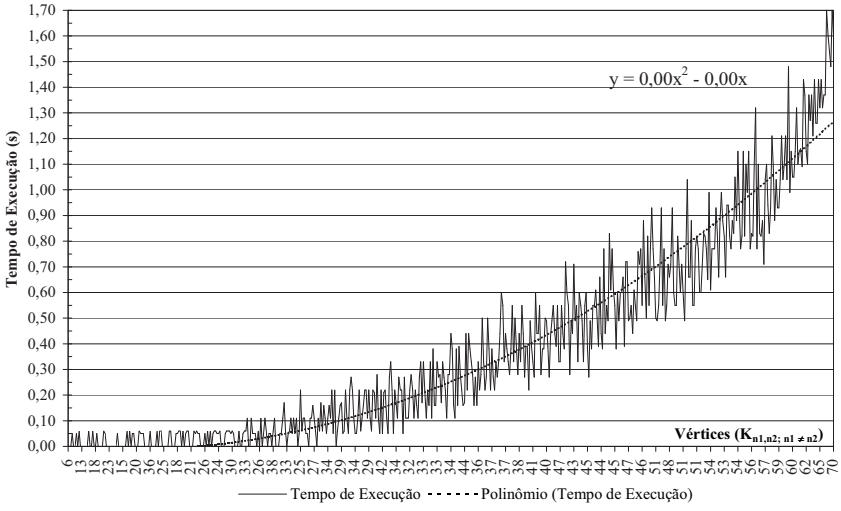


Gráfico 6.18: Tempo de execução do $K_{3,3}$ ao $K_{35,35}$, sendo $n_1 \neq n_2$

6.2.3 Resultados para os grafos Y_k

Essa classe especial de grafos foi utilizada na seção 5.1 para demonstrar a diferença entre a operação de remoção de arestas e divisão de vértices, evidenciando que, para qualquer grafo, o número de vértices divididos é sempre menor ou igual ao número de arestas removidas (proposição 5.1, p. 48).

Especialmente, essa classe de grafos, mostra, conforme a Tabela 6.1, que o número de vértices divididos dos grafos completos K_n convertidos nos Y_k não se altera, e que, para os grafos K_5 e K_6 convertidos tanto no Y_2 quanto no Y_3, o resultado obtido foi igual a solução ótima.

A última coluna mostra os valores de arestas removidas para esses grafos também. Nota-se que o número de arestas a serem removidas é proporcional ao número de k caminhos adicionados ao grafos K_n original multiplicado pela número de arestas removidas do grafo original. Já na divisão de vértices, dividindo apenas aquele vértice do grafo original K_n, significa dizer que, todas as arestas dissidentes dele que, por ventura estavam causando cruzamentos, não mais estarão, após o vértice ser dividido.

Y_2					
K_n	*Vértices*	*Arestas*	vértices divididos		arestas removidas
			Fórmula	*Algoritmo PPD*	*Fórmula*
5	25	40	1	1	2
6	36	60	2	2	6
7	49	84	3	4	12
8	64	112	4	6	20
Y_3					
5	35	60	1	1	3
6	51	90	2	2	9
7	70	126	3	4	18
8	92	168	4	6	30
K_n					
5		10	1	1	1
6		15	2	2	3
7		21	3	4	6
8		28	4	6	10

Tabela 6.1: Número de vértices divididos para os grafos k_n convertidos nos Y_k

6.2.4 Resultados para os grafos Cartesianos

Para essa classe de grafos utilizamos somente um único conjunto de grafos de entrada, com idéia semelhante ao dos grafos completos bipartidos. Uma vez os grafos cartesianos também possuem $n1$ e $n2$ conjuntos distintos de vértices, foram testadas as situações com $n_1 = n_2$ e $n_1 \neq n_2$. Dessa forma este conjunto possui 276 grafos cartesianos, indo do C_3 x C_3 ao C_{25} x C_{25}, tendo como intermediários, nesse intervalo, os Grafos C_{n1} x C_{n2} ao C_{n1} x C_{25}, com $n_1 = n_2$ e $3 \leq n_1 \leq 25$.[2]

Essa classe de grafo também possibilitou evidenciar outra importante função do algoritmo PPD que diz respeito a política de escolha de qual grupo pertinente irá receber as folhas $(i + 1)$, conforme descrita no item 3 da página 3. Como para essa classe de grafos o número de arestas removidas e vértices divididos, obtidos pelas suas respectivas fórmulas, são iguais, omitimos a curva rotulada de 'Fórmula [JTS89]' e adicionados duas curvas obtida pela execução do algoritmo PPD.

A rotulada 'Obtido PPD Q+alto' é a política de escolher um grupo pertinente sem um nó-Q ancestral ou o nó-Q ancestral mais alto. Ela permite, na maioria dos casos, um número maior de possibilidades de combinações para os descendentes de $i + 1$. A curva rotulada de 'Obtido PPD Q+baixo', por sua vez, ilustra a situação totalmente inversa a essa, isto é, escolhe-se o grupo pertinente que seja filho de um nó-Q.

O Gráfico 6.19 mostra uma diferença significativa no desempenho do algoritmo PPD com relação as essas duas políticas. Nitidamente, a política de escolher o nó-Q mais alto, possibilitou melhores resultados, sendo que quanto maior o problemas de entrada, com relação ao número de vértices, maior foi a diferença de desempenho. Entretanto, essa política nem sempre apresentou os melhores resultados.

O tempo de execução aqui também corroborou com prova teórica como mostra o Gráfico 6.20. Apesar do número melhor de arestas que essa classe de grafos possui em relação as anteriores, seu número de vértices é grande, o que contribuiu com valores maiores de tempo de execução. De fato, como discutido na análise da complexidade do algoritmo, seção 5.3 da página 62, a sua complexidade está em função do número de vértices e não do número de arestas.

[2]Por exemplo: C_3 x C_3, C_3 x C_4, C_3 x C_5, ..., C_3 x C_{25}, C_4 x C_4, C_4 x C_5, ..., C_4 x C_{25}, e assim por diante, até C_{25} x C_{25}.

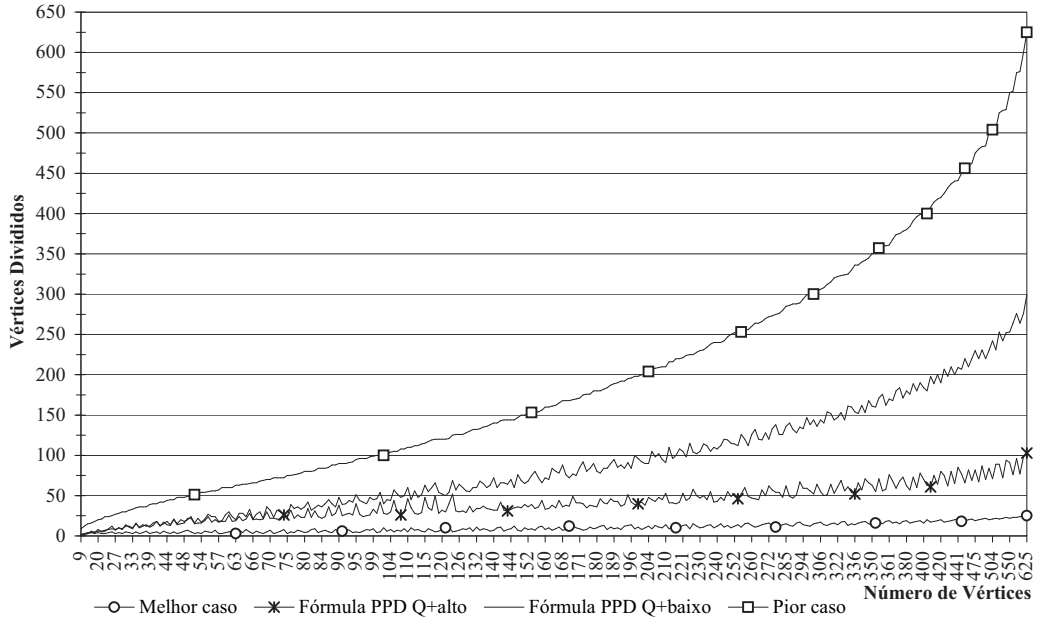


Figura 6.19: Número de Vértices Divididos do C_3 x C_3 ao $C_{25,25}$ x $C_{25,25}$, sendo $n_1 \neq n_2$

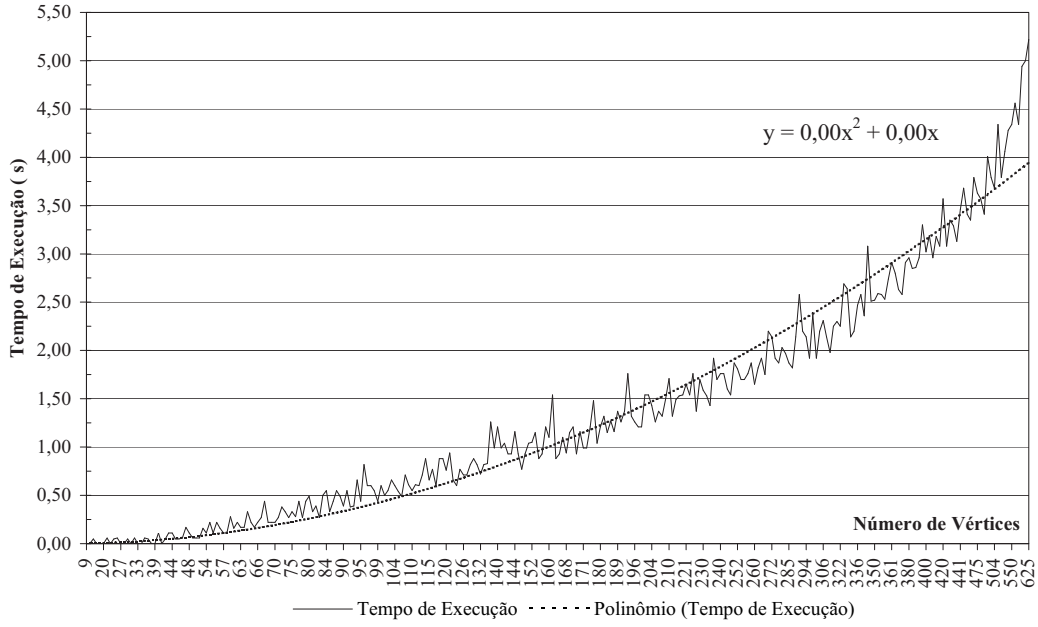


Figura 6.20: Tempo de execução do C_3 x C_3 ao $C_{25,25}$ x $C_{25,25}$, sendo $n_1 \neq n_2$

Capítulo 7
Conclusão

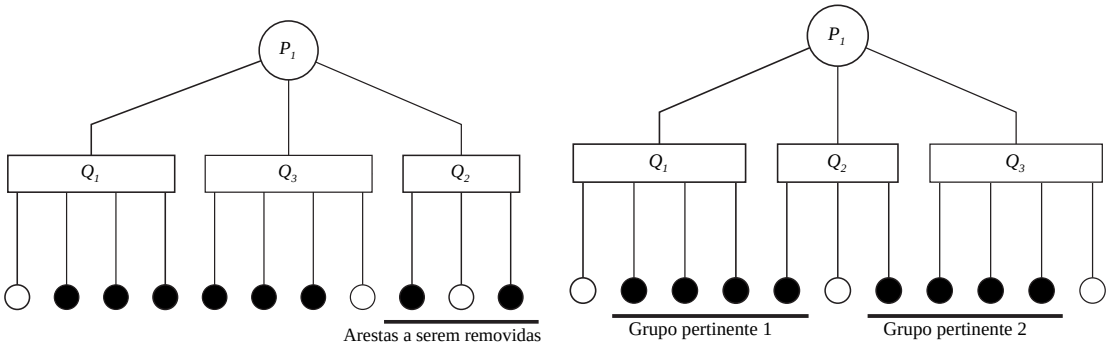
A dissertação investigou problemas relacionados a planarização de grafos e apresentou um novo método de planarização baseado na *divisão de vértices*, alternativamente aos mais conhecidos e reportados na literatura baseados na *remoção de vértices* ou *arestas*, no *particionamento planar* ou na *introdução de vértices dummy*. Independente do método empregado, qualquer grafo não planar ao ser planarizado tem sua estrutura original modificada.

Entretanto, o grau com que, cada método de planarização modifica uma grafo, ocasionando perdas de informação, varia significativamente. A vantagem da planarização de grafos pela *divisão de vértices* recai sobre o fato de que ele pode ser usado em aplicações até então não permitidas, como os vários tipos de representações utilizadas em projeto de bancos de dados, uma vez que ele não destrói a principal informação dessas representações: o relacionamentos entre as entidades.

A conclusão dessa dissertação adicionada uma etapa a mais ao trabalho iniciado na Tese de Mendonça [17], que desde aquela época, em nosso conhecimento, não havia relatos da continuação do estudo desse novo método do ponto de vista de sua implementação, a não ser a importante publicação de Luérbio, Figueiredo e Mendonça [27] comprovando que essa operação pertence a classe de problemas NP-completos.

Significativos foram os resultados obtidos na elaboração desse trabalho, ora baseados na revisão da teoria, isto é, no estudo e revisão dos conceitos e fórmulas desenvolvidas, originalmente, por Mendonça [17], ora baseados nos exaustivos testes precedidos da implementação do algoritmo PPD. A combinação dessas duas atividades foram os ingredientes que o tornaram o trabalho estimulante e desafiador, gerando grande satisfação e sentimento de 'dever cumprido' ou 'quase cumprido' no autor.

Entre os resultados importantes, um que merece ser enfatizado, é a diferença das fórmulas de Jayakumar, Thulasiraman e Swamy [44] e as aqui utilizadas, vindas de Mendonça [17], pelo algoritmo Planariza por Divisão. Enquanto as primeiras buscam, para o problema do *número de arestas removidas*, maximizar o número de folhas pertinentes consecutivas, as últimas, para o problema *número de vértices divididos*, minimizam o número de grupos pertinentes. A Figura 7.1 ilustra a diferença das abordagens.



(a) Maximizando folhas pertinentes (b) Minimizando grupos pertinentes

Figura 7.1: Exemplo da diferença entre as abordagens de [44] e [17].

Não menos importante foi, também, a abstração de um novo tipo de nó, chamado de nó tipo **V**, representando na Figura 7.1 pelo nó-Q rotulado de Q_2. Esse tipo de nó tem um papel importante na minimização do número de grupos pertinentes, através das regras de combinações e trocas, descritas na subseção 5.2.2 da página 60. Esse tipo de nó não existe na abordagem de Jayakumar, Thulasiraman e Swamy [44].

Da revisão das fórmulas originais de Mendonça e dos seguidos testes após as n versões (pelo menos umas 10), duas pequenas alterações forem realizadas. Apesar de serem modificações simples, elas tiveram grande impacto na eficiência do algoritmo, eficiência essa avaliada pelo número de vértices divididos. A primeira diz respeito sobre o cálculo para transformar um certo nó X em um nó tipo **A** e a segunda diz respeito a regra de combinação dos filhos de um nó-Q parcial.

A descrição das duas melhores formas de se combinar os filhos de um nó-P para que se tenha o menor número possível de grupos pertinentes estava correta (ver item 'Para fazer o nó X um nó tipo A', na página 58). Entretanto as fórmulas que calculam a quantidade renomeações necessárias dos grupos pertinentes para não-pertinente, após as aplicadas as operações de combinação e troca originais da árvore-PQ, para se obter o menor número de grupos pertinentes, estavam invertidas. O correto é

$$a = \begin{cases} \lceil \alpha - \theta_1 - \theta_2 \rceil & \text{se } \theta_1 \ + \ \theta_2 < 1 \text{ e } \theta_a \ = \ \frac{1}{2} \\ \lceil \alpha - 1 \rceil & \text{se } \theta_1 \ + \ \theta_2 = 1 \\ m & \text{caso contrário} \end{cases}$$

ao invés de

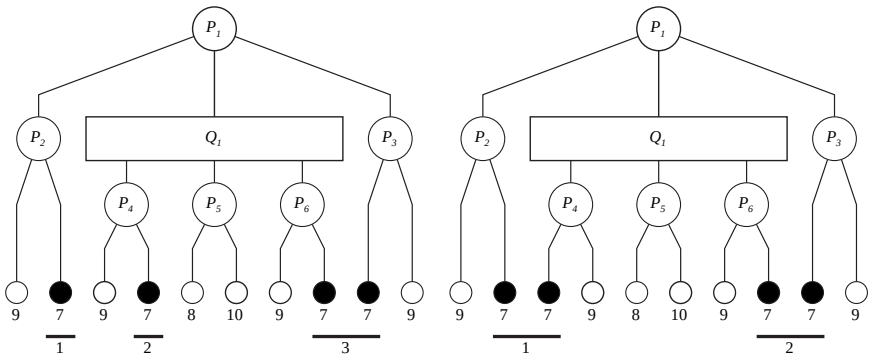
$$a = \begin{cases} \lceil \alpha - \theta_1 - \theta_2 \rceil & \text{se } \theta_1 \ + \ \theta_2 = 1 \\ \lceil \alpha - 1 \rceil & \text{se } \theta_1 \ + \ \theta_2 < 1 \text{ e } \theta_a \ = \ \frac{1}{2} \\ m & \text{caso contrário} \end{cases}$$

sendo que, isso imputava um valor a errado quando essa possibilidade acontecia durante as várias iterações do algoritmo PPD, produzindo resultados inesperados.

A segunda mudança, como comentado, altera a regra de formação dos grupos pertinentes de um nó-Q parcial, sendo que a mudança melhorou a eficiência do algoritmo por permitir que em certas situações outra possibilidade de combinação fosse contemplada. Como o item (v) ('O nó X é um nó-Q parcial') da subseção 5.2.1, da página 56, descreveu somente a nova forma adotada, explicitaremos na Tabela 7.1, sucintamente, qual foi essa mudança e daremos um exemplo, Figura 7.2, que ilustra seu impacto na formação dos grupos pertinentes.

<table>
<tr><td colspan="2">1. Atravessar os filhos de X da Esquerda ➔ Direita</td></tr>
<tr><td colspan="2">2. Para cada nó i parcial, calcular $\mu = min\{h_i + ½ \ , \ a_i + 1 \ , \ v_i + 1\}$</td></tr>
<tr><td colspan="2">3. Se $\mu = h_i + ½ = v_i + 1 \leq a_i + 1$ ou $\mu = h_i + ½ = a_i + 1 \leq v_i + 1$, então i é feito tipo H_l ou H_r com se segue:</td></tr>
<tr><td>Se o nó i é o filho mais à esquerda de X ou se seu irmão da esquerda não é um nó do tipo B, V, nem H_r
então o nó i é feito tipo H_r
senão o nó i é feito tipo H_l .</td><td>Se nó i é o filho mais a esquerda de X
então
Se seu irmão da <u>direita</u> não é um nó do tipo B, V, nem H_r
então H_l
senão H_r
senão
Se seu irmão da <u>esquerda</u> não é um nó do tipo B, V, nem H_r
então o nó i é feito tipo H_r
senão o nó i é feito tipo H_l .</td></tr>
</table>

Tabela 7.1: Regra para formação dos grupos pertinentes dos filhos de um nó-Q parcial

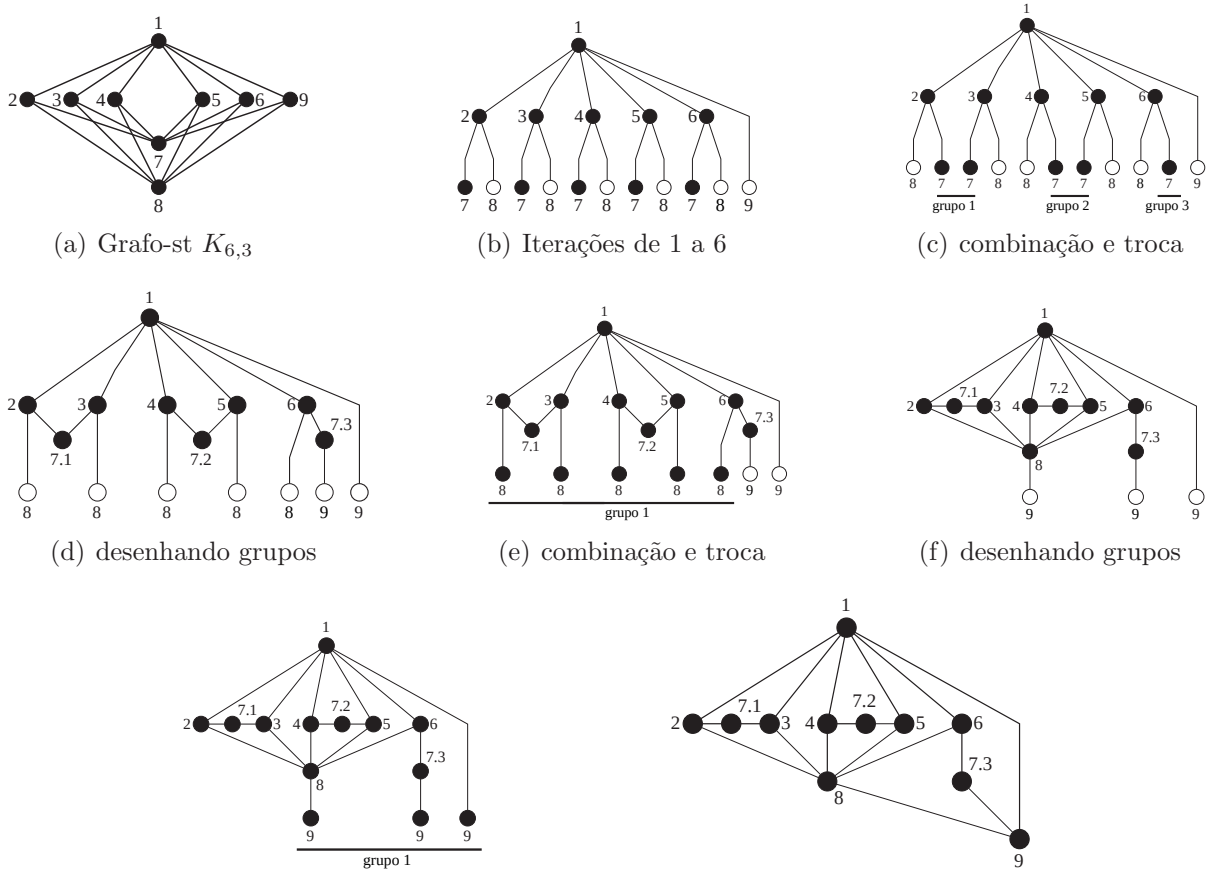


(a) Regra original – anterior (b) Regra modificada – atual

Figura 7.2: Exemplo da regra de formação dos grupos pertinentes de um nó-Q parcial

Existem também alguns resultados que abrem margem a investigações Futuras. Uma deles que, merece ser destacado apesar de sua simplicidade e pouca discussão no texto, é o da política de escolha em qual dos nós-P – criados pela substituição dos grupos pertinentes por esses – serão adicionados as próximas folhas pertinentes da iteração $i + 1$. Ele teve um grande impacto no desempenho do algoritmo. Entretanto pesquisas futuras merecem serem feitas a esse respeito, pois nos casos de teste utilizados ficou clara que o resultado variava, entre e dentro das classes de grafos utilizadas.

Outra evidência encontrada e que merece estudos futuros, ainda que o autor acredite ser pouco provável encontrar-se um regra geral *a priori* é a respeito da st-numeração. Para um mesmo grafo de entrada conforme o mesmo for numerado, poderá resultar em mais ou menos quantidade de vértices divididos. Infelizmente, por limitações de tempo e escopo, não nos aprofundamos nesse aspecto. A a Figura 7.3, ilustra essa situação para o mesmo grafo utilizado na seção 6.1, sendo que ao invés de termos 3 vértices divididos, aqui será apenas dois os necessários, refletindo a melhor solução possível. A Figura apresenta somente a Forma Arbusto para cada iteração.



(a) Grafo-st $K_{6,3}$ (b) Iterações de 1 a 6 (c) combinação e troca

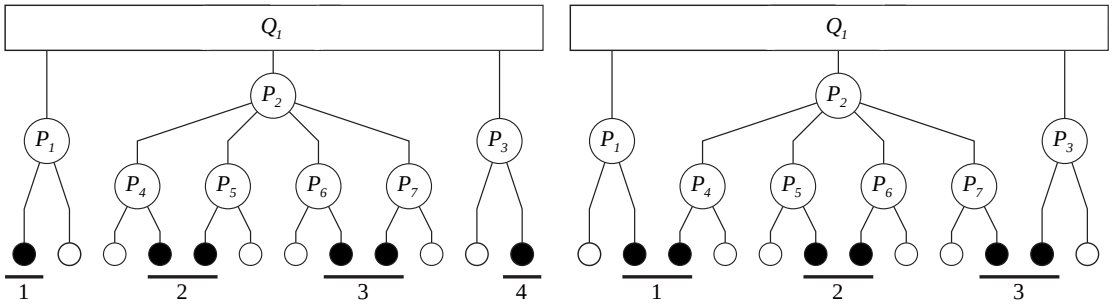
(d) desenhando grupos (e) combinação e troca (f) desenhando grupos

(g) combinação e troca (h) desenho final

Figura 7.3: Exemplo da influencia da st-numeração

Fora essa, outra situação encontrada, que o autor acredita ter possibilidades melhores de avançar, a despeito da anterior, é com relação ao melhoramento do cálculo dos valores tetas, que são a base para o cálculo dos valores *b, h, w, a* e *v*, que corresponde, em última instância, na quantidade de renomeações dos grupos pertinentes para não-pertinentes necessárias para se fazer um nó *X* do tipo *B, W, H, A* e *V*. respectivamente.

Durante os vários casos testados, aparecem situações em que a regra de formação para os filhos de um nó-Q parcial com algum(ns) de seus(s) de seus filho(s) também sendo parcial(ais) que geravam um número maior de grupo pertinentes do que realmente era necessário. Infelizmente, não conseguimos abstrair uma regra padrão genérica para esses casos, e por isso tais casos não são considerados. A Figura 7.4 ilustra tal situação.



(a) Resultado obtido (b) Resultado desejado

Figura 7.4: Caso não considerado no cálculo do valores tetas

Quando estávamos, por assim dizer, no meio do caminho de nossa dissertação, nos deparamos com duas novas estruturas de dados que também lidam com propriedades dos 1's consecutivos, como a árvore-BC de Wen-Lian Hsu [40, 68, 41, 42] e a Boyer e Myrvold [11]. Segundo os autores, ambas são estruturas muito mais simples de entendimento e implementação do que a árvore-PQ. Contudo, ambos, até o presente momento somente descreveram a base conceitual e provas de seus algoritmos, sendo ainda, pouco reportado na literatura artigos com os resultados de sua implementação para planaridade de grafos.

Assim, estudos futuros que possam avançar na aplicação desses novos conceitos para teste e planarização e grafos promete ser promisor. Até por que como já reportado por Jünger, Leipert e Mutzel [45, 46, 47] a própria estrutura da árvore-PQ apresenta sérias limitações a detectar certas configurações que vão surgindo a cada iteração e que por isso, comprometem a eficiência do algoritmo. Esses autores demonstraram essa debilidade utilizando o problema do subgrafo planar ótimo (*Maximal Planar Subgraph*).

Por fim, falta mencionarmos mais dois pontos importantes. O primeiro deles alterações foram necessárias e por isso feitas no algoritmo original da árvore-PQ de Young [81].

Infelizmente gostaríamos de ter detalhado não só nosso algoritmo PPD, como também entrado nos detalhes da implementação dos algoritmos de teste de planaridade, de busca em profundidade e da numeração-st. Todos esse foram implementados e foram um pouco alterados para lidarem com o problema de *número de vértices divididos*.

Contudo, esse nível de detalhamento adicionaria grande volume de páginas a dissertação e nos prenderia a detalhes que já foram reportados em outros trabalhos como, por exemplo, nos trabalhos internacionais de Young [81] e Leipert [53] e nacionais de Carmo [12], Baldas [3] e Eugênio [60] para a árvore-PQ, nos de Even e Tarjan [24, 25, 23, 69] e Rodrigues [19], respectivamente, para a numeração-st e busca em profundidade, e muitos outros livros que tratam desse assunto como os de Manber [58], Gibbons [30], Bondy e Murty [9], Even [23], Thulasiraman e Swamy [70], Nishizeki e Chiba [65] e Harary [35].

Dessa maneira, temos como meta futura, a elaboração de artigos e relatórios técnicos que tratam em maiores detalhes esses assuntos não tratados aqui, bem como, a elaboração de um código do algoritmo PPD muito mais comentado e documentado que possibilite sua utilização em trabalhos futuros, uma vez que das duas grandes partes que o compões, o código original do algoritmo da árvore-PQ foi alterado e o algoritmo PPD, em si, foi totalmente desenvolvido, gerando, 2.322 das 3.608 totais.

O segundo e último ponto a destacar, apesar de já termos feito isso na seção agradecimentos, mas acredito ter sido pouco, foi o fato de ter como orientador e co-orientador dois professores altamente motivados em entusiasmados com o assunto. O conhecimento de ambos, a disponibilidade e a segurança que me transmitiram foi essenciais para todo o processo.

Referências Bibliográficas

[1] Annexstein, F.; Swaminathan, R. On testing consecutive-ones property in parallel. *Discrete Applied Mathematics*, 88 (1998), 7–28.

[2] Auslander, L.; Parter, S. V. On embendding graphs in the sphere. *Journal of Mathematics and Mechanics 10*, 3 (1961), 517–523.

[3] Baldas, M. T. M. L. Reconhecimento e traçado de grafos planares. Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, COPPE, 1995.

[4] Battista, G. D.; Nardelli, E. Hierarchies and planarity. *IEEE Trans. on Systems, Man., and Cybernetics* (1988).

[5] Battista, G. D.; Tamassia, R.; Eades, P.; Tollins, I. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, 4 (1994), 235–282.

[6] Battle, J.; Harary, F.; Kodama, Y. Every planar graph with nine points has a noplanar complement. *Bulletin of the American Mathematical Society 68* (1962), 569–571.

[7] Beineke, L. W. Complete bipartide graphs: decomposition into planar subgraphs. In *A seminar on Graph Theory* (Holt, 1967), F. Harary, Ed., Rinehart and Winston.

[8] Beineke, L. W.; Harary, F.; Moon, J. W. On the thickness of the complete bipartide graph. *Proc. Camb. Phil. Soc. 60* (1964), 1–5.

[9] Bondy, J. A.; Murty, U. S. R. *Graph Theory with Applications*. North-Holland, 1979.

[10] Booth, K. S.; Lueker, G. S. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-Tree algorithms. *Journal of Computer and System Sciences*, 13 (1976), 335–379.

[11] Boyer, J.; Myrvold, W. Stop minding your P's and Q's: a simplified O (n^2) planar embedding algorithm. In *Procedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'99* (1999), pp. 104–146.

[12] Carmo, R. J. S. O problema do subgrafo planar ótimo. Dissertação de Mestrado, Instituto de matemática e Estatística da Universidade de São Paulo, São Paulo, 1994.

[13] Chiba, N.; Nishizeki, T.; Abe, S.; Ozawa, T. A linear algorithm for embedding planar graphs using PQ-Trees. *Journal of Computer and System Sciences 30*, 1 (February 1985), 54–75.

[14] Chiba, N.; Onoguchi, K.; Nishizeki, T. Drawing planar graphs nicely. *Acta Inform. 22* (1985), 187–201.

[15] Chiba, T.; Nishioka, I.; Shirakawa, I. An algorithm of maximal planarization of graphs. In *Proc. 1979 IEEE Int. Symp. on Circuits and Systems* (1979), pp. 648–652.

[16] de Fraysseix, H.; Rosenstiehl, P. A depth first characterization of planarity. *Annals of Discrete Mathematics 13* (1982), 75–80.

[17] de Mendonça Neto, C. F. X. *A Layout System for Information System Diagrams.* Tese de Doutorado, University of Queensland, Department of Computer Science, 1993.

[18] de Mendonça Neto, C. F. X.; Schaffer, K.; Xavier, E. F.; Stolfi, J.; Faria, L.; de Figueiredo, C. M. H. The splitting number and skewness of C_n x C_m. *ARS Combinatoria 63* (2002), 193–205.

[19] de Souza Rodrigues, T. A st-numeração e suas aplicações. Dissertação de Mestrado, Instituto Militar de Engenharia, Rio de Janeiro, 2001.

[20] Eades, P.; de Mendonça Neto, C. F. X. Heuristics for planarization by vertex splitting. In Proc. ALCOM Int. Workshop on Graph Drawing, GD'93, pages 83–85, 1993.

[21] Eades, P.; de Mendonça Neto, C. F. X. Vertex splitting and tension-free layout. In *Graph Drawing, Proceedings of the DIMACS International Workshop, GD'95* (1996), F. J. Brandenburg, Ed., vol. 1027 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 202–211.

[22] Eades, P.; Elgindy, H.; Houle, M.; Lenhart, B.; Miller, M.; Rappaport, D.; Whitesides, S. Dominace drawings of bipartide graphs. Manuscript, University of Newcastle, NSW, Australia, 1993.

[23] Even, S. *Graph Algorithms.* Computer Science Press, Potomac, Maryland, 1979.

[24] Even, S.; Tarjan, R. E. Computing an st-numbering. *Theoretical Computer Science 2*, 3 (september 1976), 339–344.

[25] Even, S.; Tarjan, R. E. Corrigendum: Computing an st-numbering. *Theoretical Computer Science 4*, 1 (february 1977), 123.

[26] Faria, L.; de Figueiredo, C. M. H.; de Mendonça Neto, C. F. X. Splitting number is NP-Complete. In *Theoretical Informatics. Proceedings of the 3rd Latin American Symposium, LATIN'98* (1998), C. L. L. e Arnaldo V. Moura, Ed., vol. 1380 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 141–150.

[27] Faria, L.; de Figueiredo, C. M. H.; de Mendonça Neto, C. F. X. Splitting number is NP-Complete. *Discrete Applied Mathematics 108* (2001), 65–83.

[28] Garey, M. R.; Johnson, D. S. *Computers and Intractability: a guide to the theory of NP-Completeness.* W H Freeman Co Ltd, 1979.

[29] Garey, M. R.; Johnson, D. S. Crossing number is NP-Complete. *SIAM Journal on Algebric and Discrete Methods 4*, 3 (1983).

[30] Gibbons, A. *Algorithmic Graph Theory.* Cambridge University Press, 1994.

[31] Goldstein, A. J. An efficient and constructive algorithm for testing whether a graph ca be embedded in a plane. In *Graph and Combinatorics Conference* (Office of Naval Research Logistics Proj., May 16-18 1963), Contract No NONR 1858-(21), Dept. of Mathematics, Princeton University.

[32] Gutwenger, C.; Jünger, M.; Klau, G. W.; Leipert, S.; Mutzel, P. Graph drawing algorithm engineering with AGD. Relat. Técnico 2000-394, Universität zu Köln, Institut für Informatik, Pohligstraße 1 50969 Köln, 2000.

[33] Guy, R. K. Latest results on crossing numbers. In *Recent Trends in Graph Theory* (1971), M. Capobianco; J. B. Frechen;; M. Krolik, Eds., vol. 186 of *Lecture Notes in Mathematics*, Proceedings of the First New York City Graph Theory Conference, June 11–13, 1970, Springer-Verlag, pp. 143–156.

[34] Guy, R. K. Crossing numbers of graphs. In *Graph Theory and Applications* (1972), Y. Alavi; D. R. Lick;; A. T. White, Eds., vol. 303 of *Lecture Notes in Mathematics*, Proceedings of the Conference at Western Michigan University, May 10–13, 1972, Springer-Verlag, pp. 111–124.

[35] Harary, F. *Graph Theory.* Addison Wesley, 1969.

[36] Hartsfield, N. *The toroidal splitting number of the complete graph K_n.* Tese de Doutorado, University of California, Department of Mathematics, Santa Cruz, 1984.

[37] Hartsfield, N.; Jackson, B.; Ringel, G. The splitting number of complete graphs. *Graphs and Combinatorics, an Asian Journal 1*, 2 (1985).

[38] Hobbs, A. M. A survey of thickness. In *Recent progress in combinatorics* (1969), W. T. Tutte, Ed., Proceedings of the 3rd Waterloo Conference on combinatorics, May 20-31, 1968, Academic Press, New York, pp. 255–264.

[39] Hopcroft, J.; Tarjan, R. E. Efficient planarity testing. *Journal of Association for Computing Machinery 21*, 4 (1974), 549–568.

[40] Hsu, W.-L. A linear time algorithm for finding maximal planar subgraphs. In *ISAAC: 6th SIGAL International Symposium on Algorithms* (1995).

[41] Hsu, W.-L. PC-Trees vc. PQ-Trees. In *Lecture Notes in Computer Science* (2001), no. 2108, pp. 207–217.

[42] Hsu, W.-L. A simple test for the consecutive ones property. *Journal of Algorithms 43* (2002), 1–16.

[43] Jackson, B.; Ringel, G. The splitting number of complete bipartite graphs. *Arch. Math. 42* (1985), 178–184.

[44] Jayakumar, R.; Thulasiraman, K.; Swamy, M. N. S. O (n^2) algorithms for graph planarization. *IEE Transactions on Computer-Aided Design 8*, 3 (March 1989), 257–267.

[45] Jünger, M.; Leipert, S.; Mutzel, P. On computing a maximal planar subgraph using PQ-Trees. Relat. Técnico 96.227, Universität zu Köln, Institut für Informatik, Pohligstraße 1 50969 Köln, 1996.

[46] Jünger, M.; Leipert, S.; Mutzel, P. Pitfalls of using PQ-Trees in automatic graph drawing. In *Graph Drawing* (1997), pp. 193–204.

[47] Jünger, M.; Leipert, S.; Mutzel, P. A note on computing a maximal planar subgraph using PQ-Trees. Relat. Técnico 98.320, Universität zu Köln, Institut für Informatik, Pohligstraße 1 50969 Köln, 1998.

[48] Kant, G. An O (n^2) maximal planarization algorithm based on PQ-Trees. Relat. Técnico RUU-CS-92-03, Utrecht University, Department of Computer Science, January 1992.

[49] Kant, G. *Algorithms for Drawing Planar Graphs*. Tese de Doutorado, Universiteit Utrecht, Faculteit Wiskunde en Informatica, jan. 1993.

[50] Kleitman, D. The crossing number of $k_{5,n}$. *Journal of Combinatorial Theory 9* (1970), 315–323.

[51] Kuratowski, K. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematic 15* (1930), 271–283.

[52] Leighton, F. T. New lower bound techniques for VLSI. *Math. Systems Theory 17* (1984), 47–70.

[53] Leipert, S. PQ-Trees: An implementation as template class in C^{++}. Relat. Técnico 97.259, Universität zu köln, Institut für Informatik, January 1997.

[54] Lempel, A.; Even, S.; Cederbaum, I. An algorithm for planarity testing of graphs. In *Theory of Graphs: Internat. Symposium (Rome 1966)* (New York, 1967), Gordon and Breach, pp. 215–232.

[55] Lewis, J. M.; Yannakakis, M. The node-deletion problem for hereditary properties is NP-Complete. *Journal of Computer and System Sciences 20* (1980), 219–230.

[56] Liebers, A. Planarizing graphs - a survey and annotatted bibliography. *Journal of Graph Algorithms and Applications 5*, 1 (2001), 1–74.

[57] Liu, P. C.; Geldmacher, R. C. On the deletion of nonplanar edges of a graph. In *Proc. 10th South East Conference on Combinatorics, Graph Theory, and Computing* (1979), pp. 727–738.

[58] Manber, U. *Introduction to Algorithms: a creative approach*. Addison-Wesley, 1989.

[59] Mansfield, A. Determining the thickness of graphs is NP-Hard. *Math. Proc. Camb. Phil. Soc. 93* (1983), 9–23.

[60] Marins, E. R. Traçado automático de grafos hierárquicos. Dissertação de Mestrado, Instituto Militar de Engenharia, Rio de Janeiro, Maio 1997.

[61] Mehlhorn, K.; Mutzel, P. On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. *Algorithmica 16*, 2 (August 1996), 233–242.

[62] Mehlhorn, K.; Mutzel, P.; Näher, S. An implementation of the Hopcroft and Tarjan planarity test and embedding algorithm. Relat. Técnico MPI-I-93-151, Max-Planck Institut für Informatik, Saarbrücken, 1993.

[63] Meidanis, J.; Porto, O.; Telles, G. P. On the consecutive ones property. *Discrete Applied Mathematics*, 88 (1998), 325–354.

[64] Mutzel, P.; Odenthal, T.; Scharbrodt, M. The thickness of graphs: a survey. *Graphs and Combinatorics 14* (1998), 59–73.

[65] Nishizeki, T.; Chiba, N. *Planar Graphs: Theory and Algorithms*, vol. 32 of *Annals of Discrete Mathematics*. North-Holland, Amsterdam, The Netherlands, 1988.

[66] Ozawa, T.; Takahashi, H. A graph-planarization algorithm and its applications to random graphs. In *Graph Theory and Algorithms* (1981), no. 108 in Spring-Verlag Lecture Notes in Computer Science, pp. 95–107.

[67] Schaffer, K. *The Splitting Number and Other Topological Parameters of Graphs*. Tese de Doutorado, University of California, Department of Mathematics, March 1986. Unpublished.

[68] Shih, W.-K.; Hsu, W.-L. A new planarity test. *Theoretical Computer Science 223* (1999), 179–191.

[69] Tarjan, R. E. Two streamlined deph-first search algorithms. *Fundamenta Informaticae*, IX (1986), 85–94.

[70] Thulasiraman, K.; Swamy, M. N. S. *Graphs: Theory and Algorithms*. Wiley-Interscience, February 1992.

[71] Turán, P. A note of welcome. *Journal of Graph Theory 1* (1977), 7–9.

[72] Tutte, W. T. The no-biplanar character of the complete 9-graph. *Canadian Mathematical Bulletin 6* (1963), 319–330.

[73] Vollen, G. *PQ-Trees and Maximal Planarization: an approach to skewness*. Tese de Doutorado, University of Solo, Department of Informatics, February 1998.

[74] Watanabe, T.; Ae, T.; Nakamura, A. On the NP-Hardness of edge-deletion and contraction problems. *Discrete Applied Math. 6* (1983), 63–78.

[75] Wessel, W. The non-biplanar character of the graph k_9. In *Algebra und Graphentheorie* (Beitr. Jahrestag, 1986), Siebenlehn GDR, 1985, Algebra Grenzgeb, pp. 123–126.

[76] White, A. T.; Beineke, L. W. *Selected Topics in Graph Theory*. Academic Press, New York, 1978, ch. Topological Graph Theory, pp. 15–49.

[77] Williamson, S. G. Embedding graphs in the plane - algorithmic aspects. In *Combinatorial Mathematics, Optimal Designs and their Applications* (1, 1980), J. Srivastava, Ed., vol. 6, Annals of Discrete Mathematics, North-Holland, pp. 349–384.

[78] Woodall, D. R. Cyclic-order graphs and Zarankiewics's crossing number conjecture. *Journal of Graph Theory 17* (1993), 657–671.

[79] Xavier, E. F. Invariantes de planaridade. Dissertação de Mestrado, Universidade Estadual de Campinas, Instituto de Computação, Dezembro 1999.

[80] Yannakakis, M. Node- and edge-deletion NP-Complete problems. In *10th Annual ACM Symposium on Theory of Computing, STOC´78* (1978), pp. 253–264.

[81] Young, S. M. Implementation of PQ-tree algorithms. Dissertação de Mestrado, University of Washington, Computer Science Department, February 1977.