GERAÇÃO DE BICLIQUES DE UM GRAFO

Vânia Maria Félix Dias

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

Aprovada por:

Prof. Jayme Luiz Szwarcfiter, Ph.D.

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

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Ricardo Cordeiro Corrêa, Ph.D.

Prof. Edson Norberto Cáceres, D.Sc.

Prof. Ruy Luiz Milidiú, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
OUTUBRO DE 2004

DIAS, VÂNIA MARIA FÉLIX

Geração de bicliques de um grafo [Rio de Janeiro] 2004

X, 85 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2004)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 – Bicliques

2 – Complexidade de atraso polinomial

I. COPPE/UFRJ II. Título (série)

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

GERAÇÃO DE BICLIQUES DE UM GRAFO

Vânia Maria Félix Dias

Outubro/2004

Orientadores: Jayme Luiz Szwarcfiter

Celina Miraglia Herrera de Figueiredo

Programa: Engenharia de Sistemas e Computação

Este trabalho consiste de um estudo sobre a geração de bicliques de um grafo. Um conjunto bipartido completo B é um subconjunto de vértices que admite uma bipartição $B = X \cup Y$, tal que X e Y são ambos conjuntos independentes, e todo vértice de X é adjacente a todo vértice de Y. Se $X, Y \neq \emptyset$, então dizemos que B é próprio. Uma biclique é um conjunto bipartido completo próprio maximal de um grafo. Se X ou Y não são conjuntos independentes de G, dizemos que B é uma biclique não induzida. Demonstramos que é NP-completo decidir se um subconjunto de vértices de um grafo é parte de uma biclique. Demonstramos também que não existe algoritmo com atraso polinomial para a geração de todas as bicliques em ordem lexicográfica reversa, a menos que P = NP. Por outro lado, descrevemos diferentes algoritmos, todos com atraso polinomial, para a geração das bicliques de um grafo. Apresentamos um algoritmo que gera todas as bicliques em ordem lexicográfica. Também descrevemos um algoritmo que gera todas as bicliques não induzidas de um grafo. Além disso, propomos algoritmos eficientes para a geração de bicliques de classes especiais de grafos.

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

GENERATION OF BICLIQUES OF A GRAPH

Vânia Maria Félix Dias

October/2004

Advisors: Jayme Luiz Szwarcfiter

Celina Miraglia Herrera de Figueiredo

Department : Computing and Systems Engineering

This work describes a study on the generation of bicliques of a graph. A complete bipartite set B is a subset of vertices admitting a bipartition $B = X \cup Y$, such that both X and Y are independent sets, and all vertices of X are adjacent to those of Y. If both $X, Y \neq \emptyset$, then B is called proper. A biclique is a maximal proper complete bipartite set of a graph. When the requirement that X and Y are independent sets of G is dropped, we have a non-induced biclique. We show that it is NP-complete to test whether a subset of the vertices of a graph is part of a biclique. We also show that there is no polynomial-time delay algorithm for generating all bicliques in reverse lexicographic order, unless P = NP. On the other hand, we describe different polynomial-time delay algorithms for the generation of bicliques of a graph. We present an algorithm that generates all bicliques of a graph in lexicographic order. We also describe an algorithm that generates all non-induced bicliques of a graph. In addition, we propose specialized efficient algorithms for generating the bicliques of special classes of graphs.

Agradecimentos

A realização de um trabalho como este não seria possível sem a preciosa orientação do professor Jayme Szwarcfiter. Agradeço-o pela sábia e decisiva orientação em todas as etapas cruciais desta tese e pela generosa troca de idéias.

À professora Celina Figueiredo, também minha orientadora, que colocou-se sempre cuidadosa, interessada e disponível a ajudar com seriedade e competência às revisões desta tese.

Enfim, agradeço a todos que participaram direta ou indiretamente à realização deste trabalho.

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação

A geração de objetos de uma coleção, que satisfazem determinadas propriedades, é um problema bastante estudado na literatura de Algoritmos, Combinatória e Teoria dos Grafos. Entre outros, gerar todas as cliques (respectivamente conjuntos independentes) de um grafo é um problema que tem sido alvo de atenção através dos anos [22, 25, 42, 5, 28, 40, 41, 8]. Tsukiyama et al. [42], descreveram um algoritmo eficiente para gerar todas as cliques de um grafo com atraso polinomial e usando um espaço de memória também polinomial. Johnson, Yannakakis e Papadimitriou [22] demonstraram que não existe algoritmo com atraso polinomial para gerar todas as cliques de um grafo em ordem lexicográfica reversa a menos que P=NP. No entanto, os autores descreveram um algoritmo que gera todas as cliques em ordem lexicográfica, com tempo polinomial entre a enumeração

de duas cliques consecutivas na saída do algoritmo.

Conhecer as bicliques de um grafo é útil na solução de problemas em áreas como linguagens formais e autômatos [17], ordens parciais [26], inteligência artificial [44] e redes de computadores [38]. Além dessas, uma área que naturalmente compreende bastante aplicações é a própria teoria dos grafos.

Bicliques são empregadas na caracterização de certas classes de grafos, como por exemplo, grafos bipartidos cordais [19]. Além disso, a enumeração das bicliques de um dado grafo constitui um procedimento auxiliar no algoritmo de reconhecimento do mesmo. Nesse caso, estão incluídos grafos de caminho e grafos subjacentes a digrafos linha [37].

Nesta tese de doutorado tratamos da geração de bicliques de um grafo e da complexidade computacional deste problema combinatório. Como contribuição original, esta tese propõe algoritmos eficientes e provas de NP-completude. Apresentamos algoritmos eficientes para os seguintes problemas: (1) geração de bicliques induzidas de um grafo em ordem lexicográfica; (2) geração de bicliques não induzidas de um grafo; (3) geração de bicliques de grafos bipartidos convexos e bipartidos biconvexos. Além disso, demonstramos a NP-completude dos problemas de decisão

PARTE DE BICLIQUE e MAIOR BICLIQUE LEXICOGRÁFICA. Reportamos que não há menção na literatura à existência de algoritmos eficientes para a geração de bicliques induzidas em grafos gerais, nem para as classes especiais aqui abordadas. Por outro lado, o algoritmo proposto para a geração de bicliques não induzidas em grafos gerais possui complexidade assintoticamente menor do que um algoritmo recentemente reportado [1].

Durante o curso de doutorado produzimos os seguintes trabalhos: *On the generation of bicliques of a graph* [11]; *Generating bicliques of a graph in lexicographic order* [9]; *Generating all non-induced bicliques of a graph* [10]; *The stable marriage problem with restricted pairs* [13]; *Stable matchings with restricted pairs* [12]; *Casamentos estáveis com casais forçados e casais proibidos* [14]; *Grafos clique ponderados* [15].

1.2 Organização desta tese

Na seção 1.3 daremos as definições básicas, além de conceitos e notações, que serão utilizados no decorrer do texto. Na seção 1.4 resumimos alguns dos principais resultados conhecidos sobre bicliques em grafos.

Aspectos de complexidade referentes à geração de objetos de uma coleção são abordados no capítulo 2. Na seção 2.2, examinamos alguns critérios

utilizados para a análise de complexidade de algoritmos para os quais o tamanho da saída é possivelmente exponencial no tamanho da entrada. Além disso, consideramos alguns pré-requisitos que podemos fazer em relação à ordenação dos objetos na saída de um algoritmo. Alguns algoritmos existentes para a geração de bicliques de um grafo são brevemente descritos na seção 2.3.

No capítulo 3, consideramos dois problemas de decisão cujos resultados implicam em restrições aos algoritmos de enumeração de bicliques. Na seção 3.2, demonstramos que o problema PARTE DE BICLIQUE é NP-completo. Na seção 3.3, demonstramos a NP-completude do problema MAIOR BICLIQUE LEXICOGRÁFICA.

A geração das bicliques induzidas de um grafo qualquer é o assunto do capítulo 4. Na seção 4.2, descrevemos um algoritmo polinomial para gerar a menor biclique lexicográfica contendo um dado conjunto bipartido completo. Na seção 4.3, descrevemos um algoritmo que gera as bicliques de um grafo em ordem lexicográfica, com tempo polinomial entre a enumeração de duas bicliques consecutivas. O espaço requerido pelo algoritmo, contudo, pode ser exponencial.

O capítulo 5 trata da geração de bicliques não induzidas de um grafo. Na seção 5.2, mostramos como um algoritmo para gerar as bicliques (induzidas)

de um grafo bipartido pode ser usado para gerar todas as bicliques não induzidas de um grafo qualquer. A seção 5.3 apresenta algumas propriedades de bicliques em grafos bipartidos. Na seção 5.4, descrevemos um algoritmo que lista as bicliques de um grafo bipartido, que requer tempo polinomial entre a enumeração de duas bicliques consecutivas. O algoritmo descrito requer espaço de memória polinomial.

No capítulo 6, apresentamos algoritmos para a geração de bicliques aplicados a certas classes de grafos. Na seção 6.1, consideramos características particulares de grafos bipartidos convexos, que possibilitam alterações no algoritmo descrito no capítulo 5, de modo a melhorar a eficiência do mesmo, para a classe dos bipartidos convexos. Analogamente, na seção 6.2 consideramos alterações do algoritmo aplicado a grafos bipartidos biconvexos.

Por último, no capítulo 7, fazemos um estudo comparativo dos algoritmos para geração de bicliques, com o objetivo de avaliar os desempenhos dos algoritmos apresentados. Além disso, apresentamos alguns problemas a serem investigados no futuro.

1.3 Definições básicas

Um *grafo* é um par ordenado $G = (V, E)$ onde V é um conjunto finito não vazio de *vértices*, e E é um conjunto de pares não ordenados de vértices distintos, denominados *arestas*. Denotamos por $|V| = n$ e $|E| = m$. Quando conveniente, e por simplicidade de notação, consideraremos $V = \{1, \ldots, n\}$. Dada uma aresta $e \in E$, onde $e = \{v_i, v_j\}$, dizemos que o vértice v_i é *adjacente* ao vértice v_j, que v_i e v_j são *vizinhos* em G, e que v_i e v_j são *incidentes* a e. Denotaremos por N_i os vizinhos de v_i e por $\overline{N}_i = (V \setminus N_i) \setminus \{v_i\}$.

Dado um conjunto $X \subseteq V$, definimos por *foco* de X o conjunto de vértices adjacentes simultaneamente a todos os vértices de X, isto é, $F(X) = \cap_{i \in X} N_i$. Se os vértices de X são todos mutuamente adjacentes, então X é um *conjunto completo* de G. No grafo da Figura 1.1, o foco de $\{1, 4, 5\}$ é exatamente o vértice 2, pois o vértice 3 não é adjacente ao vértice 5, e o vértice 6 não é adjacente ao vértice 1.

Uma *clique* C é um subconjunto de V tal que C é um conjunto completo e é maximal em relação a essa propriedade, isto é, não existe completo $X \neq C$ tal que C está contido em X.

Um subconjunto U de V é um *conjunto independente* se não existe aresta

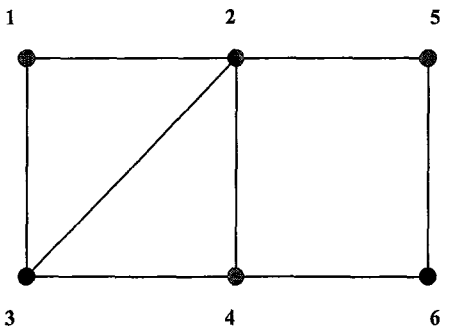


Figura 1.1: Exemplo de Foco

entre quaisquer dois vértices de U.

Um grafo $G = (V, E)$ é *bipartido* se V pode ser particionado em dois conjuntos U e W tal que $U \cup W = V$, $U \cap W = \emptyset$ e, além disso, ambos são conjuntos independentes. Um grafo $G = (U \cup W, E)$ é *bipartido completo* se todo vértice de U é adjacente a todo vértice de W. Denotamos por $K_{r,r}$ um grafo bipartido completo tal que $|U| = |W| = r$.

Para um dado conjunto $H \subseteq V$ o *subgrafo induzido* de G por H, denotado por $G[H]$, tem como conjunto de vértices H, e dois vértices são adjacentes em $G[H]$ se e somente se o são em G.

Seja $G = (V, E)$ um grafo qualquer e $X, Y \subseteq V$ tais que $X \cap Y = \emptyset$. Dizemos que $B = X \cup Y$ é um *conjunto bipartido completo* de G, quando ambos X e Y são independentes e todo vértice de X é adjacente a todo vértice de Y. Isto é, B induz um subgrafo bipartido completo em G. Dizemos que X e Y são *partes* do conjunto bipartido completo B. Se $X, Y \neq \emptyset$, então B é um *conjunto bipartido completo próprio*, i.e., B induz um subgrafo bipartido completo contendo pelo menos uma aresta de G. Caso contrário, B é chamado *conjunto bipartido completo degenerado*. No grafo da Figura 1.2, $B = \{1, 6\}$ é um conjunto bipartido completo degenerado.

Por outro lado, dizemos que $B = X \cup Y$ é um *conjunto bipartido completo não induzido* de G, se $X \cap Y \neq \emptyset$, e todo vértice de X é adjacente a todo vértice de Y.

Dizemos que B é uma *biclique induzida*, ou simplesmente uma *biclique*, de G, se B é um conjunto bipartido completo (induzido) próprio de G e é maximal em relação a esta propriedade.

Se um conjunto bipartido completo não induzido $B = X \cup Y$ é próprio e maximal, então dizemos que B é uma *biclique não induzida* de G. Logo, toda biclique (induzida) é biclique não induzida. Observe que se G é um grafo bipartido, então toda biclique não induzida é também biclique (induzida) de

G. Denotamos por $|\mathcal{B}|$ o número de bicliques de G.

A Figura 1.2 mostra uma biclique e uma biclique não induzida. Note que $B = \{1,4,5\} \cup \{2\}$ é um conjunto bipartido completo próprio maximal do grafo G, isto é, uma biclique. Por outro lado, $B = \{1,4,5\} \cup \{2\}$ é um conjunto bipartido completo próprio contido em $B' = \{1,3,4,5\} \cup \{2\}$, uma biclique não induzida de G.

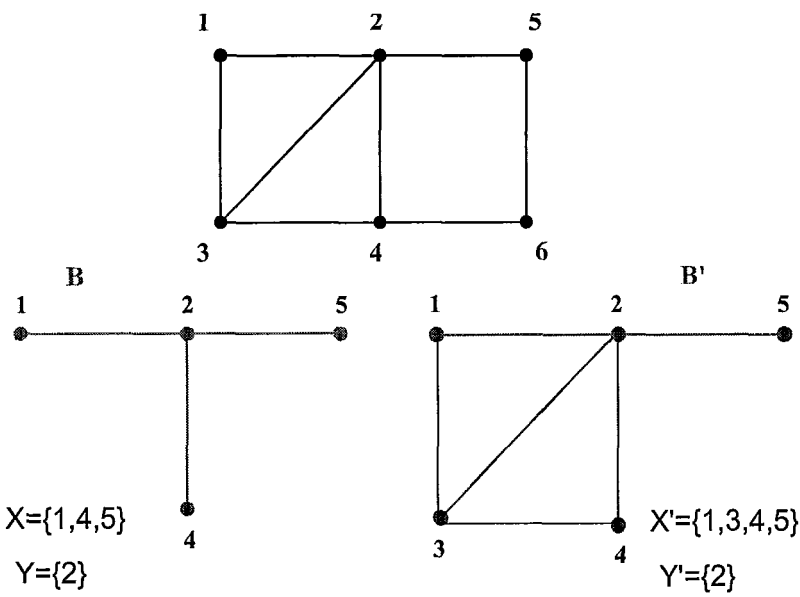


Figura 1.2: Biclique B induzida e biclique B' não induzida

Uma *cobertura por bicliques* de um grafo $G = (V, E)$ é um conjunto de bicliques de G tal que toda aresta de E pertence a pelo menos uma de suas bicliques. A *cardinalidade* da cobertura é o número de bicliques

da cobertura. A cobertura será *mínima* se não houver nenhuma outra de cardinalidade menor.

1.4 Resultados de problemas sobre bicliques

Uma das primeiras referências sobre bicliques encontra-se em *Computers and Intractability*, de Garey e Johnson [18]. Lá são abordados os seguintes problemas de decisão, referentes a bicliques.

BICLIQUE BALANCEADA

Entrada: Grafo Bipartido $G = (V, E)$, inteiro positivo $k \leq |V|$.

Pergunta: Existe biclique $B = X \cup Y$ de G tal que $|X| = |Y| = k$?

BICLIQUE MÁXIMA

Entrada: Grafo $G = (V, E)$, inteiro positivo $k \leq |V|$.

Pergunta: Existe biclique $B = X \cup Y$ de G tal que $|X| + |Y| \geq k$?

Os autores demonstraram que o problema BICLIQUE BALANCEADA é NP-completo, através da seguinte redução do problema CLIQUE.

CLIQUE

Entrada: Grafo $G = (V, E)$, inteiro positivo $k \leq |V|$.

Pergunta: Existe $C \subseteq V$ tal que C é uma clique de G e $|C| \geq k$?

Dado um grafo $G = (V, E)$ qualquer, constrói-se o seguinte grafo bipartido $G' = (U \cup W, E')$. A cada vértice v_i de V corresponde um vértice u_i em U e outro w_i em W. Um vértice u_i de U é adjacente ao vértice w_j de W se a aresta $\{v_i, v_j\}$ pertence a E ou se $i = j$. É fácil ver que existe uma correspondência um para um entre as cliques de G e as bicliques balanceadas de G'.

Yannakakis [45] demonstrou que o problema BICLIQUE MÁXIMA é NP-completo para grafos gerais. No entanto, este problema pode ser solucionado em tempo polinomial se o grafo dado é bipartido [7], como veremos a seguir.

Uma terceira variação do problema considerado, abaixo definida, também foi demonstrada ser NP-completo mais recentemente por Peeters [36]. A prova usa uma redução do problema CLIQUE para BICLIQUE MÁXIMA

EM ARESTAS.

BICLIQUE MÁXIMA EM ARESTAS.

Entrada: Grafo Bipartido $G = (V, E)$, inteiro positivo $k \leq |V|$.

Pergunta: Existe biclique $B = X \cup Y$ de G tal que $|X|.|Y| \geq k$?

Dawande et al. [7] consideraram os problemas de otimização correspondentes aos problemas de decisão anteriormente descritos. No caso do problema BICLIQUE MÁXIMA, além do grafo bipartido $G = (V, E)$, acrescenta-se à entrada do problema um peso p_i a cada vértice $v_i \in V$. Sejam V_1 e V_2 as partições do grafo bipartido dado. O problema de encontrar uma biclique de peso máximo em um grafo bipartido pode ser formulado através da seguinte programação inteira:

Maximizar $\sum p_i x_i + \sum p_j x_j$

Sujeito a $\quad x_i + x_j \leq 1,\ i \in V_1,\ j \in V_2,\ (i,j) \notin E$ (i)

$x_i \in \{0,1\}$, para todo $i \in V_1 \cup V_2$ (ii)

Relaxando a restrição (ii) para $0 \leq x_i \leq 1$ para todo $i \in V_1 \cup V_2$, obtém-se um problema de programação linear. A matriz definida pelo conjunto de

restrições em (i) corresponde à matriz de incidência do complemento de G, que é totalmente unimodular, e portanto a solução do problema correspondente será inteira [33]. Logo, o problema de encontrar a biclique de peso máximo pode ser solucionado em tempo polinomial. Ainda em [7], os autores demonstram que encontrar a biclique de peso máximo em arestas é NP-difícil. No entanto, a redução utilizada não implica na NP-completude do problema BICLIQUE MÁXIMA EM ARESTAS.

Em [21], Hochbaum apresenta um algoritmo de aproximação para o problema BICLIQUE MÁXIMA EM ARESTAS. Limites superiores para o número de arestas de uma biclique num grafo bipartido foram estabelecidos independentemente em [20] e [21].

Resultados de limites superiores para o número de bicliques de um grafo foram estabelecidos por Prisner [38]. No caso geral, o número de bicliques de um grafo é no máximo $n^{5/2}(1.618034)^n + O(1)$. Para um grafo G da classe dos grafos bipartidos é demonstrado que o número máximo de bicliques de G é $2^{n/2}$. O caso extremo é a classe $CP(r)$. Um grafo G é dito $CP(r)$ se G é resultante da remoção de um emparelhamento perfeito de um grafo $K_{r,r}$. Sejam U e W as partições de um grafo $CP(r)$, $|V| = 2r = n$. Cada subconjunto I de $\{1, ..., r\}$ define uma biclique do tipo $\{u_i : i \in I\} \cup \{w_j : j \in \{1, ..., r\} \setminus I\}$ em $CP(r)$. Logo, existem $2^{n/2}$

bicliques desse tipo. Ainda nesse trabalho são descritas duas famílias de grafos fechadas por subgrafos proibidos que apresentam limites polinomiais, sendo que uma delas pertence à classe dos grafos gerais e a outra à classe dos bipartidos.

Outro problema bastante investigado, que compreende um grande número de aplicações, é o de encontrar uma cobertura por bicliques que seja mínima. Tuza [43] apresenta limites para o número de bicliques necessárias para cobrir as arestas de um grafo qualquer. O problema de decisão correspondente foi demonstrado ser NP-difícil para grafos bipartidos [34] e para grafos bipartidos cordais [30]. No entanto, o problema se torna polinomial para as classes de grafos bipartidos sem dominó [2] (isto é, sem subgrafo induzido isomorfo ao grafo da Figura 1.3) e grafos bipartidos convexos [27].

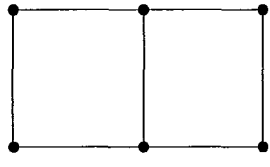


Figura 1.3: Grafo dominó

O estudo da geração de bicliques de um grafo é ainda um pouco recente. Alexe et al. [1] mostraram como um algoritmo para gerar todos os conjuntos

independentes maximais de um grafo pode ser usado para gerar todas as bicliques não induzidas de um grafo. Além disso, descreveram um algoritmo específico para enumeração das bicliques não induzidas de um grafo, com resultados de tempo empiricamente melhores do que o obtido pela transformação descrita. Contudo, este último não representou uma melhoria da complexidade analítica. Além disso, o mesmo requer espaço exponencial.

Kloks e Kratsh [24] descreveram um algoritmo para listar todas as bicliques de um grafo bipartido cordal em tempo polinomial, enquanto que Eppstein [16] descreveu um algoritmo para gerar as bicliques não induzidas de grafos de arboricidade limitada. Em ambos os casos, os grafos pertencem a classes com número de bicliques polinomial em n, o número de vértices do grafo de entrada.

Em [11], demonstramos a NP-completude dos problemas PARTE DE BICLIQUE e MAIOR BICLIQUE LEXICOGRÁFICA. Além disso, descrevemos um algoritmo polinomial para gerar a menor biclique lexicográfica de um grafo. Apresentamos em [9] um algoritmo para gerar todas as bicliques induzidas em ordem lexicográfica, com atraso polinomial entre a enumeração de duas bicliques consecutivas na saída. Decrevemos em [10] algoritmos para gerar as bicliques não induzidas de um grafo com atraso polinomial, e que

requer espaço de memória polinomial. Ainda neste último trabalho, descrevemos algoritmos eficientes para gerar as bicliques de grafos bipartidos convexos e biconvexos.

Capítulo 2

Geração de objetos

2.1 Introdução

A geração de objetos de uma coleção é um problema bastante investigado em Teoria dos Grafos e Combinatória. Entre tais problemas, a geração de conjuntos independentes (respectivamente cliques) de um grafo já foi bastante estudado [22, 25, 35, 42, 5, 28, 40, 41, 8].

Os algoritmos propostos para a geração de objetos costumam empregar diferentes noções de complexidade. Como o número de objetos a serem gerados pode ser exponencial no tamanho da entrada, é preciso considerar noções de complexidade mais flexíveis nestes casos.

Na seção 2.2, examinamos alguns critérios conhecidos para a análise de

complexidade de algoritmos para os quais o tamanho da saída é possivelmente exponencial no tamanho da entrada. Consideramos ainda alguns pré-requisitos que podemos fazer em relação à ordenação dos objetos na saída de um algoritmo. Na seção 2.3 citamos alguns algoritmos existentes para a geração de bicliques de um grafo.

2.2 Complexidade

Seja A um algoritmo para a geração de objetos de uma coleção. Denotamos por n o tamanho da entrada de A e por k o número de objetos a serem gerados. A complexidade considerada consiste no número de passos necessários para disponibilizar cada objeto na memória. A seguir descrevemos alguns critérios de análise de complexidade de algoritmos deste tipo.

2.2.1 Tipos de complexidade

1. Tempo Polinomial Total

   Dizemos que A tem complexidade de tempo polinomial total quando o tempo requerido para gerar os k objetos é polinomial em n e k.

2. Tempo polinomial incremental

   Seja $k' \leq k$ o número de objetos de um subconjunto do total de

objetos a serem gerados por A. Se A tem complexidade de tempo polinomial incremental então, a partir da entrada e de k' objetos, A enumera um novo objeto ou determina que não existe outro objeto a ser gerado em tempo polinomial no tamanho de n e k'. Certamente um algoritmo que satisfaça tal critério de complexidade gera todos os objetos em tempo polinomial em n e k. Logo, se A tem tempo polinomial incremental então A tem tempo polinomial total.

3. Tempo de atraso polinomial

   Um algoritmo A satisfaz este critério de complexidade quando o tempo necessário entre a enumeração de dois objetos consecutivos na saída de A, isto é o *atraso*, é polinomial em n. Além disso, o tempo necessário para gerar o primeiro objeto e o tempo decorrido após a geração do último são também polinomiais em n.

O algoritmo de geração de conjuntos independentes maximais apresentado em [35] é um exemplo de algoritmo com tempo polinomial total. Este algoritmo dispende tempo $O(n^2k)$ sem gerar nenhum conjunto independente e, então, gera todos estes conjuntos em tempo $O(k)$. O espaço de memória requerido pelo algoritmo é $O(m + nk)$. Um algoritmo mais eficiente de geração de conjuntos independentes maximais, este apresentado em [42], tem complexidade de tempo de atraso polinomial $O(n^3)$ e requer espaço linear em n.

Na seção 2.3 descrevemos o algoritmo de geração de bicliques não induzidas de um grafo qualquer, proposto em [1], que requer tempo polinomial incremental.

Em relação à complexidade de espaço, em geral os algoritmos que requerem tamanho polinomial de memória têm complexidade de tempo de atraso polinomial.

2.2.2 Algoritmos eficientes

Os algoritmos considerados mais eficientes neste contexto são aqueles que apresentam complexidade de atraso polinomial. Estes por sua vez podem ser divididos segundo os seguintes critérios. Denotamos por $t(A)$ o tempo total dispendido pelo algoritmo A.

1. CAT (*Constant average time*)

   Um algoritmo é classificado como CAT se $t(A)/k$ é constante.

2. Sem-laço (*Loop-free*)

   Um algoritmo A satisfaz este critério de complexidade quando o tempo

dispendido entre a enumeração de dois objetos consecutivos na saída de A é constante.

Como exemplo de algoritmo *loop-free* citamos o algoritmo para listar as extensões lineares de um conjunto parcialmente ordenado, descrito em [6]. Outros exemplos deste tipo de algoritmos incluem algoritmos para a geração de permutações, combinações, etc.

2.2.3 Ordenação

Uma outra consideração que se pode fazer é em relação à ordenação dos objetos na saída do algoritmo. Citamos dois tipos de ordenação muitas vezes utilizadas.

1. *Gray Code*

   Este termo é usado para métodos de geração de objetos nos quais dois objetos sucessivos na enumeração diferem "localmente" de modo pré-especificado.

2. Ordenação Lexicográfica

   Sejam S e T dois subconjuntos de um conjunto ordenado. Diz-se que S é *lexicograficamente menor* que T, quando (i) o menor elemento em

que S e T diferem pertence a S, ou (ii) os $|S|$ menores elementos de T coincidem com S, e $|S| < |T|$. Nesse caso, o interesse é gerar os objetos em ordem lexicográfica (crescente), ou em ordem lexicográfica reversa, isto é, do lexicograficamente maior para o lexicograficamente menor.

Como exemplos de *Gray codes* podemos citar: (1) listar todas as permutações de $1, \ldots, n$ tal que permutações consecutivas na enumeração diferem em apenas duas posições adjacentes; (2) listar todas as árvores binárias tal que duas árvores consecutivas diferem unicamente pela rotação de um nó; (3) listar todas as árvores geradoras de um grafo tal que árvores consecutivas diferem de uma única aresta. Em [39], Savage fornece um estudo abrangente e uma bibliografia vasta sobre o assunto, incluindo os exemplos acima citados. O algoritmo *loop-free* descrito em [6], que lista as extensões lineares de um conjunto parcialmente ordenado, usa o método Gray code. As extensões lineares são listadas de modo que duas extensões consecutivas na saída diferem apenas pela transposição de dois elementos adjacentes.

O algoritmo de Johnson, Yannakakis e Papadimitriou [22] gera os conjuntos independentes de um grafo em ordem lexicográfica com atraso $O(n^3)$. Os autores demonstram ainda que não existe algoritmo com tempo de atraso polinomial para gerar os conjuntos independentes em ordem lexicográfica reversa a menos que P=NP. Isto é, mesmo utilizando critérios de complexidade mais flexíveis, podem não existir algoritmos para a solução

deste tipo de problema, quando se considera algum critério de ordenação para a geração dos objetos.

2.3 Geração de bicliques

O algoritmo de geração de bicliques proposto por Alexe et al. [1] faz uso de duas definições que descrevemos a seguir. Sejam $B = X \cup Y$ e $B' = X' \cup Y'$ dois conjuntos bipartidos completos, não induzidos, de um dado grafo G. Então B *absorve* B' se $X' \subseteq X$ e $Y' \subseteq Y$, ou se $X' \subseteq Y$ e $Y' \subseteq X$. Se $Y \cap Y' \neq \emptyset$, então o *consenso* de B e B' é o conjunto bipartido completo definido por $(X \cup X') \cup (Y \cap Y')$. Observe que pode haver mais de um consenso entre B e B', definidos por quaisquer duas partes que se interseptem.

Seja $G = (V, E)$ um grafo com $|V| = n$. O algoritmo tem como entrada, além do grafo G, uma lista C de k' bicliques, $k' \leq n$, que forma uma cobertura das arestas de G. Enquanto houverem duas bicliques B e B' distintas de C tal que existe um consenso B'' entre estas que não é absorvido por nenhuma biclique de C, o algoritmo estende B'' a uma biclique de G e inclui a mesma em C.

O algoritmo tem tempo polinomial incremental em $|C| = O(k)$ e n. O tempo total do algoritmo é $O(nk(m + nlog_2k)) = O(n^3k)$. O espaço requerido pelo algoritmo é $O(k)$.

Alexe et al. [1] observaram que um algoritmo para gerar todos os conjuntos independentes maximais pode ser usado para a geração das bicliques não induzidas de um grafo $G = (V, E)$ do seguinte modo. Seja G' um grafo consistindo de duas cliques V_1 e V_2, cada uma com $|V| = n$ vértices, onde cada $v_i \in V$ aparece como $v_i' \in V_1$, e como $v_i'' \in V_2$. Dois vértices v_i' e v_j'' são adjacentes em G' se e somente se $(v_i, v_j) \in E$. Existe uma correspondência 2-1 entre as cliques (completos maximais) de G' e as bicliques não induzidas de G, com exceção das duas cliques V_1 e V_2. Consequentemente, as bicliques não induzidas de um grafo podem ser geradas com atraso $O(n^3)$.

Uma estratégia similar pode ser usada para a geração de todos os conjuntos bipartidos completos (próprios ou degenerados) maximais de um grafo $G = (V, E)$. Seja G' um grafo consistindo de dois complementos do grafo G, V_1 e V_2, cada um com $|V| = n$ vértices, onde cada $v_i \in V$ aparece como $v_i' \in V_1$, e como $v_i'' \in V_2$. Dois vértices v_i' e v_j'' são adjacentes em G' se e somente se $(v_i, v_j) \in E$. Existe uma correspondência 2-1 entre as cliques de G' e os conjuntos bipartidos completos maximais de G. Seja C uma clique de G'. É fácil ver que C está inteiramente contida em V_1 (ou V_2) se e somente

se o conjunto bipartido completo maximal correspondente de G é degenerado.

Claramente, o método descrito acima pode ser usado para gerar todos os conjuntos bipartidos completos próprios maximais, isto é, as bicliques de G. No entanto, o número de conjuntos bipartidos completos degenerados maximais de G pode ser exponencial em n, mesmo que o número de bicliques em G seja polinomial em n. Consequentemente, o algoritmo descrito no parágrafo anterior não tem complexidade de tempo de atraso polinomial.

A Figura 2.1 representa uma classe de grafos com $n/2$ bicliques (o conjunto de bicliques coincide com o conjunto de arestas) e 2^n conjuntos independentes maximais, isto é, conjuntos bipartidos completos degenerados maximais.

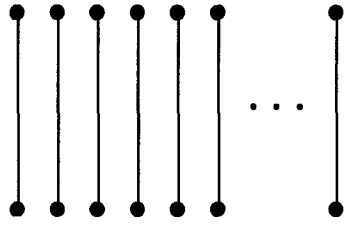


Figura 2.1: Classe de grafos com número de conjuntos bipartidos completos degenerados maximais exponencial no número de bicliques

Quando o número de objetos a serem gerados é polinomial em n então um algoritmo que satisfaça qualquer um dos critérios anteriormente descritos requer tempo polinomial em n. Exemplos destes casos no problema de geração de bicliques podem ser vistos em [16, 24]. Eppstein [16] descreve um algoritmo para gerar as bicliques não induzidas de grafos de arboricidade limitada, enquanto que em [24] é apresentado um algoritmo que gera as bicliques de um grafo bipartido cordal.

2.4 Conclusão

Noções de complexidade mais flexíveis de eficiência são necessárias quando o número de objetos a serem gerados por um algoritmo é exponencial no tamanho de sua entrada. No entanto, mesmo adotando esses critérios de complexidade nem sempre é possível a descrição de um algoritmo eficiente para solucionar estes problemas.

No capítulo seguinte demonstramos que não existe algoritmo para gerar as bicliques de um grafo em ordem lexicográfica reversa, com complexidade de atraso polinomial, a menos que P=NP. Por outro lado, no capítulo 4 descrevemos um algoritmo para gerar as bicliques de um grafo em ordem lexicográfica, com atraso $O(n^3)$. No capítulo 5 decrevemos um outro algoritmo que gera as bicliques não induzidas de um grafo com atraso $O(nm)$

e que requer espaço polinomial. No capítulo 6, tratamos de geração de bicliques de algumas classes de grafos com número polinomial de bicliques, especificamente, bipartidos convexos e biconvexos.

Capítulo 3

Problemas de bicliques NP-completos

3.1 Introdução

Problemas de decisão de bicliques têm se mostrado tão difíceis quanto os problemas de clique análogos. Por exemplo, problemas como BICLIQUE MÁXIMA, BICLIQUE MÁXIMA EM ARESTAS e BICLIQUE BALANCEADA foram demonstrados ser NP-completos para grafos gerais em [45], [36] e [18] respectivamente. Johnson, Papadimitriou e Yannakakis [22] demonstraram que o problema de encontrar a maior clique lexicográfica é NP-completo. Como veremos, o problema biclique correspondente é também NP-completo.

Neste capítulo consideramos dois problemas de decisão. Na seção 3.2, definimos e demonstramos que o problema PARTE DE BICLIQUE é

NP-completo. Na seção 3.3 demonstramos a NP-completude do problema MAIOR BICLIQUE LEXICOGRÁFICA. Nos dois casos, através de uma transformação polinomial do bem conhecido problema de SATISFABILIDADE. Ambos os problemas são relevantes quando consideramos algoritmos de geração de bicliques. Em particular, a NP-completude do problema MAIOR BICLIQUE LEXICOGRÁFICA tem como consequência a inexistência de um algoritmo para gerar as bicliques de um grafo em ordem lexicográfica reversa, com atraso polinomial, a menos que P=NP.

3.2 Parte de biclique

Dado um grafo $G = (V, E)$, dizemos que $S \subset V$ é *parte* de uma biclique $B = X \cup Y$ se $S = X$ ou $S = Y$. A seguir definimos os problemas de decisão SATISFABILIDADE e PARTE DE BICLIQUE.

SATISFABILIDADE

Entrada: Conjunto $A = \{a_1, ..., a_k\}$ de variáveis lógicas, coleção $C = \{C_1, ..., C_n\}$ de $n > 1$ cláusulas sobre A, onde cada cláusula é uma conjunção de literais nas variáveis de A.

Pergunta: Existe atribuição de valor às variáveis de A, tal que cada cláusula em C tem pelo menos um literal com valor verdadeiro?

PARTE DE BICLIQUE

Entrada: Grafo $G = (V, E)$, subconjunto $S \subset V$.

Pergunta: Existe biclique de G com parte S?

No primeiro problema, diz-se que existe uma *atribuição verdade* para C, ou ainda que C é *satisfatível*, quando a pergunta pode ser respondida afirmativamente. Observe que no problema da SATISFABILIDADE podemos assumir $n > 1$, caso contrário o problema é trivialmente polinomial.

Teorema 3.1 *Dado um grafo $G = (V, E)$ e um subconjunto $S \subset V$, é NP-Completo decidir se S é parte de uma biclique.*

Demonstração:

Inicialmente, provamos que o problema PARTE DE BICLIQUE pertence a NP. Como certificado, basta fornecer uma biclique da qual S é parte. Seja Y a outra parte da biclique dada. Claramente, pode-se verificar em tempo polinomial que $B = S \cup Y$ é um conjunto bipartido completo próprio de G e, além disso, que todo vértice $v \in V \setminus (S \cup Y)$ satisfaz uma das condições:

(i) se $\{v\} \cup Y$ é um conjunto independente, então v não é adjacente a pelo menos um vértice de S.

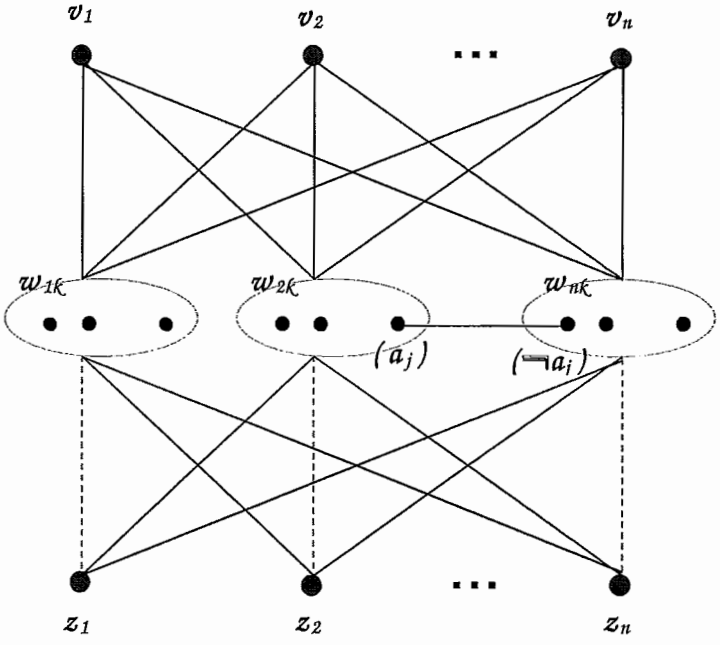


Figura 3.1: Grafo obtido na prova do teorema 3.1

(ii) se $\{v\} \cup S$ é um conjunto independente, então v não é adjacente a pelo menos um vértice de Y.

Mostraremos uma transformação do problema de SATISFABILIDADE para o problema PARTE DE BICLIQUE. Seja (A, C) uma entrada para SATISFABILIDADE, com cláusulas $C_1, C_2, ..., C_n$, onde l_{ij} é o j-ésimo literal da cláusula C_i. Lembramos que $n > 1$.

A seguir, definimos como obter o grafo $G = (V, E)$ e o conjunto $S \subset V$, isto é, a entrada particular para PARTE DE BICLIQUE, a partir da entrada de SATISFABILIDADE (ver figura 3.1).

O conjunto V de vértices de G é a união de três subconjuntos, $V = W \cup S \cup Z$, tal que:
(i) Para cada literal l_{ij} de C_i, $1 \leq i \leq n$, existe um vértice w_{ij} correspondente em W;
(ii) O conjunto S tem n vértices, denotados por $v_1, ..., v_n$;
(iii) O conjunto Z tem n vértices, denotados por $z_1, ..., z_n$.

O conjunto E de arestas de G é a união de três subconjuntos, $E = W' \cup S' \cup Z'$, tal que:

(i) $W' = \{(w_{ij}, w_{pq}) : \ell_{ij} = \neg\ell_{pq}, i \neq p\}$;

(ii) $S' = \{(v_i, w_{pq}) : v_i \in S, w_{pq} \in W\}$;

(iii) $Z' = \{(z_i, w_{pq}) : z_i \in Z, w_{pq} \in W, i \neq p\}$.

Claramente, essa construção pode ser realizada em tempo polinomial.

Suponha que (A, C) seja satisfatível. Seja T uma atribuição de valor às variáveis de A tal que cada cláusula de C tem pelo menos um literal verdadeiro. Vamos selecionar um conjunto independente $Y \subseteq W$ tal que $S \cup Y$ é uma biclique de G. Primeiro defina $Y' \subseteq W$ selecionando todos os vértices de W que correspondem a literais com valor verdadeiro em T. Por definição, Y' é um conjunto independente de G, contendo pelo menos um vértice w_{ij} para cada $i = 1, ..., n$. Agora, tome Y como sendo um conjunto independente maximal contendo Y'. Note que $Y \subseteq W$. Claramente, $S \cup Y$ induz um subgrafo bipartido completo em G. Para verificar que $S \cup Y$ é uma biclique, observe que todo vértice de $W \setminus Y$ corresponde a um literal cujo complementar pertence a Y e, como $n > 1$, cada z_i é adjacente a algum um vértice w_{kq} de Y, $i \neq k$. Logo, a partir da atribuição T, obtemos uma biclique em G com parte S.

Agora, seja $S \cup Y$ uma biclique de G. Como $Z \cup S$ é um conjunto independente em G, tem-se que $Y \subseteq W$. Pela maximalidade de $S \cup Y$, para

cada $z_i \in Z$ existe um vértice $w_{ij} \in Y$. Escolha então para cada cláusula C_i um vértice w_{ij} que pertence a Y. Cada vértice w_{ij} corresponde a um literal l_{ij}. Atribua valor *verdadeiro* a cada um desses literais. Como Y é um conjunto independente, cada vértice w_{ij} corresponde a um literal l_{ij} tal que se $l_{pq} = \neg l_{ij}$ e $l_{pq} \in C_p$, com $p \neq i$, então $w_{pq} \notin Y$. Portanto, a cada variável é atribuído um único valor. Logo, obtemos assim uma atribuição de valores que satisfaz C. ∎

O Teorema 3.1 diz que dado um conjunto independente S (maximal ou não) de um grafo G é aparentemente difícil decidir se G tem uma biclique com parte S, resultado este não muito intuitivo. O corolário a seguir torna este resultado ainda mais forte.

Corolário 3.2 *É NP-Completo decidir se um vértice v de G é parte de uma biclique, ou, equivalentemente, se existe $U \subseteq V \setminus \{v\}$ tal que (v, U) é uma biclique de G.*

Demonstração:

Tome $S = \{v\}$ na prova do Teorema 3.1.■

3.3 Maior biclique lexicográfica

Seja $G = (V, E)$ um grafo com $V = \{1, \ldots, n\}$. Considere duas bicliques $B = X \cup Y$ e $B' = X' \cup Y'$ de G. Seja $S = \{s_1, ..., s_t\}$ a ordenação crescente dos vértices de $X \cup Y$, analogamente S' em relação a $X' \cup Y'$. Seja i a primeira posição na qual s_i e s'_i são discordantes em S e S'. Então B é *lexicograficamente menor* que B', se $s_i < s'_i$. Caso contrário, B é *lexicograficamente maior* que B'.

A seguir, definimos o problema de decisão relacionado.

MAIOR BICLIQUE LEXICOGRÁFICA

Entrada: Grafo $G = (V, E)$, biclique B e uma ordem total nos vértices de V.

Pergunta: Existe biclique B' de G que seja lexicograficamente maior que B?

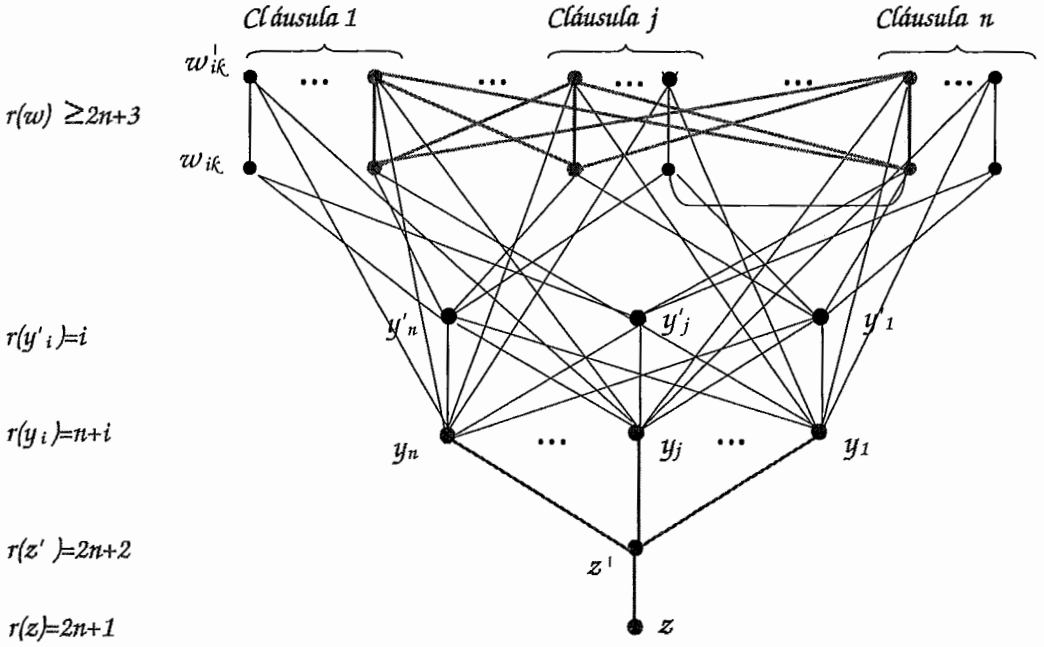


Figura 3.2: Grafo obtido na prova do teorema 3.3

Teorema 3.3 *Dado um grafo $G = (V, E)$, uma biclique B, e uma ordem sobre V, é Co-NP-completo decidir se B é a maior biclique lexicográfica de G.*

Demonstração:

Inicialmente, provamos que o problema pertence a Co-NP. Como certificado basta fornecer uma biclique B' que seja lexicograficamente maior que B, o que claramente pode ser verificado em tempo polinomial.

Para demonstrar a completude, fornecemos a seguinte transformação a partir do problema de SATISFABILIDADE. Seja (A, C) uma entrada para SATISFABILIDADE, com cláusulas $C_1, \ldots, C_n$, onde l_{ij} é o j-ésimo literal da cláusula C_i. Seja m o número total de ocorrências de literais que aparecem em $\bigcup_{i=1}^{n} C_i$. Construímos em tempo polinomial um grafo $G = (V, E)$, uma biclique B, e uma ordem sobre V, tal que C é satisfatível se e somente se B não é a maior biclique lexicográfica de G (Figura 3.2).

O conjunto de vértices V é a união de três subconjuntos, $V = Y \cup Z \cup W$, tal que:

(i) O conjunto Y tem $2n$ vértices: a cada cláusula C_i, $1 \leq i \leq n$, correspondem dois vértices y_i e y_i';

(ii) O conjunto Z tem 2 vértices: z e z';

(iii) O conjunto W tem $2m$ vértices: para cada literal l_{ij} de C_i existem dois vértices w_{ij} e w'_{ij} correspondentes em W.

O conjunto de arestas de G é definido por $E = Y^* \cup Z^* \cup W^*$, tal que:

(i) $Y^* = \{(y_i, y'_j) : i = 1, \ldots, n \text{ e } j = 1, \ldots, n\} \cup \{(y_i, w'_{pq}) : i = 1, \ldots, n \text{ e } p \neq i\} \cup \{(y'_i, w_{pq}) : i = 1, \ldots, n \text{ e } p \neq i\}$;

(ii) $Z^* = \{(z, z')\} \cup \{(z', y_i) : i = 1, \ldots, n\}$;

(iii) $W^* = \{(w_{ij}, w'_{ij}) : w_{ij}, w'_{ij} \in W\} \cup \{(w_{ij}, w'_{pq}), (w'_{ij}, w_{pq}) : i \neq p \text{ e } \ell_{ij} \neq \neg\ell_{pq}\} \cup \{(w_{ij}, w_{pq}) : i \neq p \text{ e } \ell_{ij} = \neg\ell_{pq}\}$.

Defina $B = \{z'\} \cup (\{z\} \cup \{y_1, y_2, ..., y_n\})$. Note que B é uma biclique, já que $N_z = \{z'\}$ e $N_{z'} = \{z\} \cup \{y_1, y_2, ..., y_n\}$.

Finalmente, terminamos a construção estabelecendo a seguinte ordenação total dos vértices de G. A cada vértice y'_i atribuímos rótulo i, e a cada y_i, $n + i$, para $i = 1, \ldots, n$; Aos vértices z e z', atribuímos $2n + 1$ e $2n + 2$, respectivamente; Cada vértice $w \in W$ recebe um rótulo $j \geq 2n + 3$.

À cada cláusula C_i corresponde uma aresta (y_i, y'_i). Observe que B é a única biclique que contém o vértice z. Toda biclique $B' \neq B$ que contém z' é tal que $y'_i \in B'$, para algum $i = 1, \ldots, n$.

A estrutura do grafo G obtido fornece a seguinte equivalência. *Existe uma biclique B' lexicograficamente maior que B em G se e somente se existe uma atribuição verdade que satisfaz C.*

Suponha que G contém uma biclique B' lexicograficamente maior que B. Nesse caso, $y_i' \notin B'$, para todo $i = 1, \ldots, n$, o que implica que B' contém w_{ij}, para cada $i = 1, \ldots, n$. Como este conjunto de vértices w_{ij} é um conjunto independente, os literais ℓ_{ij} correspondentes fornecem uma atribuição verdade que satisfaz C.

Por outro lado, a toda atribuição T que satisfaz C corresponde uma biclique B' de G inteiramente contida em W. Por construção, B' é lexicograficamente maior que B.■

3.4 Conclusão

Os algoritmos apresentados no capítulo a seguir contrastam bastante com os resultados aqui apresentados. Pelo Teorema 3.3, vimos que não existe algoritmo para encontrar a maior biclique lexicográfica de um dado

grafo a menos que P=NP.

No capítulo 4, descrevemos um algoritmo que dado um grafo G e um conjunto bipartido completo B' - próprio ou não - fornece em tempo polinomial a menor biclique lexicográfica de G contendo B'. Mostramos ainda como este algoritmo pode ser usado para encontrar a menor biclique lexicográfica de G. Além disso, descrevemos um algoritmo que enumera as bicliques de um grafo G em ordem lexicográfica com atraso polinomial entre as saídas de duas bicliques consecutivas.

Capítulo 4

Geração de bicliques

4.1 Introdução

Neste capítulo tratamos da geração das bicliques de um grafo qualquer. Na seção 4.2, descrevemos um algoritmo para gerar a menor biclique lexicográfica contendo um dado conjunto bipartido completo, em tempo $O(n^2)$. Apresentamos, na seção 4.3, um algoritmo que enumera as bicliques de G em ordem lexicográfica, que requer tempo polinomial entre a enumeração de duas bicliques consecutivas. A existência destes algoritmos, com suas respectivas complexidades, contrasta com os resultados de NP-completude na versão desses problemas nas quais se considera a ordem lexicográfica reversa, demonstrados no capítulo anterior.

O algoritmo que enumera as bicliques de G em ordem lexicográfica utiliza uma técnica inspirada naquela empregada no algoritmo apresentado

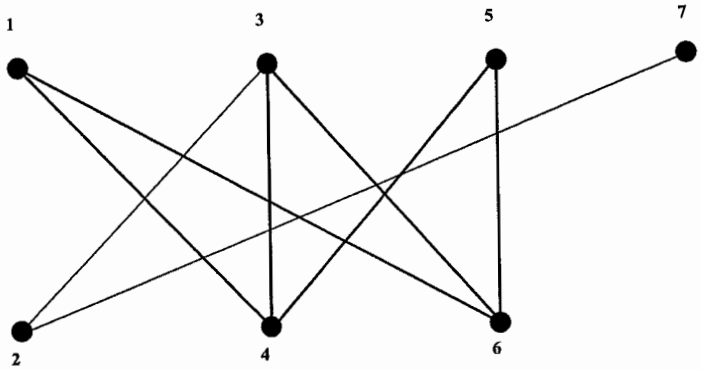
em [22], que gera os conjuntos independentes maximais de um grafo em ordem lexicográfica com complexidade de tempo de atraso polinomial.

4.2 Algoritmo para gerar a menor biclique lexicográfica

Seja $G = (V, E)$ um grafo, cujos vértices têm uma ordenação total em $V = \{1, \ldots, n\}$. Dado um conjunto bipartido completo $B = X \cup Y$, com $X \neq \emptyset$, o algoritmo MBLB retorna a menor biclique lexicográfica B' de G que contém B, ou $B' = \emptyset$, caso não exista biclique de G contendo B'.

Denotamos por $B_j = B \cap \{1, \ldots, j\}$, $X_j = X \cap \{1, \ldots, j\}$, e por $Y_j = Y \cap \{1, \ldots, j\}$. Representamos por B^* a menor biclique lexicográfica de G.

A primeira fase do algoritmo MBLB encontra o menor conjunto bipartido completo próprio B' contendo B, caso exista. Caso contrário, o foco de X é vazio, o algoritmo retorna $B' = \emptyset$. Na segunda fase, o algoritmo estende B' a um conjunto bipartido completo próprio maximal de G, isto é, uma biclique de G. Nas duas fases, os vértices são considerados em ordem crescente, o que garante que B' seja a menor biclique lexicográfica contendo B.



$B^{*}=\{1,3,5\}\cup\{4,6\}$ $B^{'}=\{2,4,6\}\cup\{3\}$ $B^{''}=\{2\}\cup\{3,7\}$

Figura 4.1: Grafo G

Algoritmo MBLB

Entrada: Grafo $G = (V, E)$, ordenação total de V,
conjunto bipartido completo $B = X \cup Y$

Saída: Menor biclique lexicográfica B' contendo B, caso exista;
$B' = \emptyset$, caso contrário

1. Se $Y = \emptyset$ então
   Se $F(X) = \emptyset$, então $B' = \emptyset$
   Senão $i \leftarrow 1$
   Repita $j \leftarrow i$-ésimo menor vértice de $V \setminus B$
   Se $\overline{N_j} \supseteq X$ e $F(X) \cap N_j \neq \emptyset$ então $X \leftarrow X \cup \{j\}$
   Se $N_j \supseteq X$ então $Y \leftarrow \{j\}$
   $i \leftarrow i + 1$
   Até que $Y \neq \emptyset$
2. $X' \leftarrow X$; $Y' \leftarrow Y$
   Para $k \leftarrow j$ até n e $k \in V \setminus (X \cup Y)$
   Se $\overline{N_k} \supseteq X'$ e $N_k \supseteq Y'$ então $X' \leftarrow X' \cup \{k\}$
   Se $\overline{N_k} \supseteq Y'$ e $N_k \supseteq X'$ então $Y' \leftarrow Y' \cup \{k\}$
   Retorne $B' = X' \cup Y'$

Figura 4.2: Algoritmo para fornecer a menor biclique lexicográfica

A fim de encontrar B^*, a menor biclique lexicográfica de G, basta aplicar o algoritmo MBLB com $B = X \cup Y$, $X = \{k\}$ e $Y = \emptyset$, onde k é o menor vértice não isolado de G.

O algoritmo MBLB é descrito na Figura 4.2. A Figura 4.1 mostra um grafo e suas bicliques em ordem lexicográfica e B^* em destaque. Considere uma execução deste algoritmo para o grafo da Figura 4.1. Seja $X = \{1, 2\}$ e $Y = \emptyset$. Nesta situação, o algoritmo retorna $B' = \emptyset$, pois o foco de $X = \{1, 2\}$

é vazio. Com $X = \{1\}$ e $Y = \emptyset$, o algoritmo retorna $\{1,3,5\} \cup \{4,6\} = B^*$.

A complexidade de tempo do algoritmo MBLB é $O(n^2)$. É fácil ver que qualquer operação descrita no algoritmo, tanto na primeira como na segunda fase, pode ser realizada em tempo $O(n)$ para cada vértice examinado. Como cada vértice é considerado no máximo uma vez, o algoritmo pode ser implementado em tempo $O(n^2)$.

4.3 Geração de bicliques em ordem lexicográfica

Seja $G = (V,E)$ um grafo, com $V = \{1,\ldots,n\}$. Sejam $B_j = B \cap \{1,\ldots,j\}$, $X_j = X \cap \{1,\ldots,j\}$, e $Y_j = Y \cap \{1,\ldots,j\}$. Seja B^* a menor biclique lexicográfica de G.

O Lema 4.1 caracteriza a menor biclique lexicográfica de um grafo através de uma condição de maximalidade. Esta condição é empregada no algoritmo apresentado para geração de bicliques em ordem lexicográfica. O Teorema 4.2 mostra como esta condição é usada para assegurar que todas as bicliques são enumeradas em ordem lexicográfica.

Lema 4.1 *Seja $B = X \cup Y$ uma biclique de G. Então $B = B^*$, se e somente se não existe biclique de G contendo $B_j \cup \{\ell\}$, para todo $j \in \{1, \ldots, n\}$ e $\ell \in \{1, \ldots, j\} \setminus B_j$.*

Demonstração:

Seja $B = X \cup Y$ e $B = B^*$. Por contradição, suponha que G contém uma biclique $B' \supseteq B_j \cup \{\ell\}$, $\ell \in \{1, \ldots, j\} \setminus B_j$. Seja ℓ' o primeiro vértice no qual B e B' são discordantes. Claramente, $\ell' \in B'$ e $\ell' \leq \ell \leq j$. Logo, B' é menor lexicograficamente que B, o que é uma contradição.

Por outro lado, seja B uma biclique tal que não exista outra biclique de G contendo $B_j \cup \{\ell\}$, com $j \in \{1, \ldots, n\}$ e $\ell \in \{1, \ldots, j\}$. Novamente por contradição, suponha que $B \neq B^*$. Seja j o menor índice tal que $B_j \neq B_j^*$. Nesse caso, $j \in B^*$ e $j \notin B$. Consequentemente, B^* contém $B_j \cup \{j\}$, o que contraria a hipótese inicial.∎

Seja B uma biclique de G e $B_j = B \cap \{1, \ldots, j\}$. Dizemos que B_j é *fracamente maximal* se não existe biclique $B' \neq B$ tal que B' contém $B_j \cup \{\ell\}$, para algum $\ell \in \{1, \ldots, j\} \setminus B_j$.

Por exemplo, considere a biclique $B'' = \{2\} \cup \{3, 7\}$ do grafo da Figura 4.1. $B''_6 = \{2\} \cup \{3\}$ não é fracamente maximal, pois $B' = \{2, 4, 6\} \cup \{3\}$ contém

$B''_6 \cup \{4\}$. Nesse caso, $j = 6$ é o maior inteiro para o qual B''_j não é fracamente maximal, já que $B''_7 = B''$. Pelo Lema 4.1, B é a menor biclique lexicográfica de G se e somente se B_j é fracamente maximal, para todo $j \in \{1, \ldots, n\}$.

O algoritmo descrito na Figura 4.3, tendo como entrada um grafo $G = (V, E)$ e uma ordenação total sobre os vértices de V, enumera as bicliques de G em ordem lexicográfica. A Figura 4.4 mostra um trecho da execução do algoritmo aplicado ao grafo da Figura 4.1, após a enumeração da biclique $B^* = \{1, 3, 5\} \cup \{4, 6\}$.

Algoritmo GBOL

Entrada: Grafo $G = (V, E)$, ordenação sobre V

Saída: Enumeração de todas as bicliques de G em ordem lexicográfica

Encontre a menor biclique B^* de G
$Q \leftarrow \emptyset$
Inclua B^* na fila Q
Enquanto $Q \neq \emptyset$ faça
    Retire a menor biclique lexicográfica $B = X \cup Y$ de Q
    Enumerar B
    Para cada vértice $j \in V \setminus B$ faça
        $X_j \leftarrow X \cap \{1, \ldots, j\}$
        $Y_j \leftarrow Y \cap \{1, \ldots, j\}$
        Repita duas vezes
            Se $X_j \cap N_j \neq \emptyset$ ou $Y_j \cap \overline{N}_j \neq \emptyset$ então
                $X'_j \leftarrow (X_j \setminus N_j) \cup \{j\}$
                $Y'_j \leftarrow Y_j \setminus \overline{N}_j$
                Se $X'_j \cup Y'_j$ é fracamente maximal então
                    Encontre a menor biclique B' de G contendo $X'_j \cup Y'_j$
                Se $B' \neq \emptyset$ e $B' \notin Q$ então inclua B' em Q
            Troque os conteúdos de X_j e Y_j

Figura 4.3: Algoritmo para geração de bicliques em ordem lexicográfica

Enumerar $B = \{1,3,5\} \cup \{4,6\}$

$j = 2$

$X_2 \leftarrow \{1\}$ e $Y_2 \leftarrow \emptyset$
$X_2 \cap N_2 = \emptyset$ e $Y_2 \cap \overline{N}_2 = \emptyset$

$X_2 \leftarrow \emptyset$ e $Y_2 \leftarrow \{1\}$
$X_2 \cap N_2 = \emptyset$ e $Y_2 \cap \overline{N}_2 = \{1\}$
$X_2' \leftarrow (X_2 \setminus N_2) \cup \{2\} = \{2\}$
$Y_2' \leftarrow Y_2 \setminus \overline{N}_2 = \emptyset$
$(X_2' \cup Y_2') = \{2\}$ é fracamente maximal
$B' \leftarrow \{2,4,6\} \cup \{3\}$
$B' \notin Q$ então $Q \leftarrow B'$

$j = 7$

$X_7 \leftarrow \{1,3,5\}$ e $Y_7 \leftarrow \{4,6\}$
$X_7 \cap N_7 = \emptyset$ e $Y_7 \cap \overline{N}_7 = \emptyset$

$X_7 \leftarrow \{4,6\}$ e $Y_7 \leftarrow \{1,3,5\}$
$X_7 \cap N_7 = \emptyset$ e $Y_7 \cap \overline{N}_7 = \{1,3,5\}$
$X_7' \leftarrow (X_7 \setminus N_7) \cup \{7\} = \{7\}$
$Y_7' \leftarrow Y_7 \setminus \overline{N}_7 = \emptyset$
$(X_7' \cup Y_7') = \{7\}$ não é fracamente maximal

Figura 4.4: Execução do algoritmo após enumerar $B = \{1,3,5\} \cup \{4,6\}$

Teorema 4.2 *Dado um grafo $G = (V, E)$, o algoritmo GBOL enumera todas as bicliques de G em ordem lexicográfica.*

Demonstração:

Claramente, se G tem uma única biclique o algoritmo funciona corretamente. Suponha que G tenha mais que uma biclique. Seja B' uma biclique incluída em Q dentro do laço

Enquanto $Q \neq \emptyset$ faça

na iteração j após uma biclique $B = X \cup Y$ ser listada. Nesse caso, B' é a menor biclique lexicográfica contendo $X'_j \cup Y'_j$, onde $X'_j = (X_j \setminus N_j) \cup \{j\}$ e $Y'_j = Y_j \setminus \overline{N}_j$, para algum $j \in V \setminus B$ que satisfaz $X_j \cap N_j \neq \emptyset$ ou $Y_j \cap \overline{N}_j \neq \emptyset$. Como B'_j é fracamente maximal, concluímos que $B'_j = X'_j \cup Y'_j$. Como $X_j \cap N_j \neq \emptyset$ ou $Y_j \cap \overline{N}_j \neq \emptyset$, a primeira posição $i < j$ na qual as bicliques B e B' são discordantes satisfaz $i \in B \setminus B'$. Consequentemente, B' é lexicograficamente maior que B. Além disso, uma biclique é listada quando é a menor lexicográfica em Q. Logo, as bicliques são listadas em ordem lexicográfica estritamente crescente. Sendo assim, nenhuma biclique é enumerada duas vezes.

Falta mostrar que todas as bicliques são listadas. A prova é feita por indução na extensão da lista de bicliques enumeradas. A primeira biclique listada pelo algoritmo é B^*. No caso geral, seja $B = X \cup Y$, $B \neq B^*$, a

próxima biclique na ordenação lexicográfica das bicliques de G. Mostraremos que $B \in Q$ nesta ocasião e que, nesse caso, B será escolhida pelo algoritmo como a próxima a ser listada, o que conclui a prova. A idéia central é identificar uma biclique $\tilde{B}$ a partir da qual B é incluída em Q, caso $B \notin Q$.

Seja j o maior inteiro tal que $B_j \cup \{\ell\}$ pertence a alguma biclique de G, para algum $\ell \in \{1, \ldots, j\} \setminus B_j$. O Lema 4.1 garante a existência de $j > 0$. Dado que B é um conjunto bipartido completo maximal, $j < n$. Seja $\tilde{B}$ uma biclique de G que contém $B_j \cup \{\ell\}$.

Nesse caso, $\tilde{B}_j \supseteq B_j \cup \{\ell\}$. Claramente, $\tilde{B}_j = \tilde{X}_j \cup \tilde{Y}_j$ é tal que $\tilde{X}_j \supseteq X_j$ e $\tilde{Y}_j \supseteq Y_j$. Além disso, como $\ell \in \tilde{B}_j \setminus B_j$, tem-se que $\tilde{X}_j \neq X_j$ ou $\tilde{Y}_j \neq Y_j$. A maximalidade de j implica que $j + 1 \in B$, pois B_{j+1} é fracamente maximal. Sem perda de generalidade, suponha que $j + 1 \in X$. Novamente pela maximalidade de j, e como $j + 1 \in B$, segue que $N_{j+1} \supseteq \tilde{X}_j \setminus X_j$ e $\overline{N}_{j+1} \supseteq \tilde{Y}_j \setminus Y_j$. Como $\tilde{X}_j \neq X_j$ ou $\tilde{Y}_j \neq Y_j$, a primeira posição i na qual B e $\tilde{B}$ discordam é tal que $i \in \tilde{B} \setminus B$. Logo, $\tilde{B}$ é menor lexicograficamente que B. Pela hipótese de indução, $\tilde{B}$ foi listada pelo algoritmo. A seguir mostramos que na iteração na qual $\tilde{B}$ é listada, o algoritmo inclui uma biclique $\tilde{B}'$ em Q, satisfazendo $\tilde{B}' = B$, caso $\tilde{B}'$ ainda não esteja em Q.

Vamos examinar o vértice $j + 1$ da iteração na qual $\tilde{B}$ foi listada. Como

$N_{j+1} \supseteq \tilde{X}_j \setminus X_j$ e $\overline{N}_{j+1} \supseteq \tilde{Y}_j \setminus Y_j$ e, além disso, $\tilde{X}_j \neq X_j$ ou $\tilde{Y}_j \neq Y_j$, temos que $j+1 \in V \setminus \tilde{B}$. Pela mesma razão, concluímos que é impossível ocorrer simultaneamente $\tilde{X}_j \cap N_j = \tilde{Y}_j \cap \overline{N}_j = \emptyset$ e $\tilde{Y}_j \cap N_j = \tilde{X}_j \cap \overline{N}_j = \emptyset$, o que implica que o bloco

Repita duas vezes

será executado ao menos uma vez para $j+1$. Sem perda de generalidade, assuma que $\tilde{X}_j \cap N_j \neq \emptyset$ ou $\tilde{Y}_j \cap \overline{N}_j \neq \emptyset$. Seja $\tilde{X}'_{j+1} = (\tilde{X}_{j+1} \setminus N_{j+1}) \cup \{j+1\}$ e $\tilde{Y}'_{j+1} = \tilde{Y}_{j+1} \setminus \overline{N}_{j+1}$. Claramente, $\tilde{X}'_{j+1} = X_j \cup \{j+1\}$ e $\tilde{Y}'_{j+1} = Y_j$. A maximalidade de j diz que $\tilde{X}_{j+1} \cup \tilde{Y}_{j+1} \cup \{\ell\}$ é fracamente maximal em G. Portanto, o algoritmo encontra a menor biclique lexicográfica $\tilde{B}'$ de G que contém $\tilde{X}'_{j+1} \cup \tilde{Y}'_{j+1} = B_{j+1}$. Como $B_{j+1} \subseteq B$, concluímos que $\tilde{B}' \neq \emptyset$. Assim, o algoritmo inclui $\tilde{B}'$ em Q, caso $\tilde{B}' \notin Q$.

Completamos a prova mostrando que $\tilde{B}' = B$. Já vimos que $\tilde{B}_{j+1} = B_{j+1}$. Por contradição, suponha que $\tilde{B}' \neq B$. O fato de $\tilde{B}'$ ser a menor biclique lexicográfica contendo B_{j+1}, implica que $\tilde{B}'$ é lexicograficamente menor que B. Consequentemente, a primeira posição $k > j$ na qual $\tilde{B}'$ e B discordam, satisfaz $k \in \tilde{B}'$ e $k \notin B$. Nesse caso, $B_k \cup \{k\}$ é fracamente maximal em G, o que contradiz a maximalidade de j. Logo, $\tilde{B}' = B$.■

O Teorema 4.3 mostra que o algoritmo GBOL dispende tempo polinomial entre a saída de duas bicliques consecutivas na ordenação lexicográgica.

Teorema 4.3 *O algoritmo GBOL enumera as bicliques de G com tempo de atraso polinomial $O(n^3)$.*

Demonstração:

Seja $G = (V, E)$ a entrada do algoritmo GBOL, onde $|V| = n$. O número de bicliques $|\mathcal{B}|$ de G é da ordem de 2^n. Implementando Q como uma fila de prioridade, as operações correspondentes a verificar se uma biclique $B \in Q$, inserir uma biclique B em Q e remover uma biclique B de Q podem ser realizadas em tempo $O(log_2|\mathcal{B}|)$, ou seja, $O(n)$. Ao listar uma biclique B, o algoritmo GBOL realiza $O(n)$ tentativas de gerar bicliques contendo B_j, através de chamadas sucessivas do algoritmo MBLB. Como uma execução do algoritmo MBLB requer tempo $O(n^2)$, a diferença de tempo entre as saídas de duas bicliques de G listadas consecutivamente pelo algoritmo GBOL é $O(n^3)$. O mesmo limite de tempo se aplica tanto para gerar a menor biclique lexicográfica como após listar a maior biclique lexicográfica de G. Portanto, as bicliques são geradas com atraso polinomial $O(n^3)$.■

4.4 Conclusão

O espaço requerido pelo algoritmo corresponde ao tamanho da fila Q, isto é, $O(2^n)$. Como no caso de geração dos conjuntos independentes maximais (respectivamente, cliques) em ordem lexicográfica, um algoritmo para gerar as bicliques de um grafo em ordem lexicográfica, em tempo de atraso polinomial, usando espaço polinomial de memória permanece um problema em aberto.

No entanto, este problema é mais básico no tocante a geração de bicliques. Até agora não se conhece algoritmo com tempo de atraso polinomial e espaço polinomial para gerar as bicliques de um grafo geral, mesmo quando o requerimento da ordenação lexicográfica é desprezado.

Capítulo 5

Geração de bicliques não induzidas

5.1 Introdução

Este capítulo aborda a geração de bicliques não induzidas de um grafo. Na seção 5.2 mostramos uma transformação de um grafo G qualquer em um certo grafo bipartido G' tal que existe uma correspondência um-para-dois entre as bicliques não induzidas de G e as bicliques de G'.

A seção 5.3 apresenta algumas propriedades de bicliques em grafos bipartidos, tais propriedades são utilizadas pelo algoritmo descrito na seção seguinte. Descrevemos, na seção 5.4, um algoritmo que lista as bicliques de um grafo bipartido, que requer tempo polinomial entre a enumeração de duas bicliques consecutivas. Além disso, o espaço de memória utilizado pelo algoritmo é polinomial.

5.2 Transformação

Seja $G = (U \cup W, E)$ um grafo bipartido. Então toda biclique não induzida de G é uma biclique, sendo que uma das partes está totalmente contida em U e a outra totalmente contida em W. A seguir mostramos como um algoritmo para gerar as bicliques de um grafo bipartido pode ser usado para gerar as bicliques não induzidas de um grafo $G = (V, E)$.

Dado um grafo $G = (V, E)$, onde $V = \{1, \ldots, n\}$, o grafo bipartido G' é construído do seguinte modo: G' consiste de dois conjuntos independentes $U = \{u_1, \ldots, u_n\}$ e $W = \{w_1, \ldots, w_n\}$, respectivamente, onde dois vértices u_i e w_j são adjacentes se e somente se $(i, j) \in E$. Observe que $(u_i, w_i) \notin E'$, para $i = \{1, \ldots, n\}$, e que cada biclique $B = X \cup Y$ de G aparece em G' como $B' = (U \cap X) \cup (W \cap Y)$ e como $B'' = (W \cap X) \cup (U \cap Y)$.

Claramente, existe uma correspondência um-para-dois entre as bicliques não induzidas de G e as bicliques de G'.

A Figura 5.1 mostra um grafo G e suas bicliques $\{1, 4\} \cup \{2, 3\}$, $\{2\} \cup \{1, 3, 4, 5\}$ e $\{3\} \cup \{1, 2, 4\}$. As duas bicliques do grafo G' obtido, correspondentes à biclique $\{1, 4\} \cup \{2, 3\}$ de G, estão em destaque na Figura 5.2.

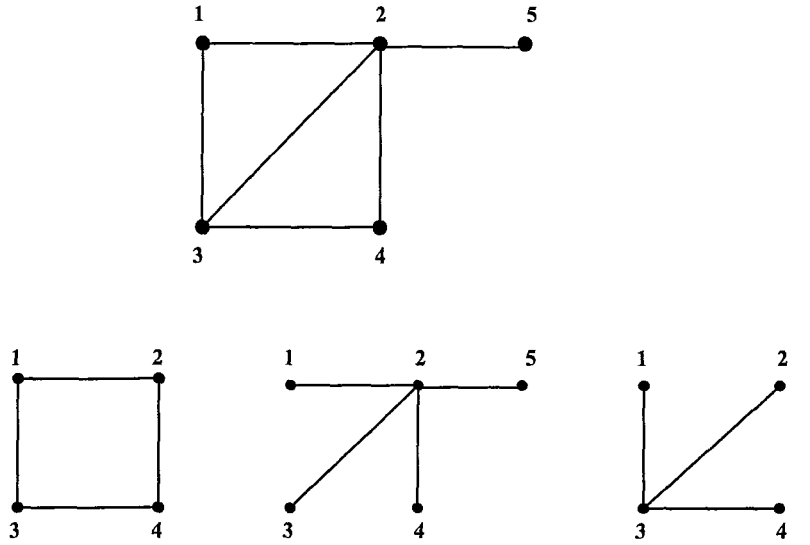


Figura 5.1: Grafo G

Sejam i e j os menores índices, respectivamente, nas partes X e Y de uma biclique de G. Certamente $i \neq j$. Seja A um algoritmo que enumere as bicliques de um grafo bipartido. A fim de listar uma única vez as bicliques não induzidas de G, basta realizar a seguinte alteração em A. Se $i < j$, lista-se a biclique $(U \cap X) \cup (W \cap Y)$ correspondente em G', caso contrário deve ser listada a biclique $(W \cap X) \cup (U \cap Y)$. Com esta convenção: $\{1,4\} \cup \{2,3\}$ será listada como $\{u_1, u_4\} \cup \{w_2, w_3\}$, enquanto $\{2\} \cup \{1,3,4,5\}$ será listada como $\{u_1, u_3, u_4, u_5\} \cup \{w_2\}$. Sendo assim, o problema de geração de bicliques não induzidas em grafos gerais pode ser reduzido ao problema de geração de bicliques em grafos bipartidos, assunto das seções seguintes.

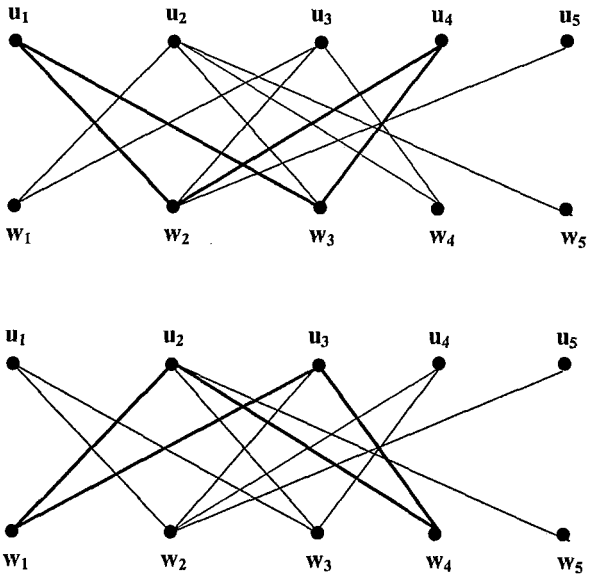


Figura 5.2: Grafo G'

5.3 Propriedades de bicliques em grafos bipartidos

Seja $G = (V, E)$ um grafo bipartido, com partições $U = \{u_1, \ldots, u_p\}$ e $W = \{w_1, \ldots, w_q\}$. Denotamos por $B = X \cup Y$ um conjunto bipartido completo próprio de G tal que $X \subseteq U$ e $Y \subseteq W$. Evidentemente, se $B = X \cup Y$ é um conjunto bipartido completo próprio de G então $X \subseteq F(Y)$ (equivalentemente, $Y \subseteq F(X)$). A Proposição 5.1 caracteriza esta condição de maximalidade.

Proposição 5.1 *Um conjunto bipartido completo próprio* $B = X \cup Y$ *de um grafo bipartido* G *é uma biclique se e somente se* $X = F(Y)$ *e* $Y = F(X)$.

Demonstração:

Seja $B = X \cup Y$ uma biclique de G. Como U e W são conjuntos independentes, não existe $u \in U \setminus X$ tal que $u \in F(Y)$ e não existe $w \in W \setminus Y$ tal que $w \in F(X)$. Logo, $X = F(Y)$ e $Y = F(X)$. Por outro lado, se $X = F(Y)$ e $Y = F(X)$ não existe conjunto bipartido completo próprio contendo estritamente $X \cup Y$. Logo, $X \cup Y$ é biclique de G.■

A Proposição 5.2 fornece uma outra condição de maximalidade, desta vez em relação aos outros conjuntos bipartidos completos próprios maximais do grafo.

Proposição 5.2 *Seja* $B = X \cup Y$ *um conjunto bipartido completo próprio de* G. *Então* B *é uma biclique de* G *se e somente se para toda outra biclique* $B' = X' \cup Y'$ *de* G *valem:*

(i) $X \neq X'$ *e* $Y \neq Y'$; *e*

(ii) $X \subset X'$ *se somente se* $Y' \subset Y$.

Demonstração:

Sejam $B = X \cup Y$ e $B' = X' \cup Y'$ bicliques distintas de G.

(*i*) Sem perda de generalidade, suponha, por absurdo, $X = X'$. Pela Proposição 5.1, $F(X) = Y$ e $F(X') = Y'$, o que implica $B = B'$, uma contradição.

(*ii*) Agora suponha que $X \subset X'$. Claramente, $F(X) \supseteq F(X')$. Novamente, pela Proposição 5.1, $F(X) = Y$ e $F(X') = Y'$. Logo, $Y \supseteq Y'$ e, por (i), $Y \supset Y'$.

Por outro lado, seja $B = X \cup Y$ um conjunto bipartido completo próprio de G tal que para toda biclique $B' = X' \cup Y'$ distinta de G valem (i) e (ii). Por contradição, suponha que B não seja uma biclique de G. Nesse caso, pela Proposição 5.1, $X \subset F(Y)$ ou $Y \subset F(X)$. Sem perda de generalidade, assuma $X \subset F(Y)$. Então, existe um vértice $u \in U \setminus X$ tal que $u \in F(Y)$. Consequentemente, $(X \cup \{u\}) \cup Y$ é um conjunto bipartido completo próprio contido em alguma biclique $B' = X' \cup Y'$ de G. Nesse caso, $X \subset F(Y')$ e $Y \subseteq F(X')$, o que contradiz (ii). ∎

Proposição 5.3 *Sejam $B = X \cup Y$ and $B' = X' \cup Y'$ dois conjuntos bipartidos completos próprios distintos de G, tais que $X \cap X' \neq \emptyset$, $X \not\subset X'$ e $X' \not\subset X$. Então, existe um conjunto bipartido completo próprio $B'' = X'' \cup Y''$ que satisfaz $X'' = X \cap X'$ e $Y'' = Y \cup Y'$.*

Demonstração:

Como $X \cup Y$ e $X' \cup Y'$ são conjuntos bipartidos completos próprios de G, temos $F(X) \supseteq Y$ e $F(X') \supseteq Y'$. Por hipótese, $X \cap X' \neq \emptyset$, portanto $F(X \cap X') \supseteq Y$ e $F(X \cap X') \supseteq Y'$. Consequentemente, $F(X \cap X') \supseteq (Y \cup Y')$. Isto é, $(X \cap X') \cup (Y \cup Y')$ é um conjunto bipartido completo próprio de G.■

Corolário 5.4 *Sejam $B = X \cup Y$ and $B' = X' \cup Y'$ duas bicliques distintas de G, tal que $X \cap X' \neq \emptyset$, $X \not\subset X'$ e $X' \not\subset X$. Então $(X \cap X') \cup F(X \cap X')$ é também uma biclique, contendo $(X \cap X') \cup (Y \cup Y')$.*

Na seção seguinte veremos como estas propriedades são usadas para gerar as bicliques de um grafo bipartido.

5.4 Algoritmo

Seja $U_j = \{u_1, \ldots, u_j\}$. Denotamos por G_j o subgrafo de $G = (U \cup W, E)$ induzido por $U_j \cup W$, com $j \leq p$. Por $\mathcal{B}_j$, o conjunto de bicliques de G_j.

O grafo G_j pode ser obtido a partir de G_{j-1} pela inclusão do vértice u_j e das arestas $\{e = (u_j, w_i) | w_i \in N_{u_j}\}$. Seja $X \cup Y$ uma biclique de $\mathcal{B}_{j-1}$. Pela Proposição 5.1, em G_{j-1}, $F(X) = Y$ e $F(Y) = X$, onde $X \subseteq \{u_1, \ldots, u_{j-1}\}$ e $Y \subseteq W$. Veja que, como todos os vértices de W estão em G_{j-1}, a condição $F(X) = Y$ continua valendo em G_j.

Os Lemas 5.5 e 5.6 indicam como obter as bicliques de $\mathcal{B}_j$ a partir de $\mathcal{B}_{j-1}$. Estes resultados são usados para garantir que toda biclique de G é listada exatamente uma vez pelo algoritmo.

Lema 5.5 *Seja $B = X \cup Y$ uma biclique de G_{j-1}. Então, ao considerarmos as bicliques de G_j, ocorre exatamente uma dentre as seguintes situações:*
(i) $B \notin \mathcal{B}_j$. Neste caso, temos $Y \subseteq N_{u_j}$ e $(X \cup \{u_j\}) \cup Y \in \mathcal{B}_j$.
(ii) $B \in \mathcal{B}_j$. Neste caso, temos $Y \not\subseteq N_{u_j}$. Além disso, se $Y \cap N_{u_j} \neq \emptyset$ e todo $u_k \in \{u_1, \ldots, u_j\} \setminus X$ satisfaz $N_{u_k} \not\supseteq (N_{u_j} \cap Y)$, então $(X \cup \{u_j\}) \cup (Y \cap N_{u_j}) \in \mathcal{B}_j$.

Demonstração:

Seja $B = X \cup Y$ uma biclique de G_{j-1}. Como todos os vértices de W estão em G_{j-1} e G_j, temos que $F(X)$ é o mesmo em G_j e em G_{j-1}.
(i) Suponha que B não seja biclique de G_j. Neste caso, em G_j, necessaria-

mente $F(Y) = X \cup \{u_j\}$. Logo, $Y \subseteq N_{u_j}$ e $(X \cup \{u_j\}) \cup Y$ é uma biclique de G_j.

(ii) Claramente, B é biclique em G_j se e somente se $u_j \notin F(Y)$, i.e, $Y \not\subseteq N_{u_j}$. Agora, suponha $Y \cap N_{u_j} \neq \emptyset$. Então, pela Proposição 5.3, $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ é um conjunto bipartido completo próprio. Claramente, $F(X \cup \{u_j\}) = Y \cap N_{u_j}$. A condição todo $u_k \in \{u_1, \ldots, u_j\} \setminus X$ satisfaz $N_{u_k} \not\supseteq (N_{u_j} \cap Y)$, garante que $F(Y \cap N_{u_j}) = X \cup \{u_j\}$. Logo, pela Proposição 5.1, $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ é biclique de G_j.■

O próximo lema faz a recíproca do Lema 5.5:

Lema 5.6 *Seja $B = X \cup Y$ uma biclique em $\mathcal{B}_j \setminus \mathcal{B}_{j-1}$. Então ocorre exatamente uma dentre as seguintes situações:*

(i) $X = \{u_j\}$. Neste caso, $N_{u_j} \not\subseteq Y'$ para toda biclique $B' = X' \cup Y'$ em $\mathcal{B}_{j-1}$.

(ii) $X \supset \{u_j\}$. Neste caso, $(X \setminus \{u_j\}) \cup F(X \setminus \{u_j\})$ é uma biclique de $\mathcal{B}_{j-1}$.

Demonstração:

Como $V(G_j) \setminus V(G_{j-1}) = \{u_j\}$, toda biclique $B = X \cup Y$ de $\mathcal{B}_j \setminus \mathcal{B}_{j-1}$ satisfaz $u_j \in F(Y)$, i.e, $u_j \in X$. (i) Se $X = \{u_j\}$ então $Y = N_{u_j}$. Suponha, por absurdo, que existe biclique $X' \cup Y'$ de G_{j-1} tal que $N_{u_j} \subseteq Y'$. Neste

caso, $(X' \cup \{u_j\}) \cup N_{u_j}$ é um conjunto bipartido completo próprio que contém estritamente $\{u_j\} \cup N_{u_j}$, uma contradição.

(ii) Certamente $(X \setminus \{u_j\}) \cup F(X \setminus \{u_j\})$ é um conjunto bipartido completo próprio de $\mathcal{B}_{j-1}$, pois $X \setminus \{u_j\} \neq \emptyset$. Se $(X \setminus \{u_j\}) \cup F(X \setminus \{u_j\})$ não é uma biclique em $\mathcal{B}_{j-1}$ então existe vértice $u_k \neq u_j$, $k < j$, tal que u_k é adjacente a todos os vértices de $F(X \setminus \{u_j\})$ mas não pertence ao conjunto $(X \setminus \{u_j\})$. Como $F(X) \subseteq F(X \setminus \{u_j\})$, temos que u_k é adjacente a todos os vértices de $Y = F(X)$, o que contraria $B = X \cup Y$ ser biclique de G_j.■

Na Figura 5.3, descrevemos o algoritmo para listar as bicliques de um dado grafo bipartido, usando os Lemas 5.5 e 5.6.

A construção direta de $\mathcal{B}_j$ a partir de $\mathcal{B}_{j-1}$ fornecida pelos Lemas 5.5 e 5.6 implica em armazenar todo o conjunto $\mathcal{B}_{j-1}$, o que dispenderia um espaço de memória $O(|\mathcal{B}|)$, que é possivelmente exponencial no tamanho do grafo dado. A fim de tornar o espaço necessário polinomial, o algoritmo usa a seguinte estratégia.

As bicliques geradas através de uma chamada $\text{BIC}(\{u_i\}, N_{u_i}, i+1)$, $i = \{1, \ldots, p\}$, são as bicliques $B = X \cup Y$ de G tal que u_i é o menor vértice em X. Claramente, se $N_{u_i} \subseteq N_{u_j}$ e $j < i$ então nenhuma biclique de G

Algoritmo BIC(X, Y, j)

Entrada: Grafo bipartido $G = (V, E)$, biclique $B = X \cup Y$ de G_{j-1}

Saída: Enumeração das bicliques $B = X \cup Y$ de G,
tal que j é o menor vértice em X.

Se $j > p$ então Enumerar $B = X \cup Y$ .....{$B \in G_p$}
Senão
Se $N_{u_j} \subseteq Y$ então $Tree[j] \leftarrow falso$ .....{Lema 5.6(i)}
Se $Y \subseteq N_{u_j}$ então BIC$(X \cup \{u_j\}, Y, j+1)$ .....{Lemas 5.5(i); 5.6(ii)}
Senão BIC$(X, Y, j+1)$ .....{Lema 5.5(ii)}
Se $Y \cap N_{u_j} \neq \emptyset$ então
$f := 0$
Para $u_k \notin X$ e $k < j$ faça
Se $N_{u_k} \supseteq N_{u_j} \cap Y$ então $f := 1$
Se $f = 0$ então BIC$(X \cup \{u_j\}, Y \cap N_{u_j}, j+1)$ ...{Lemas 5.5(ii); 5.6(ii)}

Chamada externa do algoritmo BIC

Para $i = 1$ até p faça $Tree[i] \leftarrow verdadeiro$
Para $i = 1$ até p faça
Se $N_{u_i} \neq \emptyset$ e $Tree[i]$ então BIC$(\{u_i\}, N_{u_i}, i+1)$

Figura 5.3: Algoritmo para gerar as bicliques de G'

satisfaz a condição descrita. Nesse caso, $Tree[i]$ recebe $falso$ e a chamada correspondente não é executada. Caso contrário, as bicliques contendo u_i como menor vértice em X são listadas na árvore de recursão correspondente a $Tree[i]$, implicitamente gerada pela chamada $\text{BIC}(\{u_i\}, N_{u_i}, i+1)$.

Como visto na seção 5.2, uma biclique $B = X \cup Y$ de G aparece no grafo bipartido obtido pela transformação como $B' = (U \cap X) \cup (W \cap Y)$ e como $B'' = (W \cap X) \cup (U \cap Y)$. A fim de listar uma única vez a biclique B, basta armazenar os menores vértices u_i e w_j, respectivamente, de X e Y em cada recursão e listar uma biclique somente se $i < j$. A Figura 5.4 ilustra a árvore de recursão gerada pela chamada $\text{BIC}(\{u_1\}, N_{u_1}, 2)$ para o grafo bipartido da Figura 5.2.

Teorema 5.7 *Seja $G = (V, E)$ um grafo bipartido, com $|V| = n$ e $|E| = m$. O algoritmo lista todas as bicliques de G, exatamente uma vez, com tempo de atraso polinomial $O(nm)$. Além disso, o espaço total requerido pelo algoritmo é $O(n^2)$.*

Demonstração:

Seja $B = X \cup Y$ uma biclique de G tal que u_i é o vértice de menor índice em X. É fácil de ver, usando os Lemas 5.5 e 5.6, que B será listada pelo

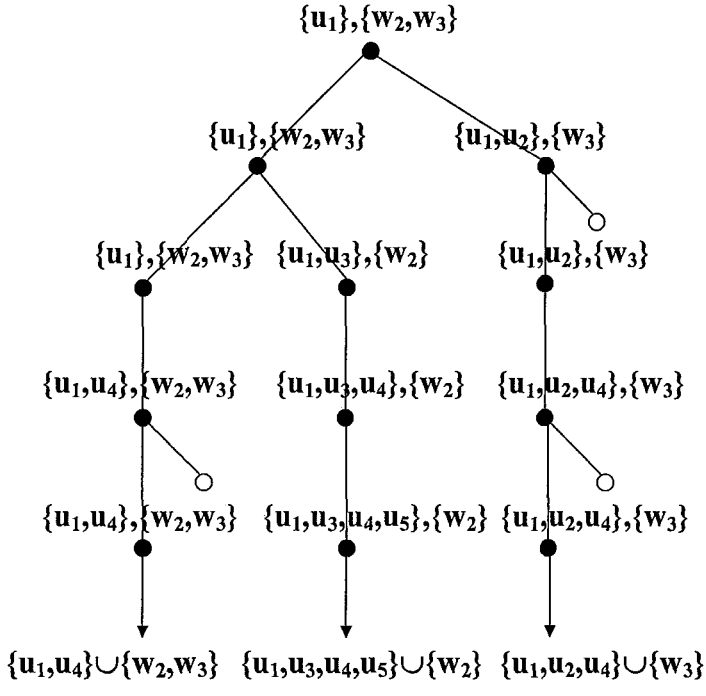


Figura 5.4: Árvore gerada por BIC($\{u_1\}, N_{u_1}, 2$)

algoritmo através da chamada BIC($\{u_i\}, N_{u_i}, j$), $j = \{i+1, \ldots, p\}$. Isto é, o algoritmo lista todas as bicliques de G. Para provar que cada biclique é listada exatamente uma vez, basta provar que cada biclique é listada uma única vez na árvore de recursão $Tree[i]$, implicitamente gerada pela execução da chamada BIC($\{u_i\}, N_{u_i}, i+1$). Observe que cada nó interno correspondente a uma chamada recursiva tem exatamente um ou dois filhos. No segundo caso, sejam B' e B'' os conjuntos bipartidos completos correspondentes aos dois filhos gerados. Claramente, $X' \subset X''$ e $Y'' \subset Y'$, o que implica que todas as folhas de $Tree[i]$ são distintas. Logo, nenhuma biclique é listada mais que uma vez.

Cada iteração da função BIC requer tempo $O(n+m)$, pois a operação dominante (laço Para) pode ser realizada percorrendo-se uma vez o grafo G. Como uma biclique é listada quando $j = p+1$, temos que o tempo decorrente entre a enumeração de duas bicliques sucessivas na saída do algoritmo é $O(nm)$. A fim de avaliar o espaço requerido, observe que para cada chamada recursiva é armazenado exatamente um conjunto bipartido completo próprio $B = X \cup Y$, cujo tamanho é $O(n)$. Logo, o espaço total dispendido é $O(n^2)$.■

5.5 Conclusão

Quando consideramos classes de grafos com número polinomial de bicliques então o algoritmo proposto tem tempo total polinomial. Entre as classes conhecidas com número polinomial de bicliques estão a classe dos grafos bipartidos convexos e a classe dos grafos bipartidos biconvexos. Além disso, algoritmos para classes particulares podem ser descritos de acordo com características estruturais das mesmas a fim de melhorar a complexidade de tempo e/ou espaço.

No capítulo seguinte mostraremos como a estrutura particular de cada uma das classes dos grafos bipartidos convexos e dos bipartidos biconvexos pode ser utilizada para reduzir a complexidade de tempo do algoritmo BIC.

Capítulo 6

Geração de Bicliques em Casos Especiais

6.1 Introdução

Entre as classes de grafos bipartidos que têm número polinomial de bicliques estão as classes bipartido biconvexo, bipartido convexo e bipartido cordal, nesta ordem de inclusão. Em [24], foi estabelecido um limite de $O(m)$ para o número de bicliques de um grafo bipartido cordal. Em [1], foi mencionada a classe dos grafos bipartidos convexos, como um exemplo de classes com número polinomial de bicliques. Algoritmos para classes particulares podem ser descritos de acordo com características estruturais das mesmas, a fim de diminuir o tempo de atraso entre a enumeração de suas bicliques.

Nas seção 6.1, mostramos algumas alterações na descrição do algoritmo

BIC aplicado a grafos convexos. Nesta nova versão, o tempo entre a enumeração de duas bicliques sucessivas na saída do algoritmo é $O(n^2)$. No caso de grafos biconvexos este tempo pode ser reduzido para $O(n)$, através de modificações de acordo com as características destes grafos. Este último algoritmo é descrito na seção 6.2.

6.2 Grafos bipartidos convexos

Seja $G = (U \cup W, E)$ um grafo bipartido. Uma ordenação $<$ de W em G tem a *propriedade de adjacência* se para todo vértice $u \in U$, N_u consiste de vértices que são consecutivos (formam um intervalo) na ordem $<$ de W. Um grafo bipartido G é *convexo* se existe uma ordenação de U ou W que satisfaz a propriedade de adjacência.

A Figura 6.1 mostra um grafo bipartido convexo, com uma ordenação de W que satisfaz a propriedade de adjacência.

Pode-se determinar se um grafo G é convexo em tempo linear aplicando o algoritmo *PQ-tree* [4]. Sendo convexo o mesmo pode ser representado em espaço $O(n)$: para cada vértice u de U basta armazenar os limites do intervalo de seus vizinhos na ordenação correspondente em W.

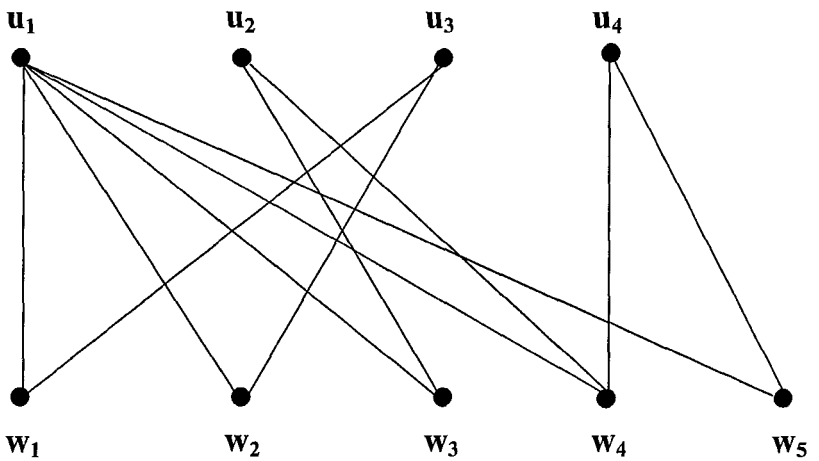


Figura 6.1: Grafo bipartido convexo

Seja $G = (U \cup W, E)$ um grafo bipartido convexo tal que W satisfaz a propriedade de adjacência. Denotamos por $I_k = [i_k, s_k]$ o intervalo correspondente a N_{u_k}. Seja $B = X \cup Y$ um conjunto bipartido completo de G, com $X \subseteq U$ e $Y \subseteq W$. Claramente, Y pode ser representado por um intervalo. Denotaremos por $I_Y = [i_y, s_y]$ o intervalo correspondente a Y.

Para grafos bipartidos convexos, um vértice u_k satisfaz $N_{u_k} \supseteq Y \cap N_{u_j}$, para algum u_j, se e somente se $i_k \leq max\{i_y, i_j\}$ e $s_k \geq min\{s_y, s_j\}$. A operação dominante para o cálculo de complexidade do algoritmo BIC é o laço Para. No laço, testamos para cada $u_k \notin X$, $k < j$, a condição

$N_{u_k} \not\supseteq (N_{u_j} \cap Y)$. Logo, neste caso podemos decidir se $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ é uma biclique em G_j em tempo $O(n)$. Lembramos que uma biclique é enumerada pelo algoritmo quando $j = |U| + 1$. Logo, o tempo entre a enumeração de duas bicliques consecutivas na saída do algoritmo BIC é $O(n^2)$ para grafos bipartidos convexos.

6.3 Grafos bipartidos biconvexos

Um grafo bipartido $G = (U \cup W, E)$ é *biconvexo* se existem ordenações de U e de W que satisfazem a propriedade de adjacência.

O grafo G ilustrado na Figura 6.1 não é bipartido biconvexo, pois não existe ordenação de U que satisfaça à propriedade de adjacência. A Figura 6.2 mostra o grafo G' obtido de G pela remoção do vértice w_5, sendo este um grafo bipartido biconvexo.

Seja $G = (U \cup W, E)$ um grafo bipartido biconvexo. Aplicando o algoritmo *PQ-tree* para cada uma das partições U e W obtemos as ordenações desejadas. Seja $B = X \cup Y$ um conjunto bipartido completo de G. Nesse caso, ambas as partes X e Y podem ser representadas por intervalos. Denotamos por $I_X = [i_x, s_x]$ e $I_Y = [i_y, s_y]$, respectivamente, os intervalos

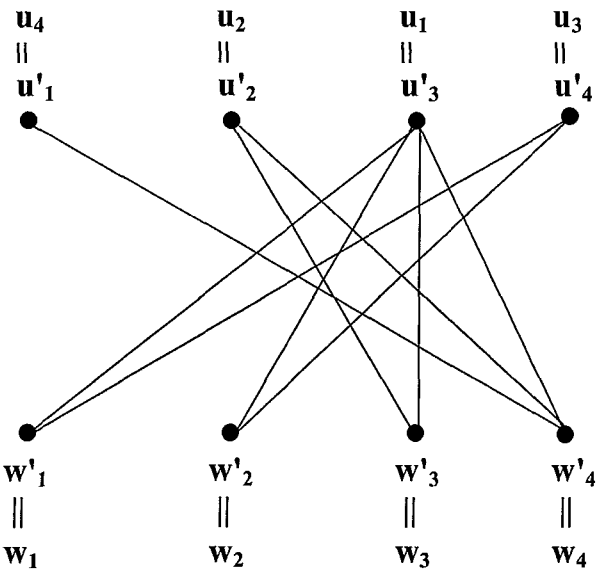


Figura 6.2: Grafo bipartido biconvexo

correspondentes a X e Y.

Seja $B = X \cup Y$ uma biclique de G_{j-1}. O Lema 6.1 caracteriza uma biclique $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ de G_j em relação aos intervalos I_X, I_Y e I_j.

Lema 6.1 *Seja $G = (U \cup W, E)$ um grafo bipartido biconvexo, onde U e W correspondem a ordenações que satisfazem a propriedade de adjacência. Seja $B = X \cup Y$ uma biclique de G_{j-1}. Então $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ é uma biclique de G_j se e somente se (i) $j = s_x + 1$, e (ii) $N_{u_{(i_x-1)}} \not\supseteq Y \cap N_{u_j}$ ou $i_x = 1$.*

Demonstração:

Suponha que $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ é uma biclique de G_j.

(i) Por absurdo, assuma $j > s_x + 1$. Como U satisfaz a propriedade de adjacência, $u_{(s_x+1)} \in F(N_{u_j} \cap Y)$. Logo, $F(N_{u_j} \cap Y) \supset \{u_{i_x}, \ldots, u_{s_x}\} \cup \{u_j\}$, o que contradiz $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ ser uma biclique de G_j.

(ii) Agora, seja $j = s_x + 1$. Por hipótese, $F(N_{u_j} \cap Y) = \{u_{i_x}, \ldots, u_{s_x}\} \cup \{u_j\}$. Pelo Lema 5.6, não existe $u_k \notin X$, $k < j$, tal que $N_{u_k} \supseteq N_{u_j} \cap Y$. Particularmente, $N_{u_{(i_x-1)}} \not\supseteq Y \cap N_{u_j}$.

Por outro lado, assuma que (i) $j = s_x + 1$ e (ii) $N_{u_{(i_x-1)}} \not\supseteq Y \cap N_{u_j}$ ou $i_x = 1$. Por contradição, suponha que $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ não é uma biclique de G_j. Então, pelo Lema 5.5, existe um vértice $u_k \notin X$, $k < j$, tal que $N_{u_k} \supseteq N_{u_j} \cap Y$. Consequentemente, por (i) e (ii), temos que $k < (i_x - 1)$, o que contradiz a propriedade de adjacência de U. ∎

Usando o Lema 6.1, podemos facilmente verificar em tempo $O(1)$ quando $(X \cup \{u_j\}) \cup (Y \cap N_{u_j})$ é uma biclique em G_j. Obviamente, qualquer outra operação do algoritmo BIC pode ser realizada em tempo constante para grafos bipartidos biconvexos. Portanto, o tempo dispendido entre a enumeração de duas bicliques consecutivas na saída do algoritmo é $O(n)$.

6.4 Conclusão

O tempo de atraso entre a enumeração de duas bicliques através do algoritmo BIC apresentado no capítulo 5 é $O(n.m)$. Neste capítulo vimos que este tempo pode ser reduzido para $O(n^2)$ e $O(n)$, respectivamente, quando consideramos as classes de grafos BIPARTIDO CONVEXO e BICONVEXO.

Vimos, portanto, como características estruturais de classes particulares de grafos podem ser utilizadas na descrição de algoritmos obtendo assim melhores resultados de complexidade. Neste sentido, seria interessante investigar outras classes de grafos a fim de apresentar algoritmos que se beneficiem de suas relativas características.

Capítulo 7

Conclusões

O foco desta tese consistiu na geração de bicliques de um grafo. Com este objetivo, realizamos no capítulo 2 uma revisão das diferentes noções de complexidade empregadas na geração de objetos de uma coleção. Neste caso, existem três classificações básicas de análise de complexidade de algoritmos para os quais o tamanho da saída é possivelmente exponencial no tamanho da entrada: Tempo Polinomial Total, Tempo Polinomial Incremental e Tempo de Atraso Polinomial. Observamos que os algoritmos considerados eficientes neste contexto são aqueles que apresentam complexidade de atraso polinomial.

No capítulo 3 consideramos dois problemas de decisão relacionados à geração de bicliques: PARTE DE BICLIQUE e MAIOR BICLIQUE LEXICOGRÁFICA. Demonstramos que o primeiro é NP-completo e o último Co-NP-completo, resultados estes que sugerem restrições na

existência de algoritmos eficientes para a geração de bicliques segundo critérios específicos. Em particular, a NP-completude do problema MAIOR BICLIQUE LEXICOGRÁFICA implica que não é possível descrever um algoritmo com atraso polinomial para a geração de bicliques em ordem lexicográfica reversa, a menos que P=NP. Em contraste, apresentamos no capítulo 4 um algoritmo polinomial que, dado um conjunto bipartido completo B', encontra a menor biclique lexicográfica contendo B', caso exista tal biclique. O tempo total do algoritmo é $O(n^2)$.

Em uma das principais contribuições desta tese fornecemos, no capítulo 4, um algoritmo que gera as bicliques de um grafo em ordem lexicográfica, que requer tempo $O(n^3)$ entre a enumeração de duas bicliques consecutivas em sua saída. Até o momento não era conhecido nenhum algoritmo eficiente para a geração de bicliques (induzidas) em um grafo qualquer. No capítulo 5, descrevemos um algoritmo com a mesma complexidade de tempo para gerar as bicliques não induzidas de um grafo, que requer espaço de memória polinomial.

Apresentamos ainda, no capítulo 6, dois algoritmos para a geração de bicliques de duas classes de grafos especiais: bipartidos convexos e bipartidos biconvexos. Nas duas classes, o número de bicliques a serem geradas é polinomial em n. O algoritmo para gerar as bicliques de um grafo bipartido

convexo requer tempo $O(n^2)$ entre a enumeração de duas bicliques, enquanto que o algoritmo para grafos bipartidos biconvexos requer tempo $O(n)$. Como o número de bicliques a serem geradas é polinomial, estes algoritmos são de fato polinomiais.

Como recomendação para estudos futuros, sugerimos a investigação de um algoritmo de atraso polinomial para gerar as bicliques (induzidas) de um grafo que faça uso de espaço polinomial de memória.

Mostramos, no capítulo 2, como usar um algoritmo para a geração de conjuntos independentes para gerar as bicliques de um grafo. No entanto, mostramos também que existem classes de grafos com número exponencial de conjuntos independentes e número polinomial de bicliques. Nesse sentido, propomos a investigação de classes de grafos nas quais as ordens de grandeza do número de conjuntos independentes e do número de bicliques sejam ambas polinomiais ou ambas exponenciais no tamanho do grafo.

Sugerimos ainda, a investigação de classes com número polinomial de bicliques e a descrição de algoritmos polinomiais para gerar as bicliques destas classes.

Referências Bibliográficas

[1] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P.L. Hammer, and B. Simeone. Consensus algorithms for the generation of all maximal bicliques. DIMACS Technical Report 52, Rutgers University, 2002. Aceito para a publicação em Discrete Applied Math.

[2] J. Amilhastre, M. C. Vilarem, and P. Janssen. Complexity of minimum biclique cover and minimum biclique decomposition for bipartite dominofree graphs. *Discrete Applied Math.*, 86:125–144, 1998.

[3] H. J. Bandelt, M. Faber, and P. Hell. Absolute reflexive retracts and absolute bipartites retracts. *Discrete Applied Math.*, 44:9–20, 1993.

[4] K. S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algortihms. *J. Comput. System*, 13:335–379, 1976.

[5] J. M. Byskov. Algorithms for k-colouring and finding maximal independent sets. In *Proc. 14th Symp. Discrete Algorithms*, pages 456–457. ACM and SIAM, 2003.

[6] E. R. Canfield and S. G. Williamson. A loop-free algorithm for generating the linear extensions of a poset. *Order*, 12:1–18, 1995.

[7] M. Dawande, J. Swaminathan, P. Keskinocak, and S. Tayur. On bipartite and multipartite clique problems. *Journal of Algorithms*, 41:388–403, 2001.

[8] D.Eppstein. All maximal independent sets and dynamic dominance for sparse graphs. In *16th ACM-SIAM Symp. Discrete Algorithms*, Vancouver, 2005. ACM and SIAM.

[9] V. M. F. Dias, C. M. H. Figueiredo, and J. L. Szwarcfiter. Generating bicliques of a graph in lexicographic order. *Submetido ao Theoretical Computer Science*, em abril de 2004.

[10] V. M. F. Dias, C. M. H. Figueiredo, and J. L. Szwarcfiter. Generating all non-induced bicliques of a graph. *Submetido ao Discrete Applied Math*, em setembro de 2004.

[11] V. M. F. Dias, C.M.H. Figueiredo, and J.L. Szwarcfiter. On the generation of bicliques of a graph. *Eletronic Notes in Discrete Mathematics*, 2004.

[12] V. M. F. Dias, G. D. Fonseca, C. M. H. Figueiredo, and J. L. Szwarcfiter. Stable matchings with restricted pairs. *Eletronic Notes in Discrete Mathematics*, 7:5–9, 2001.

[13] V. M. F. Dias, G. D. Fonseca, C. M. H. Figueiredo, and J. L. Szwarcfiter. The stable marriage problem with restricted pairs. *Theoretical Computer Science*, 306:391–405, 2003.

[14] V. M. F. Dias and J. L. Szwarcfiter. Casamentos estáveis com casais forçados e casais proibidos. *Tendências Em Matemática Aplicada e Computacional*, 1:361–372, 2000.

[15] V. M. F. Dias and J. L. Szwarcfiter. Grafos clique ponderados. In *Anais do XXXIII Simpósio Basileiro de Pesquisa Operacional*, Campos do Jordão, SP, 2001.

[16] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Inform. Process. Lett*, 51:207–211, 1994.

[17] V. Froidure. *Rangs des Relations Binaries, Semigroupes de Relations non Ambigues.* PhD thesis, Univ. Paris VI, June 1995.

[18] M. Garey and D. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[19] M.C. Golumbic and C.F. Goss. Perfect elimination and chordal bipartite graphs. *J. Graph Theory*, 2:155–163, 1978.

[20] W. H. Haemers. Bicliques and eigenvalues. *J. Combin. Theory*, B 82:56–66, 2001.

[21] D. S. Hochbaum. Approximating clique and biclique problems. *Journal of Algorithms*, 29(1):174–200, 1998.

[22] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Inform. Process. Lett.*, 27:119–123, 1988.

[23] P. Keskinocak, M. Dawande, and S. Tayur. On the biclique problem in bipartite graphs. Gsia working paper, Carnegie Mellon University, Pittsburgh, USA, 1996.

[24] T. Kloks and D. Kratsch. Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph. *Inform. Process. Lett*, 55:11–16, 1995.

[25] E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM J. Comput.*, 9(3):558–565, 1980.

[26] A. Lubiw. The boolean basis problem and how to cover some polygons by rectangles. *SIAM J. Discrete Math.*, 3(1):98–115, 1990.

[27] A. Lubiw. A weighted min-max relation for intervals. *J. Combin. Theory*, 53(2):151–172, 1991.

[28] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Proc. 9th Scand. Worksh. Algorithm Theory (SWAT2004)*, pages 260–272. Lectures Notes in Nomputer Scince 3111, 2004.

[29] T. A. McKee and F. R. McMorris. Topics in intersection graph theory. *SIAM Monographs on Discrete Math. And Appl.*, 2:121–154, 1999.

[30] H. Muller. On edge perfectness and classes of bipartite graphs. *Discrete Math.*, 149:159–187, 1996.

[31] H. Muller. Recognizing interval digraphs and interval bigraphs in polynomial time. *Discrete Applied Math.*, 78:189–205, 1997.

[32] Dana S. Nau, George Markowsky, Max A. Woodbury, and D. Bernard Amos. A mathematical analysis of human leukocyte antigen serology. *Mathematical Biosciences*, 40:243–270, 1978.

[33] G. Nemhauser and L. Wosley. *Integer and Combinatorial Optimization.* John Wiley and Soons, 1988.

[34] J. Orlin. Containment in graph theory: covering graphs with cliques. *Nederl. Akad. Wetensch. Indag. Math.*, 39:211–218, 1997.

[35] Marvin C. Paull and Stephen H. Unger. Minimizing the number of states in incompletely specified sequential switching functions. *IRE Trans. Electron. Comput.*, EC-8:356–367, 1959.

[36] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Appl. Math.*, 131:651–654, 2003.

[37] E. Prisner. Bicliques in graphs 2: Recognizing k-path graphs and underlying graphs of line digraphs. In *Proceedings of WG97*, pages 273–287, Berlin, 1997. LNCS 1335.

[38] E. Prisner. Bicliques in graphs 1: Bounds on their number. *Combinatorica*, 20(1):109–117, 2000.

[39] C. Savage. A survey of combinatorial gray codes. *SIAM review*, 39(4):605–629, 1997.

[40] V. Stix. Finding all maximal cliques in dynamic graphs. *Computational Optimization Appl.*, 27(2):173–186, 2004.

[41] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques. In *Proc. 10th Int. Computing and Combinatorics Conf. (COCOON 2004)*, 2004.

[42] S. Tsukiyama, M. Ide, H. Arujoshi, and H. Ozaki. A new algorithm for generating the maximal independent sets. *SIAM J. Comput.*, 6(3):505–517, 1997.

[43] Z. Tuza. Covering of graphs by complete bipartite subgraphs; complexity of 0-1 matrices. *Combinatorica*, 4(1):111–116, 1984.

[44] R. Wille. Restructuring lattice theory: Na approach based on hierarchies of contexts. *NATO ASI*, 83:445–470, 1982.

[45] M. Yannakakis. Node and edge deletion NP-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 253–264, Ney York, 1978. Association for Computing Machinery.