PARTICIONAMENTO E EXTENSÃO DE GRAFOS CORDAIS EM CONJUNTOS INDEPENDENTES E CLIQUES

Loana Tito Nogueira

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:

Profª. Sulamita Klein, D.Sc.

Prof. Fábio Protti, D.Sc.

Dr. Pavol Hell, Ph.D

Profª. Yoshiko Wakabayashi, Dr.rer.nat.

Profª. Célia Picinin de Mello, D.Sc.

Prof. Jayme Luiz Szwarcfiter, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
NOVEMBRO DE 2003

NOGUEIRA, LOANA TITO

Particionamento e extensão de grafos cordais em conjuntos independentes e cliques [Rio de Janeiro] 2003

IX, 124 p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2003)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Grafos Cordais

2 - Grafos-(k, l)

3 - M-partição

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.)

PARTICIONAMENTO E EXTENSÃO DE GRAFOS CORDAIS EM CONJUNTOS INDEPENDENTES E CLIQUES

Loana Tito Nogueira

Novembro/2003

Orientadores: Sulamita Klein

Fábio Protti

Programa: Engenharia de Sistemas e Computação

Este trabalho descreve um estudo sobre particionamento em grafos. Estudamos os grafos-(k, l), grafos cujo conjunto de vértices pode ser particionado em k conjuntos independentes e l cliques. Este problema é uma generalização natural do problema de reconhecer grafos split, e é NP-completo para grafos em geral, a menos que $k \leq 2$ e $l \leq 2$. Apresentamos um teorema de caracterização de grafos cordais-(k, l) por subgrafos induzidos e algoritmos eficientes para reconhecimento desta classe de grafos. Além disso, consideramos M-partições gerais para a classe dos grafos cordais. Para cada matriz simétrica M definida sobre $0, 1, *$, o problema da M-partição procura por uma partição do grafo de entrada em conjuntos independentes, cliques, ou conjuntos arbitrários, com algumas exigências entre alguns pares de conjuntos, por exemplo, que sejam completamente adjacentes ou que não possuam nenhuma aresta entre eles, tal como especificado na matriz M. Tais partições generalizam o problema da coloração de grafos e o problema do homomorfismo, e aparecem frequentemente na teoria de grafos perfeitos. Mostramos que muitos problemas de M-partição que são NP-completos em geral tornam-se polinomiais quando restritos à classe dos grafos cordais, mesmo na presença de listas. Por outro lado, mostramos que existem problemas de M-partição que permanecem NP-completos mesmo para grafos cordais.

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

PARTITION AND EXTENSION OF CHORDAL GRAPHS INTO INDEPENDENT SETS AND CLIQUES

Loana Tito Nogueira

November/2003

Advisors: Sulamita Klein

Fábio Protti

Department : Computing and Systems Engineering

This work describes a study on graphs partitions. We study the (k, l)-graphs, graphs whose set of vertices can be partitioned into k independent sets and l cliques. This is a natural generalization of the problem of recognizing split graphs, and it is NP-complete for graphs in general, unless $k \leq 2$ and $l \leq 2$. We present a theorem of characterization of chordal (k, l)-graphs by forbidden subgraphs and efficient algorithms for recognizing chordal (k, l)-graphs. Moreover, we consider general M-partitions for the class of chordal graphs. For each symmetric matrix M over $0, 1, *$, the M-partition problem seeks a partition of the input graph into independent sets, cliques, or arbitrary sets, with some pairs of sets being required to have no edges, or to have all edges joining them, as encoded in the matrix M. Such partitions generalize graph colorings and homomorphisms, and arise frequently in the theory of perfect graphs. We show that many M-partition problems that are NP-complete in general become polynomial time solvable for chordal graphs, even in the presence of lists. On the other hand, we show that there are M-partition problems that remain NP-complete even for chordal graphs.

A quem tudo devo, os meus pais

"Foi o tempo que dedicaste à tua rosa que fez tua rosa tão importante."

(Saint-Exupéry, em O Pequeno Príncipe)

Agradecimentos

Além do Criador de tudo, são muitas as pessoas às quais devo agradecer, pois nenhuma atividade científica é isolada. Ao longo do caminho, tive o apoio e a orientação de todas as pessoas que depositaram sua confiança em mim. Gostaria de agradecer, em particular, ao meu querido Cacau (Carlos Henrique), pelo amor e amizade; aos meus pais, por terem me ensinado a confiar em minha intuição; aos meus irmãos, Alex, Léo e Aninha, por todo o carinho; ao meu querido Filipe, pelo prazer em ouvi-lo dizer titia nos momentos tensos; à minha querida Jajá (in *memoriam*), pelo carinho e orgulho que sempre demonstrou ter por mim.

Tenho também uma enorme dívida de gratidão para com:

Sula e Fábio, meus orientadores, cujo compromisso, sabedoria e amizade me ajudaram e me inspiraram na execução deste trabalho.

O professor Pavol, pelas discussões que contribuíram para a conclusão desta tese e cujas idéias sempre foram bem vindas.

O professor Jayme, que me acolheu nesta área de pesquisa, pelo seu apoio em todos os sentidos.

O professor e amigo Luérbio, pela amizade e pelo incentivo acadêmico.

As professoras Célia e Yoshiko, cuja atenção aos detalhes melhoraram a versão final deste trabalho.

Os meus verdadeiros amigos, que são como irmãos, compartilhando tristezas e alegrias comigo. Em particular, Carlile, um amigo com muitas afinidades; Renata, sua esposa, por sua amizade e por ter conseguido manter o Carlile no Rio. Debs e Rosa, duas amigas super poderosas, pelas ajudas prestativas, além das brincadeiras descontraídas que sempre me tiravam do eixo e do trabalho.

O amigo Mitre, pela paciente ajuda, pois as dúvidas e problemas de Latex eram inesgotáveis.

Todos os colegas de pós-graduação, pelo espírito de coleguismo.

Os professores do Programa de Engenharia de Sistemas e Computação da COPPE-Sistemas/UFRJ.

As secretárias Cláudia, Solange, Sueli, Lúcia, Mercedes.

A CAPES, pelo suporte financeiro durante todo o tempo de pós-graduação.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Este trabalho descreve um estudo em problemas de partição de grafos, estudo esse que tem despertado muito interesse devido às pesquisas em grafos perfeitos [29] e também pela procura de algoritmos eficientes de reconhecimento de determinadas classes de grafos. Muitos desses problemas (de partição de grafos) podem ser descritos como tendo por objetivo particionar o conjunto dos vértices de um grafo em subconjuntos $V_1, V_2, \ldots, V_k$ onde $V_1 \cup V_2 \cup \ldots \cup V_k = V$ e $V_i \cap V_j = \emptyset$, $i \neq j$, $1 \leq i, j \leq k$, exigindo-se, porém, algumas propriedades sobre estes subconjuntos de vértices. Estas propriedades podem ser *internas*, como por exemplo exigir que os vértices de cada subconjunto V_i sejam completamente adjacentes (neste caso, V_i é uma clique) ou completamente não-adjacentes (neste caso, V_i é um conjunto independente), ou *externas*, onde as exigências são feitas sobre os pares (V_i, V_j), isto é, V_i e V_j podem ser completamente adjacentes ou não-adjacentes entre si. Como exemplo, citamos um dos problemas mais famosos que se insere neste con-

texto, o problema da k-coloração, onde desejamos particionar os vértices de um grafo em k conjuntos independentes $V_1, \ldots, V_k$ (sem restrições externas). Sabe-se que esse problema é polinomial para $k \leq 2$ e NP-completo para $k \geq 3$.

Outro problema bastante conhecido de particionamento de grafos é verificar se um dado grafo G é split, ou equivalentemente, verificar se o conjunto dos vértices de G pode ser particionado em dois subconjuntos, dos quais um é independente e o outro é uma clique. Provou-se que o reconhecimento de grafos split pode ser realizado em tempo linear (veja por exemplo [29]).

Recentemente, uma generalização de grafos split foi proposta por Brandstädt, que definiu uma nova classe de grafos, a classe dos *grafos-*(k, l), a qual também chamamos grafos split generalizados, como sendo aquela formada pelos grafos cujo conjunto de vértices pode ser particionado em k conjuntos independentes e l cliques. Brandstädt considerou em particular as classes grafos-$(2, 1)$, grafos-$(1, 2)$ e grafos-$(2, 2)$, apresentando algoritmos polinomiais para reconhecer estas classes [11, 12]. Feder *et al.* [25] também apresentaram algoritmos polinomiais para o reconhecimento destas classes que surgiram como sub-produto de algoritmos de partição em subgrafos densos e esparsos. Por outro lado, sabe-se que reconhecer grafos-(k, l) para $k \geq 3$ ou $l \geq 3$ é NP-completo [11]. Como exemplo, podemos considerar a classe dos grafos-$(k, 0)$, que corresponde ao problema de reconhecer se um dado grafo é k-colorível (já mencionado anteriormente, cujo reconhecimento

é NP-completo para $k \geq 3$).

Sendo NP-completo o reconhecimento de grafos-(k,l), para $k \geq 3$ ou $l \geq 3$, restringimos este problema à classe dos grafos cordais. Um grafo é dito cordal se todo ciclo induzido de tamanho pelo menos 4 possui uma corda, que é uma aresta ligando dois vértices não consecutivos no ciclo. Apresentamos uma caracterização por subgrafos proibidos para a classe dos grafos cordais-(k,l), assim como um algoritmo polinomial de complexidade $O(n(m+n))$ para o reconhecimento desta classe de grafos. Mais especificamente, provamos o seguinte teorema:

Teorema 1.1 (Caracterização) *Um grafo cordal é grafo-(k,l) se e somente se não contém $(l+1)$ cópias independentes de K_{k+1}. (Um conjunto de subgrafos de um grafo G é dito independente se eles são dois a dois disjuntos e não-adjacentes em G, i.e., não existem em G arestas entre eles.)*

Um caso especial deste resultado, para grafos cordais-$(2,1)$ foi inicialmente tratado em [36].

Nosso algoritmo de reconhecimento encontra, na verdade, o menor valor de l para o qual G é um grafo-(k,l). O algoritmo se torna mais eficiente, possuindo complexidade $O(m+n)$, quando $k=1$, i.e., quando procuramos uma partição do grafo em um conjunto independente e *um* conjunto de cliques. Quando ambos k e l são iguais a um, adaptamos o algoritmo para gerar um algoritmo mais simples e eficiente para grafos split. (Observe que neste caso

não precisamos da restrição de o grafo de entrada ser um grafo cordal, pois grafos split são cordais).

Uma forma alternativa de enunciar nosso resultado baseia-se no seguinte fato: o número máximo de $K_r's$ independentes num grafo cordal é igual ao número *mínimo* de cliques necessárias que interceptam todos os $K_r's$ de G. Em outras palavras, se denotarmos por $\alpha(G,r)$ o número máximo de cópias independentes de $K_r's$ em G, e por $\kappa(G,r)$ o número mínimo de cliques que interceptam todos os $K_r's$ de G, então podemos mostrar que para um grafo cordal $\alpha(G,r) = \kappa(G,r)$. Observe que quando $r = 1$, $\alpha(G,r)$ é o número de independência de G, e $\kappa(G,r)$ é o número de cobertura por cliques de G. Portanto, este resultado generaliza o resultado de que $\alpha(G) = \kappa(G)$, para G cordal, i.e., o resultado de que grafos cordais são perfeitos. Chamaremos a igualdade $\alpha(G,r) = \kappa(G,r)$ de r-perfeição. Nosso algoritmo de complexidade $O(n(m+n))$ identifica $\alpha(G,r)$ cópias independentes de $K_r's$ e o mesmo número de cliques que interceptam todos os $K_r's$ de G.

Independentemente do teorema de caracterização e do algoritmo de reconhecimento de grafos cordais-(k,l), acima mencionados, formulamos o problema de reconhecer grafos cordais-(k,l) em termos de programação linear. Além disso, apresentamos o problema na versão com pesos, formulado da seguinte maneira: é dado um grafo cordal $G = (V,E)$, tal que cada vértice de G possui um peso a ele associado. Nosso objetivo é encontrar um conjunto de K_r's independentes de peso total máximo. Esse problema corresponde

exatamente ao problema anterior quando resolvido para grafos onde todos os vértices possuem o mesmo peso. Fornecemos dois algoritmos para resolver o problema, ambos baseados no princípio da complementaridade de folga, que resolvem esse problema, assim como o seu dual. O primeiro opera diretamente sobre o grafo G, e o segundo opera numa extensão de G, e é uma adaptação de um algoritmo de Farber [22]. Quando os algoritmos são adaptados para a situação em que todos os vértices possuem o mesmo peso, obtemos métodos mais simples para resolver o problema original, embora não sejam tão eficientes.

Ainda neste trabalho, estendemos o nosso foco e consideramos M-partições em grafos cordais. O problema da M-partição foi introduzido em [25]. Para cada matriz simétrica M definida sobre $0, 1, *$, o problema da M-partição tem como objetivo encontrar uma partição de um grafo de entrada em conjuntos independentes, cliques, ou conjuntos arbitrários, exigindo-se algumas restrições entre esses conjuntos; por exemplo, pode-se exigir que dois desses conjuntos tenham todas as arestas entre eles, ou que não tenham nenhuma aresta entre eles, conforme indicado na matriz M. Tais partições generalizam grafos-(k, l) e portanto, coloração em grafos. Outro problema também abordado, e ainda mais geral, é o problema da M-partição com listas, uma instância deste problema é um grafo G e uma coleção de listas $L(x)$, $x \in V(G)$, cada lista sendo um conjunto de partes (uma parte é um dos subconjuntos de vértices da partição). Uma solução para a instância da M-partição com listas é uma solução para a M-partição correspondente, tal que

cada vértice x pertence a uma das partes $A_i \in L(x)$. M-partições com listas generalizam coloração com listas e homomorfismos [24], e são de bastante interesse na teoria de grafos perfeitos [13, 19]. Muitos problemas conhecidos: corte clique, conjunto homogêneo, corte 'skew', etc., podem ser formulados como problemas de M-partição com listas [25]. Além disso, o estudo de problemas de M-partição pode nos conduzir a soluções eficientes de alguns desses problemas [19]. Mostramos que muitos problemas de M-partição que são NP-completos em geral tornam-se polinomiais quando restritos à classe dos grafos cordais (os grafos cordais-(k, l) podem ser vistos como um desses exemplos), mesmo na presença de listas. Por outro lado, mostramos que existem problemas de M-partição que permanecem NP-completos mesmo para grafos cordais.

1.1 Organização do Trabalho

Este trabalho está dividido em cinco capítulos. No primeiro capítulo apresentamos os objetivos, as definições e notações utilizadas no decorrer da tese. No capítulo seguinte introduzimos inicialmente os grafos-(k, l), e apresentamos uma caracterização para os grafos cordais-(k, l), assim como um algoritmo com complexidade polinomial para reconhecer esta classe. No Capítulo 3 generalizamos o conceito de grafos-(k, l), mais especificamente, apresentamos alguns resultados para os problemas de M-partição e M-partição com listas para grafos cordais. No Capítulo 4 apresentamos a versão com pesos

do problema apresentado no Capítulo 2 e investigamos maneiras de como a dualidade da programação linear pode ser utilizada como ferramenta na solução desses problemas. Finalmente, no Capítulo 5, fornecemos as conclusões deste trabalho e algumas propostas para continuidade do trabalho. Pode-se ainda encontrar neste trabalho (Apêndice A) uma pequena abordagem sobre decomposições em árvore que serve como base para parte dos resultados apresentados no Capítulo 3.

Vamos agora estabelecer algumas definições e notações que serão utilizadas ao longo da tese.

1.2 Grafos

Um *grafo* é um par ordenado $G = (V, E)$ onde V é um conjunto finito não-vazio de *vértices* e E é um conjunto de pares não-ordenados de vértices distintos, chamados *arestas*. Denotamos os conjuntos de vértices e arestas de G por $V(G)$ e $E(G)$, respectivamente. Utilizaremos a notação $n = |V(G)|$ e $m = |E(G)|$ para denotar a cardinalidade de $V(G)$ e $E(G)$, respectivamente. Embora arestas sejam pares não-ordenados, estas serão indicadas como pares ordenados: (v, w) em vez de $\{v, w\}$.

Um grafo G é dito *trivial* se $|V(G)| = 1$.

Um vértice v é *adjacente* a outro vértice w em G se $(v, w) \in E(G)$. Neste caso, dizemos também que v e w são *vizinhos* ou estão *ligados* em G,

e que a aresta $e = (v, w)$ é *incidente* a v e w, ou que tem *extremos* v e w. Denotamos por $N(v)$ o conjunto dos vértices adjacentes a v em G, e por $N[v]$ o conjunto $N(v) \cup \{v\}$; tais conjuntos são chamados *vizinhança aberta* e *vizinhança fechada* de v, respectivamente. Um vértice v é dito *universal* quando $N(v) = V(G) - \{v\}$.

O *grau* de um vértice v é a cardinalidade do conjunto $N(v)$.

O *tamanho* de um grafo G é igual a $|V(G)| + |E(G)|$.

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

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

Dois grafos G e H são *isomorfos* quando existe uma bijeção entre $V(G)$ e $V(H)$ que preserva adjacências e não-adjacências. Nesse caso, escrevemos $G \simeq H$.

Um grafo G é dito *bipartido* quando seu conjunto de vértices pode ser particionado em dois subconjuntos V_1, V_2, tais que toda aresta de G une um vértice de V_1 a outro de V_2. Um tal grafo é dito *bipartido completo* se

possui uma aresta para cada par de vértices v_1, v_2, sendo $v_1 \in V_1$ e $v_2 \in V_2$. Denotamos por K_{n_1,n_2} o grafo bipartido completo, onde $|V_1| = n_1$ e $|V_2| = n_2$.

Um *caminho* num grafo G é uma sequência $P = v_1, v_2, \ldots, v_k$ onde os v_i's são vértices (dois a dois distintos), e $(v_i, v_{i+1}) \in E(G)$, $1 \leq i \leq k-1$. Uma *corda* em P é uma aresta que liga dois vértices não-consecutivos de P. Um *caminho induzido* é um caminho sem cordas. Denotamos por P_k o caminho induzido por k vértices.

Um *ciclo* num grafo G é uma sequência $C = v_1, v_2, \ldots, v_k v_{k+1}$ onde $v_1, v_2, \ldots, v_k$ é um caminho, $v_1 = v_{k+1}$ e $k \geq 3$. O número k é o *comprimento* de C. Um ciclo é dito *par* se tem comprimento par, caso contrário é dito *ímpar*. Uma *corda* em C é qualquer corda do caminho $v_1, v_2, \ldots, v_k$. Um ciclo é um *ciclo induzido* se não possui cordas. Denotamos por C_k o ciclo induzido por k vértices. O grafo C_3 é também chamado *triângulo*.

Um conjunto S é *maximal* (*minimal*) em relação a uma determinada propriedade P se S satisfaz P, e todo conjunto S' que contém propriamente S (que está contido propriamente em S) não satisfaz P.

Um grafo G é *conexo* se para todo par de vértices distintos v e w de G existe um caminho de v a w. Caso contrário, G é dito *desconexo*. Um *componente conexo* de G é um subgrafo maximal conexo de G. Se H_1 e H_2 são componentes conexos distintos de um grafo desconexo G, dizemos que os grafos formados por H_1 e H_2 são *isolados* um do outro. Em princípio, todos

os grafos desta tese são conexos, a menos de menção explícita em contrário quando for o caso.

Uma *árvore geradora* de um grafo G é um subgrafo de G onde existe exatamente um caminho entre cada par de vértices em $V(G)$.

Um grafo G é *completo* se quaisquer dois vértices distintos de G são adjacentes. Denotamos por K_n o grafo completo com n vértices. Denotaremos pK_n o grafo correspondente a p cópias disjuntas e não-adjacentes de K_n.

Dado um grafo G, um *corte* de vértices de G é um subconjunto minimal de vértices $S \subseteq V(G)$ cuja remoção de G o desconecta ou o transforma num grafo trivial. Isto é, $G - S$ é desconexo ou trivial, e para todo subconjunto próprio $S' \subseteq S$, $G - S'$ é conexo e não-trivial.

Um conjunto de vértices M de um grafo G é uma *clique* se $G[M]$ é um grafo completo. Uma clique M é *maximal* se para qualquer clique M' tal que $M \subseteq M'$, temos $M = M'$. Duas cliques M e M' são *disjuntas* se $M \cap M' = \emptyset$. Se $M \cap M' \neq \emptyset$, diremos que M *intercepta* M', e vice-versa. Se uma aresta $e = (v, w)$ é tal que v e w pertencem a uma clique M, dizemos que e *está contida* em M. Um parâmetro bastante estudado de um grafo G é o seu *tamanho de clique máxima* $\omega(G)$, definido como a cardinalidade de uma clique de tamanho máximo em G.

Um conjunto de vértices I de um grafo G é um *conjunto independente* ou um *conjunto estável* se $G[I]$ é um grafo sem arestas, o qual chamamos *grafo*

nulo. Um conjunto independente I é *maximal* se para qualquer conjunto independente I' tal que $I \subseteq I'$, temos $I = I'$. Outro parâmetro bastante estudado de um grafo G é o seu *número de estabilidade* $\alpha(G)$, definido como a cardinalidade de um conjunto independente de tamanho máximo em G.

Uma *coloração* de um grafo G é uma partição de $V(G)$ onde cada classe da partição é um conjunto independente. Uma *k-coloração* é uma partição de $V(G)$ em k classes. O *número cromático* de G, denotado por $\chi(G)$, é o menor k para o qual existe uma k-coloração de G. Neste caso, dizemos que o grafo G é *k-cromático* ou *k-colorível.*

Um grafo G é dito *perfeito* se G e cada um de seus subgrafos induzidos tiver a propriedade de que o número cromático χ é igual ao tamanho de sua maior clique ω.

Um grafo G é dito *cordal* se todo ciclo de comprimento estritamente maior do que três possui uma corda (isto é, uma aresta unindo dois vértices não consecutivos). Equivalentemente, G não contém um subgrafo induzido isomorfo a C_n para $n > 3$. Na literatura, os grafos cordais são também chamados grafos *triangulados, de circuito rígido* e *de eliminação perfeita.* Uma propriedade importante dos grafos cordais é que estes são perfeitos. Este resultado foi provado por Berge [5], Hajnal e Surànyi [30].

Um vértice x de G é dito *simplicial* se seu conjunto de adjacência $N(x)$ induz um subgrafo completo de G, isto é, $N(x)$ é uma clique.

Seja $G = (V, E)$ um grafo e seja $\sigma = [v_1, v_2, ..., v_n]$ uma ordenação de seus vértices. Nós dizemos que σ é *um esquema de eliminação perfeita* (ou esquema perfeito) se cada v_i for um vértice simplicial do subgrafo induzido $G[\{v_i, ..., v_n\}]$. Em outras palavras, cada conjunto $X_i = \{v_j \in N(v_i) | j > i\}$ é completo. O termo *EEP* será usado daqui para frente para abreviar esquema de eliminação perfeita. Além disso, quando escrevemos $v_i < v_j$ queremos dizer que v_i aparece antes de v_j no EEP.

Seja $\mathcal{F} = (S_1, S_2, \ldots, S_n)$ uma família de conjuntos não-vazios. O grafo de interseção de $\mathcal{F}$ é obtido representando cada conjunto $S_i \in \mathcal{F}$ por um vértice i e conectando dois vértices i, j por uma aresta $e = (i, j)$ se e somente se, para seus conjuntos correspondentes $S_i, S_j \in \mathcal{F}$, $S_i \cap S_j \neq \emptyset$.

Seja G um grafo. Se $S, S' \subseteq V(G)$, denotamos por $N_S(S')$ a vizinhança de S' em S, i.e., o conjunto dos vértices de S que estão em S' ou que são adjacentes a algum vértice de S'. Além disso, se $N_S(S') \neq \emptyset$, então dizemos que S e S' são *adjacentes*, caso contrário, dizemos que S e S' são *independentes*.

Escreveremos $N_S(v)$ em vez de $N_S(\{v\})$. Observe que esta *vizinhança* de v em S contém v se $v \in S$.

1.3 Programação Linear

A programação linear visa encontrar a melhor solução para problemas representados por expressões lineares. Isto consiste na maximização ou mi-

nimização de uma função linear, denominada *função objetivo*, respeitando-se um sistema linear de igualdades ou desigualdades que recebem o nome de *restrições* do modelo. Essas restrições do modelo determinam uma região à qual denominamos *conjunto das soluções viáveis*. Se existe um ponto que satisfaz todas as restrições do problema de programação linear, denominamos tal ponto de *solução viável*. Caso contrário, dizemos que o problema de programação linear é *inviável*. A melhor das soluções viáveis, isto é, aquela que maximiza ou minimiza a função objetivo denomina-se *solução ótima*.

O termo *programação* da expressão "programação linear"refere-se ao uso e à alocação eficiente de recursos escassos (como capital, bens e serviços) e não ao conceito usual de programação utilizado em programação de computadores.

O problema de programação linear será chamado daqui por diante de PPL.

Para o desenvolvimento de um método (ou algoritmo), que determine uma solução de um PPL, torna-se necessário a redução do problema a uma forma tal que permita a aplicação direta deste algoritmo. No caso da programação linear, podemos utlizar algoritmos como o *simplex* e o *método dos pontos interiores*.

Um PPL é um problema que pode ser expresso da seguinte forma:

$$Maximizar \quad z = \sum_{j=1}^{n} c_j x_j \tag{1.1}$$

$$sujeito\ a: \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, ...m \tag{1.2}$$

$$x_j \geq 0, \qquad j = 1, 2, ..n \tag{1.3}$$

Caso o algoritmo utilizado na resolução do modelo seja o simplex, é conveniente alterar o modelo para o formato padrão, onde c_j, a_{ij} e b_i são dados (números reais) e x_j representa uma variável de decisão, para $j = 1, 2, ..n$. A função linear a ser maximizada em 1.1 é denominada *função objetivo*. As restrições 1.3 são conhecidas como *restrições de positividade*.

Cada restrição i de 1.2 pode ser substituída por uma igualdade acrescentando-se uma *variável de folga* $x_{n+i} \geq 0$. Ou seja:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \equiv \begin{cases} \sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i, \\ x_{n+i} \geq 0 \end{cases}$$

Sendo dado um PPL com restrições de igualdades e desigualdades, poderemos acrescentar variáveis de folga às desigualdades não-triviais, passando dessa maneira a trabalhar com restrições de igualdades e desigualdades triviais. Isto é, as desigualdades podem ser transformadas em igualdades através da introdução de variáves de folga. Temos então:

$$(PPL)\ Maximizar \quad z = \sum_{j=1}^{n} c_j x_j \tag{1.4}$$

$$sujeito\ a: \quad \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, 2, ...m \tag{1.5}$$

$$x_j \geq 0, \qquad j = 1, 2, ..n. \tag{1.6}$$

que poderá ser ainda apresentado sob a forma abaixo:

$$(PPL)\ Maximizar \quad z = c^T x \tag{1.7}$$

$$sujeito\ a: \quad Ax = b \tag{1.8}$$

$$x \geq 0 \tag{1.9}$$

Onde $c^T = (c_1 c_2 \ldots c_n)$, $x^T = (x_1 x_2 \ldots x_n)$, $b^T = (b_1 b_2 \ldots b_m)$ e $A = (a_1 a_2 \ldots a_n)$, isto é, $c \in R^n$, $x \in R^n$, $b \in R^m$ e $a_j \in R^m$ e $A \in R^{m \times n}$.

Este é o *formato padrão* da programação linear: qualquer problema de programação linear pode ser formulado como acima.

1.3.1 Dualidade

Os problemas de programação linear estão casados em pares: a cada problema corresponde um problema *dual* (se D é o dual de P, P é o dual de D). Um problema e o seu dual estão intimamente relacionados: ou ambos são inviáveis, ou um é inviável e o outro é ilimitado, ou ambos têm solução. Esse fato é a essência do teorema da dualidade.

Considerando o PPL (1.1), associaremos a cada restrição de (1.2) a variável $u_i \geq 0, i = 1, 2, \ldots n$. Segue a formulação do dual:

$$(D)\ Minimizar \quad d = \sum_{i=1}^{m} b_i u_i \tag{1.10}$$

$$sujeito\ a: \quad \sum_{i=1}^{m} a_{ij} u_i \geq c_j, \quad j = 1, 2, ..., n \tag{1.11}$$

$$u_i \geq 0, \qquad i = 1, 2, ..., m. \tag{1.12}$$

Um problema de programação linear, onde algumas ou todas as suas variáveis estão restritas a valores inteiros, denominamos de *problema de programação linear inteira* ($PPLI$). No caso em que algumas variáveis são inteiras mas não todas, podemos chamar este problema de *problema de programação inteira mista* ($PPIM$).

Um $PPLI$ pode ser formulado da seguinte maneira:

$$(PPLI)\ Maximizar \quad z = c^T x \tag{1.13}$$

$$sujeito\ a: \quad Ax = b \tag{1.14}$$

$$x \geq 0 \tag{1.15}$$

$$x \text{ inteiro} \tag{1.16}$$

Onde $c \in R^n$, $x \in R^n$, $b \in R^m$, $a_j \in R^m$ e $A \in R^{m \times n}$.

Quando as variáveis estão restritas aos valores inteiros 0 ou 1 em um problema de programação linear, dizemos que temos um *problema de programação linear binário.*

Os problemas primal e dual têm certas propriedades interessantes. Mostramos a seguir, dois teoremas importantes em programação linear que ilustram a relação entre os problemas primal e dual.

Teorema 1.2 *Toda solução viável x para (P) e toda solução viável u para (D) satisfazem a desigualdade $c^T x \leq b^T u$.*

A partir do Teorema 1.2 podemos fazer algumas afirmações. No caso de o problema (P) não ter solução ótima finita, temos que (D) é vazio. No caso de (D) não possuir solução ótima finita, (P) é vazio. Podemos dizer, portanto, que se (P) e (D) são não-vazios, ambos possuem soluções ótimas finitas. Veremos que, neste caso, vale o seguinte resultado, geralmente citado como o teorema de dualidade forte:

Teorema 1.3 *Se x^* é solução ótima de (P), então existe u^* solução ótima de (D) e $c^T x^* = b^T u^*$.*

Os Teoremas 1.2 e 1.3 são teoremas básicos em programação linear e estão provados nos textos clássicos, como Bazaraa *et al.* [4], Chvátal [15] e Dantzig [17].

Existe uma associação direta entre as variáveis do problema primal e as restrições do problema dual, assim como as variáveis do problema dual estão diretamente associadas às restrições do problema primal. Como se

pode perceber nas formulações de (P) e (D), o lado das restrições de um dos problemas é o vetor de coeficientes da função objetivo do outro problema e vice-versa.

Os teoremas 1.2 e 1.3 implicam vários resultados fundamentais relacionados com os problemas primal e dual. O seguinte princípio de *complementaridade de folga*, que é outra maneira de relacionar os dois problemas, torna essas relações mais explícitas.

Propriedade 1.1 (Princípio da Complementaridade de Folga) *Dizemos que o par (x, u) das soluções viáveis primal e dual satisfaz o princípio da complementaridade de folga se*

$$u_j[\sum_{i=1}^{n} a_{ij} - b_i] = 0 \quad \forall j = 1, 2, \ldots, m \tag{1.17}$$

$$x_i[c_i - \sum_{i=1}^{m} a_{ij}u_j] = 0 \quad \forall i = 1, 2, \ldots, n. \tag{1.18}$$

O princípio da complementaridade de folga afirma que para toda restrição primal ou dual, o produto da folga na restrição e de sua variável primal ou dual associada é zero. Em outras palavras, se uma restrição tem uma folga positiva, a variável primal ou dual associada deve ter valor zero, da mesma forma, se uma variável primal ou dual tem um valor positivo, a solução dual deve satisfazer as igualdades das restrições correspondentes.

Teorema 1.4 *[17]* (Condições de Otimalidade da Complementaridade de Folga) *Uma solução primal viável x e uma solução dual viável u são soluções ótimas dos problemas primal e dual se e somente se satisfazem o princípio da complementaridade de folga.*

1.4 Classes de complexidade

Um problema algorítmico π consiste de um conjunto D de todas as possíveis entradas para o problema, chamado *conjunto de instâncias*, e de uma *questão* Q sobre estas instâncias. Resolver um problema algorítmico é desenvolver um algoritmo cuja entrada é uma instância do problema e cuja saída é uma resposta à questão do problema.

Um problema é dito de *decisão* quando a questão exige uma resposta do tipo SIM ou NÃO. Como exemplo, seja π o seguinte problema: "Dado um grafo G, reconhecer se G é um grafo cordal."O conjunto de instâncias de π é obviamente o conjunto de todos os grafos. O problema π pode ser assim esquematizado:

Instância genérica de π: um grafo G.

Questão: G é cordal?

Fica evidente que o problema π acima é um problema de decisão, em particular, um *problema de reconhecimento.* Resolver π significa elaborar um algoritmo de reconhecimento de grafos cordais.

Dizemos que um algoritmo é *polinomial* quando sua complexidade de tempo (medida do número de passos que o algoritmo efetua) é uma função polinomial no tamanho da sua entrada. Os problemas de decisão para os quais existem algoritmos polinomiais constituem a classe P. Tais problemas são chamados *polinomiais.*

Um problema de decisão é *não-determinístico polinomial* quando qualquer instância que produz resposta SIM possui um *certificado* sucinto, isto é, uma comprovação de que a resposta é de fato SIM verificável em tempo polinomial no tamanho da instância. Esta classe de problemas de decisão é a classe NP.

A classe Co-NP é formada pelos problemas que possuem um certificado sucinto para as instâncias que produzem resposta NÃO.

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

- f pode ser computada em tempo polinomial;
- para toda instância $I \in D_1$, tem-se que I produz resposta SIM para π_1 se e somente se $f(I)$ produz resposta SIM para π_2.

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

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

Um problema pertencente à classe *NP-completo* é chamado *NP-completo*. Para provar que um certo problema π é NP-completo, basta mostrar que $\pi \in$ NP e que existe uma transformação de um problema NP-completo π' em π.

Analogamente, prova-se que um problema de decisão π pertence à classe *Co-NP-completo* (e, neste caso, π é dito *Co-NP-completo*) quando $\pi \in$ Co-NP e existe um problema π' (Co-)NP-completo tal que:

- se π' é NP-completo, existe uma função f computável em tempo polinomial tal que para toda instância I' de π', tem-se que I' produz SIM para π' se e somente se $I = f(I')$ produz NÃO para π;
- se π' é Co-NP-completo, existe uma função f computável em tempo polinomial tal que para toda instância I' de π', tem-se que I' produz NÃO para π' se e somente se $I = f(I')$ produz NÃO para π.

Como fonte de referências para esta seção, indicamos [28].

Capítulo 2

Grafos Cordais-(k, l)

Muitos problemas combinatórios consistem em encontrar uma partição dos vértices de um dado grafo em subconjuntos $S_1, \ldots, S_k$ que satisfazem certas propriedades. Pode-se exigir, por exemplo, que cada S_i seja uma clique ou um conjunto independente. Pode-se ainda exigir que pares S_i, S_j sejam completamente adjacentes ou não-adjacentes.

Na Seção 2.1 definimos uma classe de grafos que se insere nesse contexto, os *grafos-*(k, l), e apresentamos alguns resultados presentes na literatura sobre esta classe. Na Seção 2.2 apresentamos uma caracterização dos grafos cordais-(k, l), i.e., uma subclasse dos grafos cordais. Na Seção 2.3 apresentamos um algoritmo polinomial para reconhecer esta classe, assim como algoritmos para reconhecer subclasses, como por exemplo, reconhecer grafos cordais-(k, l) quando $k = 1$. Fornecemos também um novo algoritmo para reconhecer grafos split, que implica a famosa caracterização de grafos split [29].

2.1 Grafos-(k, l)

Definição 2.1 *Um grafo G é dito* grafo-(k, l), *ou equivalentemente, G admite uma* (k, l)-partição, *quando seu conjunto de vértices pode ser particionado em k conjuntos independentes e l cliques.*

Pela definição acima, podemos ver que os grafos-(k, l) representam uma generalização dos grafos split [29], que correspondem aos grafos-$(1, 1)$. Grafos split podem ser eficientemente reconhecidos [29]. Além disso, os problemas clássicos de otimização combinatória são também resolvidos eficientemente nessa classe. Quando $k \geq 1$ ou $l \geq 1$, os grafos-(k, l) não são em geral cordais. Em [11, 12], algoritmos polinomiais de reconhecimento para as classes $(2, 1)$, $(1, 2)$ e $(2, 2)$ foram propostos. Feder *et al.* [25] também propuseram algoritmos polinomiais de reconhecimento para essas classes, que surgiram como sub-produto de algoritmos de partição em subgrafos densos e esparsos. Além disso, Hoang e Le [32] mostraram que essas classes de grafos satisfazem o teorema Forte dos Grafos Perfeitos. Por outro lado, sabe-se que reconhecer grafos-(k, l) para $k \geq 3$ ou $l \geq 3$ é NP-completo [11]. A prova consiste de uma redução de grafos-$(k, 0)$ para grafos-(k, l). (A classe dos grafos-$(k, 0)$ corresponde ao problema de reconhecer se um dado grafo é k-colorível.)

2.2 Grafos Cordais$-(k, l)$

Como foi mencionado anteriormente, reconhecer grafos-(k, l), para $k \geq 3$ ou $l \geq 3$ é NP-completo. Por isso, propomos resolver o problema restrito à classe dos grafos cordais e caracterizar os grafos cordais-(k, l). Assim sendo, esta seção se destina a caracterizar os grafos cordais-(k, l). Tal caracterização é dada em termos de subgrafos induzidos.

2.2.1 Caracterizando Grafos Cordais-(k, l)

Com o objetivo de caracterizar os grafos cordais-(k, l), precisamos dos seguintes lemas:

Lema 2.1 *Sejam C e C' duas cliques num grafo cordal G. Então algum vértice de C' é adjacente a todos os vértices de $N_C(C')$.*

Prova: Vamos provar que, na verdade, os vizinhos dos vértices de C' em C são linearmente ordenados por inclusão. Suponha que dois vértices distintos $v_1, v_2 \in C'$ têm vizinhos incomparáveis em C, i.e., nenhum dos conjuntos $N_C(v_1)$, $N_C(v_2)$ contém o outro. Então existem dois vértices distintos $u_1, u_2 \in C$ tal que u_1 é adjacente a v_1 mas não o é a v_2, e u_2 é adjacente a v_2 mas não o é a v_1. Isto é impossível, já que u_1, u_2, v_2, v_1 induziriam um ciclo de tamanho 4 sem corda, como mostra a Figura 2.1. Para concluir o lema basta considerarmos o vértice $v \in C'$ com $|N_C(v)|$ máximo. □

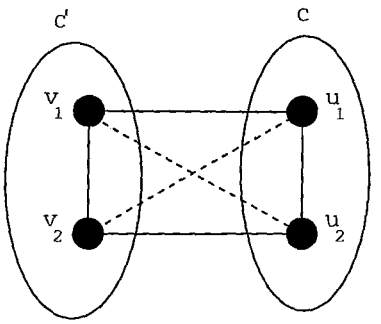


Figura 2.1: Exemplo para o Lema 1

Lema 2.2 *Sejam C e K duas cliques disjuntas de um grafo cordal G. Então existe uma clique C' com a seguinte propriedade: C' intercepta K, e também intercepta todas as cliques adjacentes a K que são interceptadas por C.*

Prova: Seja $L = N_C(K)$. Pelo Lema 2.1 algum vértice de K é adjacente a todos os vértices de L, e portanto pode ser adicionado a L para obtermos uma clique que intercepta K, assim como todas as cliques de G interceptadas por L. Considere agora uma clique K' de G que intercepta C mas é disjunta de L. Segue da definição de L que tal clique não intercepta K. Precisamos considerar tal K' se ela contiver um vértice a adjacente a algum vértice de K. Denote por A o conjunto de tais vértices, i.e., vértices que são adjacentes a K e pertencem a alguma clique que intercepta C mas não L. Afirmamos que cada $a \in A$ é adjacente a todos os vértices de L. De fato, se $b \in L$ não é adjacente a a, então existem vértices $c \in C \setminus L$, e $s, t \in K$ (possivelmente $s = t$) tais que b, c, a, s, t, b é um ciclo sem cordas. Similarmente, provamos que A é uma clique, i.e., para quaisquer $a, a' \in A$, a e a' são adjacentes em G. Caso contrário, existem vértices $c, c' \in C \setminus L$ (possivelmente $c = c'$), e

$s, s' \in K$ (possivelmente $s = s'$), tais que $a, c, c', a', s', s'a$ é um ciclo sem cordas. Logo, o conjunto $L \cup A$ induz uma clique, e o Lema ?? garante que existe um vértice $u \in K$ adjacente a todos os vértices de $L \cup A$. Agora a clique induzida por $L \cup A \cup \{u\}$ intercepta K, e por definição de A, intercepta também todas as cliques interceptadas por C que são adjacentes a K. □

Observe que o lema acima também é válido quando C e K não são disjuntas (basta tomar $C' = C$).

Lema 2.3 *Seja $\{C_1, C_2, \ldots, C_p\}$ uma coleção de cliques duas a duas adjacentes num grafo cordal G. Então existe uma clique C em G que intercepta cada $C_i, i = 1, 2, \ldots, p$.*

Prova: O resultado segue facilmente quando $p \leq 2$. Assuma agora que $p > 2$. Por indução, existe uma clique C que intercepta C_i para cada $i \in \{1, \ldots, p-1\}$. Se C intercepta C_p, nada resta a provar. Caso contrário, aplique o Lema 2.2 a C e C_p. □

Uma condição simples e necessária para que um grafo G seja um grafo-(k, l) é que ele não contenha $l+1$ cópias independentes de K_{k+1} (veja a Figura 2.2). De fato, considere qualquer partição de G em k conjuntos independentes e l cliques. Qualquer K_{k+1} em G teria de conter um vértice de uma das cliques na partição, e portanto, dentre as $(l+1)$ cópias dos K_{k+1} duas delas deveriam

interceptar a mesma clique, e logo possuiriam uma aresta ligando-as (o que significa que não eram independentes). Verifica-se que para grafos cordais a condição acima é também suficiente. (Observe que a condição diz apenas que $(l+1)K_{k+1}$ não é um subgrafo induzido de G). Veremos mais adiante que esta condição pode ser testada em tempo polinomial.

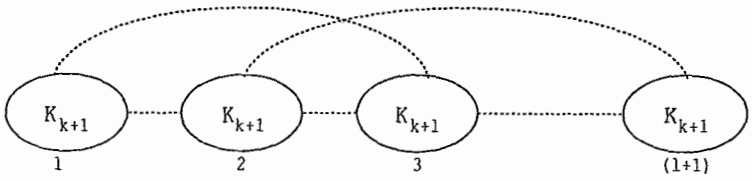


Figura 2.2: Subgrafo isomorfo a $(l+1)K_{k+1}$

Denotemos por $\alpha(G,r)$ o número máximo de $K_r's$ independentes num grafo G e por $\kappa(G,r)$ o número mínimo de cliques que interceptam todos os $K_r's$ de G. Vamos mostrar que para um grafo cordal G, $\alpha(G,r) = \kappa(G,r)$. Este fato nos fornece uma forma alternativa de enunciar nosso resultado principal. Observe que quando $r = 1$, $\alpha(G,r)$ é o número de independência de G, e $\kappa(G,r)$ é o número de cobertura por cliques de G. Nosso algoritmo de tempo $O(n(m+n))$, onde $n = |V(G)|$ e $m = |E(G)|$ identifica $\alpha(G,r)$ cópias independentes de $K_r's$ e o mesmo número de cliques que interceptam todos os $K_r's$.

Teorema 2.1 *Seja G um grafo cordal, e seja $r \geq 1$ um inteiro. Então $\alpha(G,r) = \kappa(G,r)$.*

É claro que $\alpha(G,r) \leq \kappa(G,r)$, para qualquer G e qualquer $r \geq 1$. Para provarmos a igualdade para grafos cordais, vamos proceder como segue:

Seja G um grafo. Vamos definir $\mathcal{K}^r(G)$ como o grafo com um vértice correspondendo a cada K_r em G, sendo que dois vértices são adjacentes em $\mathcal{K}^r(G)$ se e somente se os K_r's correspondentes são adjacentes em G. Veja um exemplo na Figura 2.3.

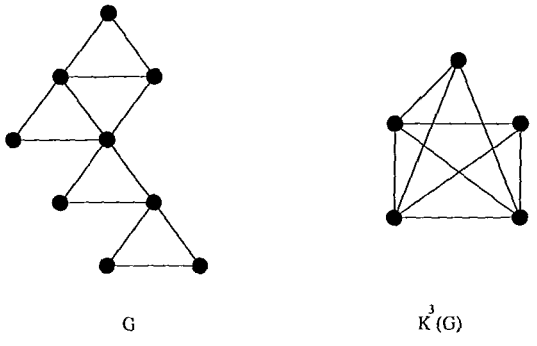


Figura 2.3: Exemplo de $\mathcal{K}^3(G)$

Lema 2.4 *Para qualquer grafo G, $\alpha(G,r)$ é o número de independência de $\mathcal{K}^r(G)$. Para um grafo cordal G, $\kappa(G,r)$ é o número de cobertura por cliques de $\mathcal{K}^r(G)$.*

Prova: A primeira asserção segue das definições de $\alpha(G,r)$ e $\mathcal{K}^r(G)$. A segunda segue da observação de que podemos transformar qualquer cobertura por cliques $\mathcal{C}$ de $\mathcal{K}^r(G)$ em um conjunto com $|\mathcal{C}|$ cliques em G que interceptam todos os K_r's de G. Para isto, basta aplicar em G o Lema 2.3 a cada coleção de cliques de G correspondente a uma clique em $\mathcal{C}$. □

Lema 2.5 *Se G é cordal então $\mathcal{K}^r(G)$ também o é.*

Prova: Assuma que $W_1, W_2, \ldots, W_q, W_1$ $(q \geq 4)$ é um ciclo sem cordas em $\mathcal{K}^r(G)$. Isto significa que W_i e W_j são consecutivos no ciclo se e somente se os K_r's correspondentes em G são adjacentes. Considere uma sequência S de vértices de G, $S = (u_1, w_1, u_2, w_2, \ldots, u_q, w_q)$ tal que u_i, $w_i \in W_i$, u_i é adjacente ao K_r correspondente a W_{i-1}, e w_i é adjacente ao K_r correspondente a W_{i+1}, para todo $i \in \{1, \ldots, q\}$ (índices são tomados circularmente no intervalo $1 \ldots q$). É claro que se i e j são índices não-consecutivos, então os subconjuntos $\{u_i, w_i\}$ e $\{u_j, w_j\}$ são não-adjacentes. Ocasionalmente, pode ocorrer $u_i = w_i$ ou $w_i = u_{i+1}$ para algum i, mas estas igualdades não podem ocorrer simultaneamente. Isto significa que todo vértice ocorrendo em S aparece no máximo duas vezes, e duas ocorrências de um mesmo vértice usam necessariamente posições consecutivas em S. Estas observações mostram que podemos construir um ciclo C_0 em G a partir de S removendo ocorrências repetidas de vértices. Esta construção assegura que pelo menos um vértice de $\{u_i, w_i\}$ é escolhido, para todo $i \in \{1, \ldots, q\}$. Logo, C_0 contém pelo menos 4 vértices. Além disso, C_0 é claramente um ciclo sem cordas, uma contradição.

□

O Teorema 2.1 segue naturalmente dos Lemas 2.4 e 2.5.

Prova do Teorema 2.1: Pelo Lema 2.5 $\mathcal{K}^r(G)$ é cordal, e logo perfeito. Portanto o número de independência de $\mathcal{K}^r(G)$ é igual ao número de cobertura por cliques. O Lema 2.4 completa a prova. □

A caracterização de grafos cordais-(k, l) por subgrafos proibidos segue como uma consequência do teorema anterior.

Teorema 2.2 *Um grafo cordal é um grafo-(k, l) se e somente se não contém $(l+1)K_{k+1}$ como um subgrafo induzido.*

Prova: Mostramos que um grafo chordal-(k, l) não pode conter $l+1$ cópias independentes de K_{k+1}, i.e., não pode conter $(l+1)K_{k+1}$ como um subgrafo induzido. Por outro lado, o Teorema 2.1 implica que se um grafo cordal G não contém $l+1$ cópias independentes de K_{k+1}, então $\kappa(G, k+1) \leq l$. Isto significa que G contém l cliques cuja remoção resulta em um subgrafo G' sem K_{k+1}. Como G é perfeito, G' é k-colorível, portanto G admite uma partição em k conjuntos independentes e em l cliques. □

2.2.2 Algoritmo para reconhecer Grafos Cordais-(k, l)

Para k e l fixos, existe um número polinomial de subgrafos de G com $(l+1)(k+1)$ vértices, e portanto o Teorema 2.2 nos fornece um algoritmo polinomial para reconhecer grafos cordais-(k, l). Existem, contudo, algoritmos mais eficientes. O algoritmo que apresentamos aqui possui complexidade $O(n(m+n))$ e nos fornece uma segunda prova para o Teorema 2.1.

Inicialmente revisamos o algoritmo padrão de coloração para grafos cordais, denominado *Algoritmo Guloso Reverso*. (Observe que testar a existên-

cia de uma k-coloração é equivalente a reconhecer grafos-$(k,0)$). Suponha que os vértices de G obedecem a um esquema de eliminação perfeita (EEP) $[1,2,\ldots,n]$. Observamos que encontrar um EEP pode ser feito em tempo polinomial [29]. O *Algoritmo Guloso Reverso* procede na ordem $n, n-1, \ldots, 1$, associando a cada vértice a menor cor disponível. Em outras palavras, para colorir G com as cores $s_1, s_2, \ldots,$ colorimos o vértice n com a cor s_1, e tendo colorido os vértices $n, n-1, \ldots, i+1$, colorimos o vértice i com a cor s_d onde d é o menor índice tal que nenhum vizinho de i dentre $i+1, i+2, \ldots, n$ foi colorido com a cor s_d. Observe que até este ponto, i faz parte de um K_d, já que i tem um vizinho de cada uma das cores $s_1, s_2, \ldots, s_{d-1}$, que são mutuamente adjacentes. (Quaisquer dois vizinhos de i dentre $i+1, i+2, \ldots, n$ são adjacentes, já que $[1,2,\ldots,n]$ é um EEP). Podemos concluir, então, que dado um EEP, o Algoritmo Guloso Reverso fornece, em tempo $O(m+n)$, uma coloração mínima e uma clique máxima.

Estamos prontos para descrever nosso algoritmo. Seja $k \geq 0$ um inteiro. O algoritmo encontra o menor valor de l (possivelmente $l = 0$) para o qual G é um grafo-(k,l). É importante ressaltar aqui que estaremos usando o termo *colorir* não no sentido usual, mas para especificar em que partição um dado vértice foi alocado. Desejamos particionar o conjunto dos vértices $\{1,2,\ldots,n\}$ de um grafo cordal de entrada G em k conjuntos independentes, $s_1, s_2, \ldots, s_k$ e em l cliques, $c_1, c_2, \ldots, c_l$. Assim, por exemplo, se dois vértices, a e b forem "coloridos"s_i, $i = 1, \ldots, k$, isto significa que $(a,b) \notin E(G)$, uma

vez que s_i representa um dos k conjuntos independentes. Da mesma forma, se dois vértices c e d forem "coloridos"c_j, $j = 1, \ldots, l$, isto significa que $(a, b) \in E(G)$, já que c_j representa uma das l cliques. Em resumo, através da execução do algoritmo, os vértices coloridos com uma mesma cor s_i formarão um conjunto independente, e os vértices coloridos com uma mesma cor c_j formarão uma clique. Denotaremos por S_i o conjunto que consiste de i, e mais todos os vértices dentre $1, 2, \ldots, i-1$ coloridos com $s_1, s_2, \ldots, s_k$.

O seguinte resultado é facilmente obtido das seguintes definições, usando as propriedades do esquema de eliminação perfeita:

Lema 2.6 *Seja* $[1, 2, \ldots, n]$ *é um EEP de* G. *Se o vértice* i *é adjacente ao primeiro vértice* j *colorido com a cor* c_a, *e* $j < i$, *então* i *é adjacente a todos os vértices* $x < i$ *coloridos com a cor* c_a.

Prova: De fato, seja j o primeiro vértice colorido com a cor c_a, com $i, j \in E(G)$, e $j < i$. Como j é adjacente a todos os vértices k coloridos com a cor c_a, onde $j < k < i$, então, pelo EEP, i é adjacente a todos esses vértices. □

Este Lema nos permitirá facilmente testar se um dado vértice i pode ser adicionado ou não à clique formada pelos vértices coloridos com a cor c_a.

Algoritmo 2.1 *Algoritmo para reconhecer grafos cordais-(k, l)*

Assuma que G é um grafo cordal com um EEP $[1, 2, \ldots, n]$.

- *Atribua ao vértice 1 a cor s_1.*

- *Tendo colorido os vértices $1, 2, \ldots, i-1$ sem usar a cor c_1:*

  - *remova as cores dos vértices $1, 2, \ldots, i-1$ e faça uma coloração dos vértices $1, 2, \ldots, i$ com as cores $s_1, s_2, \ldots, s_k$ (usando o Algoritmo Guloso Reverso), se possível, ou*

  - *mantenha a coloração dos vértices $1, 2, \ldots, i-1$, e atribua ao vértice i a cor c_1.*

- *Tendo colorido os vértices $1, 2, \ldots, i-1$ e tendo usado as cores $c_1, c_2, \ldots, c_a$:*

  - *Atribua ao vértice i a cor c_b, onde $b \leq a$ é o menor índice tal que i é adjacente ao primeiro vértice colorido com a cor c_b, se tal índice existe, ou*

  - *remova as cores dos vértices de $S_i \backslash i$ e faça uma coloração dos vértices em S_i com as cores $s_1, s_2, \ldots, s_k$ (usando o Algoritmo Guloso Reverso), se possível, ou*

  - *mantenha a coloração de $1, 2, \ldots, i-1$, e atribua ao vértice i a cor c_{a+1}.*

No algoritmo acima, l é o maior valor de a tal que existe um vértice colorido c_a, ou $l = 0$ se todos os vértices estão coloridos com as cores $s_1, s_2, \ldots, s_k$.

Como o funcionamento do algoritmo é dominado por no máximo n aplicações do Algoritmo Guloso Reverso, segue que o tempo de execução é $O(n(m+n))$. Para provarmos a corretude do algoritmo precisaremos do seguinte resultado.

Proposição 2.1 *Seja G um grafo cordal. Aplique em G o Algoritmo 2.1. Se o algoritmo usa a cor c_p, então G contém um pK_{k+1} como subgrafo induzido.*

Prova: Seja v_a o primeiro vértice (no EEP) colorido com a cor c_a. O subgrafo de G induzido por S_{v_a-1} é k-colorível, mas nosso algoritmo não conseguiu adicionar v_a de forma que S_{v_a} continuasse k-colorível. Logo existe um subgrafo X_a isomorfo a K_{k+1} contendo v_a e k vértices de S_{v_a-1}. Resta apenas mostrar que os subgrafos $X_1, X_2, \ldots, X_p$ são independentes. Suponha que um vértice x do subgrafo X_a seja adjacente ou igual a um vértice x' de um subgrafo $X_{a'}$. Assuma que $a < a'$.

Se $x' \leq x$, então devido ao fato de que x' é adjacente ou igual a $v_{a'}$, concluímos que x é adjacente ou igual a $v_{a'}$. Agora v_a e $v_{a'}$ devem ser adjacentes, já que $x \leq v_a$ e $x \leq v_{a'}$. Isto significa que $v_{a'}$ é adjacente ao primeiro vértice colorido com a cor c_a, e portanto nosso algoritmo deveria ter colorido $v_{a'}$ com a cor c_a.

Se $x' > x$, então x' é adjacente ou igual a v_a por um argumento similar. Se $x' \leq v_a$ então $v_{a'}$ e v_a devem ser adjacentes, e $v_{a'}$ deveria ter sido colorido com a cor c_a, como no caso anterior. Por outro lado, se $x' > v_a$, então x' é adjacente ao primeiro vértice colorido com a cor c_a. Portanto nosso algoritmo deveria ter colorido x' com a cor c_a, contradizendo o fato dele ter sido colorido com alguma cor c_d (no caso de x ser um elemento de $S_{v_{d'}-1}$) ou com a cor $c_{d'}$ (no caso $x' = v_{d'}$). □

Corolário 2.1 *Se G é um grafo cordal então as seguintes afirmações são equivalentes:*

1 - *O Algoritmo 2.1 particiona G em k conjuntos independentes e l cliques;*

2 - *o grafo G é um grafo-(k, l);*

3 - *o grafo G não contém $(l+1)K_{k+1}$ como subgrafo induzido.*

Prova: As implicações 1 $\Rightarrow$ 2, e 2 $\Rightarrow$ 3 são óbvias, e Proposição 2.1 prova que 3 $\Rightarrow$ 1.

Observe que a equivalência entre 1 e 2 prova a corretude do algoritmo, enquanto que a equivalência entre 1 e 3 nos fornece uma segunda prova do Teorema 2.2. □

Encerramos esta seção observando que o algoritmo encontra, para qualquer k e para qualquer grafo cordal G, o menor valor l tal que G é um grafo-(k, l).

2.2.3 Algoritmo para reconhecer Grafos Cordais-$(1, l)$, enfatizando Grafos Split

Quando $k = 1$, podemos de alguma forma simplificar o algoritmo, já que não precisamos do Algoritmo Guloso Reverso para testar se um vértice pode ser adicionado a um conjunto independente, mantendo a propriedade de independência.

Algoritmo 2.2 *(Algoritmo para reconhecer grafos cordais-$(1, l)$)*

Assuma que G é um grafo cordal com um EEP $[1, 2, \ldots, n]$.

- *Atribua ao vértice 1 a cor s_1,*
- *e continue colorindo os vértices $i = 2, 3, \ldots$ com a cor s_1 enquanto for possível (i não tem aresta para $1, 2, \ldots, i-1$),*
- *e então atribua a cor c_1 ao primeiro j que não pode ser colorido com a cor s_1.*
- *Tendo colorido os vértices $1, 2, \ldots, i-1$ usando as cores $s_1, c_1, c_2, \ldots, c_a$,*
  - *atribua ao vértice i a cor c_b, onde $b \leq a$ é o primeiro índice tal que i é adjacente ao primeiro vértice colorido com a cor c_b, se tal índice existir, ou*
  - *atribua ao vértice i a cor s_1 se i é não-adjacente a todos os vértices coloridos com a cor s_1, ou*
  - *atribua ao vértice i a cor c_{a+1}.*

É claro que este algoritmo pode ser implementado em tempo $O(m+n)$.

A situação é mais simples quando $k = l = 1$. Os grafo-$(1,1)$ são denominados grafos split. Neste caso não precisamos explicitamente assumir cordalidade. (Grafos split são cordais.) Como grafos split são de grande interesse [29, 26], nós alteramos o algoritmo uma vez mais, de forma a reconhecer grafos split:

Algoritmo 2.3 *(Algoritmo para Reconhecer Grafos Split)*

Seja G um grafo qualquer.

- *Encontre um EEP $[1, 2, \ldots, n]$ de G [29].*
- *Atribua ao vértice 1 a cor s, e continue colorindo $i = 2, 3, \ldots$ com a cor s enquanto for possível (i é não-adjacente aos vértices já coloridos), então atribua ao próximo vértice j a cor c.*
- *Se todos os vértices $1, 2, \ldots, i-1$ tiverem sido coloridos, e ambas as cores s e c já tiverem sido usadas, então atribua ao vértice i a cor c se for adjacente ao primeiro vértice j colorido com a cor c; caso contrário, atribua ao vértice i a cor s se for não-adjacente a todos os vértices já coloridos com a cor s.*

Se o algoritmo falhar devido à não-existência de um EEP, então o algoritmo dado em [29] exibe um subgrafo induzido isomorfo a C_4, C_5, ou C_k,

$k \geq 6$. Se ele falhar colorindo todos os vértices, então de acordo com a Proposição 9 e [29], G contém um subgrafo induzido isomorfo a $2K_2$. Apresentamos abaixo um pequena versão da prova, que será usada mais adiante, no Capítulo 4.

Se num certo momento um vértice i não pode ser colorido com a cor s ou com a cor c, então ele é não-adjacente ao primeiro vértice j colorido com a cor c, e é adjacente a algum vértice k que foi anteriormente colorido com a cor s. Afirmamos que j e k não podem ser adjacentes. Se $k < j$, segue das propriedades do EEP. Se $k > j$, então se k fosse adjacente a j o algoritmo teria colorido o vértice k com a cor c.

O vértice j foi colorido com a cor c, porque ele era adjacente a um vértice l previamente colorido com a cor s. Como $l < j < i$, e i, j são não-adjacentes, l deve ser não-adjacente a i. Além disso, os vértices k, l são não-adjacentes, já que eles foram ambos coloridos com a cor s. Portanto, i, j, k, l formam um subgrafo induzido isomorfo a $2K_2$ em G.

Como cada $C_k, k \geq 6$, também contém um subgrafo induzido isomorfo a $2K_2$, obtivemos a seguinte caracterização já conhecida de grafos split [29]:

Corolário 2.2 *Um grafo G é um grafo split se e somente se não contém $2K_2$, C_4 ou C_5 como subgrafos induzidos.*

□

Capítulo 3

Partição

No capítulo anterior apresentamos algoritmos eficientes para particionar grafos cordais em k conjuntos independentes e l cliques. Como mencionado anteriormente, esta é uma generalização natural do problema de reconhecimento de grafos split, e NP-completo para grafos em geral, a menos que $k \leq 2$ e $l \leq 2$. Neste capítulo estendemos o nosso foco e consideramos M-partições gerais para a classe dos grafos cordais. Para cada matriz simétrica M definida sobre $0, 1, *$, o problema da M-partição procura por uma partição do conjunto de vértices de um grafo em conjuntos independentes, cliques, ou conjuntos arbitrários, exigindo-se que alguns pares de conjuntos não tenham arestas entre eles, ou que tenham todas as arestas, tal como codificado na matriz M. Tais partições generalizam colorações em grafos e homomorfismos, e surgem frequentemente na teoria de grafos perfeitos. Mostramos que muitos problemas de M-partição que são NP-completos em geral tornam-se polinomiais quando restritos à classe dos grafos cordais, mesmo na presença

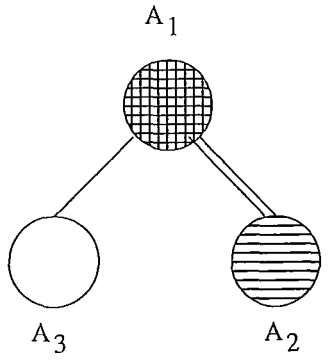
de listas. Por outro lado, mostramos que existem problemas de M-partição que permanecem NP-completos mesmo para grafos cordais.

3.1 M-partição

Em [25] Feder *et al.* formularam uma generalização do problema de grafos-(k,l) como segue: particionar o conjunto de vértices de um dado grafo em m partes $A_1, A_2, \ldots, A_m$ com certas exigências para cada A_i, $i = 1, \ldots, m$, tais como: os vértices em A_i devem ser completamente adjacentes ou completamente não-adjacentes (exigências internas) ou para cada par A_i, A_j, os vértices em A_i e A_j são completamente adjacentes ou não-adjacentes entre si (exigências externas). Com o objetivo de capturar essas exigências de uma maneira bem simples, Feder *et al.* [25] definiram uma matriz simétrica M de ordem $m \times m$ em que cada elemento $M_{i,i}$ representa a restrição interna no conjunto A_i, e um elemento fora da diagonal principal $M_{i,j}$, $i \neq j$, representa uma restrição externa entre A_i e A_j.

Mais especificamente, seja M um matriz simétrica de ordem $m \times m$ onde cada elemento $M_{i,j}$ é igual a $0, 1$, ou $*$. Uma *partição* M de um grafo G, ou uma *M-partição*, é uma partição do conjunto de vértices $V(G)$ em m partes $A_1, A_2, \ldots, A_m$, tal que A_i é estável (isto é, independente) se $M_{i,i} = 0$, completo (isto é, uma clique) se $M_{i,i} = 1$ ou sem nenhuma restrição interna se $M_{i,i} = *$, e tal que A_i e A_j são completamente não-adjacentes se $M_{i,j} = 0$, ou completamente adjacentes se $M_{i,j} = 1$, ou sem restrição externa se $M_{i,j} = *$.

Na Figura 3.1 damos um exemplo de um grafo-$(2,1)$ seguido de sua matriz M. Nesta figura, também utilizamos uma representação simbólica (utilizada em [25]) mostrando uma M-partição geral. O círculo vazio representa um conjunto independente (0 na diagonal principal de M), um círculo cruzado representa uma clique (1 na diagonal principal de M); similarmente, dois conjuntos serão ligados por uma linha dupla se eles são completamente adjacentes (1 fora da diagonal principal de M), ou sem nenhuma linha ligando-os se eles são completamente não-adjacentes (0 fora da diagonal principal de M). Além disso, representamos ausência de restrição interna por um círculo listrado ($*$ na diagonal principal de M), enquanto ausência de restrição externa é representada por uma linha simples unindo dois conjuntos ($*$ fora da diagonal principal de M).



$$M = \begin{pmatrix} 1 & 1 & * \\ 1 & * & 0 \\ * & 0 & 0 \end{pmatrix}$$

Figura 3.1: $(2,1)$-partição

Muitos conceitos conhecidos de grafos podem ser modelados por M-partições. Na Figura 3.2, ilustramos três desses conceitos, desde a noção bem conhecida de coloração de grafos e de grafos split [29], até a idéia mais recente de uma partição clique-cross [21].