Estrutura Gráfica de Matrizes

José Coelho de Pina Junior

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO

PARA OBTENÇÃO DO GRAU DE
MESTRE EM
MATEMÁTICA APLICADA

Área de Concentração: **Ciência da Computação**
Orientador: **Prof. Dr. Arnaldo Mandel**

Durante a elaboração deste trabalho,
o autor recebeu apoio financeiro do CNPq.

— São Paulo, Agosto de 1990 —

Abstract

Our objective in this work is to study the problem of converting a given matrix to an incidence matrix of a graph using elementary row operations and column-scaling, if such a conversion is possible. This problem is a particular case of the more abstract *Matroid Graph Realization* (MGR) problem, which is: given a matroid M, decide whether M is isomorphic to a matroid of a graph and, if such is the case, construct such a graph.

Tutte [1960] gave a polinomial algorithm to solve the MGR problem when M is binary, that is, given by a matrix over GF(2). Bixby & Wagner [1988] designed a faster algorithm based on a particular graph decomposition. Bixby & Cunningham [1980] showed how the MGR problem can be solved in polinomial-time when M is representable over a field, by reducing this problem to the binary case. Finally, Seymour [1981] solved the MGR problem in the general case.

These algorithms are related to the polinomial-time algorithm for testing whether a given matrix is totally unimodular, which is a consequence of Seymour's famous decomposition theorem of regular matroids, in the sense that both rely on a reduction to the binary case.

This work describes all the algorithms mentioned above, some in terms of matroids and others in terms of matrices and graphs.

Agradecimentos

Mal acredito que terminei este trabalho! Claro que não fiz sozinho! Tem muito mais *culpados* na jogada, vamos a eles (preparem-se para uma grande lista):

Arnaldo um super obrigado pela dedicação, paciência e tempo dispendido (desculpe pelo atropelo final);

os amigos do IME que agüentaram as pontas no curso de MAC-115, leram os manuscritos, me suportaram nas horas de desanimo (que não foram poucas!), corrigiram o meu inglês[1], ... em especial Alex, Carlinhos, Carlos H., Cris, Dilma, Edna, Nami, Paulo F., Steve e Yô[2].

Angela, Antonio, Beti, Célia, Fernando e Koba pelo micro, ap., leitura do manuscrito, remédios, ... e amizade;

Lisa por todo o carinho;

e meus pais pela imensa dedicação, apoio e carinho.

A todos os meus mais sinceros agradecimentos.

[1] Esta é obviamente para o Steve.

[2] Será que esqueci alguém? É tanta gente! E tô com pressa!

Conteúdo

Introdução e preliminares

1 Introdução

Problemas que requerem o emprego de programação linear ou a solução de sistema lineares são dados através de uma matriz. Quando a matriz do problema assume formas particulares algoritmos mais especializados podem ser empregados para resolve-lo com grande melhora de desempenho ou economia de memória.

Muitos problemas em grafos podem ser formulados em termos de programação linear, como, por exemplo, problemas de fluxo. Entretanto, devido à estrutura especial desses problemas e às suas relevantes aplicações, um grande número de algoritmos simples e eficientes e de teoremas elegantes têm sido desenvolvidos. Em muitos desses problemas a solução ótima é inteira, fato esse que não ocorre para programação linear em geral.

Já que problemas em grafos podem ser resolvidos eficientemente por algoritmos que exploram a estrutura do problema, e muitos desses problemas podem ser formulados em termos de programação linear, surge naturalmente a pergunta: Quando é que problemas em programação linear, que não são explicitamente problemas em grafos, podem ser convertidos em problemas em grafos?

De maneira geral não sabemos responder a essa pergunta. Entretanto, Iri [1966] foi o primeiro a observar que combinatória (ou matemática discreta) poderia ser uma ferramenta útil para uma tal conversão através do reconhecimento de matrizes que "representam" grafos. Nosso objetivo nesta dissertação foi estudar algoritmos que fazem esse reconhecimento.

No primeiro capítulo veremos várias motivações para o interesse em matrizes que representam grafos, também deixaremos claro o que queremos dizer por "representam" grafos através das definições de matrizes de grafos, de grafos sinalizados e gráficas, além de estudarmos a complexidade computacional do problema de reconhecer tais matrizes.

Um algoritmo muito eficiente para o reconhecimento de matrizes gráficas será descrito

no capítulo 2. A parte principal desse algoritmo utiliza uma técnica de decomposição bastante engenhosa de um grafo e uma estrutura de dados muito simples para armazená-la.

Os algoritmos que estudaremos para reconhecer matrizes gráficas foram originalmente escritos em termos de uma teoria extremamente rica e bela, fundada em trabalhos antológicos de H. Whitney e W. T. Tutte, para resolverem problemas mais abstratos. Essa teoria é um ramo da combinatória que tem sido estudado por mais de cinco decadas e é conhecido pelo nome de *teoria dos matróides*. No capítulo 3 faremos uma introdução rápida aos matróides. A essa altura esperamos que matróides seja algo que flua como uma abstração natural do que é apresentado nos primeiros capítulos. Ainda nesse capítulo estudaremos o problema, um pouco mais abstrato, de reconhecer-se matróides gráficos e veremos um algoritmo para resolve-lo.

Matrizes totalmente unimodulares é uma classe de matrizes para a qual problemas de programação linear possuem solução inteira. No capítulo 4 veremos como matrizes totalmente unimodulares foram caracterizadas em termos de matróides através dos trabalhos notaveis de W. T. Tutte e P. Seymour. Descreveremos ainda um algoritmo polinomial que reconhece unimodularidade total em matrizes. É interessante observarmos que esse algoritmo emprega uma estratégia semelhante à utilizada pelos algoritmos que reconhecem matrizes gráficas. Na última seção desse capítulo estudaremos como o problema de reconhecer submatrizes máximas (no sentido de linhas, colunas ou número de elementos) que satisfaçam uma propriedade Π é $\mathcal{NP}$-difícil para muitas das propriedades "boas" sobre matrizes, como, por exemplo, unimodularidade total.

Durante toda esta dissertação faremos constantemente referências aos excelentes livros de A. Recski [1989], "*Matroid Theory and its Applications*", e A. Schrijver [1986], "*Theory of Linear and Integer Programming*", pois esses, entre outras coisas, contêm uma vasta quantidade de referências originais sobre os respectivos assuntos. Usaremos as siglas MTIA e TLIP para nos referenciarmos aos livros de Recski e Schrijver, respectivamente.

2 Preliminares

Fixaremos aqui a maior parte da notação e definições que não são "muito" padronizadas e que serão empregadas ao longo desta dissertação. Os algoritmos que veremos serão apresentados numa liguagem do tipo ALGOL e usaremos uma porcentagem (%) antes de comentários.

Em teoria da complexidade de problemas será empregada a notação de Garey & Jonhson [1979].

2.1 Notação básica

É claro que usaremos todos os abusos de linguagem que já foram consagrados pelo uso (e talvez mais alguns). Usaremos os símbolos $\cup$ e $\setminus$ para a união e diferença de conjuntos, respectivamente. Entretanto, quando um dos conjuntos for unitário usaremos simplesmente $+$ ao invés de $\cup$ e $-$ ao invés de $\setminus$, respectivamente. Por exemplo, escreveremos $A + b$ ao invés de $A \cup \{b\}$.

É fácil percebermos que com isto estamos fazendo uma economia de sete caracteres no LaTeX.

2.2 Preliminares de álgebra linear

Será assumida uma certa familiaridade com os elementos da álgebra linear, tais como (sub)espaço linear, (in)dependência linear, posto, matriz, matrizes não singulares, etc. Como referência sugerimos Smith [1984] e Hoffman & Kunze [1979].

Durante esta dissertação matrizes serão indexadas por conjuntos cujos elementos não têm, a priori, nenhuma ordem especial. Assim, ao exibirmos explicitamente os elementos de uma matriz numa forma, digamos, tabular, estaremos meramente mostrando uma das diversas maneiras tabulares de exibirmos a mesma matriz. É importante que não façamos confusão entre a matriz e uma de suas formas tabular. Haverão certamente momentos em que desejaremos alguma ordem entre os elementos dos conjuntos indexadores da matriz para exibirmos a mesma numa forma tabular mais "atraente".

Se L e C são conjuntos finitos, uma $L \times C$-matriz é uma matriz com linhas e colunas indexadas por L e C, respectivamente.

Como as motivações para reconhecer-se estruturas em uma matriz vem primordialmente de problemas de programação linear (PL) e matrizes de PLs não possuem linhas ou colunas inteiramente nulas, estaremos sempre supondo que cada linha e cada coluna de uma matriz possui pelo menos um entrada não nula.

Um l-vetor é um vetor de dimensão l. Uma $l \times c$-matriz é uma matriz com l linhas e c colunas, cujas linhas e colunas são indexadas por $\{1,...,l\}$ e $\{1,...,c\}$, respectivamente.

Se A é uma matriz, e x, b, y e c são vetores, então quando usada a notação

$$Ax = b, \quad Ax \leq b, \quad yA = c \tag{1}$$

estaremos implicitamente assumindo compatibilidade de tamanhos entre A, x, b, y, e c. Assim, em (1), se A é uma $n \times m$-matriz então x é um vetor coluna de dimensão m, b é um vetor coluna de dimensão n, y é um vetor linha de dimensão n e c é um, vetor linha de dimensão m.

Analogamente, se c e x são vetores e escrevermos

$$cx \tag{2}$$

então c será um vetor linha e x um vetor coluna, ambos com o mesmo número de componentes. Logo (2) pode ser considerado como o produto interno entre c e x.

Se A é uma matriz e b um vetor coluna, chamaremos $Ax = b$ um *sistema de equações lineares*, e $Ax \leq b$ de um *sistema de inequações lineares*

Um conjunto de vetores que é l.d. e é minimal com relação a essa propriedade será dito um conjunto *l.d. minimal*

A matriz identidade será denotada por I, onde a dimensão estará clara pelo contexto.

O *suporte* de um vetor x, que será denotado por $||x||$, é o conjunto das coordenadas i tal que $x_i \neq 0$.

Uma $n \times m$-matriz é dita de *posto linha completo* (resp., *posto coluna completo*) se posto $A = n$ (resp., posto $A = m$). Denotaremos posto A por $\rho(A)$ ou simplesmente por ρA.

Seja $d = (d_1, ..., d_k)$ um vetor. Será denotado por $diag(d_1, d_2, ..., d_k)$ ou $diag(d)$ a matriz diagonal cuja entrada (i, i) é d_i.

Um vetor ou matriz é dito *racional* (resp., *inteiro*) se todos os seus elementos forem números racionais (resp., inteiros).

Se para cada $L \times C$-matriz A que satisfaz um propriedade Π temos que toda $L' \times C'$-submatriz de A, $L' \subseteq L$ e $C' \subseteq C$, satisfaz Π, após a possível remoção de linhas e colunas inteiramente nulas, então diremos que Π é uma *propriedade hereditária*. Se A não satisfaz uma propriedade hereditária Π, então toda matriz que contenha A como submatriz não satisfaz Π. Se A não satisfaz uma propriedade hereditária Π e toda submatriz própria de A satisfaz Π diremos então que A é uma *obstrução* ou *submatriz proibida* da propriedade Π. Quando não houver perigo de confusão sobre qual é a propriedade em questão diremos simplesmente que A é uma obstrução ou submatriz proibida. De fato, qualquer propriedade sobre matrizes que pode ser caracterizada em termos de submatrizes proibidas é uma propriedade hereditária.

Denotaremos por $GF(2)$ o corpo com dois elementos $\{0, 1\}$.

3 Preliminares de teoria dos grafos

Em teoria dos grafos usaremos em geral a notação de Bondy & Murty [1976] e Bollobás [1979].

Grafos não dirigidos

Um *grafo* G é uma tripla ordenada $(V(G), E(G), \psi_G)$ consistindo de um conjunto não vazio $V(G)$ de *vértices*, um conjunto $E(G)$, disjunto de $V(G)$ de *arestas*, e uma função ψ_G que associa a cada aresta de G um par não ordenado de vértices (não necessariamente distintos) de G. Se e é uma aresta e u e v são vértices tal que $\psi_G(e) = uv$, então é dito que e *conecta* ou *liga* u e v. Os vértices u e v são chamados de *pontas* de e. Algumas vezes escreveremos simplesmente $e = uv$ ao invés de $\psi_G(e) = uv$.

Um grafo G pode ser naturalmente representado através de um diagrama, como o da figura 1, onde os vértices são representados por pequenas bolas e as arestas por linhas ligando essas bolas. Normalmente não faremos distinção alguma entre o diagrama e o grafo.

As pontas de uma aresta são ditas *incidentes* com a aresta e vice-versa. Dois vértices que são incidentes com uma mesma aresta são ditos *adjacentes*.

Uma aresta com pontas iguais é dita um *laço*. Duas arestas com as mesmas pontas são ditas *paralelas*. Um grafo é *simples* se não possui laços ou arestas paralelas. Na figura 1, b

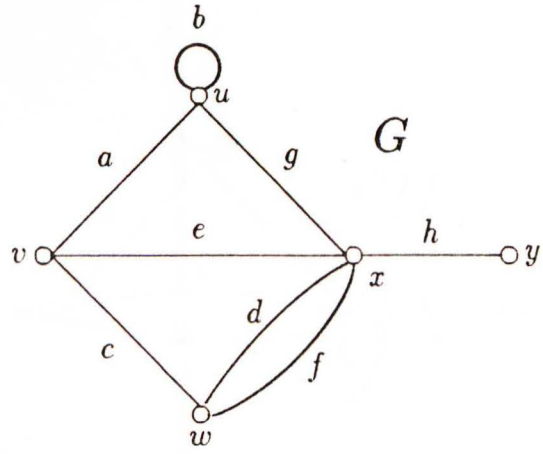

Figura 1: Um diagrama de um grafo G.

é um laço e d e f são arestas paralelas.

Um grafo é dito *bipartido* se os seus vértices podem ser particionados em dois conjuntos disjuntos X e Y, tais que cada aresta tem uma ponta em X e a outra em Y.

Suponhamos que V' é um subconjunto não vazio de V. O subgrafo de G cujo conjunto de vértices é V' e cujo conjunto de arestas são as arestas de G que possuem as duas pontas em V' é chamado de *subgrafo induzido por* V' e denotado por $G[V']$. Dizemos que $G[V']$ é um subgrafo induzido (por vértices) de G. O subgrafo induzido $G[V \setminus V']$, denotado por $G - V'$, é o subgrafo de G obtido através da remoção dos vértices em V' junto com as arestas incidentes a algum vértice de V'. Se $V' = \{v\}$ escreveremos $G - v$ ao invés de $G - \{v\}$.

Agora, suponhamos que E' é um subconjunto não vazio de E. O subgrafo de G cujo conjunto de vértices são as pontas das arestas de E' e o conjunto de arestas é E' é chamado de *subgrafo de* G *induzido por* E' e é denotado por $G[E']$. O subgrafo de G cujo conjunto de vértices é V e o conjunto de arestas é $E \setminus E'$ é denotado simplesmente por $G - E'$. Se $E' = \{e\}$ escreveremos $G - e$ ao invés de $G - \{e\}$.

Seja $G = (V, E)$ um grafo e seja $e = uv$ uma aresta de G. *Contrair a aresta* e de G consiste na operação de remover e de G e depois identificar as pontas de e. O grafo resultante é denotado por G/e. A figura 2 ilustra o efeito da operação de contração da aresta f do grafo da figura 1.

Um *passeio* em um grafo $G = (V, E)$ de v_0 a v_t é uma seqüência da forma

$$(v_0, e_1, v_1, e_2, ..., v_{t-1}, e_t, v_t) \tag{3}$$

onde $v_0, ..., v_t$ são vértices e $e_1, ..., e_t$ são arestas tais que $e_i = v_{i-1}v_i$ para $i = 1, ..., t$. O vértice v_0 e v_t são o *início* e o *fim* do passeio, respectivamente. Os vértices v_0 e v_t também são ditos as *pontas do passeio* e as arestas e_1 e e_t são as *extremidades do passeio*. Dizemos que (3) é um passeio de v_0 a v_t ou um $v_0 - v_t$-passeio que liga ou conecta v_0 e v_t. Na figura 1 $(w, c, v, e, x, d, w, f, x, h, y)$ é um $w - y$-passeio.

O comprimento do passeio (3) é t. A *distância* entre dois vértices u e v é o comprimento

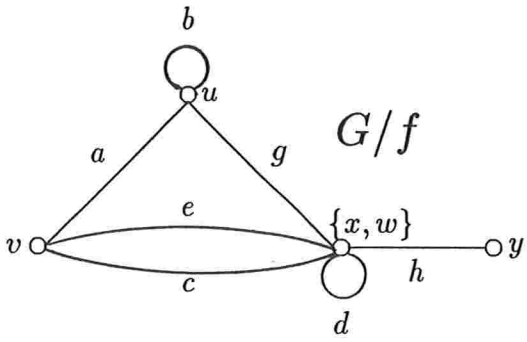

Figura 2: Grafo obtido pela contração da aresta f do grafo G da figura 1.

do menor passeio que liga u e v.

Se o passeio (3) não tem vértices repetidos então ele é chamado de um *caminho.* Na figura 1 (w, c, v, e, x, h, y) é um $w - y$-caminho.

Se $v_0 = v_t$ então o passeio (3) é chamado de *passeio fechado.* Um passeio fechado de comprimento pelo menos um e sem vértices repetidos (exceto o início e o fim) é dito um *circuito.* Na figura 1 (u, b, u), (x, d, w, f, x) e $(v, a, u, g, x, d, w, c, v)$ são exemplos de circuitos. Diremos que um *circuito é impar* (resp. *par*) se ele tiver comprimento ímpar (resp. par).

Um grafo é dito um *triângulo* se for um circuito com exatamente três arestas.

Se $G = (V + c, E)$ é um grafo tal que $| V | \geq 3$, $G[V]$ é um circuito e todo vértice $v \in V$ é adjacente a c, então G é dito um *grafo roda*[1]. Na figura 3 vemos um exemplo de grafo roda.

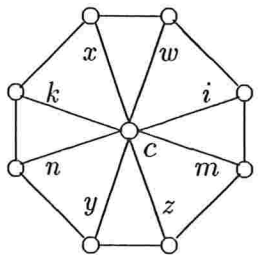

Figura 3: Um grafo roda.

A *estrela* de um vértice é o conjunto de arestas incidentes a esse vértice. Na figura 1 $\{d, e, f, g, h\}$ é a estrela do vértice x e no grafo da figura 3 $\{i, k, m, n, x, w, y, z\}$ é a estrela do vértice c.

Um grafo é *conexo* se quaisquer dois vértices puderem ser ligados por um caminho. Conexidade através de caminhos induz uma relação de equivalência entre vértices. Suas classes

[1] *Wheel graph.*

são chamadas de componentes (conexas) do grafo.

Se $G = (V, E)$ é um grafo e $V' \subseteq V$ então o conjunto das arestas que têm uma ponta em $V \setminus V'$ e a outra em V' é dito um conjunto de arestas de *corte* de G. Os conjunto de arestas $\{a, e, d, f\}$ e $\{a, e, c\}$ são cortes do grafo G da figura 1.

O *grau de um vértice* é um número de arestas incidentes a ele, laços contam como sendo duas arestas. Na figura 2 o vértice y tem grau 1 e o vértice u tem grau 4.

Uma *floresta* é um grafo sem circuitos, e uma *árvore* é uma floresta conexa. Não é difícil ver que para um grafo $G = (V, E)$ são equivalentes:

- G é uma árvore;
- G não contém circuitos e $\mid E \mid = \mid V \mid -1$;
- G é conexo e $\mid E \mid = \mid V \mid -1$;
- existe um único caminho que liga quaisquer dois vértices de G.

(4)

Um vértice de grau 1 de uma floresta é dito uma *folha*

Um subgrafo $T = (V, E')$ de $G = (V, E)$ é dito uma *árvore geradora* de G se T for uma árvore. Um grafo G possui uma árvore geradora se, e somente se, for conexo.

Um *clique* em um grafo G é um subconjunto de vértices de G dois a dois adjacentes.

Um conjunto S de V é chamado um *conjunto independente* de G se os vértices de S são dois a dois não adjacentes. O *problema do conjunto independente máximo* é o problema: dado um grafo $G = (V, E)$ encontrar um conjunto independente de tamanho máximo.

Um *emparelhamento* em um grafo é um conjunto de arestas que não contém nenhum laço e duas a duas as arestas não têm pontas em comum.

A *matriz de incidência* de um grafo $G = (V, E)$, sem laços, é uma $\{0, 1\}$-matriz com linhas e colunas indexadas por V e E, respectivamente, onde a entrada (v, e) é 1 *sse* e é incidente a v.

Grafos dirigidos

Um *grafo dirigido* D é uma tripla ordenada $(V(D), A(D), \psi_D)$ consistindo de um conjunto não vazio de *vértices* $V(D)$, um conjunto $A(D)$, disjunto de $V(D)$, de *arestas* ou *arcos* e uma função de incidência ψ_D que associa a cada aresta de D um par ordenado de vértices (não necessariamente distintos) de D. Se a é uma aresta e u e v são vértices tal que $\psi_D(a) = (u, v)$, então é dito que a liga u a v, u é dito *ponta inicial* de a e v *ponta final*. Um grafo dirigido D' é um subgrafo de D se $V(D') \subseteq V(D), A(D') \subseteq A(D)$ e $\psi_{D'}$ é a restrição de ψ_D a $A(D')$. A terminologia e notação para subgrafos dirigidos é a mesma que a usada para subgrafos não dirigidos.

A diferença de grafos dirigidos para grafos não dirigidos é a orientação que é dada aos pares. Cada grafo dirigido dá origem a um *grafo subjacente*[2] não dirigido, que é o grafo em que esquecemos das orientações dadas as arestas. Algumas vezes, quando não houver perigo de confusão, usaremos a terminologia de grafos não dirigidos para grafos dirigidos, por exemplo, T é uma árvore geradora de um grafo dirigido D se o grafo subjacente a T for uma árvore geradora do grafo subjacente a D. O conceito de *laço*, *arestas paralelas* é análogo ao conceito em grafos não dirigidos.

A *matriz de incidência de um grafo dirigido* $D = (V, A)$, sem laços, é a $\{0, \pm 1\}$-matriz cujas linhas e colunas são indexadas por V e A, respectivamente, onde a entrada (v, a) é $-1, +1$, ou 0, conforme v seja ponta inicial de a, ponta final de a, ou não seja incidente a a, respectivamente (veja figura 4).

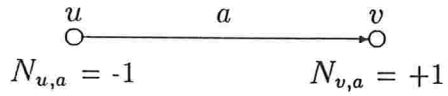

Figura 4: Uma aresta $a = uv$ e os respectivos valores nas entradas (u, a) e (v, a) da matriz de incidência N.

[2] *Underlying graph.*

Capítulo 1

Reconhecimento de grafos

Neste capítulo trataremos do problema de reconhecer em uma matriz estruturas subjacentes de um grafo. Motivações para o interesse em estrutura de grafos serão vistas na seção 1. Na seção 2 estudaremos a complexidade computacional de alguns problemas que tratam do reconhecimento de uma tal estrutura em uma matriz. Na seção 3 discutiremos como o reconhecimento da estrutura de um grafo pode ser utilizada para transformar um *problema de programação linear* (PL) em um problema de programação linear sobre um grafo.

1.1 Motivação

Diversos problemas práticos requerem a utilização de programação linear ou a solução de sistemas lineares. Dentre estes problemas alguns são naturalmente modelados por meio de grafos, problemas tais como produção e distribuição de mercadorias, rede de estradas, comunicação, roteamento de circuitos e redes elétricas. O interesse em tais problemas levou ao desenvolvimento de algoritmos que, ao explorarem a estrutura do grafo contido no problema, são mais rápidos e/ou necessitam menos memória do que algoritmos mais gerais (*cf.* Ford & Fulkerson [1962], Edmonds & Karp [1972]). Essa melhoria de eficiência chega muitas vezes a ser considerável (*cf.* Glover, Karney & Klingman [1974]).

O interesse em reconhecer-se, numa matriz, estruturas que possam ser representadas por um grafo é detectar se o problema que essa matriz representa pode ser resolvido por algum desses algoritmos específicos sobre grafos.

As motivações que veremos a seguir são provenientes da solução de sistemas lineares e, fundalmentalmente, de programação linear. A motivação proveniente de solução de sistemas lineares será mais extensivamente descrita, pois esta, na sua simplicidade, mostra claramente vantagens advindas do reconhecimento de estruturas que possam ser representadas por um grafo. Já as motivações de programação linear serão descritas brevemente, pois muitas das vantagens são em essência as mesmas que veremos na solução de sistemas lineares, serão dadas referências para estas motivações.

1.1.1 Solução de sistemas lineares

Suponha que estejamos interessados em obter uma solução do sistema

$$Nx = b, \tag{1.1}$$

onde

$$N = \begin{array}{c} \begin{array}{ccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \end{array} \\ \left[\begin{array}{ccccccccccccccccc} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{array}\right] \end{array}$$

$$b = \begin{bmatrix} 5 \\ -1 \\ 0 \\ 7 \\ 12 \\ 3 \\ 6 \\ 10 \\ 4 \end{bmatrix}$$

Poderíamos resolver o sistema acima empregando, por exemplo, o método de eliminação de Gauss. No entanto, se examinarmos um pouco mais atentamente a matriz N notaremos que ela possui em cada coluna exatamente dois elementos não nulos, um $+1$ e um -1. Assim N é a matriz de incidência de um grafo dirigido, o qual é mostrado na figura 1.1. Cientes desse fato poderíamos pensar em encontrar uma solução do sistema 1.1 utilizando algum algoritmo mais particular, algoritmo esse que tire proveito dessa estrutura.

Quando N é matriz de incidência de um grafo dirigido, o problema de encontrar uma solução do sistema 1.1 tem uma interpretação bem conhecida que é a seguinte:

> *o grafo $D = (V, A)$ pode ser pensado como uma rede de estradas, ou seja, cada vértices é uma cidade, as arestas são estradas ligando as cidades e os b_v são a demanda, de um determinado produto, em cada cidade. Um A-vetor x, solução do sistema, pode ser pensado como a quantidade de produto x_a que deve ser transportada através de cada estrada a, de tal forma que a demanda em cada cidade seja suprida. Cidades com demanda negativa, $b_v < 0$, são cidades que podem dispor de uma quantidade b_v do produto para mandarem a outras cidades. Se o valor a ser transportado numa estrada $a = (u, v)$ é negativo, $x_a < 0$, isso significa que a mercadoria está sendo transportada da cidade v para a cidade u.*

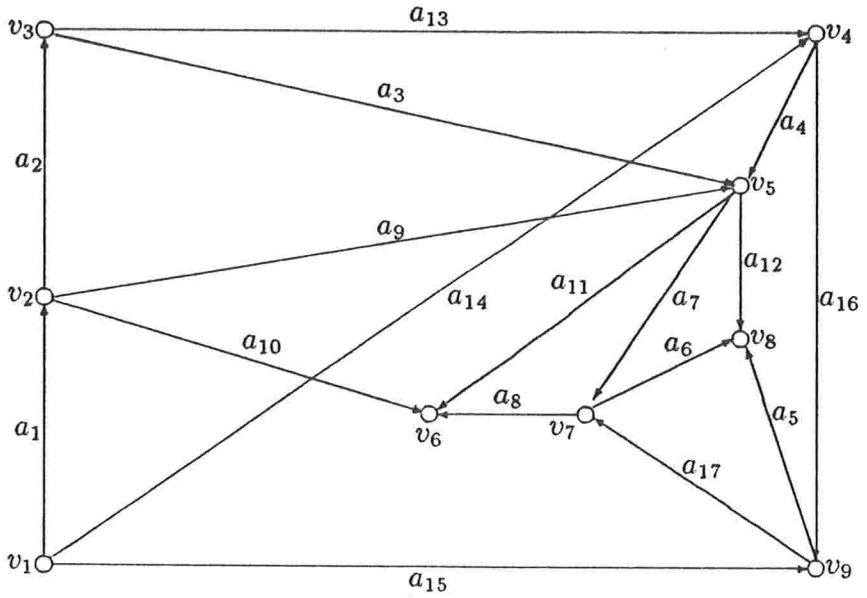

Figura 1.1: Grafo cuja matriz de incidência é N .

Se N fosse a matriz de incidência de uma árvore $T = (V, A)$ poderíamos encontrar uma solução do sistema $Nx = b$ através do algoritmo da figura 1.2. Tendo em mente a interpretação dada anteriormente, percebe-se que a estratégia do algoritmo é escolher uma cidade v, que está ligada às demais por intermédio de uma única estrada a (v é uma folha da árvore) e calcular qual deve ser a quantidade de produto x_a transportada através dessa estrada, para que a demanda na cidade v seja atendida.

Na realidade, se N é matriz de incidência de uma árvore, a solução encontrada pelo algoritmo da figura 1.2 para o sistema $Nx = b$, se existir, é única. Isso pode ser verificado facilmente por indução em $\mid VT \mid$, pois se $\tilde{a} = (u, v)$ é a aresta incidente ao vértice v (escolhido na linha 1 do algoritmo) e N' é a matriz de incidência da árvore $T' := T - a$ então x' é solução do sistema $N'x' = b'$, devolvida pela chamada *ÁrvoresSãoBacanas* (T', b') (linha 12 do algoritmo), *sse* o vetor A-vetor x dado por

$$x_a := \begin{cases} x_a' & \text{se } a \neq \tilde{a} \ ; \\ b_v & \text{se } a = \tilde{a} \end{cases}$$

é solução do sistema $Nx = b$. A figura 1.3 ilustra a situação com relação a matriz N.

É interessante observarmos que no caso de N ser a matriz de incidência de um grafo dirigido $D = (V, A)$ que possue algum circuito então o sistema $Nx = b$ não tem solução única. De fato, suponhamos que D contenha um circuito, digamos C. Consideremos uma orientação arbitrária de C e seja $\tilde{x}$ o seguinte $\{0, \pm 1\}$-vetor com

$$\tilde{x}_a := \begin{cases} +1 & \text{se } a \text{ tem o mesmo sentido da orientação de } C; \\ -1 & \text{se } a \text{ não tem o mesmo sentido da orientação de } C; \\ 0 & \text{se } a \text{ não pertence a } C. \end{cases}$$

Algoritmo
Entrada : Uma árvore $T = (V, A)$ e um V-vetor racional b.
Saída : Um A-vetor x solução do sistema $Nx = b$, se existir, onde N é a $V \times A$-matriz de incidência de T.

```
    início
1       Escolha uma folha v de T;
2       se | VT |= 1
3           então se b_v = 0
4                     então x é solução do sistema
5                     senão o sistema é incompatível
            senão início
6                     Seja u o vértice adjacente a v em T e a a aresta de T incidente a ambos;
7                     b_u ← b_u + b_v;
8                     se a = (u, v)
9                         então x_a ←   b_v;
10                        senão x_a ← −b_v;
                  fim ;
11      Seja b' o vetor b a menos da componente associada a v;
12      ArvoresSãoBacanas (T − v, b')
    fim .
```

Figura 1.2: Algoritmo para solução de sistemas da forma $Nx = b$, onde N é a matriz de incidência de um árvore.

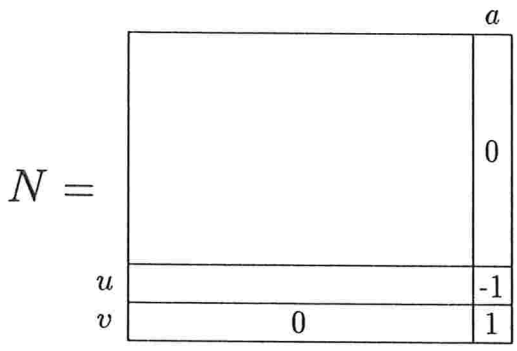

Figura 1.3: Matriz de incidência N de uma árvore dirigida, v é uma folha da árvore e a é a aresta da árvore incidente a v.

para todo $a \in A$ (a figura 1.4 mostra uma orientação de um circuito e o correspondente vetor $\tilde{x}$). Da maneira como $\tilde{x}$ foi definido temos que $N\tilde{x} = 0$ e $\tilde{x} \neq 0$, o que mostra que se o sistema $Nx = b$ tem uma solução $\overline{x}$ então ela não é única, pois $\overline{x} + k\tilde{x}$ também é solução, para todo k.

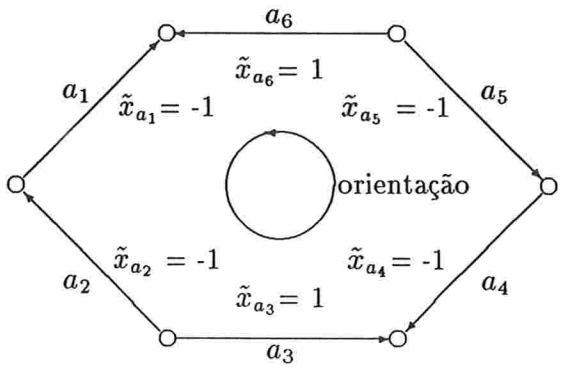

Figura 1.4: Um circuito com a sua orientação e o respectivo vetor $\tilde{x}$.

Do que vimos até agora podemos concluir que se N é a $V \times A$-matriz de incidência de um grafo dirigido conexo $D = (V, A)$ então o sistema $Nx = b$ tem solução única *sse* D for uma árvore. Por outro lado, da álgebra linear sabemos que um sistema $Nx = b$ tem solução única *sse* suas colunas forem l.i. . Logo, como toda árvore geradora T de D possui $\mid V \mid -1$ arestas, podemos resumir tudo isso na seguinte proposição:

Proposição 1.1 *Se $D = (V, A)$ é um grafo dirigido conexo, N a sua $V \times A$-matriz de incidência e $\tilde{N}$ a matriz resultante após a remoção de uma linha de N, então $\tilde{N}$ tem posto linha completo. Além disso, um conjunto K de colunas de $\tilde{N}$ forma uma base do espaço coluna de $\tilde{N}$* sse *as arestas correspondentes às colunas em K formam uma árvore geradora de D.* ■

Está proposição, que relaciona arestas de um grafo com colunas l.i. de sua matriz de incidência, nos será muito útil mais adiante.

O algoritmo da figura 1.2 encontra uma solução do sistema $Nx = b$ somente quando N é a matriz de incidência de uma árvore dirigida. Entretanto, devido a proposição 1.1, é fácil ver que para encontrarmos uma solução do sistema quando N é matriz de incidência de um grafo dirigido conexo $D = (V, A)$ basta utilizarmos o algoritmo $GrafosTambémSãoBacanas$ da figura 1.5.

Se pegassemos papel e lápis para resolver o sistema 1.1 perceberíamos facilmente o quão simples é o algoritmo $GrafosTambémSãoBacanas$ em comparação com outros algoritmos, como o método de eliminação de Gauss.

Vejamos agora qual é a complexidade do algoritmo $GrafosTambémSãoBacanas$. A linha 1 do algoritmo pode ser executada em tempo $O(\mid V \mid\mid A \mid)$. A construção da árvore geradora da linha 2 pode ser executada em tempo $O(\mid A \mid)$ (veja, Gibbons [1985: pp. 20–24]). As linhas 3 e 4 do algoritmo podem ser executadas em tempo $O(\mid A \mid)$. Durante a construção

Algoritmo GrafosTambémSãoBacanas (N, b)
Entrada : Uma $V \times A$-matriz de incidência N de um grafo dirigido conexo e um V-vetor racional b.
Saída : Um A-vetor x solução do sistema $Nx = b$, se existir.

início
1 Construa o grafo $D = (V, A)$ tal que N é a sua matriz de incidência;
2 Construa uma árvore geradora T de D;
3 para toda aresta $a \notin AT$ faça
4 $x_a \leftarrow 0$;
5 ÁrvoresSãoBacanas (T, b);
fim

Figura 1.5: Algoritmo que encontra, se existir, uma solução do sistema $Nx = b$, onde N é matriz de incidência de um grafo dirigido.

da árvore geradora T pode ser construída uma lista de vértices que forneça uma ordem em que os vértices podem ser escolhidos pela linha 1 do algoritmo *ÁrvoresSãoBacanas*. Assim, todas as linhas do algoritmo da figura 1.2 podem ser executadas em tempo constante e portanto a sua complexidade é $O(\mid V \mid)$ (uma tal lista de vértices é mostrada na figura 1.6). Portanto, a complexidade do algoritmo *GrafosTambémSãoBacanas* é $O(\mid V \mid\mid A \mid)$. Já o método de eliminação de Gauss levaria tempo $O(\mid V \mid \mid A \mid^2)$ para encontrar uma solução (veja, Humes *et al.* [1983: pp. 49 e 50]).

Convém lembrarmos que, quando o problema em questão possui a estrutura subjacente de um grafo, muitas vezes o dado do problema é o grafo e não a sua matriz de incidência. Assim, no algoritmo *GrafosTambémSãoBacanas*, a linha 1 seria descartada e a complexidade do algoritmo passaria a ser $O(\mid A \mid)$.

Na figura 1.7 vemos a simulação da resolução do sistema 1.1 através da árvore T e da ordem dada na figura 1.6.

Para armazenar um grafo o espaço requerido é $O(\mid V \mid + \mid A \mid)$, entretanto se o grafo for conexo então $\mid V \mid \leq \mid A \mid -1$, assim o espaço requerido será $O(\mid A \mid)$. Portanto, a complexidade de espaço do algoritmo *GrafosTambémSãoBacanas* é $O(\mid A \mid)$. Observemos que em nenhum instante é necessário que a matriz N esteja inteiramente na memória, pois para construir o grafo D basta o algoritmo examinar uma a uma as colunas de N . Já o espaço requerido pelo método de eliminação de Gauss seria de $O(\mid A \mid\mid V \mid)$.

As vantagens que obtemos ao utilizar o algoritmo *GrafosTambémSãoBacanas*, que encontra uma solução do sistema $Nx = b$ valendo-se da estrutura do grafo, ao invés de um outro algoritmo mais geral foram:

Economia de espaço Numa matriz de incidência os elementos não nulos são da ordem do

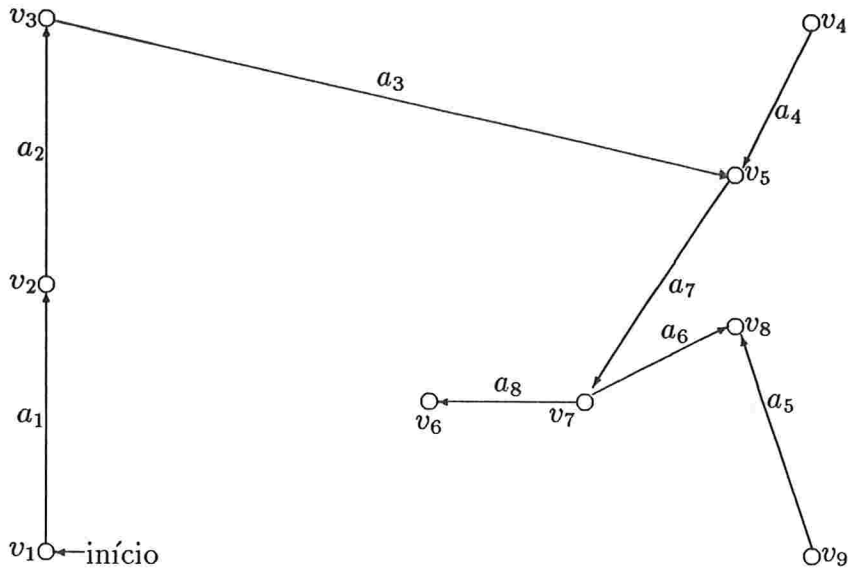

Figura 1.6: Uma árvore geradora do grafo da figura 1.1. A linha tracejada mostra uma ordem em que os vértices podem ser escolhidos pela linha 6 do algoritmo da figura 1.2.

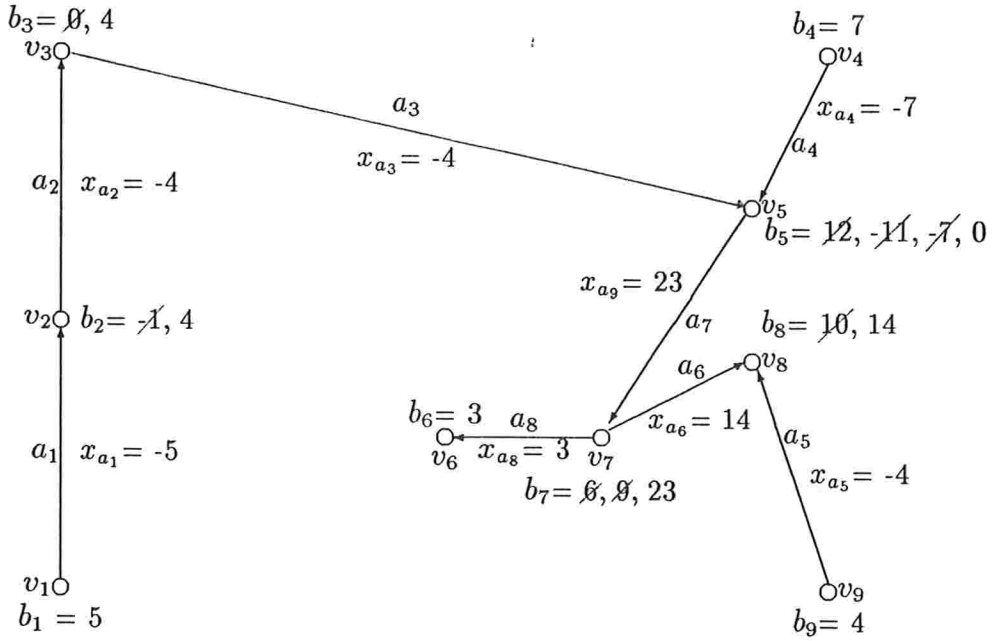

Figura 1.7: Simulação da resolução do sistema 1.1 através do algoritmo da figura 1.2. Como ao final $b_5 = 0$ então o vetor x calculado pelo algoritmo é uma solução do sistema 1.1.

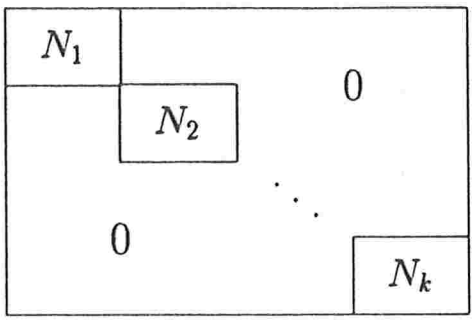

Figura 1.8: Matriz de incidência de um grafo desconexo.

número de colunas. No grafo temos representadas somente as entradas não nulas da matriz.

Eficiência As operações feitas pelo algoritmo, ao percorrer o grafo, foram somente somas e subtrações, que em geral são executadas muito rápidamente por computadores.

Redução de erros Como o algoritmo trabalha somente com o grafo, o qual representa os dados originais do problema, há uma diminuição nos erros numéricos.

Uma última observação sobre o algoritmo $GrafosTambémSãoBacanas$ é que a hipótese de conexidade do grafo D pode ser descartada. De fato, se D for desconexo então as linhas e colunas da matriz N podem ser ordenadas de tal maneira que a matriz assuma a forma tabular da figura 1.8, onde cada submatriz N_i é a matriz de incidência de uma componente conexa D_i do grafo D.

Assim é fácil ver que, para resolver um sistema linear em que o grafo D não é conexo, basta simplesmente aplicar o algoritmo $GrafosTambémSãoBacanas$ a cada um dos subsistemas associados às componentes do grafo. No seção 2.1.3 veremos como pode ser feita uma ordenação das linhas e colunas da matriz de tal forma que ela fique como na figura 1.8 bem como uma extensão do conceito de conexidade em grafos para conexidade em matrizes.

1.1.2 Programação linear

Em programação linear estruturas especiais são freqüentemente exploradas por algoritmos mais particulares objetivando assim uma maior eficiência e/ou uma economia de mémoria (*cf.* Dantzig & Wolfe [1960], Dantzig & Van Slyke [1967]). Atenção especial tem sido dedicada por diversos pesquisadores a problemas de otimização que possuem uma estrutura subjacente de grafos (*cf.* Klingman & Mulvey [1981], Gallo & Sandi [1986]).

Consideremos o problema de programação linear (PL) na sua forma canônica

$$\begin{aligned} \min \quad & cx \\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{1.2}$$

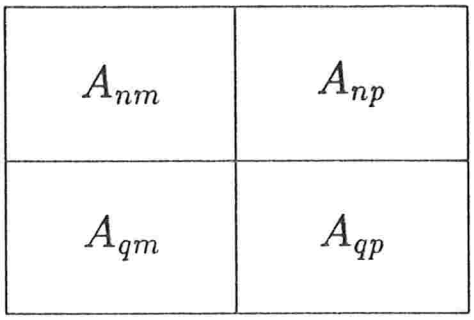

Figura 1.9: Matriz A.

Visando resolver 1.2 quando A é a matriz de incidência de um grafo, após a possível remoção de uma linha, Dantzig[1951] e Orden [1956] obtiveram uma substancial simplificação no método simplex revisado. O método resultante foi denominado de *método simplex para redes* (veja Chvátal [1983: p. 291]).

De uma maneira semelhante à que vimos na solução de sistemas lineares, o método simplex para redes obtém melhora de desempenho e economia de mémoria, em relação ao simplex revisado, ao trocar as operação sobre a matriz A por operações mais simples sobre uma árvore geradora T do grafo, cuja matriz de incidência é A.

Da mesma forma que vimos na solução de sistemas lineares aqui também a hipótese de conexidade do grafo pode ser descartada, pois caso o grafo seja desconexo as linhas e colunas de A podem ser ordenadas de maneira que a matriz assuma a forma da figura 1.8 e assim não é difícil vermos que o problema 1.2 pode ser decomposto nos subproblemas

$$\begin{aligned} \min \quad & c_i x_i \\ & N_i x = b_i \quad \text{para } i = 1, ..., k \\ & x_i \geq 0 \end{aligned} \tag{1.3}$$

onde N_i é a matriz de incidência de uma componente conexa de D e c_i, x_i e b_i são as componentes de c, x e b correspondentes à submatriz N_i.

O método simplex para redes pode ser encontrado em Chvátal [1983: cap. 19].

Suponhamos agora que a matriz A em 1.2 não seja a matriz de incidência de um grafo dirigido. Entretanto, suas linhas e colunas podem ser ordenadas de tal forma que A fique como na figura 1.9, onde A_{ij} é uma $i \times j$-submatriz e A_{nm} é a matriz de incidência de um grafo dirigido após a possível remoção de algumas linhas. Para resolver o problema 1.2 quando a matriz A tem a forma descrita, Glover & Klingman [1981] desenvolveram um método do tipo simplex chamado simplex SON *(Special ordered network)*. A motivação que ambos tiveram para desenvolver tal método foram problemas práticos em que A_{nm} é relativamente grande.

A a estrutura de A induz uma partição na submatriz básica[1] B (i.e., a submatriz

[1]Este tipo de técnica é chamada de *partitioning methods*. Veja também Bixby [1984] e Aho, Hopcroft &

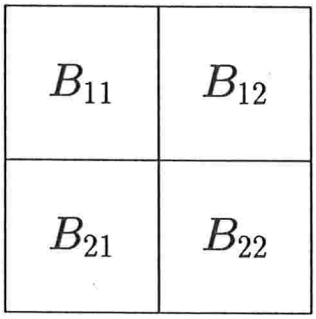

Figura 1.10: Matriz básica B.

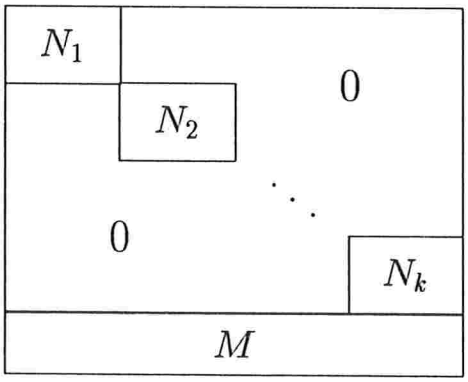

Figura 1.11: Matriz no formato de quase blocos.

formada por um conjunto de colunas l.i. que formam uma base do espaço coluna de A), como mostra a figura 1.10, onde B_{11} é a matriz de incidência de uma árvore dirigida T após a remoção de uma linha. No método SON, as operações envolvendo a submatriz B_{11} são feitas percorrendo-se T. As vantagens advindas são as mesmas mencionadas para os problemas anteriores.

O princípio de decomposição de Dantzig & Wolfe [1960] aplica-se para resolver o PL 1.2 quando A é uma matriz cujas linhas e colunas podem ser ordenadas de tal forma que ela assuma uma forma tabular como a mostrada na figura 1.11, as submatrizes N_i são associados a PLs chamados de *subproblemas* e a submatriz M é associada ao bem conhecido PL chamado *problema mestre*. Uma matriz com a forma tabular da figura 1.11 é dita uma *matriz de quase blocos*[2].

O princípio de decomposição de Dantzig & Wolfe tem suas vantagens e desvantagens em relação ao método simplex revisado. A chave do sucesso de algoritmos de decomposição reside numa partição criteriosa da matriz associada ao problema. Idealmente, na aplicação da decomposição de Dantzig & Wolfe à matrizes de quase blocos, os subproblemas deveriam poder ser resolvidos de uma maneira relativamente rápida e a submatriz associada ao problema

Ullman [1976: pp. 232 e 233].

[2] *Block-angular matrix.*

mestre deveria ter poucas linhas. Quando as matrizes associadas aos subproblemas são matrizes de incidência de grafos dirigidos, após a possível remoção de algumas linhas, os subproblemas podem ser resolvidos eficientemente pelo método simplex para redes. Neste caso a decomposição de Dantzig & Wolfe pode ser aplicada com uma grande melhora de desempenho.

Para uma descrição detalhada do príncipio de decomposição de Dantzig & Wolfe veja Chvátal [1983: cap. 26].

A última motivação que mencionaremos aqui vem de *programação linear inteira* (PLI), i.e., problemas em que estamos interessados em resolver o problema 1.2 com a restrição adicional de que todas as componentes do vetor x sejam números inteiros. Programação linear inteira é em geral um problema $\mathcal{NP}$-completo. Entretanto, quando restringimos PLI a instâncias em que a matriz de restrições é a matriz de incidência de um grafo dirigido o problema passa a poder ser resolvido em tempo polinomial (*cf.* TLIP, p. 279). O que está por trás disso é o fato de que, se N é a matriz de incidência de um grafo dirigido então, o poliedro

$$P := \{Nx = b \mid x \geq 0\}$$

tem somente vértices com coordenadas inteiras (i.e., o poliedro é *inteiro*) para todo vetor b inteiro. No capítulo 4 nos aprofundaremos um pouco no estudo e reconhecimento das matrizes que têm essa propriedade bastante desejável.

1.2 Qual é o problema?

A resposta para pergunta acima é "São vários!". O que faremos nesta seção será explicitar e discutir sobre a complexidade computacional de alguns problemas que tratam do reconhecimento de estrutura em matrizes que possam ser representadas por um grafo. Olharemos estes problemas sobre o ponto de vista de complexidade computacional, i.e., o nosso interesse será saber quais desses problemas são "fáceis" (podem ser resolvidos em tempo polinomial) e quais são "difíceis" ($\mathcal{NP}$-completos). Para os problemas que podem ser resolvidos em tempo polinomial não estaremos muito preocupados, pelo menos nesta seção, com qual é o grau do polinômio.

Lembremos que, ao longo desta dissertação, matrizes são indexadas por conjuntos que, a priori, não têm nenhuma ordem entre seus elementos. Assim uma mesma matriz pode ser exibida de diferentes formas tabulares.

Devido às motivações que vimos na seção anterior é bastante razoável que a estrutura que representa um grafo que pensemos em reconhecer primeiro seja a de matrizes de incidência de grafos dirigidos.

1.2.1 Matrizes de grafos

Da proposição 1.1 temos que se N é a $V \times A$-matriz de incidência de um grafo D com componentes $D_1, D_2, ..., D_k$, então a matriz $\hat{N}$ resultante da remoção de linhas $v_1, v_2, ...v_k$ de N, onde

v_i é um vértice qualquer de D_i, tem posto linha completo. Isto nos diz que $\tilde{N}$ contém essencialmente a mesma informação que N , pelo menos quando estamos interessados em resolver sistemas lineares ou PLs. Na realidade a partir de $\tilde{N}$ podemos obter a matriz de incidência de um grafo simplesmente acrescentando-se a $\tilde{N}$, uma linha que é soma de todas as linhas de $\tilde{N}$. É fácil vermos que a matriz resultante tem dois elementos não nulos distintos em cada coluna, um +1 e um -1. Assim, chamaremos de *matriz de um grafo*[3] a qualquer $\{0, \pm 1\}$-matriz que contenha em cada coluna no máximo duas entradas não nulas distintas e chamaremos de *matriz restrita de um grafo*[4] a matriz de um grafo que tenha posto linha completo. Diremos ainda que $\tilde{N}$ é (uma) *matriz do grafo* D.

O problema de decidir se uma matriz é matriz de um grafo, i.e. :

dados	:	Uma $L \times C$-matriz A.
pergunta	:	A é matriz de um grafo? (1.4)

é um problema fácil de ser resolvido por um algoritmo de complexidade $O(\mid L \mid\mid C \mid)$.

Percebe-se facilmente que a propriedade Π_g = "ser matriz de um grafo" é uma propriedade hereditária e que as obstruções de Π_g são as matrizes:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Na prática, é pouco provável que matrizes de PLs ou de sistemas lineares sejam matrizes de grafo. O que podemos esperar, com alguma sorte, é que essas matrizes sejam parcialmente matrizes de grafos, como mostra a figura 1.12.

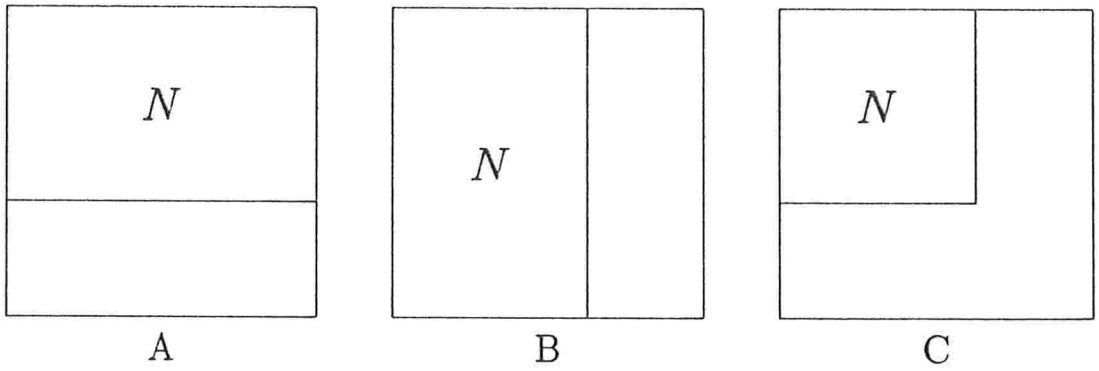

Figura 1.12: Matrizes que são parcialmente matrizes de grafos. A região com um N denota a submatriz que é matriz de um grafo.

Reconhecimento de matrizes que são parcialmente matrizes de grafos, como na figura 1.12, são úteis para: o método simplex (SON) de Glover & Klingman [1981], decomposição

[3] *Node-arc incidence matrix* ou *network matrix*.
[4] *Restricted node-arc incidence matrix*.

de Dantzig & Wolfe [1960], decomposição de Benders (veja TLIP, pp. 371 e 372]) e relaxação lagrangeana (veja TLIP, pp. 367-370]).

A figura 1.12 nos sugere três problemas, a saber:

dados : Uma $L \times C$-matriz.
encontrar : Uma $L' \times C$-submatriz de A que seja matriz de um grafo tal que $\mid L' \mid$ seja máximo. (1.5)

dados : Uma $L \times C$-matriz A.
encontrar : Uma $L \times C'$-submatriz de A que seja matriz de um grafo tal que $\mid C' \mid$ seja máximo. (1.6)

dados : Uma $L \times C$-matriz A.
encontrar : Uma $L' \times C'$-submatriz de A que seja matriz de um grafo tal que $\mid L' \mid \times \mid C' \mid$ seja máximo. (1.7)

O problema 1.6 pode ser resolvido facilmente pelo mesmo algoritmo que resolve o problema 1.4. Infelizmente os problemas 1.5 e 1.7 são $\mathcal{NP}$-difíceis, tendo em vista que, pela proposição 1.2 abaixo, o problema de decisão associado ao problema 1.5, i.e.,

dados : Uma $L \times C$-matriz A e um inteiro positivo k.
pergunta : Existe uma $L' \times C$-submatriz de A que seja matriz de um grafo e $\mid L' \mid \geq k$? (1.8)

é $\mathcal{NP}$-completo. Aqui, bem como em muitos problemas em ciência da computação, parece que a linha que separa o muito fácil do muito difícil é bastante tênue.

Proposição 1.2 *O problema 1.8 é $\mathcal{NP}$-completo.*

Prova: Mostraremos que mesmo se A é uma $\{0,1\}$-matriz o problema 1.8 é $\mathcal{NP}$-completo.

É fácil ver que o problema 1.8 pertence a $\mathcal{NP}$, pois um algoritmo não determinístico necessita somente escolher um conjunto $L' \subseteq L$ e verificar, em tempo polinomial, se a $L' \times C$-submatriz de A é matriz de um grafo.

Transformaremos o problema do conjunto independente máximo no problema 1.8.

Seja $G = (L, C)$ uma instância arbitrária do *problema do conjunto independente máximo*. A matriz A que contruiremos é simplesmente a $L \times C$-matriz de incidência de G. É claro que A pode ser construída em tempo polinomial.

Como A é a matriz de incidência G então é imediato que L' é um conjunto independente de G *sse* a $L' \times C$-submatriz de A for a matriz de um grafo. O que prova que a transformação que fizemos é de fato uma redução. ■

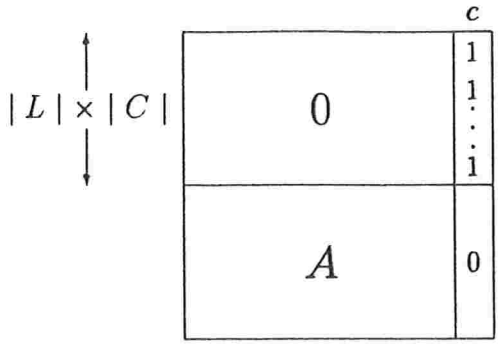

Figura 1.13: Matriz da transformação que reduz o problema 1.5 ao problema 1.7.

Para verificar que a proposição 1.2 de fato implica que o problema 1.7 é $\mathcal{NP}$-difícil, basta verificar que a transformação que dada uma $L \times C$-matriz A, instância do problema 1.12, constrói a matriz mostrada na figura 1.13 é uma redução polinomial do problema 1.5 (que, pela proposição 1.2, é $\mathcal{NP}$-difícil) ao problema 1.7.

Mesmo a estrutura de matrizes que são parcialmente matrizes de grafos é uma estrutura muito particular. Na próxima subseção trataremos do problema de reconhecer estruturas um pouco mais gerais que as tratadas até agora.

1.2.2 Matrizes de grafos sinalizados

Um problema um pouco mais geral que o problema 1.4, é o de decidir se uma matriz A pode ser transformada numa matriz de um grafo através da multiplicação de algumas linhas[5] por -1, ou seja:

dados	:	Uma $L \times C$-matriz A.	
pergunta	:	Existe uma $\{0, \pm 1\}$-matriz diagonal não singular Δ tal que $N = \Delta A$, onde N é a matriz de um grafo?	(1.9)

Se existir uma tal matriz diagonal Δ diremos que A é *matriz de um grafo sinalizado*[6]. A origem do nome é devido ao seguinte fato: se A é uma $L \times C - \{0, \pm 1\}$-matriz com no máximo duas entradas não nulas por coluna e $G = (L, C'), C' \subseteq C$, é o grafo tal que $c = l_1 l_2 \in C'$ *sse* $A_{l_1,c} = A_{l_2,c} \neq 0$, então resolver o problema 1.9 é equivalente a encontrar, se existir, uma atribuição de sinais $(+,-)$ aos vértices de G tal que ao multiplicarmos por -1 as linhas cujos vértices correspondentes receberam o sinal $(-)$, transformamos A na matriz de um grafo. É claro que toda matriz de um grafo é matriz de um grafo sinalizado, basta tomarmos $\Delta = I$.

Abaixo vemos duas $\{0, \pm 1\}$-matrizes que claramente não são matrizes de grafos sina-

[5] *Scaling* das linhas por 1 ou -1

[6] *Embedded network matrix.*

lizados:

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & 1 \\ -1 & 1 & 0 & \dots & \dots & 0 \\ 0 & -1 & 1 & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 & 0 \\ 0 & 0 & \dots & \dots & -1 & 1 \end{bmatrix} \qquad M_{2k+1} = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & 1 \\ 1 & 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 1 & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 & 0 \\ 0 & 0 & \dots & \dots & 1 & 1 \end{bmatrix} \tag{1.10}$$

onde M_1 é uma matriz de ordem maior ou igual a 2 e M_{2k+1} é uma matriz de ordem $2k+1, k > 0$. Observemos que matrizes obtidas após a multiplicação de algumas colunas das matrizes 1.10 por -1 também não são matrizes de grafos sinalizados.

É fácil vermos que a propriedade Π_s = "ser matriz de um grafo sinalizado" é uma propriedade hereditária. Portanto, as matrizes 1.10, bem como as matrizes obtidas a partir dessas após a multiplicação de algumas colunas por -1, são obstruções de Π_s.

O problema 1.9, a exemplo do que ocorre com o problema 1.4, pode ser resolvido em tempo polinomial. A figura 1.14 mostra um algoritmo polinomial para o problema 1.9.

A idéia em que se baseia o algoritmo da figura 1.14 é simples. Antes de mais nada é claro que se uma $L \times C$-matriz A não é uma $\{0, \pm 1\}$-matriz com no máximo duas entradas não nulas por coluna então A não é a matriz de um grafo subjacente, daí a razão do teste na linha 1 do algoritmo. Caso contrário, na linha 3 do algoritmo é construído o grafo $G = (L, E)$, onde uma aresta $l_1 l_2$ pertence a E *sse* existe uma coluna c tal que $A_{l_1,c} \neq 0 \neq A_{l_2,c}$. Na linha 5 são contraídas as arestas do grafo G que correspondem a colunas que não são *conflitantes,* i.e., colunas com duas entradas não nulas distintas. Assim, os vértices $\{\tilde{l}_1, \tilde{l}_2, ..., \tilde{l}_k\}$ do grafo G' induzem uma partição no conjunto L. Seja $\tilde{l} = \{l_{i_1}, l_{i_2}, ..., l_{i_p}\} \subseteq L$ um vértice de G', se alguma linha $l_{i_j} \in \tilde{l}$ for multiplicada por $\alpha \in \{1, -1\}$, para transformar A na matriz de um grafo, então todas as linhas em $\tilde{l}$ deverão ser multiplicadas por α. Mais ainda, se $\tilde{l}_1 \tilde{l}_2$ é uma aresta de G' isto significa que se as linhas em $\tilde{l}_1$ forem multiplicadas por $\alpha \in \{1, -1\}$, então as linhas em $\tilde{l}_2$ devem ser multiplicadas por $-\alpha$, para que A seja transformada na matriz de um grafo. Desta forma, é claro que se o grafo G' possuir um circuito ímpar, então A não é a matriz de um grafo subjacente, isto é verificado na linha 6 do algoritmo. Mais ainda, se G' contém um circuito ímpar de comprimento 1 (um laço), então não é difícil encontrarmos uma submatriz de A que seja igual a matriz M_1 em 1.10 (a menos da multiplicação de algumas colunas por -1), e se G' contém um circuito ímpar de comprimento maior que 1 podemos encontrar uma submatriz de A que seja igual a matriz M_{2k+1} em 1.10 (a menos da multiplicação de algumas colunas por -1) onde $2k + 1$ é o comprimento do circuito ímpar. Observemos que os índices nas matrizes 1.10 são o comprimento do circuito ímpar. Logo, se G' for um grafo bipartido, então A é a matriz de um grafo subjacente e a matriz Δ é calculada pelas linhas 7–13 do algoritmo.

Das observações que fizemos acima e da corretude do algoritmo da figura 1.14 (que não foi provada mais não é difícil de verificarmos) temos a seguinte proposição:

Proposição 1.3 *Uma $\{0, \pm 1\}$-matriz com no máximo duas entradas não nulas por coluna é a matriz de um grafo sinalizado* sse *não possuir as matrizes 1.10 (a menos da multiplicação de algumas colunas por -1) como submatrizes.* ■

Algoritmo MatrizDeGrafoSinalizado (A)
Entrada : Uma $L \times C$-matriz A.
Saída : Uma $\{0, \pm 1\}$-matriz diagonal não singular Δ tal que $N = \Delta A$, onde N é a matriz de um grafo, se A for a matriz de um grafo sinalizado.

```
    início
1       se A é uma {0,±1}-matriz com no máximo duas entradas não nulas em cada coluna
            então início
2                   Seja C' ⊆ C o conjunto das colunas com duas entradas não nulas;
3                   Construa o grafo G = (L, C'), onde c ∈ C' liga l1 e l2 se A_{l1,c} ≠ 0 ≠ A_{l2,c};
4                   Seja C'' ⊆ C as colunas com duas entradas não nulas distintas;
5                   G' ← G/C'';
6                   se G' é bipartido
                        então início
7                               A é a matriz de um grafo sinalizado;
                                % calculo da matriz Δ
8                               Seja (X, Y) uma bipartição de G';
9                               para cada vértice l ∈ L faça
10                                  se l pertence a algum vértice l̃ de X
11                                      então d_l ← −1
12                                      senão d_l ←  1 ;
13                              Δ = diag(d_{l1}, d_{l2}, ..., d_{l|L|}) é tal que ΔA
                                é a matriz de um grafo;
                            fim
14                      senão A não é matriz de um grafo sinalizado;
                    fim
15          senão A não é matriz de um grafo sinalizado;
    fim
```

Figura 1.14: Algoritmo que verifica se uma matriz é matriz de um grafo sinalizado.

O que a proposição 1.3 também está nos dizendo é que, a exemplo do que ocorre com matrizes de grafos, o problema de decidir se uma matriz é matriz de um grafo sinalizado tem uma boa caracterização (i.e. está em $\mathcal{NP} \cap$ co-$\mathcal{NP}$).

Apesar de matrizes de grafos sinalizados serem uma estrutura um pouco (talvez bem pouco) mais geral que a de matrizes de grafos, ainda assim é pouco provável que em problemas práticos nos defrontemos com matrizes que sejam matrizes de grafos sinalizados, assim é natural pensarmos nos seguintes problemas análogos aos problemas 1.5, 1.6 e 1.7, respectivamente:

dados	:	Uma $L \times C$-matriz.	
encontrar	:	Uma $L' \times C$-submatriz de A que seja matriz de um grafo sinalizado tal que $\mid L' \mid$ seja máximo.	(1.11)

dados	:	Uma $L \times C$-matriz A.	
encontrar	:	Uma $L \times C'$-submatriz de A que seja matriz de um grafo sinalizado tal que $\mid C' \mid$ seja máximo.	(1.12)

dados	:	Uma $L \times C$-matriz A.	
encontrar	:	Uma $L' \times C'$-submatriz de A que seja matriz de um grafo sinalizado tal que $\mid L' \mid \times \mid C' \mid$ seja máximo.	(1.13)

Como os problemas 1.6 e 1.7, devido a proposição 1.2, são $\mathcal{NP}$-difíceis então é claro que os problemas 1.12 e 1.13 são $\mathcal{NP}$-difíceis. A novidade agora é que apesar do problema 1.5 não ser $\mathcal{NP}$-difícil, o problema análogo para matriz de grafos sinalizados, problema 1.11, é $\mathcal{NP}$-difícil, pois, como veremos na proposição 1.4, o problema de decisão associado ao problema 1.11, ou seja:

dados	:	Uma $L \times C$-matriz A e um inteiro positivo k.	
pergunta	:	Existe uma $L' \times C$-submatriz de A que seja matriz de um grafo sinalizado e $\mid L' \mid \geq k$?	(1.14)

é $\mathcal{NP}$-completo.

Proposição 1.4 *O problema 1.14 é $\mathcal{NP}$-completo.*

Prova: É fácil vermos que o problema 1.14 pertence a $\mathcal{NP}$, pois um algoritmo não determinístico necessita somente escolher um conjunto $L' \subseteq L$ e verificar em tempo polinomial, com o algoritmo da figura 1.14 por exemplo, se a $L' \times C$-submatriz de A é matriz de um grafo sinalizado.

Transformaremos o *problema do subgrafo bipartido* no problema 1.14. O problema do subgrafo bipartido consiste em:

dados	:	Dado um grafo $G = (V, E)$ e um inteiro positivo k.	
pergunta	:	Existe um subconjunto $E' \subseteq E$ com $\mid E' \mid \geq k$ tal que o subgrafo $G[E']$ é bipartido.	(1.15)

Para mais informações sobre o problema 1.15 veja Garey & Johnson [1979: p. 196].

Seja $G = (L, C)$ uma instância do problema do subgrafo bipartido. Podemos supor sem perda de generalidade que G não possui laços. Construiremos uma $L \times C - \{0,1\}$-matriz A tal que $G[C'], C' \subseteq C$ é um subgrafo bipartido de G *sse* a $L \times C'$-submatriz de A é uma matriz de um grafo sinalizado.

A matriz A que contruiremos é nada mais do que a matriz de incidência de G, i.e., para cada aresta c de G construiremos uma L-coluna da matriz A da seguinte maneira:

$$A_{l,c} := \begin{cases} 1 & \text{se } c \text{ é incidente a } l; \\ 0 & \text{caso contrário} \end{cases}$$

para cada vértice l de G.

É claro que A pode ser construída em tempo polinomial. Agora resta apenas provarmos que $G[C']$ é um subgrafo bipartido *sse* a $L \times C'$-submatriz A' de A é uma matriz de um grafo sinalizado. De fato, suponhamos que $G[C']$ é bipartido. Seja (X, Y) um bipartição de $G[C']$ e Δ a $L \times L$-matriz diagonal onde o elemento da entrada (l, l) é -1 se $l \in X$ e 1 caso contrário. Não é difícil verificarmos que $\Delta A'$ é a matriz de um grafo. Reciprocamente, suponhamos que A' é uma matriz de um grafo sinalizado. Sejam Δ uma matriz diagonal não singular tal que $\Delta A'$ é a matriz de um grafo, $X = \{l \in L \mid \Delta_{l,l} = -1\}$ e L' os vértices de $G[C']$. Também não é difícil verificarmos que $(X, L' \setminus X)$ é uma bipartição de $G[C']$. ■

Assim, pelo que vimos, todos os problemas que tratam do reconhecimento de submatrizes máximas (máximas no sentido de linhas ou colunas ou número de elementos, veja figura 1.12) que são matrizes de grafos sinalizados são $\mathcal{NP}$-difíceis.

Na próxima subseção ainda faremos uma última generalização dos problema tratados até agora.

1.2.3 Matrizes gráficas

Na seção anterior tratamos do problema de tentar transformar uma matriz A na matriz de um grafo através da multiplicação de algumas linhas por -1. Nesta subseção trataremos novamente do problema de transformar uma matriz A na matriz de um grafo só que desta vez as operações permitidas serão combinação linear das linhas e multiplicação das colunas por números não nulos[7]. Mais precisamente:

dados	:	Uma $L \times C$-matriz A.	
pergunta	:	Existem matrizes não singulares π e Δ tais que $N = \pi A \Delta$, onde Δ é uma matriz diagonal e N é a matriz de um grafo?	(1.16)

Dizemos que duas matrizes A e B são *projetivamente equivalentes (p.e.)*, escreve-se $A \sim B$, se $B = \pi A \Delta$, onde π e Δ são matrizes não singulares e Δ é uma matriz diagonal. A

[7] *Scaling* não nulo das colunas.

origem do nome p.e. é proveniente de espaços projetivos. Se pensarmos nas colunas de A como pontos do espaço projetivo P^L então Δ leva esses pontos do espaço em pontos projetivamente equivalentes e π é um automorfismo do espaço.

Em outras palavras, o que o problema 1.16 está perguntando é se uma matriz A dada é p.e. a matriz de um grafo. É claro que, se A é a matriz de um grafo sinalizado, então A é p.e. a matriz de um grafo. Mais ainda, as matrizes M_1 e M_{2k+1} em 1.10 também são p.e. a matriz de um grafo, pois são matrizes não singulares e portanto podem ser transformadas, através de combinação linear das linhas, na identidade. Se A é uma matriz p.e. a matriz de um grafo diremos então que A é *gráfica*.

Analogamente ao que vimos nas subseções anteriores é fácil ver que a propriedade Π_{graf} = "ser uma matriz gráfica" = "ser uma matriz p.e. a matriz de um grafo" é uma propriedade hereditária.

Uma boa caracterização, bastante profunda, de matrizes gráficas é devida a Tutte e encontra-se no teorema 1.1 abaixo.

Teorema 1.1 (Tutte [1959,1965]) *Uma matriz é gráfica* sse *não possui submatrizes p.e. às seguintes matrizes:*

$$U_{2,4} = \begin{bmatrix} 1 & 0 & \alpha & \beta \\ 0 & 1 & \gamma & \delta \end{bmatrix} \quad F_7 = \begin{bmatrix} 1 & 0 & 0 & 0 & * & * & * \\ 0 & 1 & 0 & * & 0 & * & * \\ 0 & 0 & 1 & * & * & 0 & * \end{bmatrix} \quad F_7^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & * & * \\ 0 & 1 & 0 & 0 & * & 0 & * \\ 0 & 0 & 1 & 0 & * & * & 0 \\ 0 & 0 & 0 & 1 & * & * & * \end{bmatrix}$$

$$M^*(K_{3,3}) = \begin{bmatrix} 1 & 0 & 0 & 0 & * & 0 & 0 & * & * \\ 0 & 1 & 0 & 0 & * & * & 0 & 0 & * \\ 0 & 0 & 1 & 0 & 0 & * & * & 0 & * \\ 0 & 0 & 0 & 1 & 0 & 0 & * & * & * \end{bmatrix} \quad M^*(K_5) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & * & 0 & * & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & * & 0 & 0 & * \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & * & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & * & * \end{bmatrix}$$

onde α, γ, β, δ são números não nulos, (α, γ) e (β, δ) são l.i. e $*$ denota entradas não nulas, não necessariamente iguais. ■

Para outras demonstrações do teorema 1.1 veja Seymour [1980], Truemper [1985b], Wagner [1985] e Bixby & Cunningham [1980].

Diversos algoritmos polinomiais foram propostos para resolver o problema 1.16 (Tutte [1960], Iri [1966], Bixby & Cunningham [1980], Fujishige [1980] e Bixby & Wagner [1988]). Como os problemas 1.11, 1.12 e 1.13 são $\mathcal{NP}$-difíceis então é claro que qualquer problema que diga respeito a reconhecer numa matriz uma submatriz máxima (máxima no sentido de linhas ou colunas ou ainda número de elementos, veja figura 1.12) que seja gráfica é $\mathcal{NP}$-difícil.

Na próxima seção veremos como o reconhecimento de matrizes gráficas pode ser útil em programação linear.

1.3 Redes escondidas

Na última seção vimos alguns problemas que tratam do reconhecimento de estruturas subjacentes de um grafo em uma matriz. Dentre estes problemas aquele que dedicaremos nossa atenção será o último, i.e. o reconhecimento de (sub)matrizes gráficas.

Como já foi dito, a motivação para reconhecer tais estruturas subjacentes vem principalmente de programação linear. As motivações que vimos na seção 1 diziam respeito ao reconhecimento de matrizes de grafos. Veremos agora como o reconhecimento de matrizes gráficas pode ser útil em programação linear.

Um PL

$$\begin{aligned} \min \quad & cx \\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{1.17}$$

é dito uma *rede escondida*[8] se a matriz A é gráfica. Observemos que nesse caso $N = \pi A \Delta$, onde N é uma $\{0, \pm 1\}$-matriz de um grafo, então se tivermos conhecimento explicito de Δ, N e de uma solução $\tilde{x}$ do sistema $Ax = b$ podemos produzir o PL

$$\begin{aligned} \min \quad & c'y \\ & Ny = b' \\ & y \geq 0 \end{aligned} \tag{1.18}$$

tal que y é solução de 1.18 *sse* Δy é solução de 1.17. Em particular, como pode ser visto no apêndice A, não é preciso que conheçamos π para transformar o PL 1.17 no PL 1.18.

O problema de encontrar redes escondidas é na realidade o problema de encontrar matrizes gráficas em PLs.

A motivação para encontrar-se redes escondidas é primordialmente computacional. Como foi visto na seção 1 o PL 1.18 pode ser resolvido mais eficientemente que PLs mais gerais, como o PL 1.17. Também há motivação proveniente de uma certa "atração" por problemas de otimização em grafos, os quais sugerem que uma nova visão (modelagem) do problema pode ser ganha ao transformarmos problemas que, a priori, não parecem conter a estrutura de um grafo em problemas em grafos.

Bixby [1984] sugere os seguintes passos para encontrar redes encondidas:

1. *Simplificação.* Matrizes de PLs apresentam com freqüência variáveis fixas, restrições explícitas que têm o efeito de forçar implicitamente limites inferiores ou superiores em algumas variáveis. Devido a isso algumas linhas da matriz podem ser eventualmente descartadas.

2. *Escala.*[9] Se A é uma matriz, o objetivo aqui é encontrar matrizes diagonais não singulares Δ_1, Δ_2, tal que $\Delta_1 A \Delta_2$ tenha o maior número possível de $\{0, \pm 1\}$-linhas, as quais serão candidatas a serem linhas de um grafo sinalizado.

[8] *Hidden network*
[9] *Scaling.*

3. Grafos sinalizados. Neste passo deve-se tentar encontrar a maior submatriz de um grafo sinalizado que for possível.

4. Rede escondida. Agora o objetivo é estender a submatriz de um grafo sinalizado do item anterior para a maior matriz gráfica que for possível.

Referências sobre diversas heurísticas para os passos 1–3 e alguns resultados computacionais obtidos por R. Fourer e Bixby aplicando os passo acima a 10 PLs podem ser encontrados em Bixby [1984].

Bixby sugere os passos acima por considerar que a aplicação direta de técnicas para encontrar redes escondidas (que veremos no próximo capítulo) são "relativamente" ineficientes em comparação com as técnicas utilizadas nos passos 1–3.

Seja A a $L \times C$-matriz do PL 1.17. Se encontrarmos uma $L' \times C$-submatriz gráfica de A que seja "grande" poderíamos então aplicar o princípio de decomposição de Dantzig & Wolfe ao PL. As linhas que não fazem parte da submatriz gráfica formariam o problema mestre.

Pelo que vimos na seção passada sabemos que o problema de encontrar redes escondidas maximas é $\mathcal{NP}$-difícil. Por outro lado, é claro que qualquer algoritmo polinomial que decida se um PL é uma rede escondida da origem a um algoritmo polinomial para encontrar redes escondidas maximais. De fato, se dada uma $L \times C$-matriz queremos, por exemplo, encontrar uma $L' \times C$-submatriz gráfica maximal basta começarmos o processo com uma submatriz gráfica, digamos, uma simples linha, e irmos adicionando uma a uma as linhas da matriz à submatriz e testar, em tempo polinomial, se a submatriz resultante é gráfica. É claro que esse método do tipo "força bruta" não deve ser muito eficiente. Na realidade a sua complexidade será $\mid L \mid$ vezes pior que a complexidade do algoritmo polinomial que decide se uma matriz é gráfica. É de se esperar que um algoritmo que tire proveito da submatriz gráfica já construída para construir a próxima deva ter um desempenho bem melhor.

No próximo capítulo veremos um algoritmo que decide se uma matriz é gráfica e no apêndice A é ilustrado como esse algoritmo pode ser usado para converter o PL original em um PL como em 1.18. Na realidade o algoritmo que veremos faz mais do que isso, dada uma matriz $L \times C$-matriz ele encontra uma $L \times C'$-submatriz gráfica maximal.

Capítulo 2

Algoritmo para reconhecimento de matrizes gráficas

O objeto central deste capítulo e apresentar um algoritmo que reconhece matrizes gráficas. Na realidade o algoritmo que veremos resolve um problema mais geral, que é apresentado a seguir:

$$\begin{array}{lll} \textit{dados} & : & \text{Uma } L \times C\text{-matriz } A. \\ \textit{encontrar} & : & \text{Uma } L \times R\text{-submatriz gráfica } \textit{maximal} \text{ de } A. \end{array} \tag{2.1}$$

Observemos que o algoritmo encontra uma submatriz gráfica *maximal* e não *máxima*, o que é de se esperar já que no capítulo anterior vimos que o problema de encontrar submatrizes gráficas máximas é $\mathcal{NP}$-difícil.

Este capítulo está organizado como segue. Na seção 1 veremos uma primeira tentativa de algoritmo para resolver o problema 2.1 bem como uma decomposição do problema. Na seção 2 descreveremos o algoritmo de Bixby & Wagner que resolve o problema 2.1 quando A é uma matriz sobre $GF(2)$. Finalmente, na seção 3, mostraremos um algoritmo que testa equivalência projetiva entre matrizes e a versão final do algoritmo que resolve o problema 2.1.

[Durante todo este capítulo será utilizada, de uma maneira muito forte, a correspondência entre conjuntos l.i (resp., l.d. minimal) de colunas de uma matriz gráfica e florestas (resp., circuitos) de um grafo, correspondência essa que é conseqüência da proposição 1.1 (p. 5). No próximo capítulo tornaremos explícita e estudaremos a estrutura que está subjacente a este fato, a saber: "matróides".]

2.1 Algoritmo guloso

Para resolvermos o problema 2.1 é natural que pensemos em um algoritmo guloso que, a partir de um conjunto de colunas de A que formem uma submatriz gráfica, tente expandir essa submatriz acrescentando uma a uma as demais colunas de A e testando se a submatriz resultante é gráfica. O algoritmo, ou pelo menos uma primeira tentativa, pode ser visto na figura 2.1.

Algoritmo Guloso (A)
Entrada : Uma $L \times C$-matriz A.
Saída : Uma $L \times R$-submatriz gráfica maximal de A.

início
1 $R \leftarrow$ "algum" subconjunto de C tal que a $L \times R$-submatriz de A seja gráfica;
2 $\tilde{C} \leftarrow C \setminus R$;
3 para cada coluna $c \in \tilde{C}$ faça
4 se a $L \times (R + c)$-submatriz de A é gráfica
5 então $R \leftarrow R + c$;
% A $L \times R$-submatriz de A é uma submatriz gráfica maximal
fim

Figura 2.1: Algoritmo *guloso* para encontrar uma submatriz gráfica maximal.

Nós falamos que o algoritmo *Guloso* é uma primeira tentativa pois é claro que o algoritmo da figura 2.1 encontra um conjunto maximal de colunas R tal que a $L \times R$-submatriz de A é gráfica. O que está longe de estar claro é como executar as linhas 1 e 4 do algoritmo, mais precisamente:

- Como encontrar um subconjunto de C cuja submatriz seja gráfica para inicializarmos R?
- Como testar se uma matriz é gráfica?

Nas próximas subseções estaremos buscando respontas a estas perguntas.

2.1.1 Submatriz gráfica inicial

Uma matriz $L \times C$-matriz A está na *forma canônica* se as suas colunas estão ordenadas de tal maneira que A assuma a forma tabular $[I \mid A']$, onde I denota a matriz identidade e A' o restante da matriz A.

Todos algoritmos conhecidos para reconhecer matrizes gráficas assumem que a matriz dada está na forma canônica. Se a matriz dada não está na forma canônica podemos colocá-la nesta forma de duas maneiras diferentes:

1. acrestando a identidade à matriz dada, i.e. trabalhariamos com a matriz $[I \mid A]$ ao invés da matriz A. Muitos pacotes de programação linear fazem isso de qualquer maneira.
2. fazendo pivotações (operações elementares nas linhas) para gerar a identidade (suponhamos que a matriz tem posto linha completo).

Infelizmente, ambos os metódos podem destruir a estrutura de matrizes gráficas. Entretanto o segundo método parece ser particularmente ruim, pois uma linha pode reduzir uma matriz que era quase que inteiramente gráfica a uma matriz que é quase que inteiramente não gráfica, como a matriz abaixo tenta ilustrar:

$$\begin{array}{c} \begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{array} \\ \left[\begin{array}{cccccccccccc} 1 & & & 1 & & 1 & -1 & & & & & \\ & 1 & & 1 & 1 & & 1 & & & & & \\ & & 1 & 1 & -1 & -1 & & & & & & \\ & & & 1 & & & & 1 & 1 & 1 & 1 & 1 \end{array}\right] \end{array}$$

É facil vermos que ao removermos a coluna 4 da matriz acima, a submatriz restante é inteiramente gráfica. Entretanto se pivotarmos a matriz em relação ao elemento da posição (4,4) e tivermos $R = \{1,2,3,4,5,6,7\}$ então nenhuma das colunas entre 8–12 poderão ser incluidas à submatriz gráfica (veja teorema 1.1, p. 19).

Se A é uma $L \times C$-matriz, convecionaremos que $\hat{A} := [I \mid A]$ é a $L \times (L \cup C)$-matriz mostrada na figura 2.2.

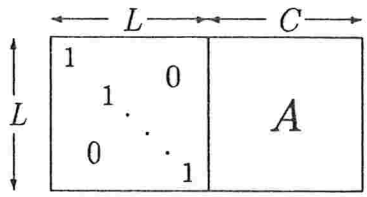

Figura 2.2: Matriz $\hat{A} := [I \mid A]$.

Assim, daqui por diante o problema que algoritmo *Guloso* estará tentando resolver não será exatamente o problema 2.1, mas sim o problema

dados : Uma $L \times C$-matriz A.
encontrar : Uma $L \times R$ submatriz gráfica *maximal* de $\hat{A}$. (2.2)

Ou seja, estaremos assumindo que a matriz de entrada para o algoritmo *Guloso* está na forma canônica.

Como é claro que a matriz identidade é gráfica, então uma maneira natural de inicializarmos o conjunto R (na linha 1 do algoritmo da figura 2.1), é inicializa-lo com o conjunto dos índices das colunas que formam a identidade de $\hat{A}$, i.e inicializaremos R com L.

2.1.2 Como testar se uma matriz é gráfica?

Lembremos que devido ao que vimos na anterior, a saída do algoritmo *Guloso* deverá ser uma submatriz gráfica maximal de $\hat{A}$ $(= [I \mid A])$.

Para executarmos as linhas 3–5 do algoritmo do algoritmo *Guloso* a idéia é tentarmos tirar proveito da $L \times R$-submatriz de $\hat{A}$ que já sabemos ser gráfica. Para isso iremos empregar a proposição 1.1, que nos dá a relação entre colunas da matriz de um grafo e as arestas do grafo.

Da proposição 1.1 sabemos que se N é uma $L \times (L \cup C)$-matriz do grafo dirigido $D = (V, L \cup C)$, então um conjunto de colunas K de N é l.i. *sse* as arestas correspondentes às colunas em K formam uma floresta em D, ou equivalentemente, um conjunto K de colunas N é *l.d. minimal sse* as arestas correspondentes às colunas em K formam um circuito em D.

Da discussão acima e do fato de combinação linear de linhas e escala de colunas serem operações que preservam dependência linear das colunas, temos que se A é uma $L \times C$-matriz tal que $\hat{A} \sim N$ (i.e. $N = \pi \hat{A} \Delta$, onde π e Δ são matrizes não singulares e além disso Δ é diagonal) onde N é matriz do grafo $D = (V, L \cup C)$, então um conjunto de colunas K de $\hat{A}$ é *l.d. minimal sse* as arestas correspondentes às colunas em K formam um circuito no grafo D. Observermos que como $\hat{A}$ tem posto linha completo então N é a matriz restrita de um grafo (i.e. N é a matriz de um grafo com posto linha completo).

Acabamos de ver uma condição *necessária* para que uma matriz seja gráfica, a saber:

> Se A é uma $L \times C$-matriz tal que $\hat{A}$ é gráfica então existe um grafo $D = (V, L \cup C)$ tal que um conjunto de colunas K de M é *l.d. minimal sse* as arestas correspondentes às colunas em K formam um circuito em D. (2.3)

Veremos mais adiante, na seção 2.3, que a condição 2.3 também é suficiente, ou seja, vale o seguinte teorema:

Teorema 2.1 (Teorema da realização) *Se A é uma $L \times C$-matriz então $\hat{A}$ é gráfica* sse *existir um grafo $D = (V, L \cup C)$ que satisfaça a seguinte propriedade:*
K é um conjunto de colunas l.d. minimal de $\hat{A}$ sse *as arestas correspondentes às colunas em K formam um circuito de D.*

Um grafo D nas condições do teorema 2.1 é dito uma realização[1] de A.

Pelo teorema da realização vemos que o problema de decidir se uma matriz $\hat{A}$ é gráfica é equivalente ao problema de encontrar uma realização para $\hat{A}$.

Devido ao que foi discutido na seção anterior sabemos que a matriz $\hat{A}$ $(= [I \mid A])$ está na forma canônica e R (linha 1 do algoritmo da figura *Guloso*) é inicializado com o conjunto de índices das colunas que formam a identidade de $\hat{A}$. Assim, no começo da execução do algoritmo, uma realização da $L \times R$-submatriz gráfica de $\hat{A}$ é qualquer floresta $D = (V, R)$. Para executarmos a linha 4 do algoritmo precisamos testar se pode ser acrescentada uma aresta c (correspondente a uma coluna de A) à D de tal forma que o grafo resultante seja uma realização da $L \times (R + c)$-submatriz de $\hat{A}$. Entretanto, logo de início temos um problema. Para percebermos qual é o problema vamos aplicar o algoritmo (o que temos até o momento)

[1] *Realization.*

à seguinte matriz:

$$\hat{A} = \begin{array}{c} \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \end{array} \tag{2.4}$$

De início $R = \{1, 2\}$ e uma realização da $L \times R$-submatriz de A pode ser uma árvore com 2 arestas (orientação arbitrária). Ao examinarmos a coluna de índice 3 vemos que essa coluna é combinação linear das colunas de índices 1 e 2, ou seja, as colunas de índices 1, 2 e 3 formam um conjunto de colunas l.d. minimal de $\hat{A}$ logo as arestas correspondentes às colunas de índices 1, 2 e 3 devem formar um circuito em qualquer realização da matriz $\hat{A}$. Ao examinarmos a coluna de índice 4 vemos que ela também é combinação linear das colunas de índices 1 e 2, assim em qualquer realização de $\hat{A}$ as arestas correspondentes às colunas de índices 1, 2 e 4 devem formar um circuito. Poderiamos pensar no grafo da figura 2.3 como uma realização de A, mas infelizmente as arestas 3 e 4 formam um circuito no grafo e as colunas de índices 3 e 4 não são colunas l.d. da matriz $\hat{A}$. Logo o grafo da figura 2.3 não é uma realização de $\hat{A}$. Na realidade pelo teorema 1.1 (p. 19) sabemos que a matriz $\hat{A}$ não é gráfica.

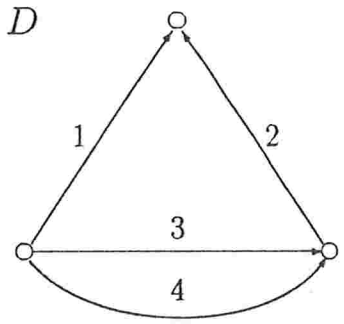

Figura 2.3: Tentativa de realização para a matriz $\hat{A}$.

Qual foi o problema ? O problema foi que acrescentamos arestas ao grafo sem verificar se as colunas correspondentes aos circuitos que estavam sendo criados formavam um conjunto l.d. minimal da matriz $\hat{A}$. Infelizmente, se tivermos que olhar para todos os circuitos que surgem ao acrescentarmos uma aresta ao grafo D e verificarmos se as colunas de $\hat{A}$ correspondentes às arestas do circuito são l.d. então o nosso algoritmo não será eficiente, pois este procedimento é certamente exponencial. O que podemos então fazer para construirmos pouco a pouco uma realização para uma submatriz de $\hat{A}$ sem que precisemos ficar nos preocupando com os eventuais circuitos que possam vir a ser criados durante o processo? A resposta para essa pergunta não é óbvia, mas o fato é que se estivessemos trabalhando com o corpo $GF(2)$ ao invés de $\mathbb{R}$ (ou qualquer outro corpo), i.e. $\hat{A}$ fosse uma matriz sobre $GF(2)$ e estivessemos considerando dependência linear sobre $GF(2)$, não haveria esse problema, como mostra a proposição 2.1 a seguir.

Proposição 2.1 *Se B é uma $L \times C$-matriz sobre $GF(2)$ tal que $\hat{B}$ $(= [I \mid B])$ é gráfica com realização $G = (V, L \cup C)$ e b é um L-vetor sobre $GF(2)$ com suporte P $(P = \| b \| \subseteq L)$ tal que as arestas de G em P formam um $u_1 - u_2$-caminho, então $[\hat{B} \mid b]$ é gráfica com realização $G + u_1u_2$.*

Prova: Seja N a matriz do grafo G. Como $\hat{B}$ é gráfica com realização G então $N = \pi B$, onde π é uma matriz não singular ($\Delta = I$, pois estamos trabalhando sobre $GF(2)$). Para provarmos que $G + u_1u_2$ é uma realização de $[\hat{B} \mid b]$ é suficiente provarmos que $|| \pi b ||= \{u_1, u_2\}$, i.e. $[N \mid \pi b]$ é a matriz do grafo $G + u_1u_2$.

Seja $\overline{B}$ e $\overline{N}$ as submatriz de $\hat{B}$ e N, respectivamente, correspondente as arestas em P (observemos que $\overline{B}$ é formada por colunas que formam a identidade de $\hat{B}$). É claro que $\overline{N} = \pi\overline{B}$ e as colunas de $[\overline{B} \mid b]$ são l.d. minimal, mas as colunas da matriz $[\overline{B} \mid b]$ são l.d. minimal *sse* a soma das colunas de $\overline{B}$ é igual a b, *sse* a soma das colunas de $\pi\overline{B}$ $(= \overline{N})$ é igual a πb, *sse* $||\pi b||= \{u_1, u_2\}$, pois as arestas em P formam um caminho em D. ■

Antes de continuarmos convém observarmos que o grafo do enunciado da proposição 1.1 é dirigido. Entretanto ao trabalhamos com matrizes de grafos sobre $GF(2)$ a orientação das arestas do grafo torna-se irrelevante (diferentes orientações do mesmo grafo dão origem a mesma matriz), estamos interessados em circuitos no grafo subjacente e não circuitos dirigidos, por isso usamos G, ao invés de D, na proposição 2.1 para denotar a realização de uma matriz B sobre $GF(2)$. Assim, ao denotarmos por G a realização de uma matriz isto significa que a orientação das arestas é arbitrária.

Uma conseqüência imediata da proposição 2.1 é o

Colorário 2.1 *Se $\hat{B}$ é uma matriz sobre $GF(2)$ e T é uma realização da submatriz identidade de $\hat{B}$ (T deve ser uma floresta) tal que $P =|| b ||$ é um caminho em T, para toda coluna b de B, então $\hat{B}$ é gráfica.* ■

Como usar a proposição 2.1 e colorário 2.1 para decidir se uma matriz A é gráfica? Parte da reposta a esta pergunta é dada na proposição 2.2 abaixo.

Proposição 2.2 *Se A é uma $L \times C$-matriz sobre $\mathbb{R}$ tal que $\hat{A}$ é gráfica e B é a $L \times C$-matriz sobre $GF(2)$ obtida a partir de A substituindo-se as entradas não nulas por 1, então $\hat{B}$ é gráfica. Mais ainda, o grafo subjacente à realização de $\hat{A}$ é uma realização de $\hat{B}$.*

Prova: Seja D uma realização de $\hat{A}$, G o grafo subjacente de D e F a floresta de G formada pelas arestas correspondentes à identidade de $\hat{A}$. Pelo colorário 2.1, para provarmos que $\hat{B}$ é gráfica é suficiente provarmos que, para toda coluna b de B, $P =|| b ||$ é o conjunto de arestas de um caminho em F.

Sejam b uma coluna qualquer de B e $P =|| b ||$. Seja ainda k o índice de b. Da definição de B não é difícil vermos que as colunas de $\hat{A}$ de índices em $P + k$ formam um conjunto de colunas l.d. minimal, logo existe em G um circuito formado pelas arestas em $P + k$. O que prova que as arestas em P são arestas de um caminho em G. ■

Uma matriz B nas condições da proposição 2.2 é dita a *imagem booleana* de A.

É fácil vermos que a recíproca da proposição 2.2 não é verdadeira, já que a matriz $\hat{A}$ em 2.4 é um contra-exemplo. Apesar de $\hat{A}$ não ser gráfica (veja teorema 1.1), a sua imagem booleana $\hat{B}$ é gráfica e o grafo da figura 2.3 é uma realização de $\hat{B}$.

A estratégia para executarmos a linha 4 do algoritmo *Guloso* será a seguinte. Suponhamos que $[I \mid A']$ é uma submatriz gráfica de $\hat{A}$. Estamos interessados em descobrir se $[I \mid A' \mid a]$ é gráfica, onde a é uma coluna de $\hat{A}$ que não está em $[I \mid A']$. Pela proposição 2.2 sabemos que se $[I \mid A' \mid a]$ é gráfica então a sua imagem booleana $[I \mid B' \mid b]$ também é gráfica. Assim, o que faremos é, a partir de uma realização de $[I \mid B']$, tentar construir uma realização para $[I \mid B' \mid b]$. Se não conseguirmos então $[I \mid A' \mid a]$ não é uma submatriz gráfica de $\hat{A}$, nesse caso descartamos a coluna a. Agora, suponha que conseguimos construir uma realização, digamos G, de $[I \mid B' \mid b]$. Infelizmente não podemos concluir que $[I \mid A' \mid a]$ é gráfica. Entretanto, devido ao lema 2.4 que veremos na seção 2.3, temos que se $[I \mid A' \mid a]$ é gráfica então o grafo D, obtido a partir de G fixando-se uma orientação arbitrária das arestas, é uma realização de $[I \mid A' \mid a]$.

Assim, o problema de decidir se uma matriz $\hat{A}$ é gráfica fica decomposto em:

1. construir, se existir, uma realização $G = (V, L \cup C)$ para a imagem booleana $\hat{B}$ de $\hat{A}$, onde $V = L + v$; e
2. testar se o grafo dirigido $D = (V, L \cup C)$, obtida a partir de G fixando-se uma orientação arbitrária de suas arestas, é uma realização de $\hat{A}$, ou equivalentemente, testar se $\hat{A}$ é p.e. a matriz de incidência do grafo D após a remoção de uma linha.

Observemos que como $\hat{A}$ tem posto linha completo e $\hat{A}$ é p.e. à matriz do grafo D, digamos N, então N tem posto linha completo. Isto quer dizer, na linguagem da seção 1.2, que N é a *matriz restrita de um grafo*.

Como veremos a dificuldade em reconhecer-se matrizes gráficas está em resolver o problema associado ao ítem 1. A resolução do item 2 é apenas um teste de equivalência projetiva entre duas matrizes, não será usado o fato de uma das matrizes ser a matriz de incidência de um grafo.

Na próxima seção descreveremos o algoritmo de Bixby & Wagner [1988] que resolve o item 1 de uma maneira muito eficiente. E na seção 2.3 veremos como pode ser testada a equivalência projetiva entre duas matrizes quaisquer.

Antes do algoritmo de Bixby & Wagner será útil que vejamos como o conceito de conexidade em grafos pode ser estendido para conexidade em matrizes. Faremos isso na próxima subseção.

2.1.3 Conexidade de matrizes

A cada $L \times C$-matriz A podemos associar o grafo bipartido $\mathcal{H}(A)$ com bipartição (L, C) onde um vértice $l \in L$ está ligado a um vértice $c \in C$ *sse* $A_{l,c} \neq 0$. Diremos que uma *matriz A é conexa sse* o grafo $\mathcal{H}(A)$ for conexo. Analogamente ao que é feito no caso de grafos, diremos que uma submatriz $\tilde{A}$ de A é uma *componente conexa*, ou simplesmente *componente*, de A se $\tilde{A}$ for uma submatriz conexa maximal de A.

Na figura 2.4 vemos o grafo $\mathcal{H}(A)$ onde A é matriz 2.5 abaixo.

$$A = \begin{array}{c} \begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{array} \\ \left[\begin{array}{cccccccccccc} 2 & & & 1 & & -3 & & & & & & \\ & & 1 & & -1 & & & & & & & \\ & & & & & & & & & & -1 & 1 \\ 1 & & & & & & & & & 1 & & \\ & & & & & & & & & -1 & & \\ & -1 & & & & & -1 & & -1 & & & \\ & & -2 & & 1 & & & & & & & \\ & & & & & & & -1 & & & & \\ -1 & & & -1 & & 1 & & & & & & \\ & 1 & & & & & 1 & & 1 & & & \\ & & & & & & & & & & -1 & 1 \end{array}\right] \end{array} \tag{2.5}$$

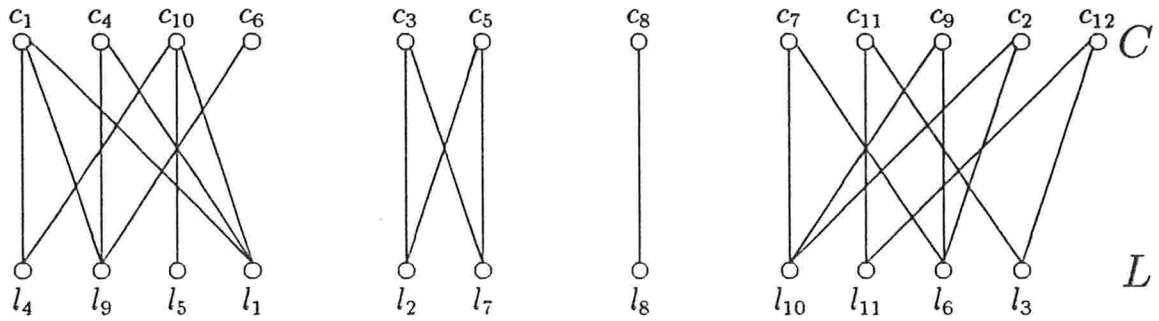

Figura 2.4: Grafo $\mathcal{H}(A)$ onde A é a matriz 2.5.

Esta definição de conexidade nada mais é do que uma extensão do conceito de conexidade em grafos, pois se D é um grafo com componentes conexas $D_1, D_2, ..., D_k$ então as linhas e colunas da matriz de D podem ser ordenadas de tal maneira que ela fique com o formato da figura 2.5, onde $N_1, ..., N_k$ são as componentes da matriz do grafo de D. Assim, não é difícil ver que, se N é a matriz de um grafo D e N é conexa, então D é conexo.

Uma ordenação que deixa evidente as componentes conexas de uma matriz, como na figura 2.5 é obtida pelo algoritmo da figura 2.6.

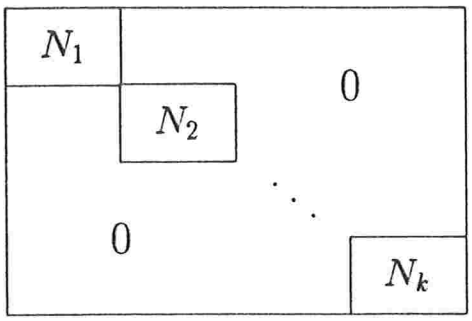

Figura 2.5: Formato da matriz de incidência de um grafo desconexo.

Algoritmo SubmatrizesConexas (A)
Entrada : Uma $L \times C$-matriz A.
Saída : Uma decomposição de A em componentes conexas $A_1, A_2, ..., A_k$.

início
1 Construa o grafo $\mathcal{H}(A)$;
2 Encontre as componentes conexas do grafo $\mathcal{H}(A)$;
3 Sejam $L_1 \cup C_1$, $L_2 \cup C_2$, ... , $L_k \cup C_k$ os vértices das componentes conexas de $\mathcal{H}(A)$.
4 As $L_i \times C_i$-submatrizes A_i são as componentes conexas de A
fim

Figura 2.6: Algoritmo que encontra as componentes conexas de uma matriz.

Abaixo vemos a matriz 2.5 após aplicarmos o algoritmo da figura 2.6.

$$A = \begin{array}{c} \begin{array}{cccccccccccc} 1 & 4 & 10 & 6 & 3 & 5 & 8 & 7 & 11 & 9 & 2 & 12 \end{array} \\ \left[\begin{array}{cccccccccccc} 1 & & 1 & & & & & & & & & \\ -1 & -1 & & 1 & & & & & & & & \\ & & -1 & & & & & & & & & \\ 2 & 1 & & -3 & & & & & & & & \\ & & & & 1 & -1 & & & & & & \\ & & & & -2 & 1 & & & & & & \\ & & & & & & -1 & & & & & \\ & & & & & & & 1 & & 1 & 1 & \\ & & & & & & & & 1 & & & -1 \\ & & & & & & & -1 & & -1 & -1 & \\ & & & & & & & & -1 & 1 & & 1 \end{array}\right] \end{array}$$

A complexidade da linha 1 do algoritmo da figura 2.6 é $O(\mid L \mid\mid C \mid)$ e a da linha 2 é $O(\mid \nu \mid)$, onde ν é o número de entradas não nulas da matriz A, i.e., o número de arestas do grafo $\mathcal{H}(A)$. Assim, a complexidade do algoritmo é $O(\mid L \mid\mid C \mid)$.

Observemos que se a entrada do algoritmo da figura 2.6 for dada em termos de uma lista de linhas e colunas das entradas não nulas, então a linha 1 pode ser executada em tempo $O(\mid \nu \mid)$ e a complexidade do algoritmo passa a ser $O(\mid \nu \mid)$.

A ligação entre matrizes conexas e matrizes gráficas é dada pela seguinte proposição:

Proposição 2.3 *Se A é uma matriz com componentes conexas $A_1, A_2, ..., A_k$, então A é gráfica* sse *A_i é gráfica, $i = 1, ..., k$.*

Prova: Se $A_1, A_2, ..., A_k$ são gráficas com realizações $D_1, D_2, ..., D_k$, então não é difícil vermos que o grafo D, cujas componentes conexas são $D_1, D_2, ..., D_k$ é uma realização de A. Reciprocamente, se A é gráfica com realização D então, devido ao teorema da realização, o subgrafo D_i formado pelas arestas correspondentes as colunas de A_i é uma realização de $A_i, i = 1, ..., k$. ■

Devido a proposição acima, podemos supor durante o restante deste capítulo que a matriz de entrada $\hat{A}$ $(= [I \mid A])$ do problema 2.1 é conexa, pois se $\hat{A}$ não é conexa podemos usar o algoritmo *SubmatrizesConexas* para encontrar as suas componentes conexas.

2.2 Algoritmo de Bixby & Wagner

O objetivo desta seção é descrever uma algoritmo eficiente para resolver o problema:

dados	:	Uma $L \times C$-matriz B sobre $GF(2)$.	
encontrar	:	Uma realização $G = (V, L \cup C)$ para $\hat{B}$ $(= [I \mid B])$, se $\hat{B}$ for gráfica.	(2.6)

Lembremos que, devido à proposição 2.3 da seção 2.1.3, estamos assumindo que B (e portanto $\hat{B}$) é uma matriz conexa.

Devido ao colorário 2.1 é fácil vermos que o problema 2.6 é equivalente ao problema:

$$\begin{array}{lll} \textit{dados} & : & \text{Uma } L \times C\text{-matriz } B \text{ sobre GF(2).} \\ \textit{encontrar} & : & \text{Uma árvore } T = (V, L), \text{ tal que } P = \| b \| \text{ é o conjunto de} \\ & & \text{arestas de um caminho em } T, \text{ para toda coluna } b \text{ de } B, \text{ se } \hat{B} \\ & & \text{for gráfica.} \end{array} \tag{2.7}$$

Vários algoritmos foram propostos para resolver o problema 2.7 (*cf.* Bixby & Cunningham [1980], Iri [1966], Truemper [1985a], Tutte [1960,1964]).

O algoritmo de Bixby & Wagner[2] [1988] que será descrito nesta seção tem a mesma complexidade de tempo que o algoritmo de Fujishige [1980], ambos têm complexidade de tempo "quase" linear no número de entradas não nulas da matriz. Esta é a melhor complexidade dentre os algoritmos conhecidos.

Os algoritmos de Fujishige e Bixby & Wagner são baseados em um procedimento exponencial proposto por Löfgren (veja Bixby & Wagner [1988]) e em maneiras "compactas" de representar todas as realizações de uma matriz gráfica. A principal diferença entre os dois algoritmos é que Fujishige utilizou fortemente uma estrutura de dados chamada de *grafo-PQ*, baseada em *árvore-PQ* (sobre árvores-PQ veja, por exemplo, Golumbic [1980: pp. 176–178]), para representar essas realizações e tornar o procedimento Löfgren polinomial enquanto que Bixby & Wagner empregaram uma técnica de decomposição em grafos.

Antes de continuarmos precisaremos de algumas definições.

2.2.1 Definições

Se G é um grafo e C é o conjunto de arestas tal que $G[C]$ é um circuito, então faremos o abuso de linguagem de também chamar C de *circuito*[3]. Faremos um abuso análogo com relação a árvores e seu conjunto de arestas. Deixaremos com o contexto o trabalho de desfazer a ambigüidade.

Um grafo conexo, sem laços e com dois vértices será chamado de um *elo*[4].

Uma árvore dirigida $T = (V, A)$ é dita uma *arborescência* se o seu grafo subjacente é uma árvore e todo vértice $v \in V$, exceto um, é o início de alguma aresta (i.e., tem uma aresta cuja ponta final é ele). O vértice que não é início de alguma aresta será chamado de *raiz* de T e será denotado por $raiz(T)$. Sejam u e v pontas inicial e final, respectivamente, de uma aresta de T. Então u é dito o *pai* de v e v um *filho* de u. Notemos que todo vértice u exceto $raiz(T)$ tem um único pai, o qual será denotado por $pai(u)$.

[2] O algoritmo é baseada na tese de doutorado de Wagner em 1983 sob a orientação de Bixby.
[3] Algumas vezes a distinção é feita chamando-se o conjunto de arestas de *cycle* ou o grafo de *polygon*.
[4] *Bond*.

Seja $G = (V, E)$ um grafo conexo e T o conjunto de arestas de uma árvore geradora de G. Para cada aresta $e \in E \setminus T$ o grafo $G[T + e]$ contém um único circuito que será chamado de *circuito fundamental* de T determinado por e. Seja M a $\{0,1\}$-matriz, cujo conjunto de índices de colunas é $E \setminus T$ e de linhas é T e o elemento da entrada (j,k) de M, $m_{j,k}$, é definido da seguinte maneira:

$$m_{j,k} := \begin{cases} 1 & \text{se } j \in C(T,k), \\ 0 & \text{caso contrário.} \end{cases}$$

Uma tal matriz M dita uma *matriz fundamental* de G em relação a T.

Uma maneira equivalente de enunciarmos o problema 2.7 é:

$$\begin{array}{lll} \textit{dados} & : & \text{Uma } L \times C\text{-matriz } B \text{ sobre } GF(2). \\ \textit{encontrar} & : & \text{Um grafo } G = (V, L \cup C), \text{ tal que } B \text{ é uma matriz fundamental} \\ & & \text{de } G. \end{array} \tag{2.8}$$

Para toda $\{0,1\}$-matriz M, gráfica ou não, chamaremos o conjunto formado pelo suporte da coluna k junto com o índice k de um *circuito fundamental* de M e o denotaremos por C_k.

Seja $G = (V, E)$ um grafo conexo, e seja $\{E_1, E_2\}$ uma partição de E. Para $k > 0$, $\{E_1, E_2\}$ é uma *k-separação* de G se

$$| E_1 | \geq k \leq | E_2 |, \quad | V(G[E_1]) \cap V(G[E_2]) | = k.$$

Para $n > 0$, G é dito *n-conexo* se ele não possui uma k-separação para $k < n$. Grafos que possuem uma 1-separação são ditos *separáveis* e grafos 2-conexos são ditos *não separáveis* ou *blocos*. É bem sabido que um grafo é 2-conexo se quaisquer duas arestas estão contidas em um circuito.

Uma *classe de paralelismo* é um conjunto maximal de arestas paralelas de um grafo. A *simplificação* de um grafo G é o grafo obtido a partir de G removendo-se os laços e removendo-se de cada classe de paralelismo todas arestas, exceto uma (uma representante da classe). Um grafo é dito *primo* se não for um circuito, nem um elo e sua simplificação for 3-conexa.

Seja $\{E_1, E_2\}$ uma 2-separação de um grafo G, tal que $V(G[E_1]) \cap V(G[E_2]) = \{u, v\}$. Definamos o grafo G' como o grafo obtido através da troca em $G[E_1]$ das arestas incidentes a u pelas arestas incidentes a v e vice-versa. Diremos que G' foi obtido a partir de G por *reversão* de $G[E_1]$ ou por uma *reversão* em relação à $\{u, v\}$. A operação de reversão está ilustrada na figura 2.7

Dois grafos G e G' são ditos *2-isomorfos* se G' puder ser obtido a partir de G através de uma seqüência de reversões. É fácil ver que se G e G' são 2-isomorfos, então eles possuem os mesmos circuitos. O que não é tão fácil de ver é que para grafos não separáveis vale a recíproca:

Teorema 2.2 (Whitney) *Se G e G' são grafos não separáveis, então G e G' são 2-isomorfos* sse *possuirem os mesmos circuitos.* ■

Uma prova curta para o teorema 2.2 pode ser encontrada em Wagner [1985].

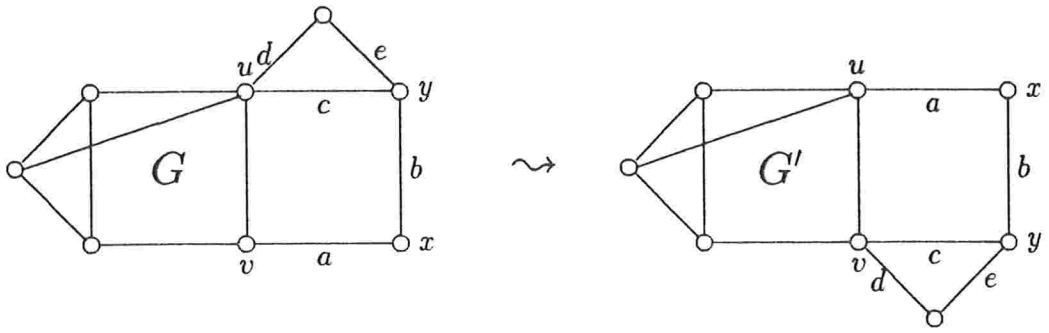

Figura 2.7: G' foi obtido a partir de G pela reversão de $G[E_1]$, onde $E_1 = \{a, b, c, d, e\}$.

2.2.2 Esboço do algoritmo

Sejam B' uma $L \times C'$-submatriz de B, tal que $\hat{B}' = [I \mid B']$ é gráfica com realização $G = (V, L \cup C')$ e b uma coluna de B que não está em B'. Um passo chave do algoritmo que veremos é a construção de uma realização para $[I \mid B' \mid b]$ a partir de G, se $[I \mid B' \mid b]$ for gráfica.

A proposição 2.1 nos dá uma dica de como a construção acima pode ser feita. Se $P = ||\, b \,||$ é o conjunto de arestas de um caminho em algum grafo 2-isormorfo a G, digamos G', então o grafo obtido a partir de G' ligando-se as pontas u_1 e u_2 do caminho P por uma aresta é uma realização de $[I \mid B' \mid b]$. Mais ainda, se G for não separável, então não é difícil vermos que, pelo teorema de Whitney, vale a recíproca, i.e., $[I \mid B' \mid b]$ é gráfica *sse* P for o conjunto de aresta de um caminho em algum grafo 2-isomorfo a G. Um caminho em algum grafo 2-isomorfo a G é dito um *hipocaminho* de G.

Assim, pelo que foi exposto acima, parece ser conveniente que em cada passo do algoritmo a realização da submatriz gráfica de $\hat{B}$ seja não separável. Por outro lado, não é muito difícil vermos que se M é uma matriz conexa com realização G, então G é não separável. Isso motivou a seguinte definição. Seja M uma $L \times C - \{0,1\}$-matriz, uma ordenação $k_1 < ... < k_{|C|}$ de C é dita *fortemente conexa* se a matriz M_{k_i}, formada pelas colunas de índice $k_1, ..., k_i$ de M, é conexa para todo i. Se M é uma matriz conexa, então não é difícil obtermos uma ordenação fortemente conexa de C. O algoritmo da *SubmatrizesConexas* (p. 30) pode computar uma tal ordenação enquanto constrói as componentes conexas de uma matriz; mais precisamente, busca em largura no grafo $\mathcal{H}(M)$ (definido na seção 2.1.3) nos fornece essa ordenação.

Se a entrada do problema 2.6 for uma $L \times C$-matriz B e uma ordenação fortemente conexa de C então ao examinarmos uma a uma as colunas de B, em ordem, tentanto expandir a submatriz gráfica $[I \mid B']$ de $\hat{B}$, teremos que a cada instante a realização de $[I \mid B']$ será não separável, a menos, eventualmente, de algumas arestas correspondentes às colunas da identidade de $\hat{B}$. Mas, isso não estraga os nossos planos do algoritmo ter uma realização conexa a cada instante, pois qualquer aresta l que corresponde a alguma coluna da identidade de $\hat{B}$ só necessita fazer parte da realização de $[I \mid B']$ se existir uma coluna b' em B' tal que $l \in ||\, b' \,||$. Isto significa que podemos fazer com que a realização, que vai sendo construída pelo algoritmo, seja sempre

não separável.

O teorema abaixo resume o que foi discutido até agora e pode ser facilmente provado a partir do teorema de Whitney.

Teorema 2.3 (Löfgren) *Se $\overline{B}$ é uma submatriz conexa de $\hat{B}$ com realização $G = (V, E)$ e b é uma coluna de B que não está em $\overline{B}$, então $[\overline{B} \mid b]$ é gráfica* sse *$P = \|b\| \cap E$ é um hipocaminho de G.* ■

Baseado no teorema 2.3, Löfgren sugeriu o algoritmo da figura 2.8 para resolver o problema 2.6.

Algoritmo Löfgren (B)
Entrada : Uma $L \times C$-matriz B e uma ordenação fortemente conexa $k_1 < ... < k_{|C|}$ de C. Observemos que é importante que a submatriz de B, que está sendo processada a cada instante, seja conexa (veja os teoremas de Whitney e Löfgren), daí a ordem das colunas de B.
Saída : Uma realização G para uma submatriz gráfica maximal de $\hat{B}$.

início
1 $R \leftarrow \{k_1\}$;
2 $E \leftarrow C_{k_1}$;
3 Seja $G = (V, E)$ um circuito;
% Diferentemente do que ocorreu até agora, aqui a ordem em que as colunas de B são
% processadas é fundamental.
4 para $i = 2, ..., |C|$ faça
início
5 $P \leftarrow C_{k_i} \cap E$
6 se P é um hipocaminho de G
então início
7 Seja G' um grafo 2-isomorfo a G, onde P é um caminho.
8 Seja G'' o grafo obtido, a partir de G' unindo-se as pontas u_1 e u_2 de P por um caminho, cujas arestas são $C_{k_i} \setminus P$;
9 $R \leftarrow R + k_i$;
10 $G \leftarrow G''$;
fim
fim
11 Seja B' a $L \times R$-submatriz de B;
% G é uma realização de $\hat{B}' = [I \mid B']$.
fim

Figura 2.8: Algoritmo de Löfgren.

A implementação da idéia de Löfgren requer um método polinomial para:

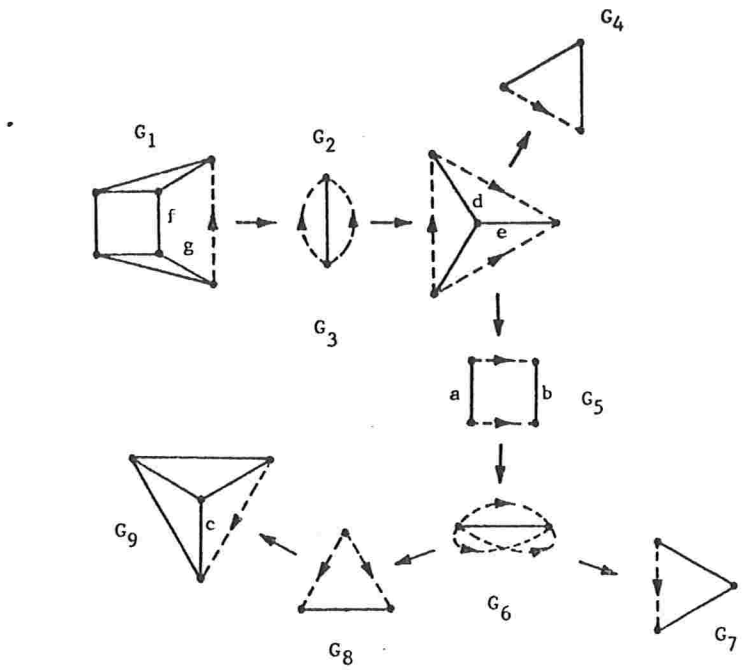

Figura 2.9: Uma t-decomposição.

i. decidir se um conjunto de arestas P de G é um hipocaminho de G (linha 6 do algoritmo) e
ii. obter o grafo G' 2-isomorfo a G (linha 7 do algoritmo).

Uma idéia natural é representar G de alguma maneira que todos os grafos 2-isomorfos a G fiquem aparentes. Para essa representação Bixby & Wagner empregaram uma decomposição do grafo G chamada de *t-decomposição.*

O restante desta seção será usada para descrever um exemplo de decomposição e ilustrar pictoricamente como ela pode ser usada para implementar a idéia de Löfgren.

Mais tarde faremos uma definição precisa de uma t-decomposição. Por enquanto uma t-decomposição é simplesmente uma arborescência cujos vértices são grafos. É assumido que cada grafo é um elo, um circuito ou primo. Dois vértices da arborescência, que são grafos, são adjacentes se eles têm uma aresta em comum, a qual é chamada de um *marcador.* Um exemplo de t-decomposição pode ser visto na figura 2.9. Uma aresta entre dois grafos indica que há um marcador comum a ambos. A raiz da decomposição é G_1. Os marcadores são representados por arestas tracejadas orientadas.

Nós associaremos com cada t-decomposição um *grafo soma* obtido pela identificação e posterior remoção dos marcadores comuns. O grafo soma da t-decomposição da figura 2.9 é mostrado na figura 2.10. A orientação dos marcadores especifica a maneira como eles devem ser identificados. Observemos que as pontas dos marcadores correspondem aos vértices do grafo soma sobre os quais podem ser feitas reversões.

Seja G o grafo da figura 2.10 e $P = \{a, b, c, d, e, f, g\}$. Cada aresta em P pertence a um grafo da t-decomposição T da figura 2.9. Consideremos a decomposição T' da figura 2.11.

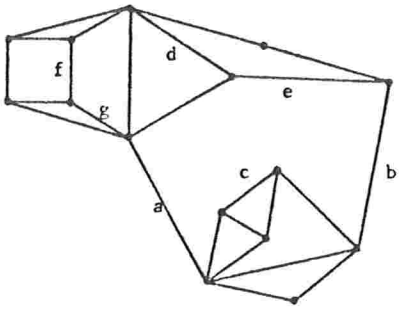

Figura 2.10: O grafo soma da t-decomposição da figura 2.9

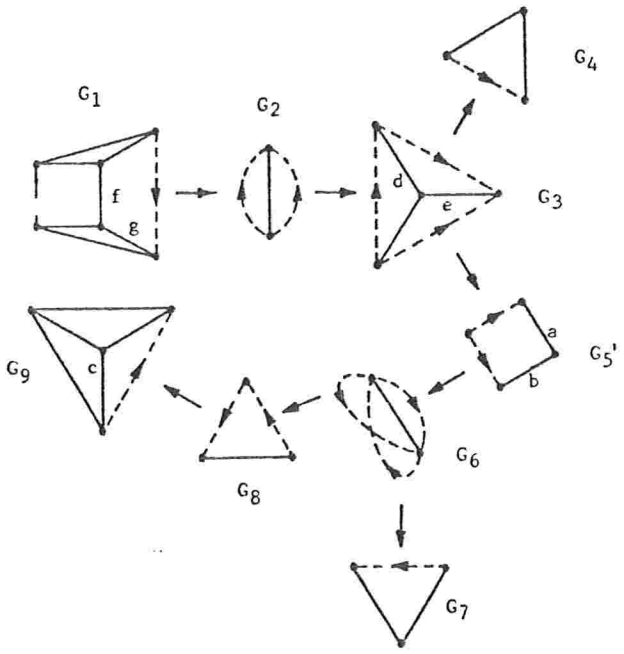

Figura 2.11: t-decomposição alterada.

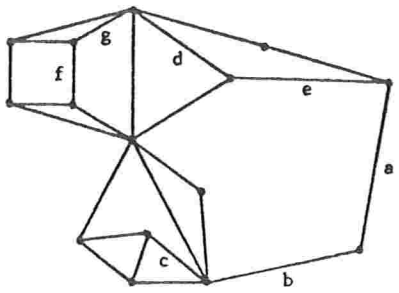

Figura 2.12: O grafo soma da t-decomposição da figura 2.11

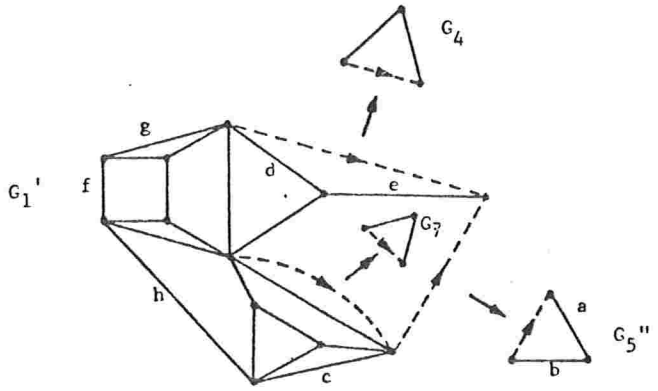

Figura 2.13: t-decomposição atualizada.

A t-decomposição T' foi obtida a partir de D através de: "reorientação" de G_1 em relação à G_2 (i.e., pela reorientação do marcador em G_1 que é comum a G_1 e G_2, isto corresponde a uma reversão no grafo soma G); "reordenação" das arestas do circuito G_5; e reorientação de G_9 em relação a G_8. O resultado final é que P é um caminho no grafo soma G' de T', o qual é mostrado na figura 2.12. O grafo G' é claramente 2-isomorfo a G, logo P é um hipocaminho de G. O algoritmo *HipoCaminho* cuidará das operações de reorientação e reordenação.

Seja h uma aresta que não está em G'. Acrescentaremos h a G' de tal maneira que h forme um circuito com as arestas de P. Uma nova t-decomposição deve ser construída para o novo grafo. Essa nova t-decomposição é mostrada na figura 2.13. Ela foi obtida fazendo-se a "soma" dos grafos G_1, G_2, G_3, G_5' de T'. O algoritmo *Atualiza* se encarregará da construção de uma nova t-decomposição.

2.2.3 O problema do hipocaminho

O problema do hipocaminho corresponde ao problema associado a linha 6 do algoritmo de

Löfgren, i.e.,

dados : Um grafo G e um conjunto de arestas P de G.
pergunta : P é um hipocaminho de G?

É fácil vermos que esse problema pode ser resolvido em tempo $O(\mid P \mid)$ se G é um grafo primo. Mais ainda, o problema pode ser resolvido em tempo constante se G for um circuito ou um elo. Veremos nessa seção um algoritmo que usando uma decomposição do grafo G, onde os membros são elos, circuitos ou primos, resolve o problema do hipocaminho.

Começaremos descrevendo uma decomposição de um grafo. Enquanto a descrição estiver sendo feita é interessante que tenhamos em mente a ilustração pictórica dada na seção 2.2.2.

Seja D uma coleção finita de grafos não separáveis, T um grafo dirigido cujos vértices são os elementos de D e dois vértices (que são grafos) são adjacentes se possuirem alguma aresta em comum. A coleção D é uma *decomposição* se T for uma arborescência e dois membros quaisquer de D têm no máximo uma aresta em comum. Se $H, K \in D$ e $\{e\} = E(H) \cap E(K)$, então e é dito um *marcador* de H e K. Se $K = pai(H)$, então e é um *marcador filho* de K e um *marcador pai* de H. O marcador pai de um membro $H \in D$ será denotado por $mp(H)$.

Será assumido pelo algoritmo que o grafo $raiz(D) = raiz(T)$ é escolhido de maneira a possuir uma aresta não pertencente a outro membro de D. Escolheremos uma aresta arbitrária, digamos m, e definiremos $m = mp(raiz(D))$. No algoritmo **sempre** será escolhida como m a aresta correspondente a primeira coluna da matriz B, a qual assumiremos ter somente uma entrada não nula. Esta escolha garantirá que m nunca estará em P (veja definição do problema do hipocaminho) e $raiz(D)$ é sempre um elo, dado que D é uma "t-decomposição".

Seja $e = mp(H)$ com pontas u_1 e u_2 em H e v_1 e v_2 em $K = pai(H)$. Uma orientação de H em relação a K é uma bijeção $f : \{u_1, u_2\} \rightarrow \{v_1, v_2\}$. Fixada f, H é dito *orientado* em relação a K. *Reorientar* H significa mudar a bijeção f. Uma decomposição é dita *orientada* se cada membro é orientado em relação ao seu pai.

Uma decomposição orientada D é dita uma *t-decomposição* se

i. todo membro de D tem pelo menos três arestas e é um circuito, um elo ou primo;
ii. somente os membros de D que são elos podem ter arestas paralelas ao seu marcador pai, e
iii. não existem dois circuitos (resp. elos) com marcadores em comum.

Reordenar um circuito de D significa transformar esse circuito em um circuito 2-isomorfo a ele, i.e., "trocar a ordem" em que as arestas aparecem no circuito.

Correspondente a decomposição acima existe um *composição*. Seja D uma decomposição orientada com $H, K \in D$, $K = pai(H)$, e $e = mp(H)$. Sejam u_1 e u_2 as pontas de e em H. *Somar* H com K corresponde a remover e de ambos os grafos e identificar u_i com $f(u_i)$, $i = 1, 2$, onde f é uma orientação de H em relação a K. O grafo resultante será denotado por $\Sigma(H, K)$.

Trocar, em D, os grafo H e K pelo grafo $\Sigma(H,K)$ dá origem a uma nova decomposição D' com um elemento a menos. Repetindo este processo mais $\mid D \mid -2$ vezes obteremos um grafo que será denotado por $\Sigma(D)$. É imediato verificar que $\Sigma(D)$ não depende da ordem em que os membros de D são somados. O grafo $\Sigma(D)$ é dito o *grafo soma* de D.

A caracteristica de uma t-decomposição que é de particular interesse aqui é a seguinte: suponha que reordenemos alguns circuitos e reorientemos alguns membros de uma t-decomposição D; como pode ser facilmente observado essas reordenações e reorientações produzirão um decomposição D' cujo grafo $\Sigma(D')$ é 2-isomorfo ao grafo $\Sigma(D)$. A seguinte versão mais refinada do teorema de Whitney, cuja prova pode ser encontrada no artigo de Bixby & Wagner, afirma que essas operações são as únicas operações necessárias para produzirmos todos os grafo 2-isomorfos a $\Sigma(D)$.

Teorema 2.4 (Whitney) *Se G e H são grafos 2-isomorfos e não separáveis e D é uma t-decomposição de G, então a decomposição D pode ser transformada, através de reordenações e reorintações de alguns de seus membros, em uma decomposição D' tal que $H = \Sigma(D')$.* ■

O teorema acima tem uma clara relevância para o problema do hipocaminho. A idéia é encontrar, se existir, uma seqüência de reordenações e reorientações dos membros da t-decomposição D de G, de tal forma a obter uma t-decomposição D', onde P seja o conjunto de arestas de um caminho em $\Sigma(D')$. Para isso será explorada a estrutura de arborescência de uma decomposição orientada de G e será feita uma "classificação" dos membros da decomposição com relação a P.

Definiremos $\hat{D}$, a *t-decomposição reduzida* de D em relação a P, como sendo a decomposição minimal contida em D que contém todos os membros de D que possuem arestas de P. É claro que P é um hipocaminho de $\Sigma(D)$ *sse* for um hipocaminho de $\Sigma(\hat{D})$. Notemos que a estrutura de arborescência de D induz uma estrutura de arborescência em $\hat{D}$.

A seguinte classificação é crucial para o algoritmo de Bixby & Wagner. Seja H um grafo, m uma aresta de H e $W_H \subseteq E(H) - m$. Considere a seguinte classificação de (H, W_H, m):

1. $W_H + m$ é um circuito;
2. W_H é um caminho e m é incidente a um vértice interno e a uma das pontas de W_H;
3. $W_H + m$ é um caminho e m está em uma das extremidades do caminho;
4. $W_H + m$ é um caminho e m não está em uma das extremidades;
5. nenhum dos casos acima ocorre.

Usando a classificação acima definiremos $Tipo(H, W_H, m)$ como sendo o menor i tal que (H', W_H, m) satisfaz (i) e H' é 2-isomorfo a H; caso $H = H'$ diremos que H é *bacana*. No que veremos adiante H será um membro de uma decomposição orientada e $m = mp(H)$.

Durante o restante desta seção usaremos a seguinte notação. $\hat{D}$ é a t-decomposição reduzida de D com relação a P, e $Q \in \hat{D}$ é fixo. Cada filho de Q é raiz de uma única t-decomposição maximal em $\hat{D}$. Sejam $D_1, D_2, ..., D_t$ essas decomposições e seja $H_i = \Sigma(D_i)$,

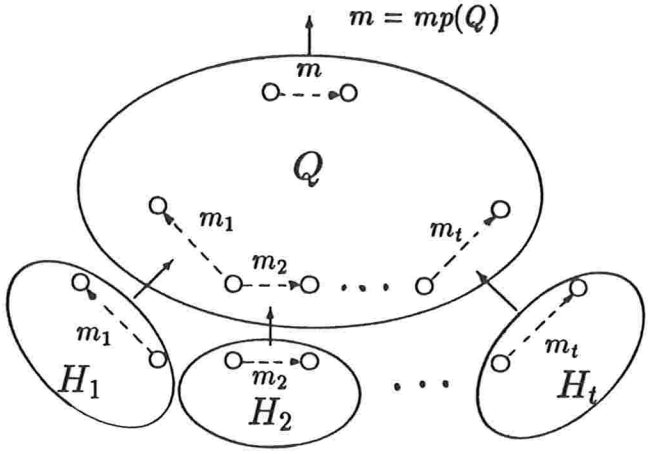

Figura 2.14: Q e os seus filhos completos $H_1, ..., H_t$.

$1 \leq i \leq t$. $H_1, ..., H_t$ são ditos *filhos completos* de Q, a situação encontra-se ilustrada na figura 2.14. Sejam $m = mp(Q)$, $H = \Sigma(Q, H_1, ..., H_2)$, $m_i = mp(H_i)$, $P_H = P \cap E(H)$ e $W_Q = (P \cap E(Q)) \cup \{m_1, ..., m_t\}$. Escreveremos $Tipo(H_i)$ para denotar $Tipo(H_i, P_{H_i}, m_i)$ e $Tipo(H)$ para denotar $Tipo(H, P_H, m)$.

Suponhamos que P é um hipocaminho de $G = \Sigma(\hat{D})$; suponhamos ainda que $Q = raiz(\hat{D})$, ou seja, $G = H = \Sigma(Q, H_1, ..., H_t)$. Como P é um hipocaminho de G, então pelo teorema 2.4, isto significa que existem grafos Q' e $H'_1, ..., H'_t$, tais que Q' é 2-isormorfo a Q, H_i é 2-isomorfo a H'_i e P é um caminho em $\Sigma(Q', H'_1, ..., H'_t)$. Observemos que $Tipo(H_i) = 2, 3$ significa que H'_i possui necessariamente uma (única) ponta do caminho P e $Tipo(H_i) = 4$ significa que as duas pontas do caminho P estão em H'_i.

Assim, é claro que se P é um hipocaminho de G então:

(a) ou $Tipo(H_i) = 2, 3$ para exatamente dois valores de i, $1 \leq i \leq t$;

(b) ou $Tipo(H_i) = 4$ para um único valor de i, $1 \leq i \leq t$.

Mais ainda, se o caso (a) acima ocorre, então W_Q é um caminho em Q' e m_j e m_k são as arestas nas extremidades do caminho, como ilustra a figura 2.15. Se o caso (b) ocorre então W_Q é um circuito em Q' (em Q também), como ilustra a figura 2.16.

As condições acima são condições necessárias para que P seja um hipocaminho de G. Devido ao teorema 2.4 não é difícil vermos que essas condições também são condições suficientes. Assim, para decidirmos se P é um hipocaminho de G precisamos somente "olhar" para o grafo Q e conhecermos os valores de $Tipo(H_1), ..., Tipo(H_t)$. Isto reduz o problema do hipocaminho ao problema de calcular o $Tipo$ dos filhos completos de membros de uma t-decomposição. Veremos a seguir, de uma maneira informal e bastante pictórica, como isso pode ser feito.

Se Q é folha da arborescência da t-decomposição $\hat{D}$, então $Tipo(H)$ $(= Tipo(Q))$ pode ser calculado facilmente (em tempo $O(\mid P_H \mid)$). No caso em que Q não é folha veremos como

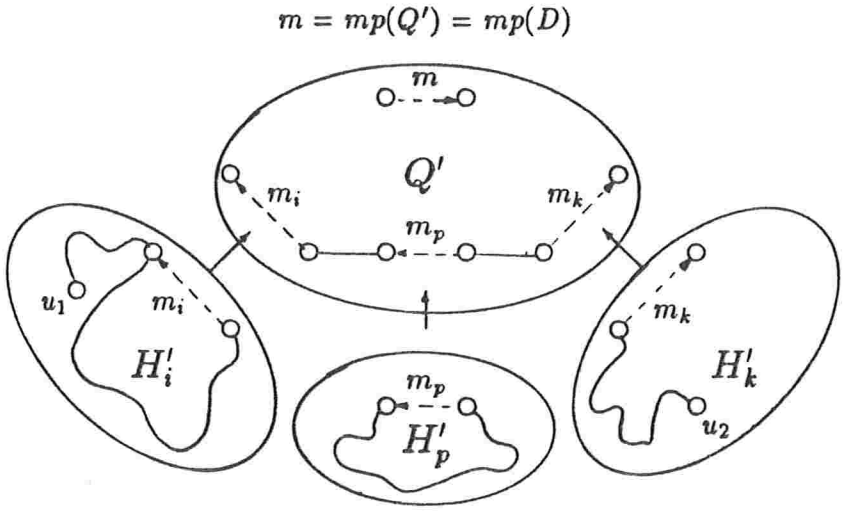

Figura 2.15: Q' e os seus filhos completos $H'_1, ..., H'_t$. Os vértices u_1 e u_2 são as pontas de P.

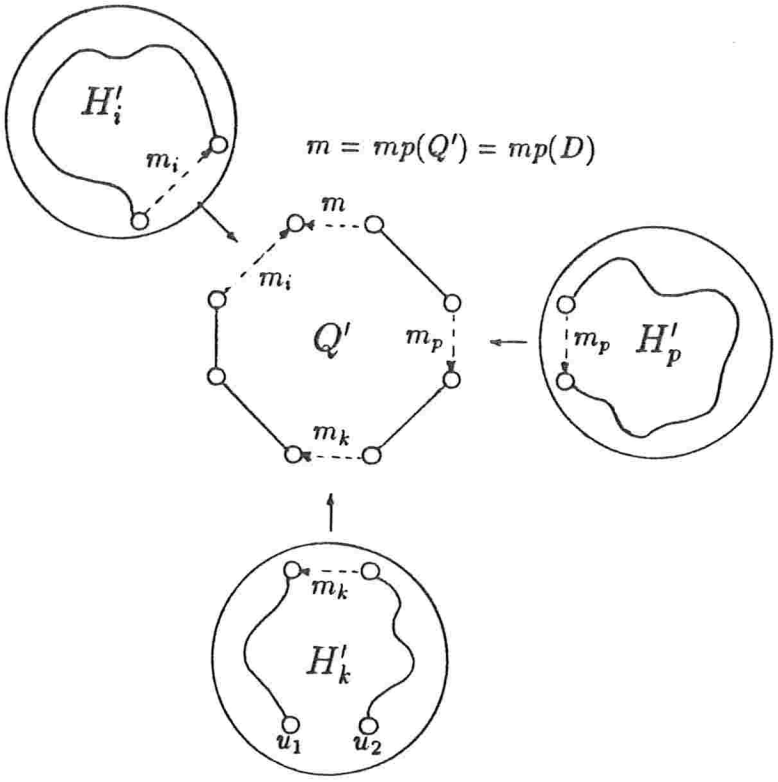

Figura 2.16: Q' e os seus filhos completos $H'_1, ..., H'_t$. Os vértices u_1 e u_2 são as pontas de P.

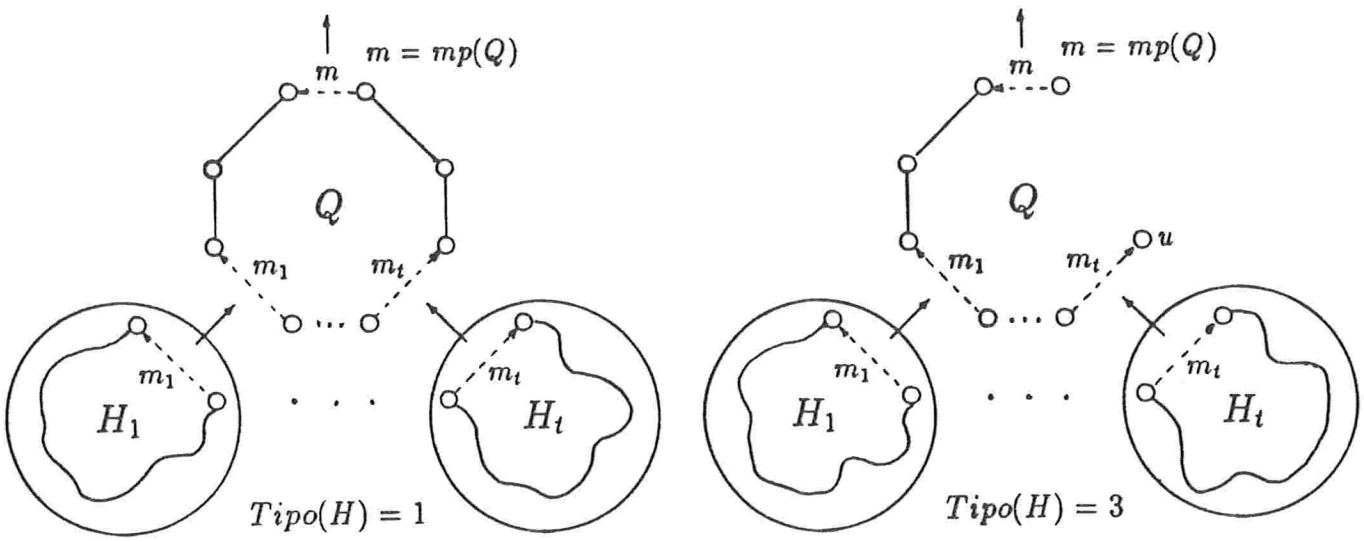

Figura 2.17: Q é um circuito e $Tipo(H_i) = 1$, $1 \leq i \leq t$. O vértice u é uma ponta de P.

$Tipo(H)$ pode ser calculado a partir de Q e do conhecimento de $Tipo(H_1), ..., Tipo(H_t)$. Vamos supor que Q é um circuito (no caso em que Q é um elo ou primo o raciocínio é semelhante).

É claro que se $Tipo(H) = 5$, então P não é um hipocaminho de G. Para que $Tipo(H)$ seja menor que 5 é necessário que:

(a) ou $Tipo(H_i) = 1$, $1 \leq i \leq t$;
(b) ou $Tipo(H_i) = 2, 3$ para no máximo dois valores de i, $1 \leq i \leq t$;
(c) ou $Tipo(H_i) = 4$ para no máximo um valor de i, $1 \leq i \leq t$.

Analisaremos, a seguir, cada um dos casos acima.

caso (a): $Tipo(H_i) = 1$, $1 \leq i \leq t$;

Podemos reordenar Q de maneira a tornar W_H um caminho com m adjacente a uma das extremidades. Neste caso se $W_Q + m = E(Q)$ então $Tipo(H) = 1$, caso contrário, $Tipo(H) = 3$. A situação está ilustrada na figura 2.17.

caso (b): $Tipo(H_i) = 2, 3$ para no máximo dois valores de i, $1 \leq i \leq t$;

Estudaremos os dois subcasos possíveis separadamente:

caso (b.1): $Tipo(H_j) = 2, 3$ e $Tipo(H_i) = 1$, $i \neq j, 1 \leq i \leq t$;

Podemos reordenar Q de tal maneira a tornar W_Q um caminho com m_j em uma das extremidades e m adjacente à outra extremidade. Suponhamos que $W_Q + m = E(Q)$, nesse caso se $Tipo(H_j) = 2$ (resp. 3), então podemos reorientar H_j, se necessário, de tal forma que $Tipo(H) = 2$ (resp. 3). A situação encontra-se ilustrada na figura 2.18. No caso em que $W_Q + m \neq E(Q)$, então podemos reorientar H_j, se necessário, de tal forma que $Tipo(H) = 3$ (independentemente de $Tipo(H_j) = 2$ ou 3). A situação está ilustrado na figura 2.19.

caso (b.2): $Tipo(H_j) = 2, 3$, $Tipo(H_k) = 2, 3$ e $Tipo(H_i) = 1$, $i \neq j, k$, $1 \leq i \leq t$;

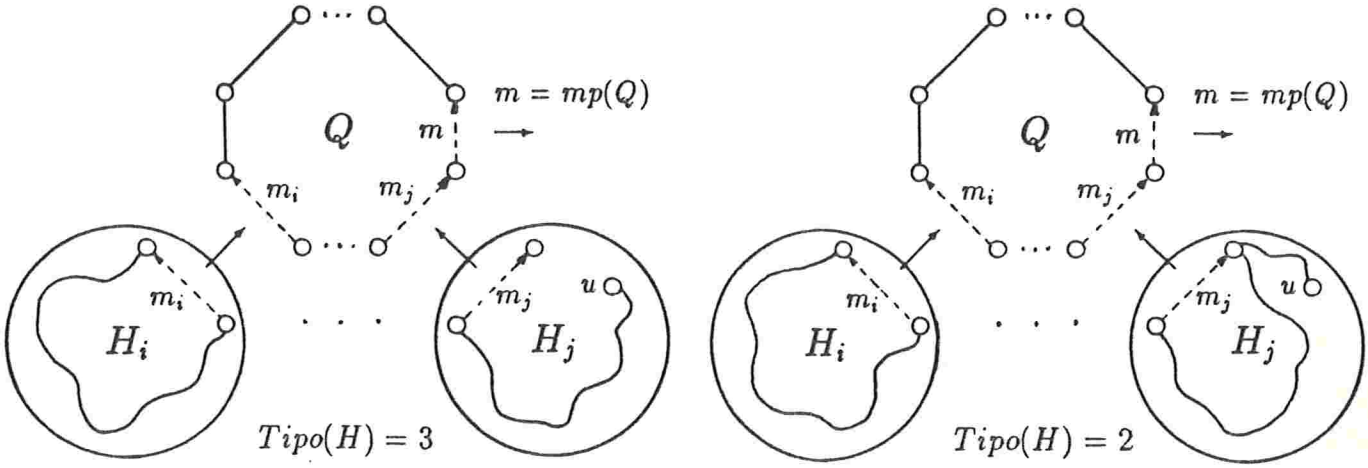

Figura 2.18: Q é um circuito, $W_Q + m = E(Q)$, $Tipo(H_j) = 3$ e $Tipo(H_i) = 1$, $i \neq j$. O vértice u é uma ponta de P.

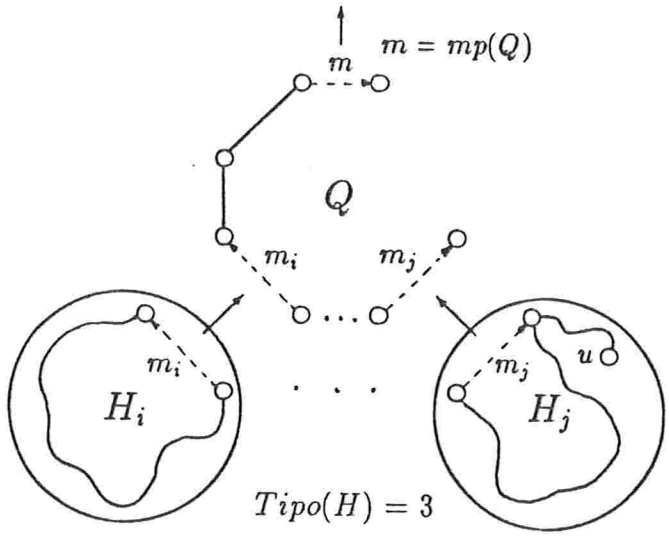

Figura 2.19: Q é um circuito, $Tipo(H_j) = 2, 3$ e $Tipo(H_i) = 1$, $i \neq j$. O vértice u é uma ponta de P.

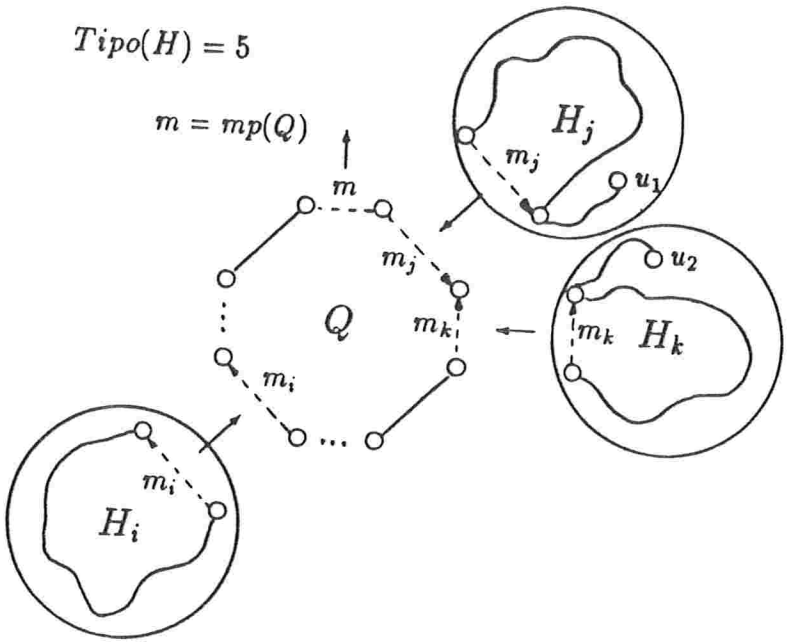

Figura 2.20: Q é um circuito, $W_Q + m = E(Q)$, $Tipo(H_j) = 2 = Tipo(H_k)$ e $Tipo(H_i) = 1$, $i \neq j, k$. Os vértices u_1 e u_2 seriam as pontas de P.

Podemos reordenar Q de maneira a tornar $W_Q - m_j$ um caminho com m_k em uma das extremidades, m adjacente a outra extremidade e m_j adjacente a m. Nesse caso, se m_j for adjacente a m_k (ou equivalentemente, $W_Q + m = E(Q)$) e $Tipo(H_j) = Tipo(H_k) = 2$, então $Tipo(H) = 5$. A situação está ilustrada na figura 2.20. Caso contrário, podemos reorientar H_j e H_k, se necessário, de tal forma que $Tipo(H) = 4$. A situação está ilustrada na figura 2.21.

caso (c): $Tipo(H_j) = 4$ e $Tipo(H_i) = 1$, $i \neq j$, $1 \leq i \leq t$;

Neste caso, se $W_Q + m \neq E(Q)$ então $Tipo(H) = 5$, caso contrário, $Tipo(H) = 4$. A situação está ilustrada na figura 2.22.

Pelo que acabamos de ver percebe-se facilmente que, durante o calculo do $Tipo$ dos filhos completos de cada membro de uma t-decomposição $\hat{D}$, é possível reordenarmos e reorientarmos os membros da t-decomposição $\hat{D}$ de tal forma a obtermos ao final uma t-decomposição $\hat{D}'$, onde todos os filhos completos de $raiz(\hat{D}')$ são bacanas. Está idéia encontra-se expressa no algoritmo $CalculaTipo$ da figura 2.23.

Com o conhecimento de $raiz(\hat{D})$ e do $Tipo$ de cada um de seus filhos completos, obtidos através do algoritmo $CalculaTipo$, é fácil decidirmos se P é um hipocaminho de $\Sigma(D)$. O algoritmo $HipoCaminho$ da figura 2.24 mostra como isto pode ser feito.

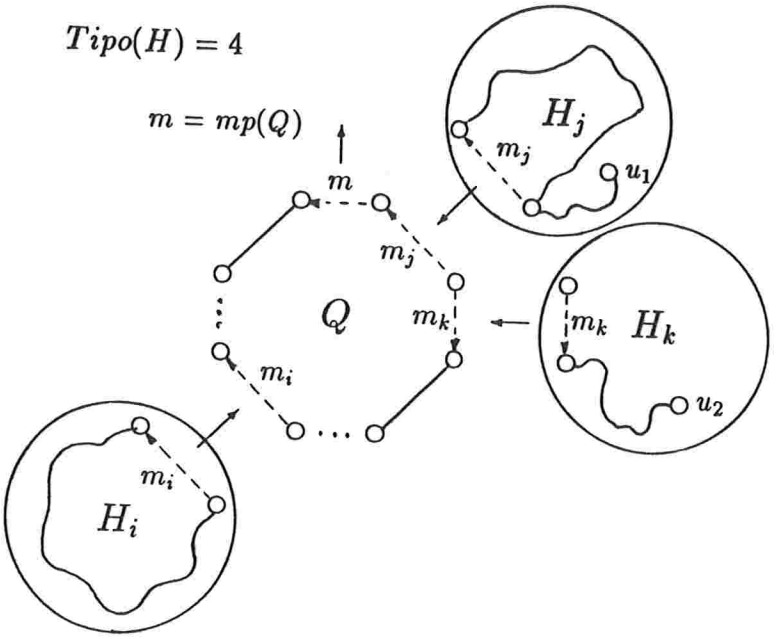

Figura 2.21: Q é um circuito, $Tipo(H_j) = 2,3$ $Tipo(H_k) = 2,3$ $Tipo(H_j) \neq Tipo(H_k)$ e $Tipo(H_i) = 1$, $i \neq j, k$. Os vértices u_1 e u_2 são as pontas de P.

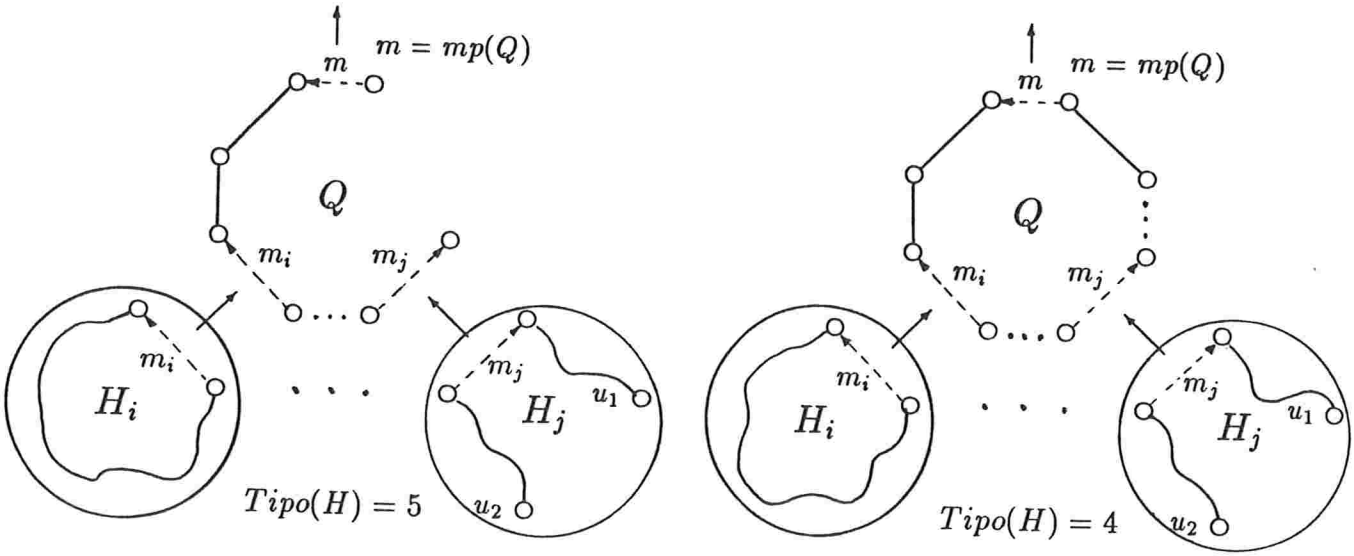

Figura 2.22: Q é um circuito, $Tipo(H_j) = 4$ e $Tipo(H_i) = 1$, $i \neq j$. Os vértices u_1 e u_2 são as pontas de P.

Algoritmo CalculaTipo $(\hat{D}, P, \pi)$
Entrada : Uma t-decomposição reduzida $\hat{D}$ correspondente a uma t-decomposição D e a um conjunto não vazio $P \subseteq E(\Sigma(D))$; a partição $\pi = (\pi_0, ..., \pi_s)$ de $\hat{D}$, onde $K \in \pi_j, 0 \leq j \leq s$, se o único caminho de K até a $raiz(\hat{D})$ na arborescência de $\hat{D}$ tem j arestas. Será assumido que $s \geq 1$.
Saída : A conclusão que P não é um hipocaminho de $\Sigma(D)$, ou uma (nova) t-decomposição $\hat{D}$ obtida a partir de reorientações e reordenações de alguns dos membros da t-decomposição $\hat{D}$ da entrada, de tal forma que cada filho completo H_i da raiz da nova decomposição $\hat{D}$ seja bacana.

início
1 para cada $H \in \pi_s$ faça
início
% H é um circuito, um elo ou primo.
2 se H é um circuito
3 então reordene H, se necessário, para transformá-lo em um circuito bacana;
4 se $Tipo(H) = 5$
5 então P não é um hipocaminho de $\Sigma(\hat{D})$;
fim

6 para $j = s - 1, ..., 1$ faça
7 para cada $Q \in \pi_j$ faça
início
8 Sejam $H_1, ..., H_t$ os filhos completos de Q.
9 se mais de dois $H_{i's}$ não são do $Tipo$ 1
10 então P não é um hipocaminho de $\Sigma(\hat{D})$;
11 se Q é um circuito
12 então reordene as arestas de Q como indicado nos casos $(a), (b)$ e (c);
13 Reoriente, se possível, $H_1, ..., H_t$ com relação a Q de tal forma a transformar $H = \Sigma(Q, H_1, ..., H_t)$ em um grafo bacana.
14 se $Tipo(H) = 5$
15 então P não é um hipocaminho de $\Sigma(\hat{D})$;
16 fim
fim

Figura 2.23: Algoritmo que determina o tipo dos filhos completos de $raiz(\hat{D})$.

Algoritmo HipoCaminho (D, P)
Entrada : Uma t-decomposição reduzida D e um conjunto $P \subseteq E(\Sigma(D))$.
Saída : Uma (nova) t-decomposição D, tal que P é um caminho de $\Sigma(D)$ e $\Sigma(D)$ é 2-isomorfo ao grafo soma da t-decomposição da entrada, ou a conclusão que P não é um hipocaminho de $\Sigma(D)$.

```
    início
1       Calcule a t-decomposição reduzida D̂ em relação a P.
2       se D̂ tem mais do que um elemento
                então início
3                       Calcule a partição π, entrada do algoritmo CalculaTipo.
4                       CalculaTipo (D̂, P, π).
                      fim

5       Q ← raiz(D̂);
6       Sejam H_1, ..., H_t o conjunto de filhos completos de Q;
7       se mais de dois H_i's não são do Tipo 1
8           então P não é um hipocaminho de Σ(D);
            senão início
9                   se Q é um circuito
10                      então Reordene Q de tal maneira que W_Q seja um caminho onde
                              os marcadores associados a filhos completos, cujo tipo é
                              diferente de 1, estejam nas extremidades do caminho;
11                  se é possível encontrar uma reorientação dos filhos completos de Q
                        de maneira que P seja um caminho de Σ(Q, H_1, ..., H_t)
12                      então P é um hipocaminho de Σ(D)
13                      senão P não é um hipocaminho de Σ(D);
15              fim
    fim
```

Figura 2.24: Algoritmo que decide se um conjunto $P \subseteq E(\Sigma(D))$ é um hipocaminho de $\Sigma(D)$. Em caso afirmativo o algoritmo obtém uma nova t-decomposição D onde P é um caminho de $\Sigma(D)$.

2.2.4 Atualização da t-decomposição

Consideremos uma t-decomposição D, um caminho P de $\Sigma(D)$ e a t-decomposição reduzida $\hat{D}$ de D em relação a P. Seja C um conjunto, tal que $C \cap E(\Sigma(D)) = P$ e $C \setminus P$ é diferente de vazio. O objetivo desta seção é mostrar como pode ser construída uma t-decomposição D^* para o grafo obtido a partir de $\Sigma(D)$, ligando-se as pontas de P por um caminho, cujo conjunto de arestas é $C \setminus P$. Isto corresponde a executar a linha 8 do algoritmo de Löfgren (pg. 35).

A idéia para construir D^* é intuitivamente simples. Primeiro selecionaremos grafos $K_1, K_2 \in D$ que contenham as pontas de P. Se $K_1 = K_2$, então D^* não será muito diferente de D. Quando $K_1 \neq K_2$ serão feitas modificações um pouco mais complicadas em D. Seja R o único caminho em D entre K_1 e K_2. Primeiro os circuitos em R serão "lapidados" e depois as arestas de $C \setminus P$ serão acrescentadas de maneira apropriada para que o grafo $\Sigma(R)$ seja primo.

Faremos agora a seguinte convenção. Para um dado vértice x de $\Sigma(D)$ poderão haver vários membros de D com um vértice, que é identificado ao vértice x no grafo soma, denotaremos qualquer um desses vértices por x.

Os grafos K_1 e K_2 podem ser obtidos naturalmente, basta apenas acrescentarmos algumas linhas aos algoritmos *CalculaTipo* e *Hipocaminho* da seção 2.2.3:

Acréscimos ao algoritmo *CalculaTipo*

5.1 se $Tipo(H) = 2, 3$
5.2 então se K_1 ainda não foi encontrado
5.3 então $K_1 \leftarrow H$
5.4 senão $K_2 \leftarrow H$;
5.5 senão se $Tipo(H) = 4$
5.6 então $K_1, K_2 \leftarrow H$;

15.1 se $Tipo(H_i) = 1$ para todo i e $Tipo(H) = 2$
15.2 então se K_1 ainda não foi encontrado
15.3 então $K_1 \leftarrow Q$
15.4 senão $K_2 \leftarrow Q$
15.5 senão se $Tipo(H_i) = 1$ para todo i e $Tipo(H) = 4$
15.6 então $K_1, K_2 \leftarrow Q$
15.7 senão se $Tipo(H_k) = 2, 3$ para algum k, $Tipo(H_i) = 1, i \neq k$ e $Tipo(H) = 4$
15.8 então $K_2 \leftarrow$ o membro no único caminho, em D entre Q e K_1 que seja mais próximo de K_1 e contenha a mesma ponta de P que está em Q;

Acréscimos ao algoritmo *HipoCaminho*

13.1 se K_1 ainda não foi encontrado
13.2 então $K_1, K_2 \leftarrow Q$
13.3 senão se K_2 ainda não foi encontrado
13.4 então $K_2 \leftarrow$ o membro no único caminho em D entre Q e K_1 que seja mais próximo de K_1 e tenha a mesma ponta de P que está em Q;

13.5 se $K_1 = K_2$
13.6 então se as pontas de P são as pontas de $mp(K_1)$ e K_1 não é um elo
13.7 então $K_1, K_2 \leftarrow pai(K_1)$
13.8 senão se as pontas de P são as pontas de um marcador m_i de K_1, K_1 é um circuito e m_i é aresta de um elo G
13.9 então $K_1, K_2 \leftarrow G$
14 Seja $u_1 \in K_1$ e $u_2 \in K_2$ as pontas de P em $\Sigma(D)$.

Não é difícil entendermos as linhas acima se tivermos em mente o que já foi observado na seção 2.2.3, i.e., $Tipo(H) = 2,3$ significa que H contém uma ponta de P e $Tipo(H) = 4$ significa que H contém as duas pontas de P.

Suponhamos que $K_1 \neq K_2$. Sejam u_1, u_2 as pontas de P em $\Sigma(D)$ com $u_i \in K_i$, $i = 1,2$ e R o único caminho em D entre K_1 e K_2. A linha 15.8 do algoritmo *CalculaTipo* e a linha 13.4 do algoritmo *HipoCaminho* têm por objetivo escolher K_2 (observemos que K_1 foi encontrado anteriormente) de maneira que não exista membro algum em R, entre K_1 e K_2, que contenha u_1 ou u_2. Observemos ainda que o fato de $m = mp(raiz(D))$ não pertencer a outro membro de D simplifica a condição da linha 13.8 do algoritmo *HipoCaminho*.

Tendo em mãos K_1 e K_2 estamos preparados para construir a nova t-decomposição D^*, mas, antes vejamos o que quisemos dizer por "lapidar" os circuitos em R.

Lapidar um circuito C de R em relação a um conjunto de arestas L, $\mid L \mid > 1$, que formam um caminho em C, significa trocar C em D^* pelos circuitos formados por $L + f$ e $(C \setminus L) + f$, onde f é um novo marcador de D^* (veja figura 2.25). No caso em que $\mid L \mid = 1$ a lapidação de C em relação a L não produz efeito algum.

Convém lembramos que uma t-decomposição possui a estrutura subjacente de uma arborescência. Ao lapidarmos um circuito os novos membros da nova t-decomposição têm uma orientação natural que é induzida pela arborescência. Todas as operações que faremos a seguir têm essa mesma caracteristica, i.e., a orientação dos novos membros é facilmente obtida da orientação da atual arborescência. A orientação dos marcadores que serão criados é arbitrária.

O apêndice A mostra um pequeno exemplo de execução do algoritmo de Bixby & Wagner, esse exemplo pode ser bastante útil para a visualização das operações feitas no algoritmo *Atualiza* da figura 2.26, bem como dos algoritmos *CalculaTipo* e *HipoCaminho*.

Observemos que pode existir no máximo um índice i, $2 \leq i \leq s$, tal que o grafo G_i

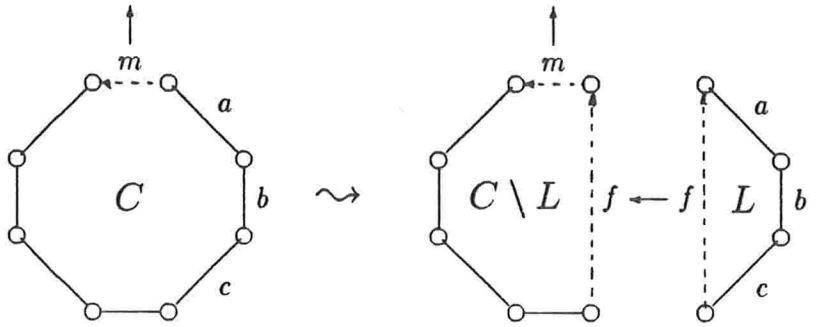

Figura 2.25: Lapidação do circuito C com relação ao caminho $L = \{a, b, c\}$. m é o marcador pai de C.

satisfaça a condição da linha 14 do algoritmo *Atualiza* e o candidato para isso é $raiz(\hat{D})$, onde $\hat{D}$ é a t-decomposição reduzida em relação a P. O objetivo da linha 15 é evitar que o grafo $\Sigma(R)$ tenha arestas paralelas ao seu marcador pai, e a lapidação dos circuitos em R atua no sentido de que, ao final do algoritmo, o grafo G da linha 19 do algoritmo seja primo.

Através de uma leitura cuidadosa do algoritmo *Atualiza* não é muito difícil nos convensermos que ao final do algoritmo $\Sigma(D^*)$ é o grafo obtido a partir de $\Sigma(D)$ acrescentando-se arestas $C \setminus P$ de tal forma que C seja o conjunto de arestas de um circuito. Podemos ainda verificar que D^* é de fato uma t-decomposição, i.e. D^* tem as seguintes propriedades:

(a) é uma decomposição;

(b) cada membro tem pelo menos três arestas;

(c) não existem circuitos (resp. elos) adjacentes;

(d) nenhum membro que é um circuito ou primo tem arestas paralelas aos seu marcador pai, e

(e) todo membro é um circuito, ou um elo ou primo.

O ponto chave para verificar-se os passos acima está no lema que se segue.

Lema 2.1 *O grafo G da linha 19 do algoritmo Atualiza é primo.* ■

Uma prova para o lema acima, bem como a verificação de que D^* é um t-decomposição, pode ser encontrada no artigo de Bixby & Wagner.

Na subseção a seguir veremos o algoritmo de Löfgren escrito em termos do trabalho de Bixby & Wagner.

Algoritmo Atualiza $(D, P, C, K_1, K_2, u_1, u_2)$
Entrada : Uma t-decomposição D, um caminho P de $\Sigma(D)$ e um conjunto C tal que $C \cap E(\Sigma(D)) = P$ e $C \setminus P$ é não vazio; membros K_1 e K_2 de D e vértices u_1, u_2 de K_1, K_2, respectivamente, tal que u_1 e u_2 são as pontas de P em $\Sigma(D)$ e não existe $G \neq K_1, K_2$ no único caminho entre K_1 e K_2 em D que contenha u_1 ou u_2.
Saída : Uma t-decomposição D^* do grafo obtido a partir de $\Sigma(D)$ após a conecção de u_1 e u_2 por um caminho de arestas $C \setminus P$ (a ordem das arestas é irrelevante).

início
1 $D^* \leftarrow D$;
2 se $|C \setminus P| = 1$
3 então Seja $\{f\} = C \setminus P$
4 senão Seja f um novo marcador. Crie um circuito, cujo conjunto de arestas é $(C \setminus P) + f$ (a ordem das arestas é irrelevante) e acrescente a D^* esse circuito.
5 se $K_1 = K_2$
6 então se K_1 não é um circuito
7 então Ligue u_1 e u_2 com a aresta f
8 senão se u_1 e u_2 são adjacentes % K_1 é um circuito
9 então Seja e a aresta em K_1 que, é incidente a u_1 e u_2 e f' um novo marcador. Troque e por f' em K_1 e acrescente o elo formado por $\{f', e, f\}$ a D^*.
10 senão Sejam L_1, L_2 os dois caminhos disjunto de u_1 a u_2 em K_1 e f_1, f_2 novos marcadores. Remova K_1 de D^* e acrescente os circuitos formados por $L_i + f_i$, $i = 1, 2$ e o elo formado por $\{f_1, f, f_2\}$.
senão início
11 Sejam $R = (K_1 = G_1, a_1, G_2, ..., a_{s+1}, G_{s+1} = K_2)$ o único caminho entre K_1 e K_2 e $\{m_i\} = E(G_i) \cap E(G_{i+1}), i = 1, ..., s$;
12 para $i = 2, ..., s$ faça
13 se G_i é primo e m_{i-1}, m_i são arestas paralelas ou G_i é um elo com pelo menos 4 arestas e $pai(G_i) \notin R$
14 então Seja f' um novo marcador. Sejam G'_i o grafo obtido a partir de G_i após a remoção de m_{i-1} e m_i e a adição de f' (com as mesmas pontas); e B o elo formado por $\{m_{i-1}, f', m_i\}$.
Troque G_i em R por B e remova G_i de D^* e acrescente G'_i e B.
15 senão se G_i for um circuito
16 então Sejam L_1 e L_2 as duas componentes de $G_i - \{m_{i-1}, m_i\}$; Lapide o circuito G_i com relação a L_1 e o circuito resultante (que contém L_2) com relação a L_2.
17 se $K_i, i = 1, 2$, é um circuito
18 então Sejam L_1, L_2 os caminhos entre as pontas do marcador pai de K_i e u_i. Lapide K_i com relação a L_1 e o circuito resultante com relação a L_2.
19 Seja G o grafo $\Sigma(R)$ com as pontas u_1 e u_2 ligadas pela aresta f.
% G é um grafo primo !
20 Remova de D^* os membros de R e acrescente o grafo G.
fim
fim

Figura 2.26: Algoritmo que acrescenta um caminho ao grafo $\Sigma(D)$ e atualiza a sua t-decomposição.

2.2.5 Algoritmo principal e estrutura de dados

Começaremos esta seção descrevendo, na figura 2.27, o algoritmo de Bixby & Wagner, que tornou o procedimento de Lõfgren polinomial através da manipulação e manutenção de uma t-decomposição D. Durante a execução do algoritmo, $\Sigma(D)$ é realização de uma submatriz gráfica de $\hat{B}$. O algoritmo de ambos se utiliza dos algoritmos *HipoCaminho* (p. 48) e *Atualiza* (p. 52).

A estrutura de dados proposta por Bixby & Wagner é elementar. Os membros da t-decomposição possuem uma estrutura de arborescência subjacente. Assim, é necessário que cada membro H da t-decomposição tenha um apontador para o membro $pai(H)$. Esses apontadores, entre outras coisas, serão utilizados no cálculo da t-decomposição reduzida $\hat{D}$ em relação a P (linha 1 do algoritmo *HipoCaminho*).

O algoritmo *Hipocaminho* e *CalculaTipo* requerem uma estrutura de dados simples. De fato, para testarmos se P é um hipocaminho de $\Sigma(D)$ não precisamos muito mais do que:

(a) a designação de cada membro de $\hat{D}$ (se ele é um elo, um circuito ou primo);

(b) o marcador pai de cada membro;

(c) para cada membro H de D, diferente de $raiz(D)$, está associado um marcador filho de $pai(H)$. Precisaremos de apontadores de H para esse marcador filho e vice-versa.

O algoritmo *Atualiza* necessita de uma estrutura de dados que permita que a operação de soma entre os membro de D seja feita rapidamente. Lembremos que, pela definição de decomposição, D é um conjunto de grafos. Quando somamos alguns membros de D, digamos $G_1, ..., G_k$, o que estamos fazendo no fundo é trocar esses elementos em D pelo elemento $\{G_1, ..., G_k\}$. Isso sugere que a estrutura de dados para a t-decomposição, além de ser uma arborescência, permita que a união (disjunta) dos seus membros possa ser feita de maneira eficiente. O mesmo ocorre com os vértices dos membros de D, pois estes eventualmente precisam ser identificados quando somamos dois membros. Assim, os membros de D bem como os vértices de seus membros devem ser armazenados de tal forma que a união disjunta de conjuntos seja feita rápidamente.

Bixby & Wagner provaram que o limite computacional do algoritmo *Grafo* vem da atualização das t-decomposições e não do problema do hipocaminho. Mais ainda, ambos provaram que a complexidade do algoritmo *Grafo* é a mesma complexidade do algoritmo utilizado para a de união disjunta de conjuntos .

A estrutura de dados mais eficiente para o problema da união disjunta de conjuntos que é conhecida é a *união disjunta de conjunto com compressão de caminhos* que pode ser encontrada, por exemplo, em Terada [1982: pp. 76–87] e Horowitz & Sahni [1984: pp. 70–79]).

[A estrutura para união disjunta de conjuntos consiste em dados dois elementos distintos x e y:

$cria(x)$: cria o conjunto $\{x\}$ de nome x. É assumido que x não pertence a outro conjunto;

$nome(x)$: encontra o nome do conjunto contendo x e

Algoritmo Grafo (B)
Entrada : Uma $L \times C$-matriz B e uma ordenação $k_1 < ... < k_{|C|}$ fortemente conexa de C. É assumido que a coluna k_1 de B tem apenas uma entrada não nula (uma tal coluna pode ser sempre acrescentada a B sem afetar o fato de uma submatriz ser gráfica ou não).
Saída : Uma realização G para uma submatriz gráfica maximal de $\hat{B}$ $(= [I \mid B])$.

início
1 Seja G um elo com duas arestas;
% G é uma realização para a submatriz de $\hat{B}$ formada pela coluna k_1 de B
% e por uma coluna da identidade de $\hat{B}$.
2 Seja $mp(G)$ a aresta de G correspondente a coluna k_1 de B;
3 $D \leftarrow \{G\}$;
% Notemos que neste ponto D é uma decomposição, mas não uma t-decomposição
% pois G tem apenas 2 arestas. Este será o único momento em que isto ocorre.

% A ordem em que as colunas são processadas é importante.
4 para $i = 2, ..., \mid C \mid$ faça
início
5 $P \leftarrow C_{k_i} \cap E(\Sigma(D))$;
6 $HipoCaminho(D, P)$;
7 se P é um hipocaminho de $\Sigma(D)$
então início
8 $Atualiza(D, P, C_{k_i}, K_1, K_2, u_1, u_2)$;
% Observemos que K_1, K_2, u_1, u_2 (entradas do algoritmo
% *Atualiza*), foram obtidos pelo algoritmo *HipoCaminho*.
9 $D \leftarrow D^*$; % D^* é saída do algoritmo *Atualiza*.
fim
fim

10 $G \leftarrow \Sigma(D)$;
% G é uma realização para uma submatriz gráfica maximal de $\hat{B}$.
fim

Figura 2.27: Algoritmo de Bixby & Wagner.

união(x, y) : faz a união dos conjuntos contendo x e y.

Os conjuntos são armazenados como arborescências onde a raiz contém o nome do conjunto e cada vértice contém o nome de um elemento. Executar *nome*(x) significa percorrer a arborescência desde o vértice contendo x até a raiz "comprimindo" esse caminho de tal forma que cada vértice nele passe a apontar diretamente para a raiz. A operação *união*(x, y) corresponde a executar *nome*(x) depois *nome*(y) e criar uma aresta que aponte da raiz da arborescência menos profunda até a raiz da mais profunda.

Suponhamos que o algoritmo *Grafo* utilize a estrutura acima para armazenar os membros de D e os vértices dos membros de D. Assim, se e é uma aresta que indica que H é o membro ao qual ela pertence e que as suas pontas são u e v, então o nome corrente do membro ao qual e pertence é *nome*(H) e de suas pontas é *nome*(u) e *nome*(v).]

Teorema 2.5 (Tarjan) *A união disjunta de conjuntos com compressão de caminhos tem complexidade de tempo* $O(m\alpha(m, n))$, *onde* m *é o número de operações: cria, nome e união; e* n *é o número de elementos do conjunto.* ■

A função $\alpha(m, n)$ que aparece no teorema de Tarjan é definida em termos da função de Ackerman $A(i, j)$ como $\alpha(m, n) = \min\{i \geq 1 \mid A(i, \lfloor m/n \rfloor) > \log_2 n\}$, $m \geq n \geq 1$, ($\lfloor x \rfloor = \max\{i \mid i \leq x\}$). A função $\alpha(m, n)$ é uma função que cresce muito lentamente, para propósitos práticos ela nunca é maior que 4 ($A(4, 1) = f(17)$, onde $f(1) = 2$ e $f(k) = 2^{f(k-1)}$ para $k \geq 2$. Portanto, $f(4) = 2^{65536}$).

Assumindo que a matriz B é dada através de uma lista de entradas não nulas por coluna e utilizando a estrutura de dados acima, principalmente o teorema de Tarjan, Bixby & Wagner provaram o teorema 2.6 a seguir, onde ν denota o número de entradas não nulas da matriz B.

Teorema 2.6 (Bixby & Wagner) *Dado uma* $L \times C - \{0, 1\}$*-matriz com* ν *entradas não nulas, o algoritmo Grafo tem complexidade de tempo* $O(\nu\alpha(\nu, \mid L \mid))$ *e complexidade de espaço* $O(\nu)$. ■

A análise completa do algoritmo, explicações de como se usar a estrutura de dados, bem como a prova do teorema 2.6 encontra-se no artigo de Bixby & Wagner.

2.3 Teste de equivalência projetiva

Lembremos que na seção 2.1 o problema associado à execução da linha 4 do algoritmo *Guloso* (p. 23), i.e., o problema de decidir se uma matriz $\hat{A}$ ($= [I \mid A]$) é gráfica, foi decomposto em dois subproblemas, a saber:

1. construir, se existir, uma realização $G = (V, L \cup C)$ para a imagem booleana $\hat{B}$ de $\hat{A}$, onde $V = L + v$ e

2. testar se o grafo $D = (V, L \cup C)$ (obtido a partir de G fixando-se uma orientação arbitrária das arestas) é uma realização de $\hat{A}$, ou equivalentemente, se $\hat{A}$ é p.e. à matriz de incidência de D após a remoção de uma linha.

O problema do item 1 foi resolvido na seção 2.2. O que faremos nessa seção é resolver o problema do item 2. Na seção 2.2 a orientação das aresta de G era irrelevante, agora é chegado o momento de olharmos para uma orientação arbitrária de G .

O problema de decidir se duas matrizes (gráficas ou não) são p.e. é bem conhecido e, como veremos, a sua solução não levará em conta se uma das matrizes é gráfica.

O método que veremos para testar equivalência projetiva entre duas matrizes baseia-se no lema 2.2 a seguir, que afirma, que equivalência projetiva entre duas matrizes na forma canônica pode ser verificada de maneira muito simples.

Lema 2.2 *Se X e Y são $L \times C$-matrizes tal que $\hat{X} \sim \hat{Y}$, então existem matrizes diagonais não singulares Δ_1 e Δ_2 tal que $\hat{X} = \Delta_1 \hat{Y} \Delta_2$.*

Prova: Se $\hat{X} \sim \hat{Y}$ então por definição existem matrizes não singulares π e Δ tal que Δ é diagonal e $\hat{X} = \pi \hat{Y} \Delta$. Como $\hat{X}$ e $\hat{Y}$ estão na forma canônica então $\hat{X} = [I \mid X]$ e $\hat{Y} = [I \mid Y]$. Segue que existe uma submatriz apropriada Δ' de Δ tal que $\pi \Delta' = I$, o que implica que π é diagonal. ■

Sejam A uma $L \times C$-matriz sobre $\mathbb{R}$, $\hat{B}$ a imagem booleana da matriz $\hat{A}$ e $G = (V, L \cup C)$ uma realização da matriz $\hat{B}$ obtida pelo algoritmo *Grafo* (p. 55), onde $V = L + v$. Chamaremos de $D = (V, L \cup C)$ o grafo obtido a partir de G após fixarmos uma orientação arbitrária de suas arestas e N a sua matriz de incidência.

O lema 2.2 sugere que para verificarmos se $\hat{A} \sim \tilde{N}$, onde $\tilde{N}$ é a matriz obtida a partir de N após a remoção da linha correspondente ao vértice $v \in V$, primeiro transformemos $\tilde{N}$ em uma matriz na forma canônica. Entretanto, uma tal transformação requer um esforço computacional muito grande ($O(\mid L \mid^2 \mid C \mid)$). Assim, trabalharemos diretamente com o grafo D, para construirmos uma matriz $\hat{M} = [I \mid M] \sim \tilde{N}$.

Observemos que como $\hat{A}$ é conexa então pela proposição 1.1 (p. 5) $\tilde{N}$ tem posto linha completo.

Definiremos a $L \times C$-matriz M como segue. Seja $T = (V, L)$ o subgrafo de D induzidos pelas arestas de L (as arestas em L corresponde às colunas da identidade de $\hat{B}$). Pela proposição 1.1, T é uma árvore geradora de D. Para cada aresta $c \in C$ temos que $T + c$ possui um único circuito, o *circuito fundamental* de T determinado por c. Seja T_c o conjunto de arestas na árvore que fazem parte do circuito. Consideremos a orientação do circuito $T_c + c$ que tem o mesmo sentido da aresta c e definamos

$$m_{l,c} := \begin{cases} +1 & \text{se } l \in T_c \text{ tem o mesmo sentido da orientação do circuito;} \\ -1 & \text{se } l \in T_c \text{ tem o sentido contrário ao da orientação do circuito;} \\ 0 & \text{se } l \notin T_c. \end{cases} \tag{2.9}$$

A matriz M é dita *matriz fundamental* de D com relação a T[5].

Abaixo vemos a matriz $\hat{M} = [I \mid M]$, onde M é a matriz fundamental do grafo D da figura 2.28 em relação a árvore $T = \{1, 2, 3\}$.

$$\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} 1 & & & +1 & \\ & 1 & & -1 & -1 \\ & & 1 & & +1 \end{array}\right] \end{array}$$

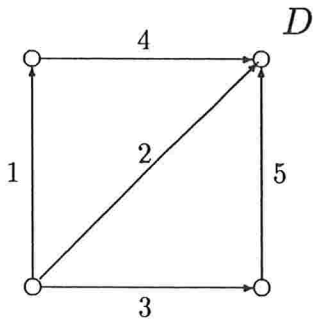

Figura 2.28: $T = \{1, 2, 3\}$ é uma árvore geradora de D.

Verifiquemos que $\hat{M}$ é de fato p.e. a $\tilde{N}$. Seja P a $V \times L$-matriz de incidência de $T = (V, L)$ e $\tilde{P}$ a matriz P após a remoção da linha correspondente ao vértice v. Pela proposição 1.1 $\tilde{P}$ é não singular. Podemos verificar facilmente que $P\hat{M} = N$. Assim, $\tilde{P}\hat{M} = \tilde{N}$ e portanto $\hat{M} = \tilde{P}^{-1}\tilde{N}$, o que prova que $\hat{M}$ é p.e. a $\tilde{N}$.

A matriz $\hat{M}$ pode ser contruída em tempo $O(\mid L \mid\mid C \mid)$, na realidade se armazenarmos $\hat{M}$ na forma de lista de suas entradas não nulas então $\hat{M}$ pode ser construída em tempo $O(\mid \nu \mid)$, onde ν é o número de entradas não nulas de A.

Tendo em mãos a matriz $\hat{M}$, que é p.e. a $\tilde{N}$, resta agora apenas testarmos se $\hat{A} \sim \hat{M}$.

Consideremos o grafo $\mathcal{H}(A)$ com bipartição (L, C) onde um vértice $l \in L$ é adjacente a um vértice $c \in C$ *sse* $A_{l,c} \neq 0$. Para cada aresta $e = lc \in V(\mathcal{H}(A))$ seja a_e e m_e os valores de $A_{l,c}$ e $M_{l,c}$, respectivamente. Como veremos, testar equivalência projetiva entre as matrizes $\hat{A}$ e $\hat{M}$ é equivalente a descobrir se existe um $L \cup C$-vetor μ tal que $\mu_x a_e \mu_y = m_e$, para toda aresta $e \in E(\mathcal{H}(A))$, $e = xy$.

Um algoritmo que testa equivalência projetiva entre duas matrizes na forma canônica é exibido na figura 2.29.

É fácil vermos que a complexidade do algoritmo $TestaPE$ é $O(\mid L \mid\mid C \mid)$.

[5] M também é dita uma *network matrix* (veja TLIP, p.276)

Algoritmo TestaPE (A, M)
Entrada : Duas $L \times C$-matriz conexas A e M.
Saída : A conclusão de que $\hat{A} \not\sim \hat{M}$ ou um $L \cup C$-vetor μ tal que $\mu_x a_e \mu_y = m_e$, para toda aresta $e \in E(\mathcal{H}(A)), e = xy$.

início
1 Construa o grafo $\mathcal{H}(A)$ com bipartição (L, C) onde um vértice $l \in L$ é adjacente a um vértice $c \in C$ *sse* $A_{l,c} \neq 0$.
Para cada aresta $e = lc \in V(\mathcal{H}(A))$ seja a_e e m_e os valores de $A_{l,c}$ e $M_{l,c}$, respectivamente.
2 Construa uma árvore geradora T de $\mathcal{H}(A)$ de raiz r.
% $\mathcal{H}(A)$ possui uma árvore geradora pois A é conexa.

% As linhas 4–8 abaixo podem ser executadas simultaneamente à construção
% da árvore T.
3 $\mu_r \leftarrow 1$;
4 Percorra a árvore T de vértice *pai* para vértice $filho$ definindo o vetor μ da seguinte maneira, onde $x = pai(y)$ e $e = xy$:
5 $\mu_y \leftarrow m_e/(\mu_x a_e)$;

6 se para cada aresta $e \in E(\mathcal{H}(A)), e = xy$ temos que $\mu_x a_e \mu_y = m_e$
7 então $\hat{A} \sim \hat{M}$
8 senão $\hat{A} \not\sim \hat{M}$
fim

Figura 2.29: Algoritmo que testa equivalência projetiva entre duas matrizes na forma canônica.

Observemos que como A é uma matriz conexa então $\mathcal{H}(A)$ é um grafo conexo (veja seção 2.1.3) e como algoritmo $TestaPE$ percorre uma árvore geradora de $\mathcal{H}(A)$ então os valores de μ_x estarão bem definidos para todo $x \in L \cup C$. Observemos ainda que se $\hat{A} \sim \hat{M}$ então, pelo lema 2.2, A e M têm exatamente as mesmas entradas não nulas.

Teorema 2.7 *Dadas duas $L \times C$-matrizes, A e M, o algoritmo $TestaPE$ decide corretamente se $\hat{A}$ é projetivamente equivalente a $\hat{M}$.*

Prova: Suponhamos que o algoritmo $TestaPE$ conclua que $\hat{A} \sim \hat{M}$. Assim, devido à linha 6 do algoritmo, é fácil verificarmos que $\hat{M} = \pi \hat{A} \Delta$, onde π é uma $L \times L$-matriz diagonal tal que $\pi_{l,l} = \mu_l$ e Δ é uma $L \cup C \times L \cup C$-matriz diagonal tal que $\Delta_{l,l} = \mu_l^{-1}$, para $l \in L$ e $\Delta_{c,c} = \mu_c$, para $c \in C$. Portanto, a conclusão do algoritmo é correta.

Suponhamos agora que o algoritmo $TestaPE$ conclua que $\hat{A} \not\sim \hat{M}$. Se essa conclusão é falsa, então pelo lema 2.2 existe um $L \cup C$-vetor η, tal que para toda aresta $e \in \mathcal{H}(A), e = lc$, $\eta_l a_e \eta_c = m_e$. Agora seja r a raiz da árvore T construída na linha 2 do algoritmo. É claro que podemos assumir, sem perda de generalidade, que $\eta_r = 1$. Mas, não é difícil vermos que, isto implica que $\eta = \mu$, que é a contradição que estavamos procurando. ■

Devido a prova do teorema 2.7 é fácil vermos que o vetor μ calculado pelo algoritmo $TestaPE$ é único a menos de multiplicação por escalar.

Um algoritmo muito semelhante ao algoritmo $TestaPE$ pode ser usado para transformar uma matriz A em uma $\{0, \pm 1\}$-matriz, ou determinar que uma tal transformação não é possível. Um tal algoritmo poderia ser empregado para detectarmos mais rápidamente que uma matriz não é gráfica. Entretanto, isso não eliminaria a necessidade do algoritmo $Grafo$, já que nem toda $\{0, \pm 1\}$-matriz é gráfica (veja teorema 1.1, p. 19).

Na proposição 2.4 vemos a prova de que se $\hat{A}$ é uma matriz gráfica então D, a matriz obtida a partir de uma orientação arbitrária das arestas do grafo G saída do algoritmo $Grafo$, é uma realização de $\hat{A}$.

Proposição 2.4 *Se A é uma $L \times C$-matriz tal que $\hat{A}$ é gráfica, B é a imagem booleana de A, G é uma realização para $\hat{B}$ e D é uma grafo obtido a partir de G fixando-se uma orientação arbitrária de sua arestas, então D é uma realização de $\hat{A}$.*

Prova: Como $\hat{A}$ é gráfica então pelo teorema da realização (p. 25) existe um grafo $D' = (V, L \cup C)$ realização de $\hat{A}$. Seja G' o grafo subjacente a D'. Para provarmos a proposição é suficiente provarmos que G e G' têm exatamente os mesmos circuitos, i.e. G e G' são grafos 2-isomorfos.

Seja M uma $L \times C$-matriz construída a partir de D' como definido em 2.9 e seja T' a árvore de G' correspondente às colunas da identidade de $\hat{M}$. Pela definição de M é claro que para cada coluna m de M temos que $\| m \|$ é o conjunto de arestas de um caminho em T'. Como, pelo lema 2.2, M e B têm exatamente as mesmas entradas não nulas então para

cada coluna b de B temos que $||b||$ é um caminho em T'. Assim, pelo colorário 2.1, G' é uma realização de $\hat{B}$ e portanto G' e G são grafos 2-isomorfos. ■

Antes de encerrarmos este capítulo resta ainda provarmos o teorema da realização (p. 25). Mas antes, provaremos o lema 2.3, a seguir, que será nos auxiliará a provar o teorema da realização.

Lema 2.3 *Se A é uma $L \times C$-matriz, A' é uma $L' \times C'$-submatriz de A, $D = (V, L \cup C)$ é uma realização de $\hat{A}$ e D' é o grafo obtido a partir de D após a remoção das arestas em $C \setminus C'$ e contração das arestas em $L \setminus L'$, então D' é uma realização de $\hat{A}'$.*

Prova: Precisamos provar que um subconjunto K' de colunas de $\hat{A}'$ é l.d. minimal *sse* K' é o conjunto de arestas de um circuito em D'. Denotaremos por A_i (resp., A'_i) a coluna de índice i da matriz A (resp., A').

Se $\{A'_{k_1}, ..., A'_{k_t}\}$ é um conjunto l.d. minimal de colunas de A' *sse* existem $\alpha_1, ..., \alpha_t \in \mathbb{R}$ tal que $\sum_{i=1}^{t} \alpha_i A'_{k_i} = 0$. Seja $P = ||\alpha_1 A_{k_1} + ... + \alpha_t A_{k_t}|| \ (\subseteq L \setminus L')$. É fácil ver que o conjunto das colunas de A de índices em $P \cup \{k_1, ..., k_t\}$ é l.d. minimal e portanto $P \cup \{k_1, ..., k_t\}$ é o conjunto de arestas de um circuito em D, donde $\{k_1, ..., k_t\}$ é o conjunto de arestas de um circuito em D'.

Reciprocamente, $\{k_1, ..., k_t\}$ é o conjunto de arestas de um circuito em D' *sse* existe um conjunto de arestas P $(\subseteq L \setminus L')$ de D tal que $P \cup \{k_1, ..., k_t\}$ é o conjunto de arestas de um circuito em D. Assim, existem $\alpha_1, ..., \alpha_t \in \mathbb{R}$ tal que $\sum_{i=1}^{t} \alpha_i A_{k_i} + \sum_{i \in P} \alpha_i A_{k_i} = 0$. Não é difícil verificar que $\sum_{i=1}^{t} \alpha_i A'_{k_i} = 0$ e que $A'_{k_1}, ..., A'_{k_t}$ é um conjunto l.d. minimal. ■

O lema 2.3 mostra que se uma matriz $\hat{A}$ possui uma realização então toda submatriz de $\hat{A}$ possui uma realização, o que era de se esperar se assumissemos o teorema da realização como verdadeiro, já que submatrizes de matrizes gráficas são gráficas.

Para provarmos o teorema da realização restava apenas provarmos a suficiência da condição 2.3 (p. 25), isso é feito através do lema 2.4 a seguir.

Lema 2.4 *Se A é uma $L \times C$-matriz e $D = (V, L \cup C)$ é uma realização de $\hat{A}$, então $\hat{A}$ é gráfica.*

Prova: É claro que podemos assumir que A é conexa. Seja M uma $L \times C$-matriz construida a partir de D como definido em 2.9. Considere a aplicação do algoritmo $TestaPE(A, M)$. Existe uma árvore T de $\mathcal{H}(A)$ e um $L \cup C$-vetor μ tal que $\mu_l a_e \mu_c = m_e$, para toda aresta $e = lc \in E(T)$. Se $\hat{A} \not\sim \hat{M}$ então existe uma aresta $e = lc \notin E(T)$, tal que $\mu_l a_e \mu_c \neq m_e$. Portanto existe um circuito Q do grafo $\mathcal{H}(A)$ tal que $\mu_l a_e \mu_c \neq m_e$ para exatamente uma aresta $e = lc \in E(Q)$. Seja L' o subconjunto de linhas e C' o subconjunto de colunas correspondentes aos vértices de Q. Se A' é a $L \times C$-matriz definida por $a'_{l,c} := \mu_l a_{l,c} \mu_c$, para $l \in L$ e $c \in C$, então os conjuntos L' e C' determinam submatrizes A'_1 de A' e M_1 de M que têm exatamente duas entradas não nulas em cada linha e coluna e que são identicas a menos de uma única entrada não nula. Seja

α e β o valor dessa entrada em A'_1 e M_1, respectivamente. Para números a e b, $a \neq 0$, temos que determinante de A'_1 é $\alpha a + b$ e determinante de M_1 é $\beta a + b$. Por outro lado, é claro que D é uma realização de A' e portanto o grafo D_1 resultante de D após a remoção das arestas $C \setminus C'$ e contração das arestas $L \setminus L'$ é uma realização de $\hat{A}'_1$ e de $\hat{M}_1$. Como A'_1 tem exatamente duas entradas não nulas por coluna então não é difícil ver que todos os circuitos fundamentais de D_1 são triângulos e que as arestas da árvore $T_1 = D_1[L]$ formam a estrela de um vértice, portanto D_1 é um grafo roda. Como D_1 é uma realização de $\hat{A}'_1$ e de $\hat{M}_1$ então o determinante de A'_1 e M_1 são ambos igual a zero, mas isso é possível *sse* $\alpha = \beta$, que é a contradição que estavamos procurando. ■

Na figura 2.30 vemos o algoritmo *Guloso*, na sua forma final, utilizando-se dos algoritmos $Grafo$ e $TestaPE$.

É claro que em cada execução dos algoritmos $Grafo$ e $TestaPE$ (linhas 6 e 10 do algoritmo, respectivamente) informações sobre a submatriz que já sabemos ser gráfica pode ser mantida (a t-decomposição construida pelo algoritmo $Grafo$ e o vetor μ calculado pelo algoritmo $TestaPE$) de tal forma a simplificar a próxima iteração. Assim, o trabalho total feito pelos algoritmos $Grafo$ e $TestaPE$, em uma execução do algoritmo *Guloso*, têm complexidade de tempo "quase" linear no número de entradas não nulas da matriz A e $O(\mid L \mid\mid C \mid)$, respectivamente. Logo, a complexidade do algoritmo *Guloso* é $O(\mid L \mid\mid C \mid)$.

No apêndice A é mostrada a execução do algoritmo *Guloso* tendo como entrada a matriz de restrições de um PL que é uma rede escondida, bem como a conversão desse PL a um PL onde a matriz de retrições é a matriz restrita de um grafo (matriz de um grafo com posto linha completo), como já foi mencionado na seção 1.3.

Algoritmo Guloso (A)
Entrada : Uma $L \times C$-matriz A.
Saída : Uma realização D para uma $L \times R$-submatriz gráfica maximal de $\hat{A}$.

início
1 Seja B a imagem booleana de A.
2 Consideremos uma ordenação $k_1 < ... < k_{|C|}$ fortemente conexa de C.
% Uma tal ordenação pode ser obtida utilizando-se uma variação do algoritmo
% *SubmatrizesConexas* (p. 30).
3 $C' \leftarrow \emptyset$;
% A ordem em que as colunas são processadas é importante.
4 para $i = 1, ..., |C|$ faça
início
5 Seja B' a $L \times (C' + k_i)$-submatriz de B.
6 $Grafo(B')$; % execução do algoritmo $Grafo$;
7 se o algoritmo $Grafo$ decidiu que B' é gráfica
então início
8 Seja D' a realização de B contruída pelo algoritmo $Grafo$ após fixarmos uma orientação arbitrária das arestas.
9 Seja M a matriz construída a partir de D' como descrito em 2.9 (p. 56) e A' a $L \times (C' + k_i)$-submatriz de A.
10 $TestaPE(A', M)$; % execução do algoritmo $TestaPE$
13 se o algoritmo $TestaPE$ decidiu que $A' \sim M$
então início
14 $C' \leftarrow C' + k_i$;
15 $D \leftarrow D'$;
16 fim
fim ;
fim ;
16 $R \leftarrow L \cup C'$;
% A $L \times R$-submatriz de $\hat{A}$ é uma submatriz gráfica maximal com realização D.
fim

Figura 2.30: Algoritmo *guloso* para encontrar uma submatriz gráfica maximal.

Capítulo 3

Matróides e o reconhecimento de matrizes gráficas

Como pudemos observar pelos capítulos anteriores, grafos e matrizes (de incidência, de grafos ou gráficas) se comportam de maneira muito semelhante. Isto pode ser notado, por exemplo, através da proposição 1.1 que afirma que um conjunto K de colunas da matriz de incidência de um grafo G é l.i. (resp., l.d. minimal) *sse* as arestas de G correspondentes às colunas em K formam uma floresta (resp., um circuito) em G.

Isso sugere que existe, subjacente a grafos e os seus circuitos e matrizes e os seus conjuntos de colunas l.d. minimal, uma estrutura abstrata comum a ambos. Este fato foi primeiro trazido a tona por Whitney [1935] em seu trabalho antológico "*On the abstract properties of linear dependence*". A esta abstração de grafos e matrizes Whitney deu o nome de *matróide*.

Neste capítulo estudaremos um pouco dessa estrutura combinatória que são os matróides e qual é o papel dessa teoria no reconhecimento de matrizes gráficas. Como veremos, muito do que fizemos nos capítulos anteriores pode ser escrito de uma maneira mais simples em termos de matróides, de fato muito do que vimos até agora foi originalmente escrito usando a linguagem de teoria dos matróides. Definições equivalentes de matróides bem como alguns exemplos encontram-se na seção 1. Alguns conceitos como isormorfismo, soma direta, conexidade e dualidade de matróides serão mostrados na seção 2. Na seção 3 veremos como matróides se relacionam com os algoritmos vitos no capítulo 2. Finalmente, na seção 4, serão descritos alguns resultados em matróides sobre um problema mais geral do que o reconhecimento de matrizes gráficas, a saber, o reconhecimento de matróides "gráficos".

3.1 Matróides: definição e exemplos

Como veremos matróides são uma abstração de: matrizes e a dependência linear entre suas colunas; e de grafos e os seus circuitos. Podemos definir um matróide de várias maneiras diferentes, muitas das quais foram descritas por Whitney em seu trabalho original, dentre essas várias maneiras veremos somente as que mais nos interessarão. A primeira dessas, inclusive

na nomenclatura, tem sua origem do ponto de vista de teoria dos grafos. Sem mais delongas vamos aos matróides.

Um *matróide* M é um conjunto finito E e uma coleção $\mathcal{C}$ de subconjuntos de E que satisfazem as propriedades (C1) e (C2) abaixo:

(C1) se $C_1 \in \mathcal{C}$ e $C_2 \subset C_1$, então $C_2 \notin \mathcal{C}$;

(C2) se C_1, C_2 são membros distintos de $\mathcal{C}$ e $e \in C_1 \cap C_2$, então existe $C_3 \in \mathcal{C}$ tal que $C_3 \subseteq (C_1 \cup C_2) - e$.

Diremos que $M = (E, \mathcal{C})$ é um matróide sobre E.

Exemplos de matróides:

Matróide gráfico: Se E é o conjunto de arestas de um grafo G e $\mathcal{C}$ é o conjunto de circuitos de G então o par $(E, \mathcal{C})$ é um matróide, chamado de *matróide gráfico* de G, o qual será denotado por $\mathcal{M}(G)$.

Matróide Linear: Se V é um espaço vetorial sobre um corpo F, E é um conjunto finito de vetores de V e os membros de $\mathcal{C}$ são os subconjuntos de vetores de E que são l.d. minimais sobre F, então o par $(E, \mathcal{C})$ é um matróide. No que veremos, E será tipicamente o conjunto de colunas de uma matriz sobre $\mathbb{R}$ ou GF(2).

Os membros do conjunto $\mathcal{C}$ são ditos os *circuitos* de M (analogia com grafos). Consideremos um conjunto $C \in \mathcal{C}$. Se $C = \{e_1\}$ diremos que e_1 é um *laço* matróide M. Se $C = \{e_1, e_2\}$, diremos que e_1 e e_2 são *elementos em paralelo* (analogia com grafos).

Motivados pela analogia com álgebra linear (principalmente independência linear) é natural que definamos um matróide M como sendo um conjunto finito E e uma coleção $\mathcal{I}$ de subconjuntos de E (chamados de conjuntos *independentes*) que satisfaçam (I1)–(I3) a seguir:

(I1) $\emptyset \in \mathcal{I}$ (i.e. $\mathcal{I} \neq \emptyset$);

(I2) se $I_1 \in \mathcal{I}$ e $I_2 \subseteq I_1$, então $I_2 \in \mathcal{I}$;

(I3) se I_1, I_2 são membros de $\mathcal{I}$ e $\mid I_1 \mid = \mid I_2 \mid + 1$, então existe $e \in I_1 \setminus I_2$ tal que $I_2 + e \in \mathcal{I}$.

Um subconjunto de E que não está em $\mathcal{I}$ é dito *dependente* (analogia com espaços vetoriais).

Ambas as definições de matróides que acabamos de ver são equivalentes. De fato, se $\mathcal{C}$ é o conjunto de circuitos de um matróide M sobre E então $\mathcal{I} = \{I \subseteq E \mid \text{não existe } C \in \mathcal{C}, C \subseteq I\}$ é o conjunto dos independentes de M. Reciprocamente, se $\mathcal{I}$ é o conjunto de independentes de um matróide M sobre E e $\mathcal{C}$ é o conjunto dos dependentes minimais de M, então $\mathcal{C}$ é o conjunto dos circuitos de M.

Assim, podemos definir matróides gráficos e matróides lineares, de maneira equivalente à que fizemos, especificando o conjunto dos independentes desses matróides da seguinte maneira:

Matróides Gráficos: Se E é o conjunto de arestas de um grafo G então o conjunto $\mathcal{I}$ das arestas de G que formam florestas em G é o conjunto dos independentes do matróide gráfico de G ($\mathcal{M}(G)$).

Matróides Lineares: Se V é um espaço vetorial sobre um corpo F e E é um conjunto finito de vetores de V então o conjunto $\mathcal{I}$ dos subconjuntos de vetores em E que são l.i. é o conjunto de independentes do matróide linear sobre E.

Seguindo a analogia com espaços vetoriais faremos as seguintes definições.

Uma *base* de um matróide M é um conjunto independente maximal, a coleção das bases é denotado por $\mathcal{B}$.

A função *posto* de um matróide m sobre E é a função do conjunto das partes de E nos inteiros definida por

$$\rho(A) := \max\{|I| \mid I \subseteq A, I \in \mathcal{I}\} \quad (A \subseteq E).$$

Muitas vezes escreveremos simplesmente ρA e ρM ao invés de $\rho(A)$ e $\rho(E)$, respectivamente.

Vimos que o conhecimento do conjunto de independentes ou de circuitos de um matróide é suficiente para defini-lo de maneira única. De maneira semelhante, não é difícil verificarmos que o conhecimento das bases ou da função posto também definem um matróide de maneira única. Assim, é de se esperar que também hajam axiomas que definam um matróide em termos desses conceitos.

Axiomas de Base: Uma coleção $\mathcal{B}$ de subconjuntos de um conjunto E é o conjunto das bases de um matróide *sse* $\mathcal{B}$ satisfaz as propriedades (B1)-(B3) a seguir:

(B1) $\mathcal{B} \neq \emptyset$;

(B2) $|B_1| = |B_2|$, para todo $B_1, B_2 \in \mathcal{B}$;

(B3) se $B_1, B_2 \in \mathcal{B}$ e $e_1 \in B_1 \setminus B_2$, então existe $e_2 \in B_2 \setminus B_1$ tal que $(B_1 - e_1) + e_2 \in \mathcal{B}$.

Axiomas de Posto: Uma função ρ do conjunto das partes de E nos inteiros é a função posto de um matróide sobre E *sse* valem (R1)-(R3), a seguir, para $X \subseteq E, e_1, e_2 \in E$:

(R1) $\rho(\emptyset) = 0$;

(R2) $\rho(X) \leq \rho(X + e_1) \leq \rho(X) + 1$;

(R3) se $\rho(X + e_1) = \rho(X + e_2) = \rho(X)$, então $\rho(X + e_1 + e_2) = \rho(X)$.

Outro exemplo de matróide:

O Matroide Uniforme $U_{k,n}$: Se E é um conjunto de cardinalidade n e $\mathcal{I}$ e a coleção de todos os subconjuntos de E de cardinalidade menor ou igual a k, então o par $(E,\mathcal{I})$ é um matróide sobre E, o qual é denotado por $U_{k,n}$, o *matróide uniforme* de posto k. As bases, circuitos e função posto de $U_{k,n}$ estão listadas abaixo:

$$\begin{aligned}
\mathcal{B}(U_{k,n}) &= \{B \subseteq E \mid |B| = k\},\\
\mathcal{C}(U_{k,n}) &= \{C \subseteq E \mid |C+1| = k+1\},\\
\rho A &= \begin{cases} |A| & \text{se } |A| \leq k \\ k & \text{se } |A| > k \end{cases}
\end{aligned}$$

A coleção dos independentes, dos circuitos e das bases de um matróide M serão denotadas por $\mathcal{I}(M),\mathcal{C}(M)$ e $\mathcal{B}(M)$, respectivamente. Quando não houver dúvidas sobre qual o matróide que está sendo tratado escreveremos simplesmente $\mathcal{I},\mathcal{C}$ e $\mathcal{B}$ ao invés de $\mathcal{I}(M),\mathcal{C}(M)$ e $\mathcal{B}(M)$.

Se $M = (E,X)$ é um matróide, onde $X = \mathcal{I}(M),\mathcal{C}(M),\mathcal{B}(M)$ ou a função posto de M, então diremos simplesmente que M é um matróide sobre E. o qual denotaremos por $M(E)$.

Em Welsh [1976] (primeira monografia e referência "canônica" sobre matróides) podem ser encontradas outras definições equivalentes e muito mais exemplos de matróides.

Na próxima seção veremos alguns conceitos sobre matróides que serão empregados até o final desta dissertação.

3.2 Alguns conceitos em teoria dos matróides

Dois matróides M_1 e M_2 sobre E_1 e E_2, respectivamente, são ditos *isomorfos* se existe uma bijeção entre E_1 e E_2 que preserva independência (e portanto preserva circuitos, bases, posto, etc.). Escreveremos $M_1 \simeq M_2$ se M_1 e M_2 são isomorfos.

Consideremos o matróide $U_{2,3}$ sobre um conjunto E qualquer. Todo subconjunto próprio de E é independente em $U_{2,3}$. Assim, se um grafo G é um circuito com três arestas, então é claro que $\mathcal{M}(G) \simeq U_{2,3}$. Por outro lado, não existe um grafo G tal que $\mathcal{M}(G) \simeq U_{2,4}$, pois não existe grafo com quatro arestas tal que qualquer três arestas formam um circuito.

Um matróide M é dito *gráfico* se existe um grafo G tal que $\mathcal{M}(G) \simeq M$. O grafo G é dito uma *realização* de M. Logo $U_{2,3}$ é gráfico e $U_{2,4}$ não é gráfico.

Dois grafos $G_1 = (V_1,E_1)$ e $G_2 = (V_2,E_2)$ são *2-isormorfos* se existe uma bijeção entre suas arestas que preserva circuitos (estamos generalizando um pouco o conceito de 2-isormorfismo feito na seção 2.2.1, pois aqui não estamos exigindo que $E_1 = E_2$). É claro que quaisquer grafos 2-isomorfos têm o mesmo matróide (i.e. os matróides são isormorfos).

Se A é uma $L \times C$-matriz sobre um corpo F, denotaremos por $M(A)$ o matróide sobre

C tal que $C' \subseteq C$ é dependente *sse* as colunas de A cujos índices estão em C' formam um conjunto l.d. minimal como vetores de F^L. É claro que $M(A)$ é isomorfo ao matróide linear sobre o conjunto dos vetores formados pelas colunas de A vistos como vetores do espaço F^L.

Os vetores $(1,1,0),(1,0,1)$ e $(0,1,1)$ são linearmente dependentes sobre GF(2). O matróide linear sobre esses vetores é isormorfo ao matróide $U_{2,3}$. Os mesmos vetores são linearmente independentes sobre $\mathbb{R}$ e o matróide linear desses vetores (sobre $\mathbb{R}$) é isomorfo ao matróide $U_{3,3}$. Um matróide M é dito *representável* sobre um corpo F se existe uma matriz A sobre F tal que $M \simeq M(A)$, a matriz A é dita uma *representação* de M. Por exemplo, $U_{2,4}$ é representável sobre $\mathbb{R}$, pois $U_{2,4} \simeq M(A)$ onde A é a matriz abaixo:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}$$

Entretanto, não é difícil verificarmos que, $U_{2,4}$ não é representável sobre GF(2).

Acima dissemos que A é uma representação de M, pois é claro que se π é uma matriz não singular, então $M(A) \simeq M(\pi A)$. A matriz π pode ser vista simplesmente como uma matriz de mudança de base dos vetores coluna de A. Assim, é claro que se M é representável sobre um corpo F então M possui uma *representação canônica* sobre F, i.e. existe uma matriz $\hat{A} = [I \mid A]$ sobre F tal que $M \simeq M(\hat{A})$. Se M é um matróide sobre E que é representável sobre GF(2) podemos construir uma representação canônica de M sobre GF(2) como segue. Seja $B = \{b_1, ..., b_r\}$ uma base qualquer de M. Para cada elemento $e \in E \setminus B$ o matróide $M(B + e)$ contém um único circuito C_e, o *circuito fundamental* de e com relação a B. Seja A uma $B \times (E \setminus B) - \{0,1\}$-matriz, onde o elemento da entrada (b, e) de A, $a_{b,e}$, é definido da seguinte maneira:

$$a_{b,e} := \begin{cases} 1 & \text{se } b \in C_e; \\ 0 & \text{se } b \notin C_e. \end{cases} \tag{3.1}$$

A matriz $\hat{A} = [I \mid A]$ é uma representação canônica de M (veja, por exemplo, Welsh [1976: pp. 165 e 166]). Observemos que a construção da matriz A acima foi feita de maneira análoga ao que fizemos em 2.9 (p. 56).

Um fato interessante sobre a representabilidade de matróides gráficos encontra-se na proposição a seguir.

Proposição 3.1 *Se M é um matróide gráfico, então M é representável sobre todo corpo.* ■

A prova da proposição acima não é difícil e pode ser encontrada, por exemplo, em MTIA, p. 186.

Uma conseqüência imediata da proposição 3.1 é que o matróide $U_{2,4}$ não é representável sobre todos os corpos. De fato, como já havíamos mencionado, não é difícil verificarmos que $U_{2,4}$ não é representável sobre GF(2).

Quase todos os fatos em teoria dos grafos que podem ser fomulados sem o emprego da palavra "vértice" possuem uma formulação análoga e natural em teoria dos matróides. A

idéia agora é generalizarmos para matróides os conceitos, muito úteis, de remoção e contração de arestas em grafos.

Se $M = (E,\mathcal{C})$ é um matróide e $X \subseteq E$ então o conjunto

$$\mathcal{C}' = \{C \in \mathcal{C} \mid C \cap X = \emptyset\}$$

é o conjunto dos circuitos de um matróide sobre $E \setminus X$, obtido pela *remoção* do conjunto X (ou *restrição* do matróide M ao conjunto $E \setminus X$). Este matróide será denotado por $M \setminus X$ ou por $M(E \setminus X)$.

Se $M = (E,\mathcal{C})$ é um matróide e $X \subseteq E$ então o conjunto

$$\mathcal{C}' = \{C \setminus X \mid C \in \mathcal{C} \text{ e não existe } C' \in \mathcal{C} \text{ tal que } C' \neq C \text{ e } C' \setminus X \subseteq C \setminus X\}$$

(i.e. $\mathcal{C}'$ é formado pelos subconjuntos minimais de $\{C \setminus X \mid C \in \mathcal{C}\}$) é o conjunto dos circuitos de um matróide sobre $E \setminus X$ obtido pela *contração* do conjunto X. Este matróide será denotado por M/X ou por $M.(E \setminus X)$.

Os matróides $M \setminus X$ e M/X generalizam para matróides os conceitos de remoção e contração de arestas em um grafo, respectivamente.

A partir de $M(E)$ vários novos matróides podem ser obtidos através da remoção e contração de subconjuntos de E. O matróide resultante de uma seqüência de remoções e contrações é dito um *menor* de M. Não é difícil provarmos que a ordem em que as contrações e remoções são feitas é irrelevante, i.e. se $M(E)$ é um matróide e M' é um menor de M, então existem $X, Y \subseteq E, X \cap Y = \emptyset$ tais que $M' = (M \setminus X)/Y = (M/Y) \setminus X$ (veja, por exemplo, MTIA, p. 158]).

Consideremos uma matriz A, representação do matróide $M(E)$, e $e \in E$. Uma representação para o matróide $M - e$, pode ser obtida a partir de A simplesmente removendo-se a coluna correspondente ao elemento e. Já se quisermos construir uma representação para M/e teremos que considerar dois casos possiveis. Se e é um laço basta removermos a coluna correspondente a e, caso contrário teremos um pouco mais de trabalho (mas não muito). Primeiro deveremos pivotar a matriz A com relação à alguma entrada não nula da coluna correspondente a e. Seja A' a matriz resultante. Se l é o índice da linha onde encontra-se a única entrada não nula de A' correspondente a x, então uma representação para M/e pode ser obtida a partir de A' removendo-se a coluna e e a linha l (o que estamos fazendo é olhar para os vetores coluna de A módulo o vetor coluna correspondente a e). Não é difícil verificarmos que as matrizes acima são de fato representações para os menores $M - e$ e M/e.

Os menores de um matróide preservam alguma estrutura do matróide original, como podemos perceber pelas duas proposições a seguir.

Proposição 3.2 *Se M é um matróide representável sobre F, então todo menor de M é representável sobre F.*

Prova: De fato, se M' é um menor de M então uma representação para M' pode ser obtida a partir de uma representação para M através de uma seqüência de pivotações e remoções de linhas e colunas como foi descrito acima. ■

Proposição 3.3 *Se M é um matróide gráfico, então todo menor de M é gráfico.*

Prova: Seja M' um menor de M. Como M é gráfico então existe um grafo G tal que $M \simeq \mathcal{M}(G)$ e como M' é um menor de M então existem conjuntos disjuntos $X, Y \subseteq E$ tais que $M' = (M \setminus X)/Y$. Agora, não é difícil verificarmos que o grafo G' obtido a partir de G após a contração das arestas correspondentes aos elementos em X e a remoção das arestas correspondentes aos elementos em Y é uma realização de M' (i.e $M' \simeq \mathcal{M}(G')$, a prova é essencialmente a mesma do lema 2.3). ■

O que as proposições anteriores estão nos mostrando é que as propriedades Π_{mr} = "ser um matróide representável" e Π_{mg} = "ser um matróide gráfico" são hereditárias. Mais ainda, uma conseqüência imediata da proposição 3.3 e do fato de $U_{2,4}$ não ser representável sobre GF(2) é que se M é um matróide que possui um menor M' tal que $M' \simeq U_{2,4}$, então M não é representável sobre $GF(2)$. Tutte deu uma caracterização de matróides binários provando que vale também a recíproca dessa afirmação, ou seja:

Teorema 3.1 (Tutte [1965]) *Um matróide é representável sobre GF(2)* sse *não possui $U_{2,4}$ como menor.* ■

Uma prova para o teorema acima pode também ser encontrada em Welsh [1976, pp. 167–169].

Um matróide que é representável sobre GF(2) é dito *binário*. Assim, o que Tutte provou foi que $U_{2,4}$ é essencialmente o "único" matróide não binário, ou seja, que $U_{2,4}$ é a única *obstrução* (*menor proibido*) da propriedade Π_{mg}.

Da proposição 3.1 e do fato de $U_{2,4}$ não ser gráfico é evidente que se M é um matróide que possui $U_{2,4}$ como menor, então M não é gráfico. Infelizmente, isto ainda não nos dá uma caracterização de matróides gráficos. Na próxima seção veremos uma caracterização bastante profunda de matróides gráficos, em termos de obstruções, a qual é também devida a Tutte.

Podemos também criar matróides, a partir de matróides já existentes, que são "maiores" que os matróides originais. Uma das maneiras de fazermos isso é a seguinte. Consideremos matróides $M_1 = (E_1, \mathcal{C}_1)$ e $M_2 = (E_2, \mathcal{C}_2)$, $\{C \mid C \in \mathcal{C}_1 \text{ ou } C \in \mathcal{C}_2\}$ é o conjunto de circuitos de um matróide M sobre $E_1 \cup E_2$. Diremos que M é matróide *soma direta*, ou a *1-soma*, dos matróide M_1 e M_2 e denotaremos essa operação por $M = M_1 +_1 M_2$. A operação de soma direta pode ser estendida para mais de um matróide. Por exemplo, se $G_1, ..., G_k$ são as componentes 2-conexas de um grafo G, então $\mathcal{M}(G) = \mathcal{M}(G_1) +_1 ... +_1 \mathcal{M}(G_k)$ e se $A_1, ..., A_k$ são as componentes conexas de uma matriz A (a definição de componente conexa de uma matriz encontra-se na seção 2.1.3), então $M(A) = M(A_1) +_1 ... +_1 M(A_k)$.

Um matróide $M(E)$ é dito *conexo* ou *não separável* se ele não pode ser escrito como soma direta, ou equivalentemente, não existe uma partição $\{E_1, E_2\}$, $E_1 \neq \emptyset \neq E_2$, de E, tal que

$$\rho E_1 + \rho E_2 \leq \rho M. \tag{3.2}$$

$M(E')$ é dito uma *componente conexa* (ou simplesmente *componente*) de M *sse* $M(E')$ é conexo. Observemos que nesse caso $\{E', E \setminus E'\}$ satisfaz 3.2 com igualdade.

Notemos que $\mathcal{M}(G)$ é conexo *sse* G é 2-conexo e $M(A)$ é conexo *sse* A é conexa.

Precisaremos ainda de mais um conceito importante antes de terminarmos esta seção, que é o conceito de dualidade em matróides.

Seja $\mathcal{B}$ o conjunto de bases de um matróide $M(E)$. O conjunto $\mathcal{B}^* = \{E \setminus B \mid B \in \mathcal{B}\}$ é o conjunto de bases de um matróide $M^*(E)$, o *matróide dual* de M. O prefixo "co", em geral, será usado para dualizar um termo. Assim, um *cocircuito* de M é um circuito do matróide M^*, um *colaço* de M é um laço de M^*, etc. Por exemplo, C^* é um cocircuito de um matróide $\mathcal{M}(G)$ *sse* as arestas de G, correspondentes aos elementos de C^*, formam um corte do grafo G. É óbvio que a relação entre M e M^* é simétrica, ou seja

$$(M^*)^* = M.$$

Um matróide M é dito *cográfico sse* M^* é gráfico. Se M é gráfico e cográfico, então M é dito *planar*. A nomenclatura aqui vem da analogia com planaridade em grafos. De fato, as definições que acabamos de ver generalizam para matróides o conceito de planaridade em grafos. O teorema a seguir é a versão para matróides da caracterização de grafos planares de Kuratowski (veja, por exemplo, Welsh [1976: pp. 93–96]).

Teorema 3.2 (Tutte [1965]) *Se M é um matróide e G é um grafo, tal que $M \simeq \mathcal{M}(G)$ então M é planar* sse *G é planar.* ■

A relação entre a representação de um matróide e a representação do seu matróide dual é mostrada na proposição 3.4.

Proposição 3.4 *Se $[I \mid A]$ é uma representação de um matróide M, então $[I \mid A^t]$ é uma representação de M^*.* ■

Uma prova da proposição acima pode ser encontrada em MTIA, pp. 186 e 187.

A tabela 3.1 pode ser útil para visualisarmos em teoria dos grafos e álgebra linear o que representam alguns conceitos vistos por nós em teoria dos matróides.

Uma tabela com mais conceitos do que a tabela 3.1 pode ser encontrada em MTIA, p. 197.

3.3 Reconhecimento de matrizes gráficas

No capítulo 2 vimos como o problema de reconhecer matrizes gráficas (i.e., matrizes projetivamente equivalentes a matrizes de grafos) pode ser resolvido. Nesta seção discutiremos este

Como podemos imaginar certos conceitos em teoria dos matróides?		
Conceito em teoria dos matróides	Suponha que o matróide é dado através de um grafo	uma matriz
elemento	aresta	vetor coluna
laço	laço	vetor nulo
elementos em paralelo	arestas paralelas	vetores l.d.
circuito	circuito	vetores l.d. minimais
conjunto independente	floresta	vetores l.i.
base	floresta geradora	base do espaço coluna
posto	número de arestas em uma floresta maximal	posto
remoção	remoção	remoção
contração	contração	espaço quociente
dual	dual para grafos planares, nada para os demais	espaço ortogonal
soma direta	veja figura 4.1, p. 84	veja figura 4.4, p. 86

Tabela 3.1:

problema do ponto de vista da teoria dos matróides. Como já foi mencionado, muito do que vimos no capítulo 2 foi originalmente escrito em termos de teoria dos matróides.

O *problema de reconhecer matrizes gráficas* é o problema

$$\begin{array}{lll} \textit{dados} & : & \text{Uma matriz } A \text{ sobre um corpo } F. \\ \textit{pergunta} & : & A \text{ é uma matriz gráfica? Ou seja, existem } N \text{ matriz de um} \\ & & \text{grafo, } \pi \text{ e } \Delta \text{ matrizes não singulares, } \Delta \text{ diagonal, tais que} \\ & & N = \pi A \Delta? \end{array} \tag{3.3}$$

Do que vimos na seção 3.2 é claro que, se A é uma matriz, π e Δ são matrizes não singulares e Δ é diagonal, então $M(A) \simeq M(\pi A \Delta)$, i.e., matrizes projetivamente equivalentes têm o mesmo matróide. Logo, o teorema 2.1 (p. 25) é equivalente ao seguinte teorema:

Teorema 3.3 *Uma matriz A é gráfica* sse *o matróide $M(A)$ é gráfico.* ■

Assim, o problema de reconhecer matrizes gráficas é equivalente ao problema:

$$\begin{array}{lll} \textit{dados} & : & \text{Uma matriz } A \text{ sobre um corpo } F. \\ \textit{pergunta} & : & \text{O matróide } M(\hat{A}) \text{ é gráfico? Ou seja, existe um grafo } G \text{ tal} \\ & & \text{que } M(\hat{A}) \simeq \mathcal{M}(G)? \end{array} \tag{3.4}$$

Ao enunciarmos o problema acima assumimos implicitamente que a matriz dada está na forma canônica, não há nenhuma perda em generalidade com isso, pois matrizes projetivamente

têm o mesmo matróide. A razão para termos assumido que a matriz dada está na forma canônica ficará evidente logo adiante quando enunciarmos a proposição 3.6.

É comum que quando um matróide não é conexo muitos problemas envolvendo esse matróide possam ser solucionados resolvendo-se o problema em questão para cada uma de suas componentes conexas. Este é precisamente o caso do problema 3.4, como pode ser visto através da proposição a seguir, que é conseqüência imediata da proposição 2.3 (p. 31) e do teorema 3.3.

Proposição 3.5 *Se A é uma matriz com componentes conexas $A_1,...,A_k$, então A é gráfica* sse *$M(A_i)$ é gráfico, $i = 1,...,k$.* ■

A proposição 2.3 está nos dizendo que podemos assumir, sem perda de generalidade, que a matriz de entrada A do problema 3.4 é conexa (se A não for conexa podemos utilizar o algoritmo *SubmatrizesConexas*, p. 30, para encontrar as componentes conexas de A). Observemos que ambas as hipóteses sobre a matriz de entrada (forma canônica e conexidade) do problema 3.4 são exatamente as mesma hipóteses que foram assumidas pelo algoritmo que vimos no capítulo 2.

Da proposição 2.2 (p. 27) e do teorema 3.3 segue imediatamente a seguinte proposição 3.6:

Proposição 3.6 *Se A é uma matriz sobre um corpo F tal que $M(\hat{A})$ é gráfico e B é a imagem booleana de A, então $M(\hat{B}) \simeq M(\hat{A})$.* ■

Até o final desta seção denotaremos por B a imagem booleana da matriz A.

É importante observarmos que na proposição 3.6 o fato de $\hat{A}$ ser uma matriz na forma canônica é fundamental, pois é claro que a matriz

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

é gráfica (qualquer árvore com duas arestas é uma realização para A) e portanto, pelo teorema 3.3, o matróide $M(A)$ é gráfico. Entretanto, é evidente que $M(A) \not\simeq M(B)$. A proposição 3.6 ainda continua verdadeira se trocarmos no enunciado a palavra "gráfico" por "binário" (veja, por exemplo, Welsh [1976: pp. 165–167] ou a proposição 4.2, p. 88).

Agora veremos a razão que nos levou a assumir que a matriz dada está na forma canônica. De maneira análoga ao que ocorreu na seção 2.1.2, a proposição 3.6 nos fornece uma condição necessária para que $M(\hat{A})$ seja gráfico, a saber, que o matróide $M(\hat{B})$ seja gráfico. Infelizmente não vale a recíproca da proposição 3.6 (qualquer representação na forma canônica de $U_{2,4}$ é um contra-exemplo). Entretanto, a proposição a seguir nos fornece num certo sentido, uma "quase" recíproca da proposição 3.6.

Proposição 3.7 *Se A é uma matriz, $M(\hat{A})$ é gráfico e G é um grafo tal que $M(\hat{B}) \simeq \mathcal{M}(G)$, então $M(\hat{A}) \simeq M(\hat{B})$.* ■

A prova da proposição acima segue facilmente da proposição 2.4 (p. 59) e do teorema 3.3.

Devido às proposições 3.6 e 3.7 e já que o algoritmo *Grafo* (p. 54) não só resolve o problema 3.4 quando o corpo F é GF(2) como também, no caso de $M(\hat{B})$ ser gráfico, devolve um grafo G tal que $M(\hat{B}) = \mathcal{M}(G)$, então é razoável que pensemos em decompor o problema 3.4 nos subproblemas:

1. testar se $M(\hat{B})$ é gráfico;
2. testar se $M(\hat{A}) \simeq M(\hat{B})$.

Devemos observar que precisamos testar se $M(\hat{A}) \simeq M(\hat{B})$ somente quando o algoritmo *Grafo* decide que $M(\hat{B})$ é gráfico. Assim, no item 2 o que estamos testando na realidade é se $M(\hat{A}) \simeq \mathcal{M}(G)$, onde G é um grafo tal que $M(\hat{B}) \simeq \mathcal{M}(G)$. Isso faz muita diferença, pois, como a proposição 3.6 permanece verdadeira quando $M(\hat{A})$ é binário, ao ínves de gráfico, então testar se $M(\hat{A}) \simeq M(\hat{B})$ é equivalente a decidir se $M(\hat{A})$ é binário e não existe algoritmo polinomial que decide se um matróide, mesmo representável, é binário (*cf.* Seymour [1981]).

Para testarmos se $M(\hat{A}) \simeq \mathcal{M}(G)$ utilizaremos a proposição 3.8 a seguir e alguns resultados da seção 2.3.

Proposição 3.8 *Se X e Y são matrizes tais que $M(\hat{X}) = \mathcal{M}(G) = M(\hat{Y})$, então $\hat{X}$ é projetivamente equivalente a $\hat{Y}$ ($\hat{X} \sim \hat{Y}$).*

Prova: Seja N a matriz construída a partir do grafo G como descrito em 2.9 (p. 56). Como $M(\hat{X}) = \mathcal{M}(G)$ então $\hat{X} \sim \hat{N}$. Da mesma forma temos que $\hat{Y} \sim \hat{N}$. Portanto, como $\sim$ é uma relação de equivalência, temos que $\hat{X} \sim \hat{Y}$. ■

Pela proposição 3.8 vemos que para testarmos se $M(\hat{A}) \simeq \mathcal{M}(G)$, basta construirmos um matriz N a partir do grafo G, como descrito em 2.9 (é claro que $\hat{N}$ é uma representação canônica de $\mathcal{M}(G)$) e usar o algoritmo *TestaPE* (p. 58) para testar se $\hat{A} \sim \hat{N}$.

Recapitularemos agora, usando a linguagem de teoria dos matróides, como o problema de reconhecer matrizes gráficas é resolvido no capítulo 2. Primeiro o algoritmo *Grafo* testa se $M(\hat{B})$ é gráfico. Se o algoritmo *Grafo* decide que $M(\hat{B})$ não é gráfico, então (pela proposição 3.6) concluímos que $M(\hat{A})$ não é gráfico e portanto (pelo teorema 3.3) $\hat{A}$ não é gráfica. Caso o algoritmo *Grafo* decida que $M(\hat{B})$ é gráfico, então o algoritmo devolve um grafo G tal que $M(\hat{B}) \simeq \mathcal{M}(G)$. O próximo passo é a construção de uma representação canônica $\hat{N}$ para $\mathcal{M}(G)$ a partir do grafo G como o descrito em 2.9 e o teste de equivalência projetiva, feito pelo algoritmo *TestaPE* (p. 58), entre a representação $\hat{N}$ e a matriz $\hat{A}$. Se o algoritmo *TestaPE* decide que $\hat{N}$ não é projetivamente equivalente a $\hat{A}$, então (pela proposição 3.7) concluímos que $M(\hat{A})$ não é gráfico e portanto $\hat{A}$ não é gráfica. Caso contrário, como matrizes projetivamente equivalentes têm o mesmo matróide, concluímos que $M(\hat{A})$ é gráfico, logo $\hat{A}$ é gráfica.

Resumindo: para resolver o problema 3.4 bastou sabermos testar equivalência projetiva entre duas matrizes e decidir se um matróide binário é gráfico. Como testar equivalência

projetiva é fácil então, no fundo, precisamos saber resolver o problema 3.4 somente quando o corpo F é GF(2).

Lembremos que o algoritmo que vimos no capítulo 2 não só decide se um matróide representável $M(E)$ é gráfico como também, no caso em que $M(E)$ não é gráfico, encontra um conjunto $E' \subseteq E$ maximal tal que $M(E')$ é gráfico. Na próxima seção veremos mais alguns resultados sobre o problema de decidir se um matróide (não necessariamente representável) é gráfico.

3.4 Reconhecimento de matróides gráficos

Discutiremos nesta seção a respeito de um problema um pouco mais geral que o problema 3.5, que é o *problema de reconhecer matróides gráficos*, ou seja:

dados	:	Um matróide $M(E)$.	
pergunta	:	O matróide $M(E)$ é gráfico? Ou seja, existe um grafo G tal que $M(E) \simeq \mathcal{M}(G)$?	(3.5)

Durante toda esta seção, M será um matróide sobre um conjunto E.

Antes de prosseguirmos precisamos ter claro o significado de um matróide ser a entrada do problema 3.5.

Em geral, aplicações práticas requerem matróides representáveis ou gráficos como modelos, os quais podem ser dados através de matrizes ou grafos. Em ambos os casos testar se um subconjunto $E' \subseteq E$ é independente em M requer no máximo $| E |^3$ operações (polinomial em $| E |$).

Mas como fazer no caso geral, quando M é um matróide qualquer? Nesses casos assume-se que o matróide M é dado através de uma *subrotina* que responde a alguma pergunta do tipo: "$E' \subseteq E$ é independente em M?"; "$E' \subseteq E$ é uma base de M?"; "$E' \subseteq E$ é um circuito de M?"; etc. Uma tal subrotina é chamada de um *oráculo*[1]. Se o oráculo responde à pergunta "$E' \subseteq E$ é independente em M?", então ele é dito um *oráculo de independência*.

Todos os algoritmos que foram vistos até agora são polinomiais, no sentido em que o número de operações é um polinômio no tamanho da entrada. Agora qual é o tamanho da entrada quando a entrada é um matróide? Quando um matróide é dado através de um grafo ou de uma matriz a resposta é obvia. Mas o que fazer quando o matróide é dado atráves de um oráculo? Em casos como esses o que é feito é medir a complexidade do algoritmo em termos do número de chamadas a esse oráculo.

Tudo o que vimos nos parágrafos anteriores e muito mais sobre oráculos para matróides está explicado de uma maneira muito clara no excelente livro MTIA, sec. 17.1.

[1] Um meio infalível através do qual os deuses nos revelam os mistérios do universo.

Algoritmos para reconhecer matróides gráficos

Na seção 2.2 vimos o algoritmo de Bixby & Wagner [1988] que resolve o problema 3.5 quando M é um matróide binário, ou seja, quando M é dado através de uma representação $\hat{B}$ de M sobre GF(2). A complexidade desse algoritmo é "quase" linear no número de entradas não nulas de B.

Bixby & Cunningham [1980] propuseram um algoritmo de complexidade $O(\rho M(\hat{A}).\nu)$ para decidir se um matróide $M(\hat{A})$ é gráfico, onde $\rho M(\hat{A})$ e ν são o posto (obviamente igual ao número de linhas de A) e o número de entradas não nulas de $\hat{A}$, respectivamente. O algoritmo de Bixby & Cunningham usa a mesma estratégia do algoritmo que vimos na seção 3.3, a diferença é que para decidir se $M(\hat{B})$ é gráfico, e em caso afirmativo construir um grafo G tal que $\mathcal{M}(G) \simeq M(\hat{B})$, Bixby & Cunningham se basearam no algoritmo de Tutte [1960]. A seguir veremos um esboço do algoritmo de Tutte, o qual é uma abstração do seu algoritmo de teste de planaridade para grafos.

Ajudará bastante se durante a descrição da motivação do algoritmo tivermos em mente que se $M \simeq \mathcal{M}(G)$ para algum grafo G, então um cocircuito Y de M corresponde a um conjunto de arestas de corte de G. Dado um cocircuito Y de um matróide M, as *pontes* do cocircuito Y são as componentes conexas de $M \setminus Y$. No caso em que $M \simeq \mathcal{M}(G)$ as pontes de um cocircuito Y correspondem a componentes 2-conexas do grafo $G \setminus Y$. Uma *Y-componente* é um matróide da forma $M.(P \cup Y)$, onde P é o conjunto de elementos de uma ponte de Y. Finalmente, os P-segmentos de Y são as *classes de paralelismo* de $M.(P \cup Y) \setminus P$, onde a classe de paralelismo de um elemento e é o conjunto de todos os elementos que são paralelos a e. O conceito de ponte está ilustrado na figura 3.1, onde $M = \mathcal{M}(G)$, $Y = \{1,2,3,4,5,6\}$ e as pontes de $M \setminus Y$ são os matróides gráficos de $G[P_1], ..., G[P_7]$. A Y-componente $\mathcal{M}(G).(P_4 \cup Y)$ é o matróide gráfico do grafo da figura 3.2.

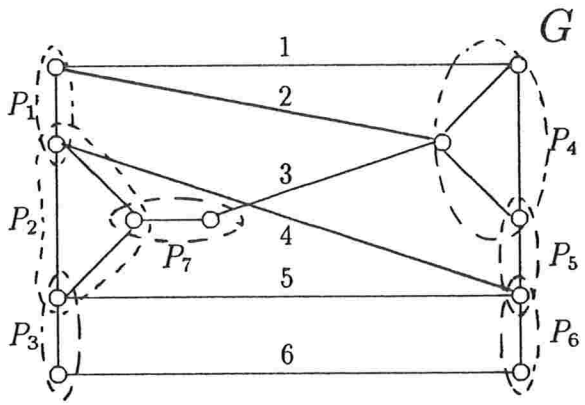

Figura 3.1: Grafo G e as pontes de $\mathcal{M}(G) \setminus \{1,2,3,4,5,6\}$.

Lembremos que um matróide M é gráfico *sse* cada uma de suas componentes conexas o forem. Assim, podemos descrever o algoritmo de *Tutte* somente para o caso em que M é conexo. Se $M = \mathcal{M}(G)$ e Y é um cocircuito de M com apenas uma ponte, então é claro que Y é a estrela de um vértice em G. Portanto, cada elemento de M pode estar em no máximo dois cocircuitos que só tenham uma ponte. O próximo fato captura a idéia que está aparente no

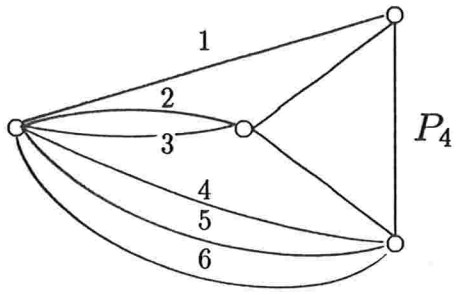

Figura 3.2: Uma Y-componente de $\mathcal{M}(G)$, onde G é o grafo da figura 3.1 e $Y = \{1,2,3,4,5,6\}$.

exemplo da figura 3.1, ou seja, em um matróide gráfico, as pontes de um cocircuito podem ser divididas em duas classes, uma classe correspondente a cada um dos lados do corte Y em G. Tutte formulou uma abstração para esta propriedade introduzindo a noção de "entrelaçamento" entre pontes. Dizemos que duas pontes P_1 e P_2 se *entrelaçam*[2] se não existe um P_1-segmento S_1 e um P_2-segmento S_2, tal que $S_1 \cup S_2 = Y$. É fácil observarmos que, no grafo G da figura 3.1, as pontes que estão do mesmo lado com relação ao corte Y não se entrelaçam duas a duas. De fato, as únicas pontes de Y que se entrelaçam são os pares de pontes (P_1, P_4), (P_2, P_5) e (P_4, P_2). Define-se o *grafo ponte* de um circuito Y como sendo o grafo com um vértice para cada ponte de Y, onde dois vértices são adjacentes *sse* as pontes correspondentes se entrelaçam. Como as observações acima sugerem o grafo ponte de um cocircuito de um matróide gráfico é bipartido.

Teorema 3.4 *Se M é um matróide e Y é um cocircuito de M, então M é gráfico* sse

i. as Y-componentes de M são gráficas, e

ii. o grafo ponte de Y é bipartido. ■

O teorema 3.4 foi primeiro provado por Tutte [1960] para matróides binários. Um prova para o teorema acima quando $M = M(\hat{A})$, onde $\hat{A}$ é uma matriz sobre um corpo F, pode ser encontrada em Bixby & Cunningham [1980]. Bixby [1984] provou o teorema acima quando M é um matróide qualquer.

Inspirado pelo teorema 3.4 Tutte [1960] propôs um algoritmo recursivo para resolver o problema 3.5, quando M é um matróide binário. O algoritmo de Tutte encontra-se na figura 3.3. Nele é utilizado o fato de que se M é binário e $\hat{B}$ é uma representação de M sobre GF(2), então o suporte de cada linha de $\hat{B}$ é um cocircuito de M (isto é imediato a partir da proposição 3.4).

O algoritmo de Bixby & Cunningham [1980] para decidir se um matróide $M(\hat{A})$ é gráfico consiste de uma implementação cuidadosa do algoritmo de Tutte, especialmente na construção do grafo ponte de Y. Bixby & Cunningham ainda apresentaram em seu trabalho um algoritmo diferente para construir uma realização G de $M(\hat{A})$, no caso em que $M(\hat{A})$ é gráfico.

[2] *Overlap.*

Algoritmo Tutte (M)
Entrada : Um matróide binário M, dado através de uma de suas representações canônicas $\hat{B}$ (dado M a representação $\hat{B}$ pode ser contruida como foi descrito em 3.1).
Saída : A conclusão se M é gráfico ou não.

```
    início
1       se cada coluna de B tem no máximo duas entradas não nulas
2           então M é gráfico
3           senão início
4                     Escolha uma coluna de B com um 1 em três linhas, correspondentes a
                      cocircuitos Y1, Y2 e Y3 de M.
5                     se Y1, Y2 e Y3 têm somente uma ponte
6                         então M não é gráfico
                          senão início
7                                   Seja Y ∈ {Y1, Y2, Y2} um cocircuito com mais de uma ponte.
8                                   se o grafo ponte de Y não é bipartido
9                                       então M não é gráfico
                                        senão início
                                                % Aplica o algoritmo de Tutte recursivamente
                                                % à cada Y-componente de M.
10                                              para cada Y-componente M' faça
11                                                  Tutte(M');
12                                              Pelo teorema 3.4 M é gráfico
                                                sse cada Y-componente de M é gráfica.
                                              fim
                                    fim
                    fim
    fim
```

Figura 3.3: Algoritmo de Tutte para decidir se um matróide binário é gráfico.

Na seção 3.3 vimos que para decidir se um matróide M é gráfico tivemos que resolver o seguinte problema:

$$\begin{array}{lll} \textit{dados} & : & \text{Um matróide } M \text{ e um grafo } G. \\ \textit{pergunta} & : & M \simeq \mathcal{M}(G)? \end{array} \tag{3.6}$$

Na realidade, só tivemos que resolver o problema 3.6 quando M era um matróide representável. O teorema de Seymour a seguir, junto com o algoritmo que ele induz, constitui a primeira solução para o problema 3.6, quando M é um matróide qualquer (dado através de um oráculo de independência).

Teorema 3.5 (Seymour [1981]) *Se $M(E)$ é um matróide e $G = (V, E)$ é um grafo, então $M = \mathcal{M}(G)$* sse

i. $\rho(M) \leq \rho(\mathcal{M}(G))$, *e*

ii. a estrela de cada vértice de G é a união de cocircuitos de M. ■

Baseado em seu teorema, Seymour propôs o primeiro algoritmo polinomial, que reconhece matróides gráficos (o matróide é dado através de um oráculo de independência). O algoritmo que Seymour propôs consiste em construirmos uma representação canônica $\hat{B}$ sobre GF(2) para M (em 3.1 está descrito como construir uma tal representação) e testarmos se $M(\hat{B})$ é gráfico (usando, por exemplo, um dos métodos em: Tutte [1960], Bixby & Cunningham [1980] ou Bixby & Wagner [1988]). Se $M(\hat{B})$ não é gráfico, então M não é gráfico (pela proposição 3.6). Caso contrário, devemos construir um grafo G tal que $M(\hat{B}) \simeq \mathcal{M}(G)$ (os algoritmos que citamos fazem isso). Se a estrela de cada vértice de G é a união de cocircuitos de M então M é gráfico e G é uma realização para M, senão M não é gráfico (pelo teorema 3.5).

O teorema 3.6 é uma versão mais forte do teorema de Seymour obtida por Truemper (*cf.* Bixby [1984]).

Teorema 3.6 (Truemper) *Sejam M_1 e M_2 matróides sobre E e G um grafo com arestas E. Se*

i. M_1 e M_2 são conexos,

ii. M_1 e M_2 possuem uma base em comum, tal que M_1 e M_2 têm os mesmos circuitos fundamentais em relação a essa base,

iii. $M_2 = \mathcal{M}(G)$ e

iv. Para cada vértice v de G, a estrela de v contém um cocircuito de M_1,

então $M_1 = M_2$. ■

O teorema de Truemper simplifica o último passo do algoritmo de Seymour. A complexidade do algoritmo passa a ser $O(\rho M. \mid E \mid)$ (chamadas do oráculo de independência). De fato, a construção de uma base K de M requer $O(\mid E \mid)$ chamadas do oráculo. Construir uma representação $\hat{B}$ para M a partir da base K requer mais $O(\rho M. \mid E \mid)$ chamadas ($O(\rho M)$ para

cada $e \in E \setminus K$). Para determinarmos se $M(\hat{B})$ é gráfico e, em caso afirmativo, construir um grafo G, tal que $M(\hat{B}) = \mathcal{M}(G)$, podemos usar o algoritmo de Bixby & Wagner descrito na seção 2.2. Finalmente, para aplicarmos o teorema de Truemper basta testarmos se a estrela de cada vértice de G contém um cocircuito de M. Este último passo requer $O(\rho M)$ chamadas do oráculo (precisamos testar se para cada estrela X de G, $E \setminus X$ não contém uma base de M).

Uma caracterização de matróides gráficos em termos de menores proibidos

Na seção 3.2 vimos que uma condição necessária para que um matróide seja gráfico é que ele seja binário, i.e., não tenha $U_{2,4}$ como menor. Entretanto, isso não é suficiente, pois como (pela proposição 3.1) $\mathcal{M}(K_5)$ e $\mathcal{M}(K_{3,3})$ são binários, então (pelo teorema 3.4) $\mathcal{M}^*(K_5)$ e $\mathcal{M}^*(K_{3,3})$ são binários. Mas devido ao teorema 3.2 é evidente que $\mathcal{M}^*(K_5)$ e $\mathcal{M}^*(K_{3,3})$ não são gráficos. Mais ainda, os mátroides lineares das seguintes matrizes sobre GF(2) são obviamente binários, mas nenhum deles é gráfico.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

O matróide linear da primeira matriz é conhecido como *matróide de Fano* e é denotado por F_7. É claro, pelo teorema 3.4, que o matróide linear da segunda matriz é F_7^*. F_7 não é gráfico, pois se existisse G, tal que $F_7 \simeq \mathcal{M}(G)$, então G teria 4 vértices , pois F_7 tem posto 3. Entretanto, qualquer grafo com 4 vértices e 7 arestas possui algum laço ou aretas paralelas, mas F_7 não possui laços ou elementos em paralelo. Analogamente, se $F_7^* \simeq \mathcal{M}(H)$ isto implicaria que H possui 5 vértices, mas como qualquer grafo com 5 vértices e 7 arestas possui um vértice de grau no máximo 2, então H deve possuir um corte com no máximo duas arestas. Entretanto, F_7^* não possui laços nem elementos em paralelo.

Tutte provou que $U_{2,4}, \mathcal{M}^*(K_5), \mathcal{M}^*(K_{3,3}), F_7$, e F_7^* são essencialmente os único matróides não gráficos, como está resumido no teorema abaixo.

Teorema 3.7 (Tutte [1965]) *Um matróide é gráfico* sse *não possui como menor um dos seguintes matróides:* $U_{2,4}$, $\mathcal{M}^*(K_5)$, $\mathcal{M}^*(K_{3,3})$, F_7, *e* F_7^*. ■

Outras provas do teorema de Tutte podem ser encontradas em Seymour [1980a] e Wagner [1985].

O teorema 3.7 é o teorema 1.1 (p. 19) reescrito em termos de matróides.

Capítulo 4

Unimodularidade total e propriedades hereditárias

Programação linear inteira (PLI) investiga a solução de problemas de programação linear onde há a restrição adicional de que o vetor solução seja inteiro, isto é:

$$\begin{array}{ll} \min & cx \\ & Ax \leq b \\ & x \text{ inteiro} \end{array} \tag{4.1}$$

Não é conhecido algoritmo polinomial para PLI. De fato, PLI é $\mathcal{NP}$-completo (veja, TLIP, sec. 18.1).

Uma matriz A sobre $\mathbb{R}$ é dita *totalmente unimodular* se o determinante de cada submatriz de A é $0, +1$, ou -1. A relevância de total unimodularidade para PLI é evidente tendo em vista o teorema a seguir.

Teorema 4.1 (Hoffman & Kruskal) *Se A é uma matriz inteira, então A é totalmente unimodular* sse *o poliedro $\{x \mid x \geq 0; Ax \leq b\}$ é inteiro para todo vetor b inteiro.* ■

Uma prova do teorema acima pode ser encontrada em TLIP, p. 268.

O teorema de Hoffman & Kruskal está nos dizendo que matrizes totalmente unimodulares é precisamente a classe de matrizes para as quais PLI pode ser resolvida por algoritmos para PL quando b é inteiro, ou seja, a restrição de x ser inteiro é redundante. Assim, é claro que se a matriz A em 4.1 é totalmente unimodular então esta instância do problema pode ser resolvida eficientemente.

Na seção 1 deste último capítulo nos dedicaremos ao estudo de matrizes totalmente unimodulares sobre o ponto de vista de teoria dos matróides. Na seção 2 descreveremos um algoritmo polinomial que testa total unimodularidade. Finalmente, na seção 3 discutiremos sobre um corolário (observado por Bartholdi [1982]) do notável trabalho de Yannakakis [1981],

o qual afirma que o reconhecimento de submatrizes máximas (no sentido de linha, colunas ou número de elementos) que satisfaz uma propriedade hereditária Π é $\mathcal{NP}$-completo, o que generaliza (e muito!) as provas que fizemos para algumas propriedades (matrizes de grafos, de grafos sinalizados e gráficas) na seção 1.2.

4.1 Unimodularidade total e matróides regulares

Nesta seção estudaremos algumas propriedades de matrizes totalmente unimodulares e matróides regulares bem como a relação entre ambos.

Lembremos que uma matriz sobre $\mathbb{R}$ é dita *totalmente unimodular* (t.u.) se o determinante da cada uma de suas submatrizes é 0, $+1$, ou -1. Da definição de unimodularidade total segue imediatamente que se A é uma matriz t.u. então: A é uma $\{0, \pm 1\}$-matriz; toda submatriz de A é t.u.; $[I \mid A]$ é t.u; e A^t é t.u., onde A^t é a transposta de A. Mais ainda, unimodularidade total é preservada sob as operações de pivotação e multiplicação de uma linha ou coluna por -1.

O teorema de Hoffman & Kruskal nos fornece uma caracterização de matrizes totalmente unimodulares. Existem diversas outras caracterizações, duas das quais são apresentadas no teorema abaixo.

Teorema 4.2 *Se A é uma $\{0, \pm 1\}$-matriz, então são equivalentes:*

i. A é totalmente unimodular.

ii. cada coleção de colunas de A pode ser particionada em dois conjuntos A_1 e A_2 tal que a soma das colunas em A_1 menos a soma das colunas em A_2 é um $\{0, \pm 1\}$-vetor.

iii. a soma das entradas não nulas de cada submatriz quadrada de A com um número par de entradas não nulas em cada linha e coluna é divisível por quatro. ■

As caracterizações *ii.* e *iii.* são devidas a Ghouila-Houri e Camion, respectivamente. Uma prova para o teorema acima, bem como muitas outras caracterizações de matrizes totalmente unimodulares podem ser encontradas em TLIP, sec. 19.2.

Matrizes de incidência de grafos dirigidos (e portanto matrizes de grafos) e matrizes de incidência de grafos bipartidos são exemplos de matrizes totalmente unimodulares. Isto pode ser facilmente provado utilizando-se a caracterização de Ghouila-Houri. Mais exemplos de matrizes t.u. (e aplicações) podem ser encontradas em TLIP, sec. 19.3.

Da definição de unimodularidade total é evidente que a propriedade Π_{tu} = "ser uma matriz t.u." é hereditária. Assim, tendo em vista o que já ocorreu para matrizes gráficas (teorema 1.1, p. 19), é natural que façamos a seguinte pergunta: Quais são as obstruções para Π_{tu}? A seguir obteremos a resposta para essa pergunta em termos de matróides "regulares".

Um matróide M é dito *regular* se é representável sobre todo corpo e ele é dito *totalmente unimodular* se existe uma matriz A t.u. que é uma representação de M. A relação entre matróides regulares e totalmente unimodulares é dada pelo seguinte teorema:

Teorema 4.3 (Tutte [1965]) *Um matróide é regular* sse *ele é totalmente unimodular.* ■

O teorema de Tutte está afirmando que uma matróide é regular *sse* ele possui uma representação t.u. . Mais ainda, ele mostra que o problema de decidir se uma $\{0, \pm 1\}$-matriz é t.u. é equivalente ao problema de determinar se o matróide dessa matriz é regular.

Devido à proposição 3.2 (p. 68) que afirma que se um matróide M é representável então todo menor de M é representável é claro que vale a seguinte proposição:

Proposição 4.1 *Se M é um matróide regular, então todo menor de M é regular.* ■

Em outras palavras o que a proposição 4.1 está mostrando é que a propriedade Π_{mreg} = "ser um matróide regular" é uma propriedade hereditária. É claro que para obtermos as obstruções de Π_{tu} basta obtermos as obstruções de Π_{mreg}.

Sabemos que matróides gráficos são representáveis sobre todo corpo (proposição 3.1, p. 67) logo todo matróide gráfico é regular. Devido às proposições 3.1 e 3.4 (p. 70) temos que matróides cográficos também são regulares. Os matróides regulares não são somente gráficos ou cográficos como nos mostra o matróide linear R_{10} representado pela seguinte matriz sobre GF(2):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.2}$$

O matróide R_{10} não é gráfico nem cográfico, embora seja regular (veja MTIA, pp. 299, 499 e 500).

$U_{2,4}$ não é binário e portanto este matróide é uma obstrução óbvia da propriedade Π_{mreg}. Sabemos ainda que F_7 e F_7^* não são gráficos nem cográficos (veja teorema 3.7, p. 79), então é claro que esses matróides são obstruções de Π_{mreg}. Tutte em mais uma de suas caracterizações de matróides em termos de menores proibidos provou que essas são as únicas obstruções de Π_{mreg}.

Teorema 4.4 (Tutte [1965]) *Um matróide é regular* sse *não possui um dos seguintes matróides como menor: $U_{2,4}, F_7$, e F_7^*.* ■

Como matrizes projetivamente equivalentes (lembremos que A_1 é projetivamente equivalente (p.e.) a A_2 se existem matrizes π e Δ não sigulares, Δ diagonal, tal que $A_1 = \pi A_2 \Delta$) têm o mesmo matróide então podemos enunciar o teorema de Tutte somente em termos de matrizes seguinte maneira:

Teorema 4.5 (Tutte [1959,1965]) *Uma* $\{0, \pm 1\}$*-matriz é regular* sse *não possui submatrizes p.e. às seguintes matrizes:*

$$U_{2,4} = \begin{bmatrix} 1 & 0 & \alpha & \beta \\ 0 & 1 & \gamma & \delta \end{bmatrix} \quad F_7 = \begin{bmatrix} 1 & 0 & 0 & 0 & * & * & * \\ 0 & 1 & 0 & * & 0 & * & * \\ 0 & 0 & 1 & * & * & 0 & * \end{bmatrix} \quad F_7^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & * & * \\ 0 & 1 & 0 & 0 & * & 0 & * \\ 0 & 0 & 1 & 0 & * & * & 0 \\ 0 & 0 & 0 & 1 & * & * & * \end{bmatrix}$$

onde α, γ, β, δ são números não nulos, (α, γ) e (β, δ) são l.i. e $*$ denota entradas não nulas, não necessariamente iguais. ■

O teorema de Tutte acima nos fornece um meio para provarmos que um matróide $M(A)$ não é regular, basta exibirmos um menor de $M(A)$ isomorfo a uma das obstruções de Π_{mreg} (isto pode ser verificado em um número polinomial de passos se o matróide é dado através de um oráculo de independência).

Agora, como podemos provar que um matróide $M(A)$ é regular? Esta resposta é fácil quando trocamos regular por gráfico. De fato, para provarmos que um matróide $M(A)$ é gráfico basta exibirmos um grafo G tal que $M(A) \simeq \mathcal{M}(G)$, pois sabemos verificar em tempo polinomial se $M(A) \simeq \mathcal{M}(G)$ (Seymour [1981] foi o primeiro a provar que esse teste pode ser feito em um número polinomial de passos, veja p. 78). Entretanto, para provarmos que $M(A)$ é regular não podemos tentar exibir representações de $M(A)$ sobre todo corpo, pois existem infinitos. Mesmo tendo em vista a seguinte caracterização de matróides regulares:

Teorema 4.6 (Seymour [1979]) *Uma matróide é regular* sse *ele é representável sobre GF(2) e GF(3).* ■

Ainda assim não saberíamos como provar que $M(A)$ é regular, pois, como já foi dito na seção 3.3, testar se $M(A) \simeq M(B)$, onde B é um canditado a representação de $M(A)$ sobre GF(2), é equivalente a testar se $M(A)$ é binário e não existe algoritmo polinomial que faça este teste (*cf.* Seymour [1981]).

A resposta à pergunta acima foi dada por Seymour [1980b]. Mas antes, para entendermos a resposta de Seymour, precisaremos estender a operação de 1-soma (soma direta) de matróides (vista na seção 3.2).

A 1-soma entre matróides gráficos pode ser visualisada pictoricamente através da figura 4.1, onde a 1-soma entre os matróides $\mathcal{M}(G_1)$ e $\mathcal{M}(G_2)$ é o matróide gráfico do grafo resultante após a identificação de um vértice do grafo G_1 com um vértice do grafo G_2. As figuras 4.2 e 4.3 sugerem as operações de 2-soma e 3-soma, respectivamente. A 2-soma entre os matróides $\mathcal{M}(G_1)$ e $\mathcal{M}(G_2)$ é o matróide que tem como realização o grafo obtido a partir de G_1 e G_2 após a identificação e posterior remoção de uma aresta de cada grafo. A operação de 3-soma é análoga às anteriores, só que desta vez identificamos um triângulo de G_1 a um triângulo de G_2 e em seguida as arestas dos triângulos são removidas. Denotaremos por $G_1 +_k G_2$ e $\mathcal{M}(G_1) +_k \mathcal{M}(G_2)$ o grafo e o matróide resultante da k-soma, $k = 1$, 2 e 3, entre $\mathcal{M}(G_1)$ e $\mathcal{M}(G_2)$, respectivamente. É interessante observamos que a operação de 2-soma entre grafos já foi empregada por nós na seção 2.2, onde era denotada por $\Sigma(G_1, G_2)$. Observemos ainda que G_1

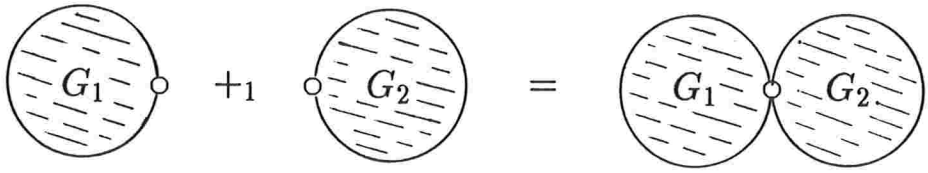

Figura 4.1: $\mathcal{M}(G_1) +_1 \mathcal{M}(G_2)$, soma direta entre os matróides $\mathcal{M}(G_1)$ e $\mathcal{M}(G_1)$.

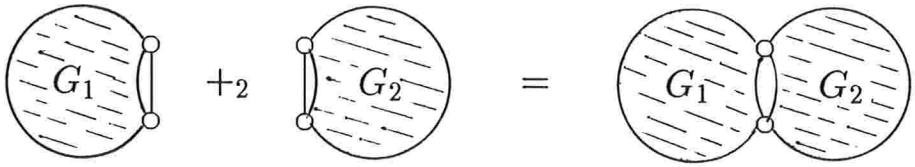

Figura 4.2: $\mathcal{M}(G_1) +_2 \mathcal{M}(G_2)$.

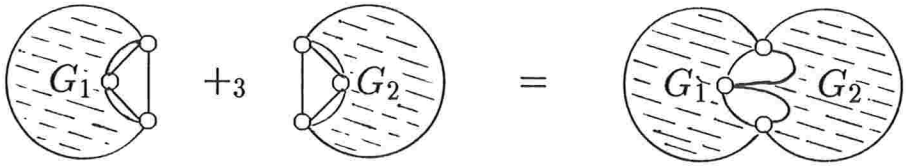

Figura 4.3: $\mathcal{M}(G_1) +_2 \mathcal{M}(G_2)$.

e G_2 podem ser obtidos a partir do grafo $G_1 +_k G_2, k = 1, 2$ e 3, através de contrações e remoções de algumas arestas. Isto significa que $M(G_1)$ e $M(G_2)$ são menores de $M(G_1) +_k M(G_2)$

[Para prosseguirmos não necessitaremos da definição formal de 1-,2- e 3-soma, mas sim da intuição do que ela representa, principalmente no que diz respeito a matróides representáveis. De qualquer forma abaixo encontra-se a definição.

A definição de 1-,2- e 3-soma para matróides binários é a seguinte. Sejam M_1 e M_2 matróides binários sobre E_1 e E_2, respectivamente. Seja $M_1 \triangle M_2$ o matróide sobre $E_1 \triangle E_2 = (E_1 \cup E_2) \setminus (E_1 \cap E_2)$ cujos circuitos são os membros minimais não vazios de

$$\{C_1 \triangle C_2 \mid C_1 \triangle C_2 \subseteq E_1 \triangle E_2; \text{ onde } C_i \text{ é a união disjunta de circuitos de } M_i, i = 1, 2\}.$$

Agora temos três casos de interesse a considerar:

i. Se $E_1 \cap E_2 = \emptyset$ e $E_1, E_2 \neq \emptyset$, então $M_1 \triangle M_2$ (denotado também por $M_1 +_1 M_2$) é a *1-soma (soma direta)* de M_1 e M_2.

ii. Se $| E_1 |, | E_2 | \geq 3$ e $E_1 \cap E_2 = \{e\}$, onde e não é um laço ou colaço de M_1 ou M_2, então $M_1 \triangle M_2$ (denotado também por $M_1 +_2 M_2$) é a *2-soma* de M_1 e M_2.

iii. Se $| E_1 |, | E_2 | \geq 7$ e $E_1 \cap E_2 = Z = \{e, f, g\}$, onde: e, f e g são distintos; Z não contém um cocircuito de M_1 ou M_2; e Z é um circuito de ambos os matróides, então $M_1 \triangle M_2$ (denotado também por $M_1 +_3 M_2$) é a 3-soma de M_1 e M_2.]

Agora estamos preparados para a resposta de Seymour que consiste em uma caracterização muito profunda de matróides regulares.

Teorema 4.7 (Seymour [1980b]) *Um matróide M é regular* sse *vale uma das seguintes afirmações:*

i. M é gráfico ou cográfico.

ii. M é isomorfo a R_{10}.

iii. M é 1-,2-, ou 3-soma de dois matróides regulares. ∎

O teorema de Seymour nos fornece uma maneira para provarmos que um matróide $M(A)$ é regular: basta explicitarmos uma coleção $\{M(A_1), ..., M(A_k)\}$ de matróides (que na realidade são menores de M), onde $M(A_i)$ é gráfico, cográfico ou R_{10} e uma seqüência de 1-,2- e 3-somas que geram o matróides $M(A)$. Com efeito, testar sabemos em tempo polinomial se um matróide é gráfico, cográfico (veja seção 3.4) ou R_{10} e calcular a 1-,2- e 3-soma entre dois matróides binários. Assim, o teorema de Seymour mostrou que o problema de decidir se um matróide $M(A)$ é regular possui uma boa caracterização (i.e. está $\mathcal{NP} \cap$ co-$\mathcal{NP}$).

Observemos que não foi feita hipótese alguma sobre o matróide M do teorema de Seymour. Entretanto, são do nosso particular interesse matróides representáveis. Assim, é importante que saibamos o que significa uma k-soma de matróides representáveis.

Quando M_1 e M_2 são matróides com representações A_1 e A_2 sobre um mesmo corpo, então uma representação $A_1 +_k A_2$ para $M_1 +_k M_2$, $k = 1, 2$, ou 3, pode ser obtida da maneira como está ilustrado nas figuras 4.4, 4.5 e 4.6. Na figura 4.5 e é o único elemento comum a M_1

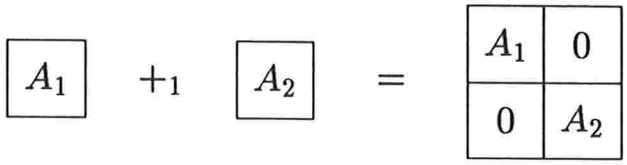

Figura 4.4: $M(A_1) +_1 M(A_2)$, soma direta dos matróides $M(A_1)$ e $M(A_2)$).

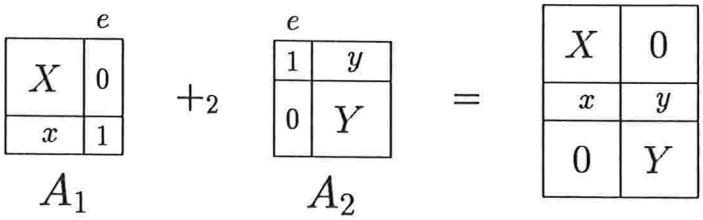

Figura 4.5: $M(A_1) +_2 M(A_2)$.

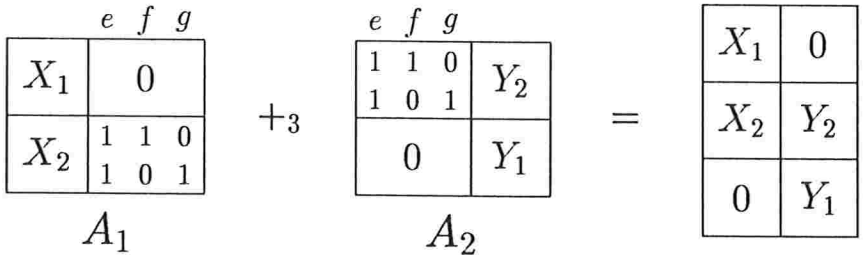

Figura 4.6: $M(A_1) +_3 M(A_2)$.

e M_2. Na figura 4.6 $\{e, f, g\}$ são os únicos elementos comum a M_1 e M_2. As submatrizes com 1's e 0's que estão ilustradas nas figuras podem ser facilmente obtidas através de pivotações.

Se A_1 e A_2 são matrizes t.u. então não é difícil verificarmos que $A_1 +_k A_2$ é t.u., basta usarmos, por exemplo, a caracterização de Ghouila-Houri de matrizes t.u. (teorema 4.2, *i.*). Ou seja, unimodularidade total é invariante sob a operação de 1-, 2- e 3-soma.

Correspondente à operação de k-soma, $k = 1, 2$ ou 3 existe uma *k-decomposição*. Se M, M_1 e M_2 são matróides tal que $M = M_1 +_k M_2$, então dizemos que $\{M_1, M_2\}$ é uma *k-decomposição* de M.

[Se um matróide $M(A)$ possui uma k-decomposição então, como nos sugere as figuras 4.4, 4.5 e 4.6, a matriz A pode ser transformada em uma matriz no formato de quase blocos. Isso é uma caracteristica de k-decomposições que pode ser útil quando estamos resolvendo um PL cuja matriz de restrições é A, pois nos permite aplicar o princípio de decomposição de Dantiz & Wolfe [1960].]

Seja M um matróide sobre E. Uma partição $\{E_1, E_2\}$ de E é chamada de uma *k-separação* de M se

$$| E_1 | \geq k \leq | E_2 |, \quad \rho E_1 + \rho E_2 = \rho M + k - 1. \tag{4.3}$$

onde ρ é a função posto de M. Se M não possui uma k'-separação $k' < k$ então M é dito *k-conexo.*

Como nos sugere as figuras 4.4, 4.5 e 4.6 a cada k-decomposição de $M(A)$ está associada uma k-separação e vice-versa.

Observemos que a definição de k-conexidade acima é uma extensão da definição de conexidade feita em 3.2 (p. 69). Um matróide é conexo *sse* ele é 1-conexo.

Na próxima seção veremos como o teorema de Seymour possibilitou que o problema de decidir se uma matriz é t.u. pudesse ser resolvido em tempo polinomial.

4.2 Reconhecimento de unimodularidade total

Nesta seção estudaremos um algoritmo polinomial que decide se uma matriz é t.u.. Como vimos na introdução deste capítulo a motivação para um tal algoritmo vem de PLI.

Durante a apresentação do algoritmo é interessante que observemos a grande semelhanças entre a estratégia usada aqui é a empregada na seção 3.3 para decidir se uma matriz é gráfica.

Devido ao teorema 4.3 que afirma que uma matriz $\hat{A}$ inteira é t.u. *sse* $M(\hat{A})$ é um matróide regular, vemos que o problema de decidir se uma matriz $\hat{A}$ é t.u. é equivalente ao problema:

dados	:	Uma $\{0, \pm 1\}$-matriz A sobre $\mathbb{R}$.
pergunta	:	$M(\hat{A})$ é regular?

(4.4)

Estamos assumindo que a matriz dada está na forma canônica, o que não constitui perda de generalidade, pois unimodularidade total é invariante sob pivotação. A razão para termos assumido está hipótese é a mesma razão que nos levou a assumi-la na seção 3.3 e encontra-se expressa na proposição a seguir, que constitui um primeiro passo no sentido de reduzir o problema 4.4 ao "caso binário".

Proposição 4.2 *Se A é uma $L \times C$-matriz sobre um corpo F tal que $M(\hat{A})$ é binário e B é a imagem booleana de A, então $M(\hat{B}) \simeq M(\hat{A})$.*

Prova: É claro que L é uma base de $M(\hat{A})$. Consideremos uma representação canônica $[I \mid B']$ de $M(\hat{A})$ onde as colunas da identidade são indexadas por L. Para provarmos a proposição é suficiente provarmos que $B = B'$.

Sejam e um elemento de $M(\hat{A})$ que não está em L e b_e e b'_e as colunas de B e B', respectivamente, correspondentes a e. Seja ainda C_e o circuito fundamental de e com relação à base L. Da unicidade de C_e segue imediatamente que $||\, b_e \,|| = C_e - e = ||\, b'_e \,||$. ■

[A proposição acima é uma versão mais forte da proposição 3.6 (p. 72) onde no lugar da palavra "binário" encontramos "gráfico". A hipótese de que $M(A)$ é binário é fundamental, como pode ser visto através das matrizes 4.5 logo adiante.]

A prova da proposição 4.2 nos mostra como podemos representar um matróide de maneira única sobre GF(2). Mais precisamente:

Teorema 4.8 (Teorema da unicidade de representação) *Se M é um matróide binário e K é uma base de M, então existe uma única representação canônica de M sobre GF(2), onde os elementos de K correspondem à identidade.* ■

Assim, se M é um matróide sobre $L \cup C$, L é uma base de M e B é uma $L \times C$-matriz sobre GF(2) tal que $\hat{B}$ representa M, então diremos que $\hat{B}$ é a representação de M em relação à base L (observemos que a representação é única contanto que fixemos a base).

O mesmo não ocorre quando representamos um matróide sobre outros corpos, como pode ser observado através das seguintes matrizes sobre $\mathbb{R}$:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & -1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \tag{4.5}$$

É fácil vermos que as duas matrizes acima têm o mesmo matróide e estão na forma canônica com relação à mesma base, entretanto é óbvio que elas não são iguais.

Até o final desta seção denotaremos por B a imagem booleana da $L \times C$-matriz A.

Como matróides regulares são em particular binários então a proposição 4.2 nos fornece uma condição necessária para que o matróide $M(\hat{A})$ seja regular (i.e. que a matriz $\hat{A}$ seja t.u.),

a saber, que a o matróide $M(\hat{B})$ seja binário. Somente essa condição não é suficiente (qualquer uma das matrizes em 4.5 é um contra-exemplo).

Veremos a seguir como o problema de decidir se $M(\hat{A})$ é regular ($\hat{A}$ é t.u.) pode ser reduzido ao problema de decidir se $M(\hat{B})$ é regular.

Redução ao caso binário

Suponha que soubessemos testar se um matróide binário é regular. Nesse caso, para resolvermos o problema 4.4, bastaria que tivessemos um algoritmo que testasse se $M(\hat{A}) \simeq M(\hat{B})$. Todavia, devido ao teorema da representação única, esse teste seria equivalente a testarmos se o matróide $M(\hat{A})$ é binário e como já foi mencionado na seção 3.3, esse problema é intrinsicamente exponencial (*cf.* Seymour [1981]). Por outro lado, só precisamos testar se $M(\hat{A}) \simeq M(\hat{B})$ quando soubermos a priori que $M(\hat{B})$ é regular. Veremos como esse fato aliado ao teorema 4.3 (p. 82) e ao teorema de Camion a seguir, nos proporcionarão um meio de contornarmos esse entrave exponencial.

Uma *sinalização* de uma $\{0, \pm 1\}$-matriz corresponde a trocarmos alguns 1's dessa matriz por -1's.

Teorema 4.9 (Camion) *Se B uma $L \times C - \{0,1\}$-matriz que possui uma sinalização, tal que a matriz resultante é t.u., então essa sinalização é única, a menos de multiplicação de linhas e colunas de B por -1.*

A prova que veremos do teorema de Camion encontra-se em TLIP, pp. 296 e 297. Ela descreve, implicitamente, um algoritmo polinomial que encontra uma sinalização para uma $\{0, \pm 1\}$-matriz B de tal forma que a matriz resultante é t.u. *sse* existe uma sinalização que transforma B em uma matriz t.u..

Prova do teorema 4.9: Seja $\mathcal{H}(B)$ o grafo bipartido com bipartição (L, C), tal que um vértice $l \in L$ é adjacente a um vértice $c \in C$ *sse* $B_{l,c} = 1$. Seja F uma floresta maximal de $\mathcal{H}(B)$ e B' uma matriz t.u. resultante de uma sinalização de B. Através de multiplicação de linhas e colunas de B' por -1 podemos fazer com que as entradas correspondentes às arestas de F sejam todas igual a 1. De fato, seja $l \in L$ (resp., $c \in C$) uma folha de F e $e = lc \in F$. Podemos multiplicar a linha l (resp., coluna c) por -1, se necessário, de tal forma que $B'_{l,c}$ seja igual a 1. Depois, podemos proceder de maneira análoga com a floresta $F - e$ em lugar de F e assim por diante, de tal forma que ao final todas as entradas de B' correspondentes a arestas em F sejam igual a 1. Logo, podemos supor, sem perda de generalidade, que todas as entradas de B' correspondentes a arestas em F são iguais a 1.

Para provarmos a proposição usaremos o fato de que num circuito K de $\mathcal{H}(B)$ sem cordas a soma das entradas de B' correspondentes às arestas em K é divisível por 4 (veja teorema 4.2 *iii.*). Consideremos uma ordenação das arestas $\{e_1, ..., e_k\}$ de $\mathcal{H}(B)$ que não estão em F tal que se o circuito em $F + e_j$ tem comprimento maior que o circuito em $F + e_i$, então $i < j$. Para cada j existe um circuito sem cordas em $F \cup \{e_1, ..., e_{j-1}\}$ que contém e_j. Portanto,

fixados os valores das entradas em B' correspondentes às arestas em $F \cup \{e_1, ..., e_{j-1}\}$ existe uma única possibilidade para atribuirmos valor para a entrada correspondente a e_j. Como essa é a situação para cada e_j o resultado segue por indução. ■

Caso $M(\hat{B})$ seja regular então pelo teorema 4.3 sabemos que existe uma representação canônica t.u. de $M(\hat{B})$ em relação a base L, digamos $\hat{S}$. Pelo teorema da representação única é claro que S e B têm as mesmas entradas não nulas, logo $\hat{B}$ possui uma sinalização que a transforma em uma matriz t.u.. Além disso, pelo teorema de Camion essa representação é única, a menos de equivalência projetiva. Tudo isso mostra que o problema 4.4 pode ser decomposto em:

1. testar se $M(\hat{B})$ é regular; e
2. testar se $\hat{A} \sim \hat{S}$ (i.e. $\hat{A}$ é p.e. a uma sinalização que transforma $\hat{B}$ em uma matriz t.u.) .

É claro que só faremos o teste do item 2 quando soubermos, a priori, que $M(\hat{B})$ é regular.

Testar equivalência projetiva entre matrizes na forma canônica em relação a uma mesma base não é difícil e pode ser resolvido de maneira eficiente pelo algoritmo *TestaPE* (p. 58). Se é possível sinalizarmos uma $\{0,1\}$-matriz de tal forma que a matriz resultante seja t.u., então podemos obter uma tal sinalização rapidamente através do algoritmo implicito na prova do teorema de Camion. Logo testar se $M(\hat{A})$ é regular ($\hat{A}$ é t.u.) fica reduzido a testar se $M(\hat{B})$ é regular. A seguir veremos um algoritmo que faz este teste.

Algoritmo para testar se um matróide binário é regular

Seja B uma $L \times C - \{0,1\}$-matriz. Segundo o teorema de Seymour vale uma das seguintes alternativas:

i. $M(\hat{B})$ é gráfico;
ii. $M(\hat{B}^t)$ é grafico (i.e. $M(\hat{B})$ é cográfico, veja proposição 3.4, p. 70);
iii. $M(\hat{B}) \simeq R_{10}$;
iv. $M(\hat{B}) = M(\hat{B}_1) +_k M(\hat{B}_2)$, onde $\{M(\hat{B}_1), M(\hat{B}_2)\}$ é uma k-decomposição de $M(\hat{B})$, $k = 1, 2$ ou 3; ou
v. $M(\hat{B})$ não é regular.

É claro que se uma das alternativas *i.–iii.* ocorre então $M(\hat{B})$ é regular.

Como vemos, o teorema de Seymour sugere naturalmente um algoritmo recursivo para testarmos se $M(\hat{B})$ é regular.

Nas seções 2.2 e 3.3 vimos algoritmos eficientes que verificaram se a alternativa *i.* ou *ii.* ocorre (i.e. testam se uma matróide binário é gráfico). É claro que podemos testar em tempo polinomial se $M(\hat{B}) \simeq R_{10}$. Assim, só resta sabermos como encontrar uma k-decomposição de $M(\hat{B})$, $k = 1, 2$ ou 3.

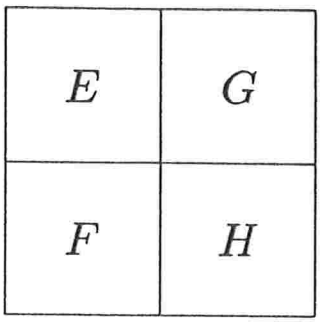

Figura 4.7: Matriz básica $\hat{B}$, onde $\rho F + \rho G \leq k - 1$, o número de colunas de F e G é pelo menos k.

Tendo em mente como obtivemos uma representação para a k-soma, $k = 1, 2$ e 3, de matróides representáveis (veja figuras 4.4, 4.5 e 4.6), não é muito difícil percebermos que encontrar uma k-decomposição de $M(\hat{B})$ é equivalente a encontrarmos uma ordenação das linhas e colunas de $\hat{B}$ tal que $\hat{B}$ assuma a forma tabular mostrada na figura 4.7, onde

$$\rho F + \rho G \leq k - 1$$

e o número de colunas de F e de G é pelo menos k.

Seja $E = L \cup C$. A matriz $\hat{B}$ possui uma ordenação como a mostrada na figura 4.7 *sse* $M(\hat{B})$ possui uma k-separação, i.e. existe uma partição $\{E_1, E_2\}$ de E tal que

$$| E_1 | \geq k \leq | E_2 |, \quad \rho E_1 + \rho E_2 = \rho M + k - 1.$$

Além disso, é claro que a partir de uma k-separação de $M(\hat{B})$ podemos obter uma ordenação das linhas e colunas de $\hat{B}$ que a coloquem no formato da matriz da figura 4.7. Assim, para obtermos uma k-decomposição de $M(\hat{B})$ é suficiente encontrarmos uma k-separação deste matróide.

Cunningham & Edmonds [1980] observaram que uma k-separação de uma matróide pode ser encontrada em tempo polinomial utilizando-se o *algoritmo de intersecção de matróides* (AIM). O AIM, que pode ser encontrado em MTIA, sec. 13.1 e Lawler [1976: cap. 8], é um algoritmo que resolve o *problema da intersecção de dois matróides* (PIM), i.e:

dados : Matróides $M_1 = (E, \mathcal{I}_1)$ e $M_2 = (E, \mathcal{I}_2)$.
encontrar : Um conjunto de I de cardinalidade máxima tal que $I \in \mathcal{I}_1 \cap \mathcal{I}_2$.

O AIM não só resolve o PIM como também devolve uma partição $\{E_1, E_2\}$ de E tal que $| I | = \rho_1 E_1 + \rho_2 E_2$, onde ρ_1 e ρ_2 são as funções posto de M_1 e M_2, respectivamente.

Para todo $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ e $X \subseteq E$ vale que $| I | \leq \rho X + \rho(E \setminus X)$, pois

$$| I | = | I \cap X | + | I \setminus X | \leq \rho_1 X + \rho_2 (E \setminus X).$$

Devido a este fato e a corretude do AIM temos que vale o seguinte teorema do tipo minimax:

Teorema 4.10 (Edmonds) *Se ρ_1 e ρ_2 são as funções posto dos matróides $M_1 = (E, \mathcal{I}_1)$ e $M_2 = (E, \mathcal{I}_2)$, respectivamente, então*

$$\max_{I \in \mathcal{I}_1 \cap \mathcal{I}_2} | I | = \min_{X \subseteq E} [\rho_1 X + \rho_2 (E \setminus X)].$$

■

Antes de continuarmos é conveniente que observemos que se X é um conjunto de elementos de $M(\hat{B})$ e ρ_X é a função posto do matróide $M(\hat{B})/X$ sobre $E \setminus X$, então

$$\rho_X Y = \rho(Y \cup X) - \rho X,$$

onde ρ é a função posto de $M(\hat{B})$. No matróide $M(\hat{B})/X$ consideramos dependência linear dos vetores colunas de $\hat{B}$ módulo o subespaço gerado pelos vetores de índice X (um conjunto V de vetores é l.i. módulo um subespaço S, se $V \cup K$ é l.i. para alguma base K de S).

Se $M(\hat{B})$ possui uma k-separação $\{E_1, E_2\}$, então existem $S \subseteq E_1$ e $T \subseteq E_2$ tal que

$$| S | = k = | T | \text{ e } S, T \in \mathcal{I}.$$

Consideremos agora os matróides $M_1 = (M(\hat{B}) \setminus T)/S$ e $M_2 = (M(\hat{B}) \setminus S)/T$. M_1 e M_2 são matróides sobre $E' = E \setminus (S \cup T)$ com função posto $\rho_1 X = \rho(X \cup S) - k$ e $\rho_2 X = \rho(X \cup T) - k$, respectivamente.

Se aplicarmos o AIM a M_1 e M_2, então ao final da sua execução o AIM nos devolverá $I' \in \mathcal{I}_1 \cap \mathcal{I}_2$ e uma partição $\{E'_1, E'_2\}$ de E' tal que $| I' | = \rho_1(E'_1) + \rho_2(E'_2)$. Assim,

$$| I' | = \rho(E'_1 \cup S) + \rho(E'_2 \cup T) - 2k.$$

Se $| I' | < \rho M(\hat{B}) - k$, então $\{E'_1 \cup S, E'_2 \cup T\}$ é uma k-separação de $M(\hat{B})$. Por outro lado, se $| I' | \geq \rho M(\hat{B}) - k$ então, pelo teorema de Edmonds, não existe uma k-separação $\{E_1, E_2\}$ de $M(\hat{B})$ tal que $S \subseteq E_1$ e $T \subseteq E_2$.

Portanto, para encontrarmos uma k-separação, $\{E_1, E_2\}$ de $M(\hat{B})$ basta repetirmos o procedimento acima para todo os possíveis subconjuntos $S, T \subseteq E, S \cap T = \emptyset$ tal que

$$| S | = k = | T | \text{ e } S, T \in \mathcal{I}.$$

Existem $O(| E |^6)$ subconjuntos S e T que podem satisfazer essas hipóteses. Assim, podemos encontrar, se existir, uma k-separação de $M(\hat{B})$ em $O(| E |^6)$ chamadas do AIM.

Cunningham [1973: cap. 5] descreve um algoritmo que encontra uma k-decomposição de uma matróide M qualquer. O algoritmo que vimos é um versão particular do algoritmo de Cunningham.

O algoritmo

Resumiremos agora, rapidamente, o algoritmo visto para reconhecer unimodularidade total em matrizes.

Podemos supor, sem perda de generalidade, que a matriz dada está na forma canônica (pois unimodularidade total é invariante sob pivotação). Para decidirmos se uma matriz $\hat{A}$ é t.u. primeiro devemos verificar se toda entrada de A contém um 0, 1 ou -1. Em caso afirmativo devemos agora verificar se $M(\hat{A})$ é regular (veja teorema 4.3).

Para verificarmos se $M(\hat{A})$ é regular a estratégia empregada foi testar se $M(\hat{B})$ é regular. Se $M(\hat{B})$ não é regular, então $M(\hat{A})$ não é regular (veja proposição 4.2). Caso contrário, construímos uma sinalização para $\hat{B}$, tal que $M(\hat{A})$ é regular *sse* $\hat{A}$ é projetivamente equivalente a essa sinalização (veja teorema Camion nesta seção).

Para testarmos se $M(\hat{B})$ é regular utilizamos de uma maneira fundamental o teorema de Seymour (teorema 4.7). O teste consiste em primeiro verificarmos se $M(\hat{B})$ é gráfica, cográfica (i.e., $M(\hat{B}^t)$ é gráfica) ou é R_{10}. Se algum desses casos ocorre concluímos que $M(\hat{B})$ é regular. O nosso próximo passo é encontrar, se existir, $\{M(\hat{B}_1), M(\hat{B}_2)\}$ uma k-decomposição, $k = 1, 2$ ou 3, de $M(\hat{B})$. Se não existir uma tal k-decomposição, então $M(\hat{B})$ não é regular. Caso contrário, pelo teorema de Seymour, $M(\hat{B})$ é regular *sse* $M(\hat{B}_1)$ e $M(\hat{B}_2)$ forem regulares.

É interessante observarmos que os algoritmos que vimos para decidir se uma matriz $M(\hat{A})$ sobre $\mathbb{R}$ é gráfica procederam de maneira análoga ao algoritmo que estudamos nesta seção. Ou seja, o problema foi reduzido a: decidir se $M(\hat{B})$ era gráfico; em caso afirmativo encontrar uma representação $\hat{A}'$ de $M(\hat{B})$ sobre $\mathbb{R}$ e testar se $\hat{A}$ e $\hat{A}'$ são projetivamente equivalentes.

O fato de termos um teorema de unicidade de representação sobre GF(2) (teorema 4.8) e $\hat{A}$ e $\hat{A}'$ serem matrizes sobre $\mathbb{R}$ que representam o matróide $M(\hat{A})$ sobre uma mesma base (em caso de $\hat{A}$ ser gráfica) foi muito importante, tanto para reconhecer matrizes totalmente unimodulares, como matrizes gráficas. Esta importância pode ser percebida através das seguintes matrizes:

$$\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} 1 & & & +1 & \\ & 1 & & -1 & -1 \\ & & 1 & & +1 \end{array}\right] \end{array}, \qquad \begin{array}{c} \begin{array}{ccccc} 1 & 5 & 3 & 4 & 2 \end{array} \\ \left[\begin{array}{ccccc} 1 & & & +1 & \\ & 1 & & -1 & -1 \\ & & 1 & -1 & -1 \end{array}\right] \end{array}.$$

Ambas as matrizes acima têm como realização o grafo D da figura 4.8 e portanto, são gráficas e projetivamente equivalentes. Entretanto, o algoritmo $TestaPE$ não detectaria que elas são projetivamente equivalentes (elas nem sequer têm as mesmas entradas não nulas), pois o algoritmo $TestaPE$ assume implicitamente que as matrizes de entrada são representações de $\mathcal{M}(D)$ em relação a uma mesma base (veja como foi feita a construção da representação para $\mathcal{M}(D)$ em 2.9, p. 56).

O livro TLIP, cap. 20, apresenta um algoritmo, descrito inteiramente em termos de matrizes, que reconhece matrizes totalmente unimodulares e $\{0, \pm 1\}$-matrizes gráficas. O algoritmo de intersecção de matróides está implícito no algoritmo apresentado para encontrar-se uma 1-,2- ou 3-decomposição de uma matriz.

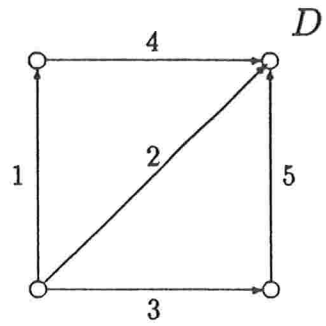

Figura 4.8: $T = \{1,2,3\}$ e $T' = \{1,3,5\}$ são bases de $\mathcal{M}(D)$.

4.3 Propriedades hereditárias

Nesta seção consideraremos o problema de encontrar submatrizes que satisfaçam alguma propriedade hereditária.

Como já foi mencionado na seção 1.1, a motivação para encontrar-se submatrizes "grandes" que satisfaçam alguma propriedade especial é permitir que o problema que essa matriz representa possa talvez ser resolvido por algoritmos particulares, que explorem essa estrutura. Por exemplo, se a matriz de restrições de um PLI possui um subconjunto grande de linhas com alguma propriedade "tratável", então algum método de decomposição como relaxação lagrangeana talvez possa ser aplicado (veja, por exemplo TLIP, pp. 367–370). Analogamente, se existir um subconjunto grande de colunas com alguma propriedade "especial", então decomposição de Benders pode vir a ser útil (veja, por exemplo, TLIP, pp. 371 e 372]).

Infelizmente, como veremos, Bartholdi [1982], utilizando os resultados de Yannakakis [1981], mostrou que o problema de encontrar em uma matriz dada uma submatriz de tamanho fixo que satisfaça alguma propriedade Π que seja hereditária, "interessante" e satisfeita por toda matriz identidade é $\mathcal{NP}$-difícil. Entre as propriedades que satisfazem essas condições encontram-se todas as propriedades vistas por nós nesta dissertação, ou seja, Π_g = "ser matriz de um grafo", Π_s = "ser a matriz de um grafo sinalizado", Π_{graf} = "ser uma matriz gráfica" e Π_{tu} = "ser uma matriz totalmente unimodular". Mesmo as propriedade Π_i = "ser a matriz identidade" satisfaz as condições acima. Assim, o que veremos nesta seção generaliza (e muito) os resultados vistos por nós na seção 1.2.

Como observou Bartholdi, não é nenhum espanto que esses problemas sejam $\mathcal{NP}$-difíceis, o que é notável é que todos podem ser tratados *en masse*.

A seguir descreveremos um pouco do trabalho de Yannakakis, que é todo em termos do "problema da remoção de vértices" em grafos, e em seguida veremos as observações feitas por Bartholdi.

O problema da remoção de vértices[1] para um propriedade Π é o problema:

dados	:	Uma grafo $G = (V, E)$ simples e um inteiro k.
pergunta	:	Existe $V' \subseteq V, \mid V' \mid < k$, tal que $G[V \setminus V']$ satisfaz Π? (4.6)

Muitos problemas, como o problema da cobertura mínima, podem ser apresentados na forma de um problema de remoção de vértices através de uma especificação conveniente da propriedade Π.

Diremos que uma propriedade sobre grafos é *não trivial* sobre um domínio U se ela é satisfeita pelo grafo com um único vértice e sem arestas e ela não é satisfeita por todos os grafos em U. Por exemplo, a propriedade Π_c = "ser um grafo completo" é não trivial sobre o conjunto dos grafos. Uma propriedade Π será dita *interessante* sobre um domínio U se existem grafos arbitrariamente grandes que satisfaçam Π. Por exemplo, Π_c não é uma propriedade interessante sobre os grafos bipartidos.

Se Π não é uma propriedade interessante o problema da remoção de vértices pode ser resolvido em tempo polinomial. De fato, se Π é satisfeita para um número finito de grafos $G_1, ..., G_k$, então podemos encontrar, em tempo polinomial, o grafo G_i, com o maior número de vértices, que é isomorfo a um subgrafo induzido por vértices de um grafo dado.

De maneira análoga ao que foi definido para matrizes diremos que uma *propriedade Π sobre grafos é hereditária*, se para cada grafo $G = (V, E)$ que satisfaça Π, tenhamos que todos os seus subgrafos induzidos por vértices, i.e., $G[V']$ com $V' \subseteq V$, também satisfazem Π. Se G não satisfaz uma propriedade hereditária Π então todo grafo que contenha G como subgrafo induzido por vértices não satisfaz Π. Diremos que um grafo que não satisfaz uma propriedade hereditária Π e é minimal com relação a isso é uma *obstrução* ou *subgrafo proibido* da propriedade Π. Qualquer propriedade sobre grafos que possa ser caracterizada em termos de obtruções é uma propriedade hereditária. Exemplos de propriedades hereditárias sobre grafos são: "G é um clique"; "G é um conjunto independente"; "G é uma floresta"; "G é bipartido"; "G é planar"; "G é um grafo de comparabilidade".

Teorema 4.11 (Yannakakis [1981]) *O problema da remoção de vértices para propriedades sobre grafos bipartidos que sejam hereditárias, não triviais, e satisfeitas por subgrafos induzidos por emparelhamentos é $\mathcal{NP}$-completo.* ■

É claro que no teorema de Yannakakis só estão sendo consideradas propriedades que possam ser reconhecidas em tempo polinomial não determinístico (i.e. estão em $\mathcal{NP}$).

Grafos bipartidos podem ser naturalmente associados a $\{0,1\}$-matrizes da seguinte maneira: se G é um grafo com bipartição (L, C), então podemos contruir $L \times C$-matriz A, onde para todo $l \in L$ e $c \in C$ temos que $A_{l,c} = 1$ se l é adjacente a c em G, ou seja $G = \mathcal{H}(A)$. Observermos que essa associação não é única, pois G pode corresponder a A ou A^t.

[1] *Node-deletion problem.* Sobre o problema da remoção de vértices veja também Lewis & Yannakakis [1980].

A correspondência acima entre grafos bipartidos e $\{0,1\}$-matrizes faz com que cada propriedade Π sobre grafos bipartidos induza uma propriedade Π' sobre $\{0,1\}$-matrizes da seguinte maneira: uma $\{0,1\}$-matriz A satisfaz Π', se $G = \mathcal{H}(A)$ satisfaz Π.

Seja Π' uma propriedade sobre $\{0,1\}$-matrizes. Analogamente ao que foi definido para propriedades sobre grafos, diremos Π' é *não trivial* se ela é satisfeita pela matriz $[0]$ e ela não é satisfeita por todas $\{0,1\}$-matrizes. Diremos ainda que Π' é *interessante* se existem matrizes arbitrariamente grandes, que satisfaçam Π'.

É evidente que, se Π é uma propriedade sobre grafos bipartidos e Π' é a propriedade induzida por Π sobre $\{0,1\}$-matrizes, então: Π' é hereditária *sse* Π é hereditária; Π' é não trivial *sse* Π é não trivial; Π' é interessante *sse* Π é interessante; e Π' é satisfeita por toda matriz identidade *sse* Π é satisfeita por qualquer grafo induzido por um emparelhamento.

Tudo que vimos até agora nos sugere que o problema da remoção de vértices para uma propriedade Π hereditária e interessante sobre grafos bipartidos é polinomialmente redutível ao problema:

$$\begin{array}{lll} \textit{dados} & : & \text{Uma } L \times C - \{0,1\}\text{-matriz } A \text{ e inteiros } k_1 \text{ e } k_2. \\ \textit{pergunta} & : & \text{Existe uma } L' \times C'\text{-submatriz de } A \text{ que satisfaça } \Pi', \text{ tal que} \\ & & \mid L' \mid \geq k_1 \text{ e } \mid C' \mid \geq k_2? \end{array} \qquad (4.7)$$

onde Π' é a propriedade sobre $\{0,1\}$-matrizes induzida por Π. O problema acima é chamado de *o problema da submatriz máxima* para a propriedade Π'.

Com efeito, consideremos um grafo bipartido $G = (V, E)$ com bipartição (L, C) e um inteiro k, instância arbitrária do problema da remoção de vértices para a propriedade Π fixa. Como Π é uma propriedade interessante e hereditária, Π é satisfeita por qualquer conjunto independente de vértices. Assim, se k é maior ou igual ao tamanho da cobertura mínima de G, a resposta ao problema da remoção de vértices sobre esta instância é obviamente sim. Logo, estaremos assumindo, sem perda de generalidade, que k é menor que o tamanho da cobertura mínima. Na realidade, para os nossos fins, é suficiente $k < \min\{\mid L \mid, \mid C \mid\}$.

A partir de G podemos construir em tempo $O(\mid L \mid\mid C \mid)$ uma $L \times C$-matriz A, tal que $G = \mathcal{H}(A)$. Agora, basta testarmos para todos os possiveis valores de k_1 e k_2, $k_1 + k_2 = \mid V \mid -k$ (observemos que o fato de k ser menor do que $\min\{\mid L \mid, \mid C \mid\}$ garante que k_1 e k_2 sejam não nulos), se A ou A^t possui uma submatriz com pelo menos k_1 linhas e pelo menos k_2 colunas que satisfaça Π'. Para isso $O(\mid L \mid + \mid C \mid)$ testes são suficientes. Não é difícil vermos que, se $L' \neq \emptyset \neq C'$, então a $L' \times C'$-submatriz de A ou a $C' \times L'$-submatriz de A^t satisfaz Π' *sse* $G[L' \cup C']$ satisfaz Π.

O que acabamos de ver mostra como, a partir de um algoritmo polinomial para o problema da submatriz máxima para Π', podemos obter um algoritmo polinomial para o problema da remoção de vértices para Π sobre grafos bipartidos. Como conseqüência dessa redução polinomial temos o seguinte colorário do teorema de Yannakakis:

Colorário 4.1 (Bartholdi [1982]) *O problema da submatriz máxima para uma propriedade*

Π' que é hereditária, não trivial, interessante e satisfeita por toda matriz identidade é $\mathcal{NP}$-completo. ■

Dentre as propriedades que satisfazem as condições do corolário acima encontra-se também Π_b = "ser uma matriz balanceada".

[Uma $\{0,1\}$- matriz A é balanceada se ela não possui uma submatriz quadrada de ordem ímpar com exatamente dois 1's em cada linha e cada coluna. Ou seja, se A não possui uma submatriz da forma:

$$\begin{bmatrix} 1 & 0 & 0 & \dots & \dots & 1 \\ 1 & 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 1 & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 & 0 \\ 0 & 0 & \dots & \dots & 1 & 1 \end{bmatrix}$$

Como a matriz acima tem determinante dois, então $\{0,1\}$-matrizes t.u. são balanceadas. Fulkerson, Hoffman & Oppenheim provaram que, se A é uma matriz balanceada então o poliedro $\{x \mid x \geq 0, Ax = 1\}$ é inteiro. Mais sobre matrizes balanceadas pode ser encontrado em TLIP, sec. 21.5.]

Através de modificações na prova do teorema de Yannakakis, Bartholdi provou que se Π' é uma propriedade hereditária, não trivial, interessante e satisfeita por toda matriz identidade, então os seguintes problemas são $\mathcal{NP}$-completos:

dados : Uma $L \times C - \{0,1\}$-matriz A e um inteiro k.
pergunta : Existe uma $L \times C'$-submatriz de A que satisfaça Π', tal que $\mid C' \mid \geq k$?

dados : Uma $L \times C - \{0,1\}$-matriz A e um inteiro k.
pergunta : Existe uma $L' \times C$-submatriz de A que satisfaça Π', tal que $\mid L' \mid \geq k$?

Consideremos uma $L \times L$-matriz A, toda $L' \times L'$-submatriz de A é dita uma *submatriz principal* de A. Utilizando o trabalho de Lewis & Yannakakis [1980], Bartholdi ainda mostrou que se Π' é uma propriedade hereditária sobre submatrizes principais, não trivial, interessante e é satisfeita por toda matriz identidade, então o problema:

dados : Uma $L \times L$-matriz A e um inteiro k.
pergunta : Existe uma $L' \times L'$-submatriz de A que satisfaça Π', tal que $\mid L' \mid \geq k$?

é $\mathcal{NP}$-completo. Uma propriedade que satisfaz essas condições é Π_t = "ser uma matriz triangular". Mais propriedades e referências podem ser encontradas no artigo de Bartholdi.

Como os problemas que vimos são $\mathcal{NP}$-completo para propriedades sobre $\{0,1\}$-matrizes, então os problemas correspondentes para matrizes sobre $\mathbb{R}$ são $\mathcal{NP}$-completo.

Assim, Bartholdi mostrou que o problema de encontrar-se numa matriz uma submatriz máxima que satisfaça uma propriedade "boa" é $\mathcal{NP}$-difícil para muitas das propriedades conhecidas, mesmo que existam algoritmos polinomiais para reconhecer-se essas propriedades.

[Truemper [1985] apresenta um algoritmo que dado uma matriz encontra submatrizes maximais que satisfazem uma propriedade hereditária que atenda a certas condições. Π_{graf} e Π_{tu} atendem essas condições.]

Apêndice A

Exemplo de reconhecimento de rede escondida

Neste apêndice veremos um exemplo em que reconhecimento de matrizes gráficas pode ser aplicado para transformar um PL

$$\begin{array}{ll} \min & cx \\ & \hat{A}x = b \\ & x \geq 0 \end{array} \tag{A.1}$$

que é uma rede escondida (i.e. $\hat{A}$ é uma matriz gráfica) em um PL

$$\begin{array}{ll} \min & c'y \\ & \tilde{N}y = b' \\ & y \geq 0 \end{array} \tag{A.2}$$

onde $\tilde{N} = \pi\hat{A}\Delta$ é uma $\{0, \pm 1\}$-matriz de um grafo, π e Δ são matrizes diagonais não singulares, tais que y é solução de A.2 *sse* Δy é solução de A.1. Em particular, como veremos, não é necessário que tenhamos conhecimento explícito de π para transformar o PL A.1 no PL A.2.

A.1 O PL original

Consideremos o PL

$$\min x_1 + 3x_2 + x_3 + x_4 + 2x_5 + x_6 + 2x_7 + 4x_8 + x_9 + 7x_{10} + 8x_{11} + x_{12} + x_{13} + 3x_{14} + x_{15} + 2x_{16} + x_{17}$$

$$\begin{array}{llllllllllll}
x_1 & +3x_9 & & & & & & +3x_{15} & & & = & 3 \\
x_2 & -9x_9 & -12x_{10} & -6x_{11} & & & & -9x_{15} & & & = & 9 \\
x_3 & -6x_9 & -8x_{10} & -4x_{11} & & & -4x_{14} & -6x_{15} & & & = & -8 \\
x_4 & +3x_9 & & & & & +2x_{14} & & -x_{16} & & = & 2 \\
x_5 & & & & & & & -15x_{15} & -5x_{16} & +5x_{17} & = & 0 \\
x_6 & & & & & +3x_{13} & & +9x_{15} & +3x_{16} & -3x_{17} & = & 9 \\
x_7 & & & -4x_{11} & -2x_{12} & -2x_{13} & & -6x_{15} & -2x_{16} & & = & -8 \\
x_8 & & & +8x_{11} & +4x_{12} & & & & & & = & 12
\end{array}$$

Denotaremos por $\hat{A}$ a matriz de restrições do PL acima e por A a matriz obtida a partir de $\hat{A}$ removendo-se as colunas da identidade.

Nossa primeira tarefa para decidir se o PL acima é uma rede escondida será verificar se a matriz $\hat{A}$ é gráfica. Isto, como foi visto no capítulo 2, pode ser feito aplicando-se o algoritmo *Guloso* (p. 62) à matriz $\hat{A}$. O algoritmo *Guloso*, por sua vez, se utiliza dos algoritmos *Grafo* (p. 53) e *TestaPE* (p. 58). Por questão de simplicidade na exposição da aplicação de tais algoritmos à matriz A, apresentaremos separadamente a execução dos algoritmo *Grafo* e *TestaPE*. Entretanto, como pode ser visto no algoritmo *Guloso*, é claro que esses algoritmos podem trabalhar concomitantemente para decidir se $\hat{A}$ é gráfica.

A.2 Execução do algoritmo *Grafo*

Seja B a imagem booleana de A. Como a primeira coluna de A tem mais de uma entrada não nula acrescentaremos a B uma coluna de índice 18 com apenas a entrada $(1,1)$ não nula e chamaremos de B a matriz resultante. Assim,

$$B = \begin{array}{c} \begin{array}{cccccccccc} 18 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \end{array} \\ \left[\begin{array}{cccccccccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0
\end{array}\right] \end{array}$$

Como foi observado na entrada do algoritmo *Grafo*, o acréscimo da coluna 18 à imagem booleana de A não altera em nada o fato desta ser ou não gráfica.

Observemos que, com a ordenação das colunas com que B foi apresentada acima, B é uma matriz totalmente conexa, pois cada coluna de B, exceto a coluna 18, possui um 1 em uma linha, que já possui um 1 em uma coluna anterior.

Veremos agora a execução do algoritmo *Grafo* tendo como entrada a matriz B. Daremos os nomes $G_1, G_2, ...$ aos grafos, que forem sendo criados durante a execução do algoritmo para formar a t-decomposição.

Coluna 18: Na linha 1–3 do algoritmo *Grafo* será criado o grafo G_1, realização para a coluna 18, com $mp(G_1) = 18$. No início $D = \{G_1\}$. Notemos que D ainda não é uma t-decomposição, pois G_1 têm apenas duas arestas. Entretanto, D é uma t-decomposição com relação a todos os outros aspectos (veja figura A.1).

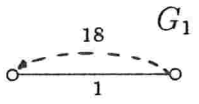

Figura A.1: Decomposição D (coluna 18).

Coluna 9: Na linha 5 do algoritmo *Grafo* teremos $P = \{1\}$. A execução da linha 1 do algoritmo *HipoCaminho* é trivial, pois D tem apenas um membro. A execução das demais linhas do algoritmo *HipoCaminho* também é trivial, pois como $\mid P \mid = 1$, é claro que P é um hipocaminho de $\Sigma(D)$ (o grafo soma de D). A execução das linhas 13.1, 13.2 e 14 do algoritmo *HipoCaminho* farão com que $K_1, K_2 \leftarrow G_1$ e u_1 e u_2 sejam as pontas da aresta 1.

Na execução da linha 4 do algoritmo *Atualiza* será criado e acrescentado a D^* o circuito G_2, cujas arestas são $\{2, 3, 4, 5, 19\}$, onde o marcador 19 tem orientação arbitrária. Finalmente, na linha 7 do algoritmo *Atualiza* as pontas da aresta 1 serão ligadas através do marcador 19. A t-decomposição $D = \{G_1, G_2\}$ resultante após a execução da linha 9 do algoritmo *HipoCaminho* pode ser vista na figura A.2.

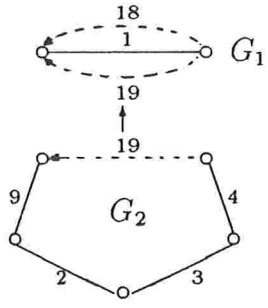

Figura A.2: t-decomposição D (coluna 9).

Coluna 10: Desta vez teremos $P = \{2, 3\}$. Na linha 1 do algoritmo *HipoCaminho* será construída $\hat{D} = \{G_2\}$, a t-decomposição reduzida de D com relação a P. As linha 3 e 4 não serão executadas, já que $\mid \hat{D} \mid$ é novamente igual a 1. Na linha 10 as arestas de G_2 serão reordenadas de maneira que $W_Q = \{2, 3\}$ seja o conjunto de arestas de um caminho em G_2.

As linhas 13.1 e 13.2 farão com que $K_1, K_2 \leftarrow G_2$ e na linha 14 teremos que u_1 e u_2 serão as pontas de W_Q em G_2.

A linha 3 do algoritmo *Atualiza* criará a aresta $f = C \setminus P = 10$. Como $K_1 = K_2$, G_2 é um circuito e u_1, u_2 não são adjacentes, então, na linha 10 do algoritmo *Atualiza*, o circuito G_2 será removido de D^*, serão criados marcadores $f_1 = 20$ e $f_2 = 21$ e criados ainda circuitos formados pelas arestas $\{4, 9, 19, 20\}$ e $\{2, 3, 21\}$, bem como um elo formado por $\{20, 10, 21\}$ os quais serão todos acrescentados a D^*. A estrutura subjacente de arborescência dos membros de D^* é facilmente obtida da orientação dos membros de D. A t-decomposição D atualizada é mostrada na figura A.3, onde o nome G_2 foi dado ao circuito formado pelas arestas $\{4, 9, 19, 20\}$.

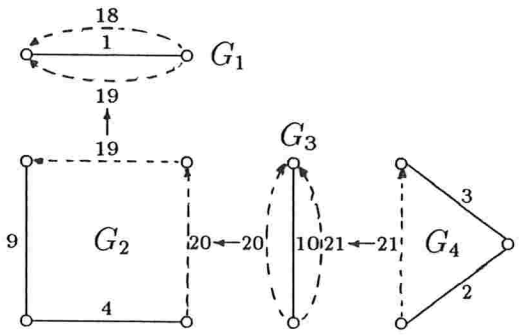

Figura A.3: t-decomposição D (coluna 10).

Coluna 11: Novamente $P = \{2, 3\}$. Na linha 1 do algoritmo *HipoCaminho* teremos a t-decomposição reduzida $\hat{D} = \{G_4\}$. Ainda no algoritmo *HipoCaminho*, na linha 10, as arestas de $Q = G_4$ serão ordenadas de maneira que $W_Q = \{2, 3\}$ seja um caminho em G_4 (desta feita este passo é redundante), na linha 13.3 teremos $K_1, K_2 \leftarrow G_4$. Entretanto, como $K_1 = K_2$, as pontas de P são as pontas de $mp(G_4)$ e G_4 não é um elo, então na linha 13.7 $K_1, K_2 \leftarrow pai(G_4) = G_3$ e devido à linha 14 u_1 e u_2 serão as pontas de P em G_3.

Durante a execução do algoritmo *Atualiza*, na linha 4, serão criados um novo marcador $f = 22$ e um novo circuito G_5, cujas arestas são $(C \setminus P) + f = \{7, 8, 11, 22\}$, e G_5 será acrescentado a D^*. Agora como $K_1 = K_2 = G_3$ é um elo, então as pontas u_1 e u_2 serão ligadas pelo marcador 22. A decomposição D resultante após a execução da linha 9 do algoritmo *Grafo* pode ser vista na figura A.4.

Coluna 12: Desta feita, $P = \{7, 8\}$. Na linha 1 do algoritmo *HipoCaminho* será calculada a t-decomposição reduzida $\hat{D} = \{G_5\}$, e na linha 10 as arestas de $Q = G_5$ serão ordenadas de tal maneira que $W_Q = \{7, 8\}$ seja um caminho em G_5 (desta vez este passo é novamente redundante). Ainda durante a execução do algoritmo *HipoCaminho*, na linha 13.2, $K_1, K_2 \leftarrow G_5$ e, na linha 14, u_1 e u_2 serão as pontas de P em G_5.

Na linha 3 do algoritmo *Atualiza* teremos que $f = C \setminus P = 12$. Como $K_1 = K_2$, G_5 é um circuito e u_1, u_2 não são adjacentes então, na linha 10 do algoritmo *Atualiza*, o circuito G_5 será removido de D^*, serão criados marcadores $f_1 = 23$ e $f_2 = 24$ e circuitos formados pelas arestas $\{11, 22, 13\}$ e $\{7, 8, 24\}$, bem como um elo formado pelas arestas $\{23, 12, 24\}$, os quais serão todos acrescentados a D^*. Como já foi mencionado, a estrutura subjacente de

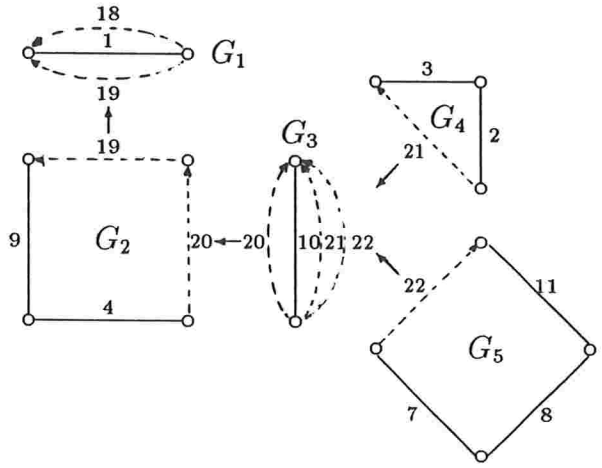

Figura A.4: t-decomposição D (coluna 11).

arborescência dos membros de D^* é facilmente obtida através da orientação dos membros de D. A t-decomposição resultante, após a execução da linha 9 do algoritmo *Grafo*, pode ser vista na figura A.5, onde o nome G_5 foi dado ao circuito formado por $\{11, 22, 23\}$.

Coluna 13 Temos que $P = \{7\}$ e $\hat{D} = \{G_7\}$. De maneira análoga ao que já foi visto, ao final do algoritmo *HipoCaminho*, teremos $K_1 = K_2 = G_7$ e u_1 e u_2 serão as pontas da arestas 7 em G_7.

Quando da execução da linha 4 do algoritmo *Atualiza*, serão criados um marcador $f = 26$ e um circuito formado pelas arestas $(C \setminus P) + f = \{6, 13, 26\}$ o qual será acrescentado a D^*. Como $K_1 = K_2$, G_7 é um circuito e u_1, u_2 são adjacentes em G_7 então, na linha 9 do algoritmo *Atualiza*, um novo marcador $f' = 25$ será criado, a aresta 7 será trocada pelo marcador 25 em G_7 e o elo $\{25, 7, 26\}$ será acrescentado a D^*. A t-decomposição resultante, após a execução da linha 9 do algoritmo *Grafo*, encontra-se na figura A.6.

Coluna 14: Nós temos $P = \{3, 4\}$ e $\hat{D} = \{G_2, G_3, G_4\}$. Na linha 3 do algoritmo *Hipocaminho* será calculada a partição π de $\hat{D}$. Assim, teremos que $\pi_0 = \{G_2\}$, $\pi_1 = \{G_3\}$ e $\pi_2 = \{G_4\}$. Durante a execução das linhas 2–5 do algoritmo *CalculaTipo* será verificado que $Tipo(G_4) = 3$ e durante a execução das linhas 8–15 será constatado que $Tipo(H) = 3$ ($H = \Sigma(G_3, G_4)$). Como $Q = G_2$ é um circuito então, durante a execução da linha 10 do algoritmo *HipoCaminho*, G_2 será ordenado de maneira que as arestas $W_Q = \{4, 20\}$ formem um caminho em G_2 com o marcador 20 em uma das extremidades (no caso em questão esse passo é redundante), e na linha 11 o marcador 20 será reorientado para que P seja um caminho de $\Sigma(G_2, G_3, G_4)$.

Na linha 3 do algoritmo *Atualiza* será criada a aresta $f = C \setminus P = 14$. Os vértices do caminho R da linha 11 serão G_2, G_3 e G_4. As linhas 12–16 não causarão efeito algum a G_3. Nas linhas 17 e 18 o circuito G_2 será lapidado com relação a $\{4, 20\}$. Finalmente, na linha 20 do algoritmo *Atualiza*, os grafos G_2, G_3, G_4 serão removidos de D^* e o grafo G, que é $\Sigma(R)$ com as

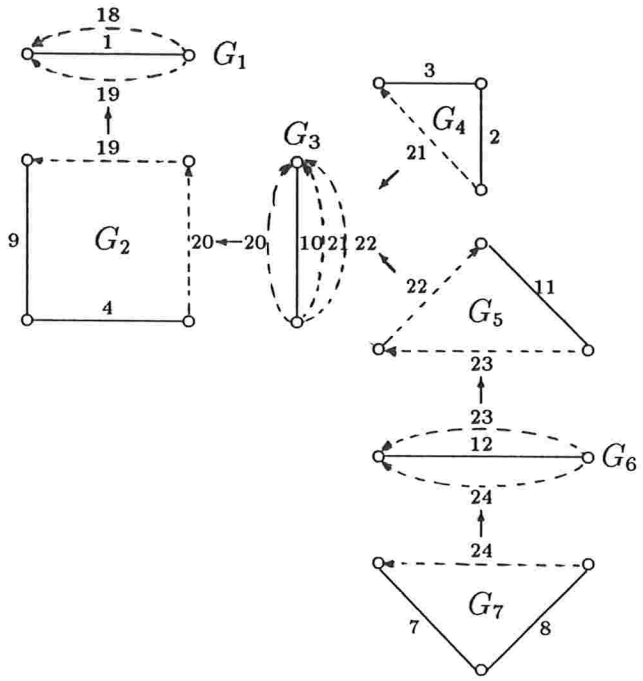

Figura A.5: t-decomposição D (coluna 12).

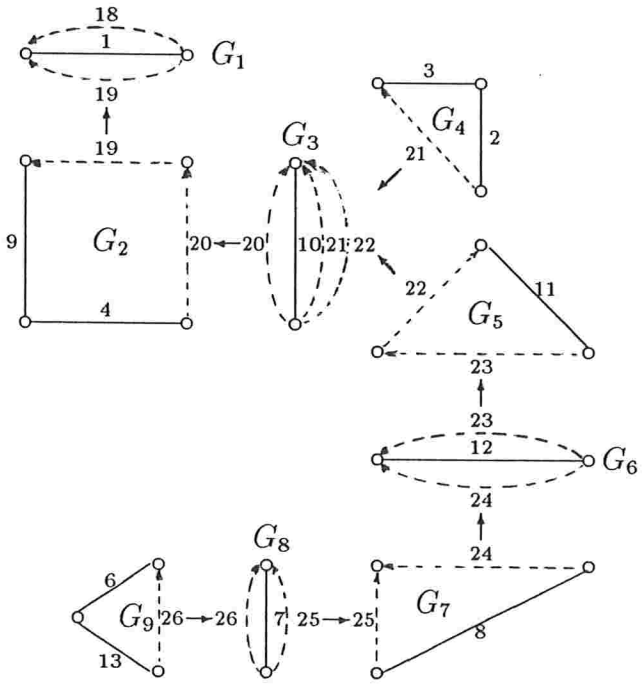

Figura A.6: t-decomposição D (coluna 13).

pontas u_1 e u_2 ligadas pela aresta aresta $f = 14$, será acrescentado. Na figura A.7 encontra-se a t-decomposição D já atualizada, onde o grafo $\Sigma(R)$ é chamado de G_2 e o grafo G_{10} é o grafo resultante da lapidação feita na linha 18 do algoritmo *Atualiza*.

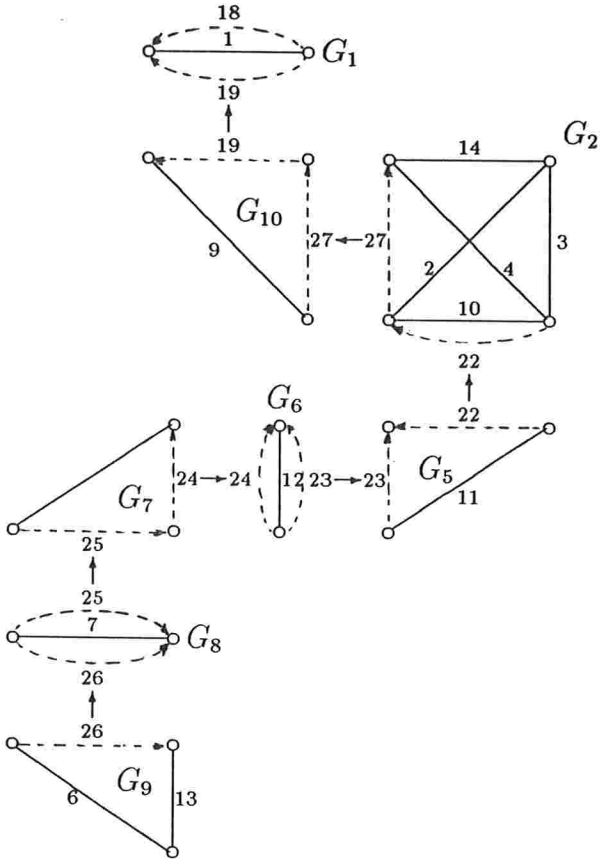

Figura A.7: t-decomposição D (coluna 14).

Coluna 15: $P = \{1,2,3,6,7\}$ e $\hat{D} = D$. A partição π calculada na linha 3 do algoritmo *HipoCaminho* será tal que $\pi_0 = \{G_1\}$, $\pi_2 = \{G_{10}\}$, $\pi_3 = \{G_2\}$, $\pi_4 = \{G_5\}$, $\pi_5 = \{G_6\}$, $\pi_6 = \{G_7\}$, $\pi_7 = \{G_8\}$ e $\pi_8 = \{G_9\}$.

Na execução das linhas 1–5 do algoritmo *CalculaTipo* será verificado que $Tipo(G_9) = 3$, pois $P_{G_9} + mp(G_9) = \{6, 26\}$ é um caminho em G_9 com o marcador 26 em uma das extremidades (as arestas de G_9 nem precisam ser reordenadas e na linha 5.3 teremos $K_1 \leftarrow G_9$). Durante a execução das linhas 6–16 do algoritmo *CalculaTipo* teremos que quando:

$j = 7$: $Q = G_8$, $H_1 = G_9$ e $Tipo(H) = 2$ ($H = \Sigma(Q, H_1)$), pois $W_H = \{6, 7\}$ é um caminho e $mp(Q) = 25$ é incidente a um vértice interno e a uma das pontas de W_H;

$j = 6$: $Q = G_7$, $H_1 = \Sigma(G_8, G_9)$ e $Tipo(H) = 3$, pois $W_H + mp(Q) = \{6, 7, 25, 24\}$ é um caminho em H com o $mp(Q) = 24$ em uma das extremidades;

$j = 5$: $Q = G_6$, $H_1 = \Sigma(G_7, G_8, G_9)$ e $Tipo(H) = 3$, pois $W_H + mp(Q) = \{6, 7, 23\}$ é um caminho em H com $mp(Q) = 24$ em uma das extremidades;

$j = 4$: $Q = G_5$, $H_1 = \Sigma(G_6, G_7, G_8, G_9)$ e $Tipo(H) = 3$, pois na linha 13 o marcador filho 23 de Q será reorientado de tal maneira que $W_H + mp(Q) = \{6, 7, 22\}$ seja um caminho em H com $mp(Q) = 22$ em uma das extremidades;

$j = 3$: $Q = G_2$, $H_1 = \Sigma(G'_5, G_6, G_7, G_8, G_9)$, onde G'_5 é o grafo G_5 após a reorientação do marcador filho 23, e $Tipo(H) = 3$, pois na linha 13 o marcador filho 22 de Q será reorientado de tal maneira que $W_H + mp(Q) = \{6, 7, 3, 2, 27\}$ sejam arestas de um caminho em H com $mp(Q) = 27$ em uma das extremidades; e finalmente

$j = 2$: $Q = G_{10}$, $H_1 = \Sigma(G'_2, G'_5, G_6, G_7, G_8, G_9)$, onde G'_2 é o grafo G_2 após a reorientação do marcador filho 22, e $Tipo(H) = 3$, pois o marcador filho 27 de Q será reorientado, na linha 13, de tal maneira que $W_H + mp(Q) = \{6, 7, 3, 2, 19\}$ seja um caminho em H com $mp(Q) = 19$ em uma das extremidades.

Assim, ao final da execução do algoritmo *CalculaTipo* a t-decomposição D terá sido alterada, de forma que o único filho completo de $raiz(D) = G_1$, i.e., $\Sigma(G'_{10}, G'_2, G'_5, G_6, G_7, G_8, G_9)$, seja um grafo bacana, onde G'_{10} é o grafo G_{10} após a reorientação do marcador filho 27.

No restante da execução do algoritmo *Hipocaminho* será verificado que P é um caminho de $\Sigma(D)$ (linhas 5–13), $K_2 \leftarrow G'_{10}$ (linha 13.4) e u_1, u_2 serão as pontas de P em G_9 e G'_{10}, respectivamente.

Na linha 4 do algoritmo *Atualiza* será criado um novo marcador $f = 28$ e um circuito de arestas $(C \setminus P) + f = \{5, 15, 28\}$, o qual será acrescentado a D^*. O caminho R da linha 11 será composto pelos membros $G'_{10}, G'_2, G'_5, G_6, G_7, G_8$ e G_9 de D^*. Na linha 20 serão removidos de D^* os membros que estão no caminho R e o grafo G, que é $\Sigma(R)$ com as pontas u_1 e u_2 ligadas pelo marcador $f = 28$, será acrescentado. A t-decomposição resultante é mostrada na figura A.8, onde chamamos de G_{10} o grafo G.

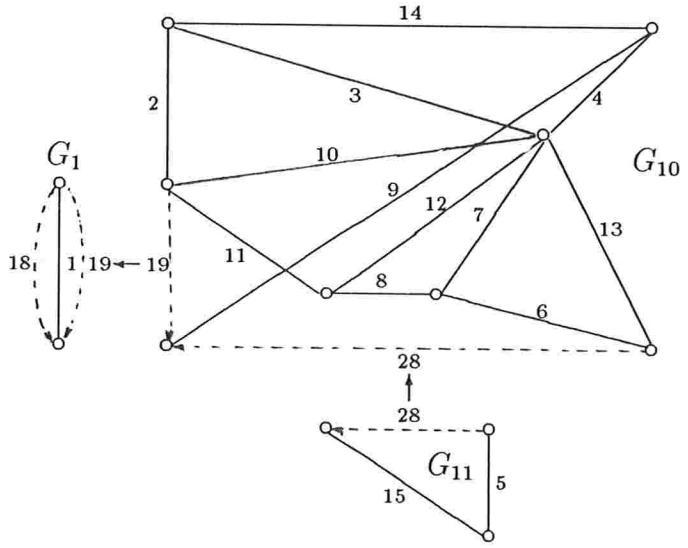

Figura A.8: t-decomposição D (coluna 15).

Coluna 16: Teremos $P = \{4, 5, 6, 7\}$, $\hat{D} = \{G_{10}, G_{11}\}$, $\pi_0 = \{G_{10}\}$ e $\pi_1 = \{G_1 1\}$. Nas

linhas 1–5 do algoritmo *CalculaTipo* será verificado que $Tipo(H) = 3$, onde $H = G_{10}$, pois $P_H + mp(H)$ é um caminho em H com $mp(H) = 28$ em uma das extremidades, e na linha 5.3 $K_1 \leftarrow G_{11}$. Nas linhas 5–13 será constatado que P é um caminho de $\Sigma(\hat{D})$. Na linha 13.4 $K_2 \leftarrow G_{10}$ e, devido à linha 14, u_1, u_2 serão as pontas de P em G_{11} e G_{10}, respectivamente.

Na linha 3 do algoritmo *Atualiza* será criada a aresta $f = C \setminus P = 16$. O caminho R da linha 11 será composto somente por K_1 e K_2, i.e. G_{11} e G_{10}, o grafo G da linha 19 será o grafo $\Sigma(R)$ com u_1 e u_2 ligados pela aresta $f = 16$ e finalmente na linha 20 os grafos G_{11} e G_{10} serão removidos de D^* e o grafo G será acrescentado. A decomposição D resultante encontra-se na figura A.9, onde foi dado o nome G_{10} ao grafo G.

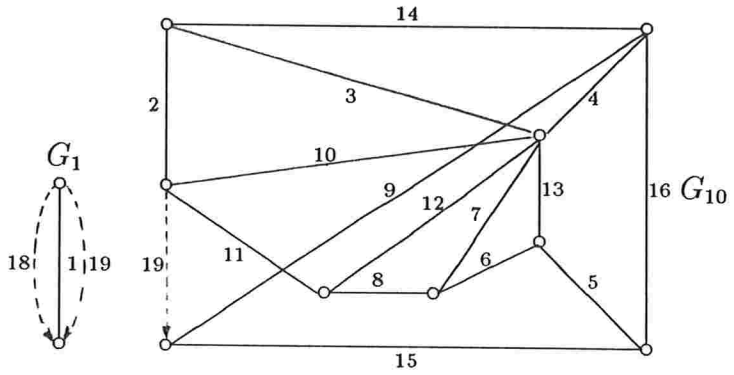

Figura A.9: t-decomposição D (coluna 16).

Coluna 17: Desta feita $P = \{5, 6\}$, $\hat{D} = \{G_{10}\}$. Como $| \hat{D} | = 1$, então a execução do algoritmo *HipoCaminho* é trivial, ao final de sua execução teremos a conclusão de que P é um caminho de $\Sigma(\hat{D}) = G_{10}$, $K_1 = K_2 = G_{10}$ e u_1, u_2 serão as pontas de P em G_{10}.

O algoritmo *Atualiza* apenas ligará as pontas u_1, u_2 de P com a aresta $f = C \setminus P = 17$, que foi criada na linha 3. A t-decomposição resultante está na figura A.10.

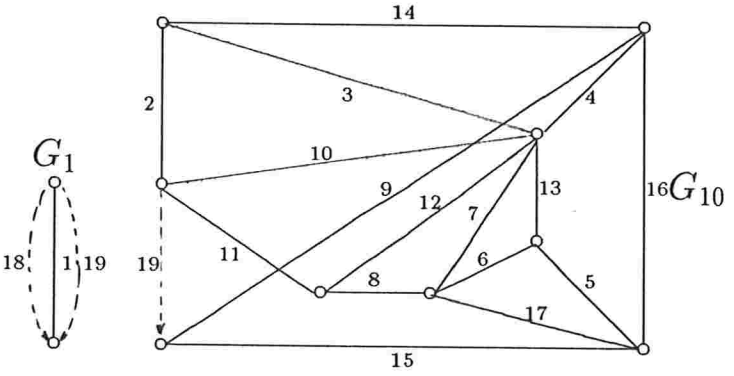

Figura A.10: t-decomposição D (coluna 17).

Assim, ao final do algoritmo *Grafo*, teremos que $\Sigma(G_1, G_{10})$ é uma realização da matriz $\hat{B}$ e portanto, pelo teorema da realização (p. 25), temos que $\hat{B}$ é uma matriz gráfica.

Em Bixby & Wagner [1988] pode ser encontrada a simulação do algoritmo *Grafo* aplicado a outra $\{0,1\}$-matriz.

Seja G o grafo $\Sigma(G_1, G_{10})$ após fixarmos arbitrariamente a orientação de suas arestas. Devido ao lema 2.4 (p. 60) sabemos que, se $\hat{A}$ é uma matriz gráfica então G é uma de suas realizações. Assim, o nosso próximo passo para sabermos se $\hat{A}$ é gráfica será testar se $\hat{A}$ é projetivamente equivalente (p.e.) à matriz do grafo G.

A.3 Execução do algoritmo *TestaPE*

Consideremos o grafo G da figura A.11, o qual foi obtido a partir do grafo $\Sigma(D)$ da figura A.10 após fixarmos uma orientação arbitrária de suas arestas. Seja N a matriz de incidência de G e $\tilde{N}$ a matriz N, após a remoção de uma de suas linhas.

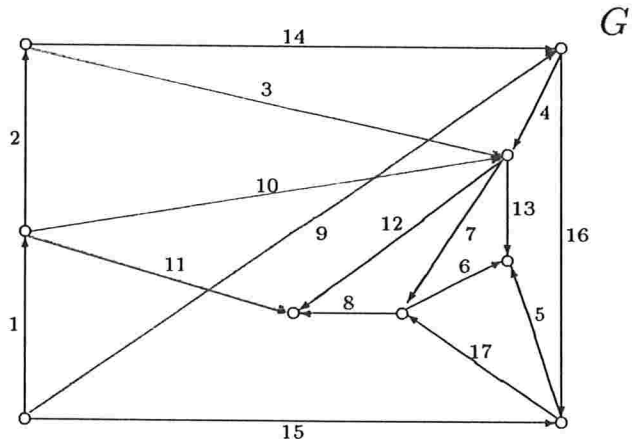

Figura A.11: Grafo obtido a partir do grafo $\Sigma(D)$ da figura A.10, após fixarmos uma orientação arbitrária de suas arestas.

Seja M a *matriz fundamental* de G com relação a $T = \{1,2,3,4,5,6,7,8\}$, contruída como foi descrito em 2.9 (p. 56). Do que foi visto na seção 2.3 sabemos que $\hat{M}$ abaixo, é p.e. a $\tilde{N}$.

$$\hat{M} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \end{matrix} \\ \left[\begin{array}{ccccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

Observemos que as arestas $\{1,...,8\}$ de G formam uma árvore geradora T de G, e que as colunas correspondentes às arestas de T, tanto em $\hat{A}$ quanto em $\hat{M}$, formam a matriz identidade. Para a aplicação do algoritmo $TestaPE$ é importante que $\hat{A}$ e $\hat{M}$ sejam matrizes canônicas em relação a uma mesma árvore geradora de G.

Para vermos como a matriz M foi construída a partir de G consideremos, por exemplo, a aresta 15 de G. O *circuito fundamental* de T determinado pela aresta 15 é $\{1,2,3,5,6,7,15\}$. Assim, a coluna 15 terá entradas não nulas nas colunas 1, 2, 3, 5, 6, e 7. Além disso, como 1, 2, 3, 6, e 7 têm a mesma orientação no circuito, então as entradas correspondentes na coluna 15 terão os mesmos valores, digamos 1, e a entrada correspondente a aresta 5 terá valor -1.

Aplicando o algoritmo $TestaPE$ às matrizes A e M obteremos os seguintes valores para o vetor μ: $\mu_1 = 1$, $\mu_2 = -1/3$, $\mu_3 = -1/2$, $\mu_4 = -1$, $\mu_5 = 1/5$, $\mu_6 = 1/3$, $\mu_7 = -1/2$, $\mu_8 = 1/4$, $\mu_9 = 1/3$, $\mu_{10} = 1/4$, $\mu_{11} = 1/2$, $\mu_{12} = 1$, $\mu_{13} = 1$, $\mu_{14} = 1/2$, $\mu_{15} = 1/3$, $\mu_{16} = 1$, $\mu_{17} = 1$.

É imediato verificar que $\hat{M} = \pi\hat{A}\Delta$, onde π é uma 8×8-matriz diagonal, tal que $\pi_{l,l} = \mu_l$ e Δ é uma 17×17-matriz diagonal, tal que $\Delta_{l,l} = \mu_l^{-1}$, $l \in \{1,...,8\}$ e $\Delta_{c,c} = \mu_c$, $c \in \{9,...,17\}$. Assim, podemos afirmar que a matriz $\hat{A}$ é de fato uma matriz gráfica.

Para encerrarmos este apêndice resta apenas exibirmos o PL sobre redes (do tipo do PL A.2) equivalente ao PL apresentado na seção A.1, é precisamente isso que faremos a seguir.

A.4 O PL sobre redes equivalente

Apesar do grafo G da figura A.11 ser o grafo subjacente do PL equivalente, ainda resta encontrarmos a demanda dos vértices e o custo sobre arestas. Como um primeiro passo para obtermos esses valores devemos observar que os valores do vetor μ correspondentes as colunas das variáveis x_2, x_3, x_4 e x_7 são negativos. Claro que gostaríamos de evitar multiplicar as colunas de $\hat{A}$ por valores negativos, pois isso faria com que as variáveis correspondentes a essas colunas no PL equivalente tivessem a restrição de não serem positivas, o que não é usual. Assim, mudaremos o sinal de μ_2, μ_3, μ_4 e μ_7, isso pode ser feito simplesmente reorientando-se as arestas 2, 3, 4 e 7 do grafo G (é claro que o grafo resultante ainda é uma realização de $\hat{A}$).

Agora nós calcularemos os custos das arestas. Consideremos por exemplo a aresta 10 correspondente à variável x_{10}. Como $\mu_{10} = 1/4$, então a coluna 10 da matriz será multiplicada por 1/4, logo o coeficiente c'_{10} de y_{10} no PL equivalente deverá ser $1/4c_{10} = 7/4$, onde c_{10} é o coeficiente de x_{10} na função objetivo do PL da seção A.1. De maneira análoga obteremos que $c'_1 = 1$, $c_2 = 3.3 = 9$ (a aresta 2 foi reorientada), $c'_3 = 2.1 = 2$ e assim por diante.

Resta apenas calcularmos a demanda nos vértices. É claro que $b' = \pi b$. Entretanto, podemos calcular b' de uma outra maneira bastante simples. Como $\hat{A}$ está na forma canônica, então temos em mãos uma solução do sistema $\hat{A}x = b$ (observemos que esta solução não é necessariamente uma solução viável do PL). Assim, $x_1 = 3$, $x_2 = 9$, $x_3 = -8$, ... é uma solução de $\hat{A}x = b$. Vamos transformar essa solução em um fluxo de G', onde G' é o grafo da figura A.11, após a reorientação das arestas 2, 3, 4 e 7. É fácil fazermos essa transformação

se lembrarmos que $y = \Delta^{-1}x$ é uma solução, não necessariamente viável, do PL equivalente. Agora, as demandas dos vértices de G' podem ser facilmente obtidas computando-se a diferença entre o fluxo que entra e o fluxo que sai de cada vértice.

O problema em redes equivalente é mostrado na figura A.12, onde as demandas estão próximas aos respectivos vértices e os custos estão próximos às respectivas aresta .

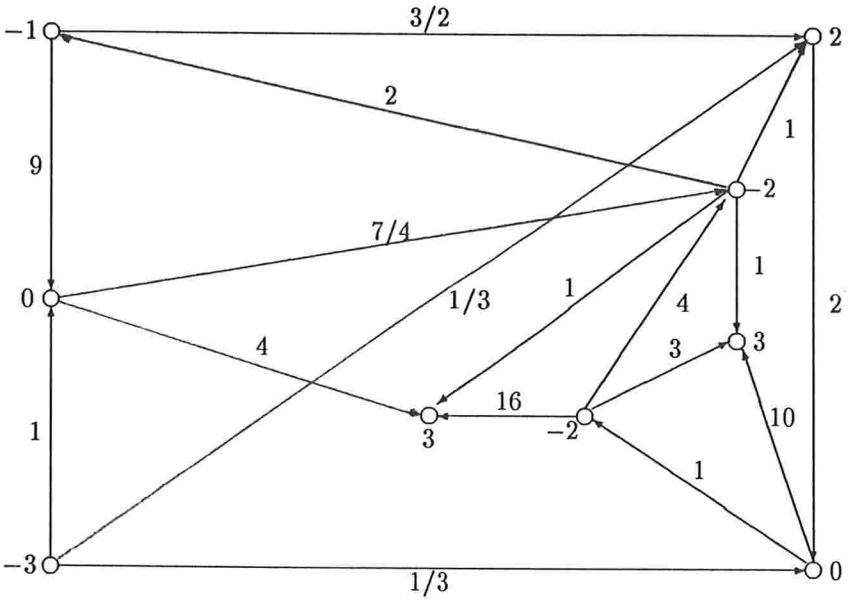

Figura A.12: O problema sobre redes equivalente ao PL da seção A.1.

É fácil verificar que a solução ótima para o PL sobre redes equivalente é $y_1 = 2$, $y_6 = 2$, $y_9 = 1$, $y_{10} = 2$, $y_{12} = 3$, $y_{13} = 1$, $y_{14} = 1$, $y_{15} = 0$ e $y_i = 0$ para os demais valores de i. A solução correspondente do PL da seção A.1 é $x_1 = 2$, $x_6 = 6$, $x_9 = 1/3$, $x_{10} = 1/2$, $x_{12} = 3$, $x_{13} = 1$ $x_{14} = 1/2$, $x_{15} = 0$ e $x_i = 0$ para os demais valores de i. A optimalidade da solução acima pode ser verificada empregando-se o teorema fraco de dualidade e os multiplicadores $\gamma_1 = 1$, $\gamma_2 = 13/18$, $\gamma_3 = -47/24$, $\gamma_4 = -29/12$, $\gamma_5 = 41/60$, $\gamma_6 = \gamma_7 = 1$ e $\gamma_8 = 3/4$.

Referências

Aho, A. V., Hopcroft, J. E. & Ullman, J. D. (1974), *The Design and Analysis of Computer Algorithms*, Addison-Weley, Reading, Mass., 1974.

Bartholdi, J.J. (1982), A good submatrix is hard to find, *Operations Research Letters* **1** (1982) 190–193.

Bixby, R. E. (1984), Recent algorithms for two versions of graph realization and remarks on applications to linear programming, *Progress in Combinatorial Optimization* (W. R. Pulleyblank ed.), Academic Press. Canada, 39–67.

Bixby, R. E. & Cunningham, W. H. (1980), Converting Linear programs to network problems, *Mathematics of Operations Research* **5** (1980) 321–357.

Bixby, R. E. & Wagner, D. K. (1988), An almost linear-time algorithm for graph realization, *Mathematics of Operations Research* **13** (1988) 99–123.

Bollobás, B. (1979), *Graph Theory: an introductory course*, Spring-Verlag, New York, 1979.

Bond, J. A. & Murty, U. S. R. (1976), *Graph Theory with Applications*, Macmillan, London, 1976.

Chvátal, V. (1983), *Linear Programming*, W. H. Freeman and Company, New York, 1983.

Cunningham, W. H. (1973), *A Combinatorial Decomposition Theory*, PhD Thesis, Waterloo, 1973.

Cunningham, W. H. & Edmonds, J. (1980), A combinatorial decomposition theory, *Canadian Journal of Mathematics* **32** (1980) 734–765.

Dantzig, G. B. (1951), Aplication of the simplex method to a transportation problem, *Activity Analysis of Production and Allocation* (Tj. C. Koopmans, ed.), Wiley, New York, 1951, 359–373.

Dantzig, G. B. & Wolfe, P. (1960), Decomposition principle for linear programs, *Operation Research* **8** (1960) 101–111.

Dantzig, G. B., & Van Slyke, R. M., (1967), Generalized upper bounding techniques, *Journal of Computer and System Sciences* **1** (1967) 213–226.

Edmonds, J. & Karp, R. M. (1972), Theoretical improvements in algorithmic efficiency for network flow problems, *Journal of the Association for Computing Machinery* **19** (1972) 248–264.

Ford, Jr, L. R. & Fulkerson, D. R. (1962), *Flows in Networks,* Princeton University Press, Princeton, New Jersey, 1962.

Fujishige, S. (1980), An efficient PQ-graph algorithm for solving the graph-realization problem, *Journal of Computer and System Sciences* **21** (1980) 63–86.

Gallo, G. & Sandi, C. (eds.) (1986), Netflow at Pisa, *Mathematical Programming Study* **26**, 1986.

Garey, M. R. & Johnson, D. S. (1979), *Computer and intractability*, W. H. Freeman and Company, San Francisco, 1979.

Gibbons, A. (1985), *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985.

Glover, F., Karney, D. & Klingman, D. (1974), Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problems, *Networks* **4** (1974) 191–212.

Glover, F. & Klingman, D. (1981), The simplex SON algorithm for LP/embedded network problems, *Mathematical Programming Study* **15** (1981) 148–176.

Golumbic, M. C. (1980), *Algorithmic Graph Theory and Perfect Graphs* , Academic Press, London, 1980.

Horowitz, E., Sahni, S. (1984), *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, Maryland, 1984.

Humes, A. F. P. de C., Homem de Melo, I. S., Yoshida, L. K. & Martins, W. T. (1983), *Noções Básicas de Cálculo Numérico*, IME-USP, São Paulo, 1983.

Iri, M. (1966), A criterion for the reducibility of a linear programming problems to a linear network-flow problem, *RAAG Research Notes,* Third Series, Nº98.

Klingman, D. & Mulvey, J. M. (eds.) (1981), Network Models and Associated Aplications, *Mathematical Programming Study* **15**, 1981.

Lewis, J. M. & Yannakakis, M. (1980), The node-deletion problem for hereditary properties is $\mathcal{NP}$-complete, *Journal of Computer and System Sciences* **20** (1980) 219–230.

Lawler, E. L. (1976), *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

Orden, A. (1956), The transhipment problem, *Management Science* **2** (1956) 276–285.

Recski, A. (1989), *Matroid Theory and its Applications*, Springer-Verlag, Budapest, 1989.

Schrijver, A. (1986), *Theory of Linear and Integer Programming*, John Wiley & Sons, Essex, 1986.

Seymour, P. D. (1979), Matroid representation over GF(3), *J. Comb. Theory Ser. B,* **26** (1979) 159–173.

Seymour, P. D. (1980a), On Tutte's characterization of graphic matroids, *Annals of Discrete Mathematics* **8** (1980) 83–90.

Seymour, P. D. (1980b), Decomposition of regular matroids, *J. Comb. Theory Ser. B,* **28** (1980) 305–359.

Seymour, P. D. (1981), Recognizing graphic matroids, *Combinatorica* **1** (1981) 75–78.

Smith, L. (1984), *Linear Algebra*, Springer-Verlag, New York, 1984.

Tutte, W.T. (1959), Matroids and graphs, *Trans. Amer. Math. Soc.* **90** (1959) 527–552.

Tutte, W.T. (1960), An algorithm for determining whether a given binary matroid is graphic, *Proceedings of the American Mathematical Society* **11** (1960) 905–917.

Tutte, W. T. (1965), Lectures on matroids, *J. Res. Bur. Stand.* **69B** (1965) 1–48.

Truemper, K. (1985a), *A flexible decomposition approach for finding submatrices with certain inherited properties,* University of Texas at Dallas report, Dallas, 1985.

Truemper, K. (1985b), A decomposition theory for matroids, II. Minimal violation matroids, *Journal of Combinatorial Theory* (B) **39** (1985) 282–297.

Wagner, D. K. (1985), On theorems of Whitney and Tutte, *Discrete Mathematics* **57** (1985) 147–154.

Welsh, D. J. A. (1976), *Matroid Theory*, Academic Press, London, 1976.

Whitney, H. (1935), On the abstract properties of linear dependence, *American Journal of Mathematics,* **57** (1935) 509–533.

Yannakakis, M. (1981), Node-deletion problems on bipartite graphs, *Siam J. Comput.* **10** Nº2 (1981) 310–327.

Índice

R

S

T

V

Y