REDUÇÃO DE PERFIL E DE LARGURA DE BANDA DE MATRIZES ESPARSAS

Nelson Soares de Rezende

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovado por:

Jayme Luiz Szwarcfiter
(Presidente)

Nelson Maculan Filho

Paulo Alcantara Gomes

Marco Antonio Raupp

Rio de Janeiro, RJ - BRASIL
Janeiro de 1979

REZENDE, NELSON SOARES DE

Redução de Perfil e de Largura de Banda de Matrizes Esparsas [Rio de Janeiro] 1979.

VII, 115 p. 29,7 cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas, 1979).

Tese - Univ. Fed. Rio de Janeiro. Depto. Engenharia de Sistemas.

1. Matrizes Esparsas I. COPPE/UFRJ
II. Título(série)

A meus pais,
minha esposa,
minhas filhas.

AGRADECIMENTOS

Aos professores da COPPE/UFRJ pelos conhecimentos transmitidos, em especial ao Prof. Jayme Luiz Szwarcfiter por sua orientação, críticas e estímulo constante durante a realização desse trabalho.

Ao Prof. Marco Antonio Raupp pela colaboração recebida.

Aos meus colegas do Laboratório de Cálculo do Centro Brasileiro de Pesquisas Físicas, pelo incentivo.

A Maria do Carmo Vivas Gonçalves pela confecção gráfica.

Aos meus irmãos: Galeno, Pedro, Eliane, Wilmar e Wiltamar, pela eterna amizade que nos une.

A minha esposa por sua abnegação, sacrifício pessoal e apoio constante.

RESUMO

Nesse trabalho são discutidos esquemas de ordenação das linhas e colunas de matrizes esparsas que produzem larguras de banda ou perfís reduzidos. Esses esquemas permitem usar técnicas de armazenamento que reduzem o espaço de memória requerido para tais matrizes.

O problema de efetivamente minimizar a largura de banda ou o perfil foram mostrados serem NP-Completos por Papadimitriou [17], portanto, os esquemas discutidos nessa tese são estratégias heurísticas para reduzir (não para minimizar) a largura de banda e/ou o perfil. São descritos os importantes algoritmos de Cuthill-McKee (CM e RCM) [7] e de Gibbs at al. (GPS) [8]. Suas análises de complexidade e implementações de tipo Algol ótimas são também apresentadas.

Esses esquemas de ordenação são importantes na resolução de sistemas de equações lineares (Ax = b) decorrentes do uso do método de elementos finitos que produzem matrizes esparsas simétricas e positivas definidas, cujo armazenamento na forma de banda (ou de envelope) permite reduzir o tempo de execução requerido no processo de resolução do sistema de equações, pela diminuição do seu número de operações.

Um esquema baseado nos algoritmos RCM e GPS, inicialmente proposto por Alan George [9] é também apresentado com uma nova modificação que visa reduzir seu tempo de execução.

SUMMARY

In this work schemes of ordering the lows and the columns of sparse matrices which yields small bandwidth and/or profile are discussed. These schemes allow the use of storage techniques which lead to reduce the storage required for those matrices.

The problem of effectively minimize the bandwidth or the profile were showed to be NP-Complete by Papadimitriou [17], therefore the schemes discussed in this thesis are heuristics strategies to reduce (not to minimize) the bandwidth and/or the profile. The important algorithms of Cuthill-McKee (CM and RCM) [7] and of Gibbs at al. (GPS)[8] are described Their complexity analysis and optimal like-Algol implementations are presented.

These ordering schemes are important in the solution of systems of linear equations (Ax = b) arising from the use of the finite element method which yields sparse symmetric and positive definite matrices whose band-oriented (or envelope-oriented) storage form allows to reduce the execution time in the linear solver, by decreasing its number of operations.

A scheme based on the algorithms RCM and GPS initially proposed by Alan George [9] is also presented with a new modification that usually reduces its execution time.

INDICE

I. INTRODUÇÃO E CONCEITOS BÁSICOS

I.1. Introdução

O método de elementos finitos é um processo numerico muito potente e efetivo para se aproximar soluções de equações diferenciais parciais [1]. Essa técnica era já amplamente utilizada por engenheiros em cálculos de estruturas muito antes de ter sido reconhecida e explicada rigorosamente pelos matemáticos.

O ponto de partida para a aplicação dessa técnica é a subdivisão da região de interesse (domínio do problema) em pequenas partes denominadas elementos. No caso de um contínuo bidimensional, o domínio R é geralmente dividido em um conjunto finito $\hat{\tau}$ de elementos quadriláteros ou triângulares, tais que (veja figura I-1):

$$(I\text{-}1)\quad T_1, T_2 \in \tau \text{ e } T_1 \neq T_2 \iff \begin{cases} T_1 \cap T_2 = \phi, \\ T_1 \text{ e } T_2 \text{ tem um lado comum,} \\ T_1 \text{ e } T_2 \text{ tem um vertice comum.} \end{cases}$$

Essa subdivisão (τ) da região R produz o que é denominado malha de elementos finitos.

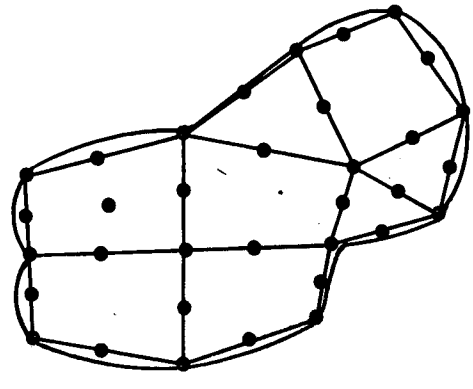


Figura I-1: Uma malha de elementos finitos, com 8 elementos e 34 nós.

Em todo vertice de cada elemento T é, então, definido um nó. Outros nós podem ainda ser definidos nos lados dos elementos e/ou em seus interiores. Na figura (I-1) os pontos representam nós.

A cada nó P pertencente a elementos de τ é então, associada uma função de forma $\phi_P : \Omega \rightarrow R$, contínua, e de suporte limitado nos elementos que contem referido nó, i.e., ϕ_P é nula em todos os pontos de R não pertencentes aos elementos que contem P. Além disso, $\phi_P(P) = 1$, e ϕ_P é nula em todos os outros nós.

A solução aproximada $\sum_{i=1}^{N} x_i\phi_i$ da equação diferencial é então construida na forma $V(x,y) = \sum_{i=1}^{N} x_i\phi_i$ onde as ϕ_i's constituem uma base de funções definida a partir das funções de forma ϕ, e os x_i's são valores nodais a serem determinados.

O passo final na aplicação do metodo de elementos finitos consiste na determinação dos valores nodais. Isso é feito de acordo com princípios variacionais ou de ortogonalidade, tais como, o metodo de Ritz ou de Galerkin, produzindo um sistema de equações algebricas lineares de ordem n

$$Ax = b \tag{I-2}$$

onde n é a dimensão da base de funções.

Quanto maior for o refinamento da malha de elementos finitos (número de elementos na malha), maior será, evidentemente, o número de equações. Frequentemente, tais sistemas são da ordem de milhares de equações.

A matriz $A = (a_{ij})$ é esparsa uma vez que os suportes de ϕ_P e ϕ_Q tem intersecção vazia sempre que P e Q não pertencerem a um mesmo elemento. Essa matriz é, ainda, simetrica e positiva definida, isto é,

$$\forall i,j \ (a_{ij} = a_{ji}) \ , \text{ e} \tag{I-3}$$

$$\forall x \in \mathbb{R}^n \ (x \neq 0 \Rightarrow x^t Ax > 0) \tag{I-4}$$

Tais matrizes podem ser fatoradas na forma

$$LL^t \tag{I-5}$$

onde $L = (l_{ij})$ é uma matriz triangular inferior e inversível,

e L^t sua transposta. Geralmente a matriz L é obtida através do metodo de fatoração de Cholesky [1].

Uma vez determinada a matriz L, resolve-se sucessivamente os sistemas triangulares

$$\begin{aligned} Ly &= b \ , \quad e \\ L^t x &= y \end{aligned} \qquad \text{(I-6)}$$

obtendo-se assim, a solução do sistema algebrico (I-2).

O metodo de Cholesky [1] requer ordem de n^3 multiplicações, para fatorar uma matriz simetrica positiva definida cheia, porém, tal complexidade pode ser sensivelmente melhorada se for levada em conta a esparsidade da matriz A.

Tal esparsidade pode ser considerada de duas formas distintas:

(i) através de metodos gerais que exploram todos os zeros da matriz,

(ii) através de metodos que exploram os zeros fora de um conjunto particular de elementos da matriz, como metodos de banda ou de envelope (que exclui todos os zeros a esquerda do primeiro elemento não nulo, em cada linha da matriz).

O objetivo dessa tese é discutir estrategias de ordenação (permutação) das variáveis e equações do sistema algebrico (I-2) que transformem a matriz A para a forma de matriz de banda (com todos os elementos não nulos próximos a diagonal) permitindo a exploração da sua esparsidade através de métodos do tipo (ii).

O processo de Cholesky para calcular a matriz L (I-6) é estável ([1],[23]), o que significa que na discussão das estrategias de ordenação de A não é necessário tecer considerações sobre sua estabilidade numerica.

Matrizes esparsas representadas na forma de banda (ou de envelope) possibilitam - pela eliminação dos zeros fora da banda (ou do envelope) da matriz -, além da redução do número de multiplicações realizadas no processo de Cholesky, uma grande redução do espaço de memória necessário para seu armazenamento, o que é bastante crítico em grandes sistemas de

equações.

Os problemas de efetivamente, minimizar a largura de banda [17] e o perfil (número de elementos no envelope) [20] de matrizes estão discutidos no capítulo II, sendo ambos NP-Completos (i.e., pertencem a uma classe de problemas para os quais, grosseiramente falando, não se conhecem algoritmos que os resolvam executando um número polinomial de operações, embora não se tenha demonstrado que tais algoritmos não existam). Em vista disso, os melhores algoritmos atualmente existentes são algoritmos heurísticos que geram ordenações das linhas e colunas de matrizes esparsas que implicam numa reduzida largura de banda e/ou perfil, porém, não necessariamente mínimos.

Embora os problemas de redução de largura de banda e de perfil sejam independentes, em geral, os algoritmos de redução de largura de banda provocam também redução do perfil.

Nos capítulos IV e V são analisados os metodos de ordenação existentes, mais importantes: Reverso de Cuthill-McKee (RCM) ([4],[6]) e de Gibbs at al. [8] que denominamos GPS.

No capítulo III são apresentados algoritmos auxiliares utilizados pelos esquemas RCM e GPS.

No capítulo VI é feita uma comparação entre os dois esquemas, sendo, também, apresentado um metodo de ordenação que utiliza características de ambos.

A análise da complexidade de um algoritmo [18] fornece informações sobre os recursos requeridos pelo mesmo, em função de características dos dados de entrada. Em geral, tais informações se referem ao tempo de execução e ao espaço de memória requeridos. A especificação dos dados de entrada é necessária para permitir a realização da análise, e uma possibilidade é assumir que os dados são os piores possíveis para determinado tamanho de problema, o que se denomina análise de pior caso. Uma medida de pior caso do tempo de execução ou do espaço de memória, como uma função do tamanho do problema, fornece uma performance garantida do algoritmo, pois ele nunca

requererá mais tempo ou espaço que o requerido para esse caso limite.

Para alguns problemas, essa medida de pior caso pode ser muito pessimista e, então, pode ser feita uma análise de caso medio ou de casos mais representativos, o que poderia fornecer medidas de complexidade mais realísticas. Essas análises de caso medio são, em geral, muito complexas.

Nesse trabalho são apresentadas, apenas, estimativas de pior caso, que, para uma grande variedade de problemas [18], é bastante útil e realística.

Serão ignorados os fatores constantes, tanto em tempo de execução quanto em espaço de memória, pois são muito difíceis de serem calculados e tendem a ser pouco importantes, pelo menos para problemas muito grandes.

Nas medidas de complexidade é usada a seguinte notação: se f e g são funções de n, então "f(n) é de $0(g(n))$" significa que $f(n) \leq c(g(n))$, para todo n, onde c é uma constante positiva adequada.

O problema de ordenação das linhas e colunas de uma matriz A pode ser reduzido ao problema de se determinar uma matriz de permutação P, n x n, de modo a transformar o sistema algebrico (I-2), no sistema

$$(PAP^t)\ Px = Pb \qquad \text{(I-7)}$$

onde a matriz PAP^t resultante é uma matriz de banda.

Uma matriz de permutação $P = (p_{ij})$ de ordem n é uma matriz binária, tal que:

(i) $\forall i \leq n\ \exists j \leq n : p_{ij} = 1$, e

(ii) $\forall i,j \leq n\ \ p_{ij} = 1 \iff \begin{cases} p_{ik} = 0, \forall k \neq j \\ p_{kj} = 0, \forall k \neq i \end{cases}$

Os lemas I.1. e I.2. seguintes mostram a relação entre a matriz PAP^t e a matriz A original.

Lema I.1.: Dada uma matriz $A = (a_{ij})$, n x n e uma matriz de permutação $P = (p_{ij})$, n x n, então a matriz $PAP^t = (a'_{ij})$ é tal que:

(i) $p_{ij} = 1 \Rightarrow a'_{ii} = a_{jj}$, e

(ii) $p_{ir} = 1$ e $p_{js} = 1 \Rightarrow \begin{cases} a'_{ij} = a_{rs}, \text{ e} \\ a'_{ji} = a_{sr} \end{cases}$

Demonstração: Basta calcular o produto de matrizes PAP^t.

Lema I.2.: Se $A = (a_{ij})$, n x n, é uma matriz esparsa, simetrica e positiva definida, e $P = (p_{ij})$ é uma matriz de permutação de ordem n, então PAP^t é esparsa, simetrica e positiva definida.

Demonstração: Imediata a partir do lema I.1. e das expressões (I-3) e (I-4).

O lema I.2. garante que o sistema (I-7) também pode ser resolvido através do metodo de Cholesky.

I.2. Matrizes

I.2.1. Definições

Seja $A = (a_{ij})$ uma matriz, n x n, simetrica e positiva definida. Para cada linha i de A, define-se

(I-8) $\quad \ell_i(A) = \min \{j : a_{ij} \neq 0\}$

(I-9) $\quad \beta_i(A) = i - \ell_i(A)$

Definição I.1.: A largura de banda da matriz A ($B(A)$) é o número:

(I-10) $\quad B(A) = \max \{|i - j| : a_{ij} \neq 0\} = \max \{\beta_i(A)\}$

Definição I.2.: O envelope* da matriz A (Env(A)), é o conjunto:

(I-11) $\quad Env(A) = \{(i,j) : \ell_i(A) \leq j < i\}$

---

* Há autores, como W. H. Liu [7] que incluem, também os elementos da diagonal no envelope da matriz.

Definição I.3.: O perfil da matriz A ($P(A)$) é o número:

$$(I\text{-}12) \qquad P(A) = \sum_{i=1}^{n} \beta_i(A) = |Env(A)|$$

onde $|\ |$ indica a cardinalidade do conjunto.

Define-se, ainda, preenchimento (Enc(A)) da matriz A como sendo o número de elementos nulos de A que tornam-se diferentes de zero após a aplicação do metodo de Cholesky, isto é,

$$(I\text{-}13) \qquad Enc(A) = \{(i,j) : a_{ij} = 0 \text{ e } l_{ij} \neq 0\}$$

onde $L = (l_{ij})$.

E é fácil mostrar que $Enc(A) \subseteq Env(A)$ [13].

Definição I.4.: O Envelope Transposto de A (Tenv(A)) é o conjunto:

$$(I\text{-}14) \qquad Tenv(A) = \{(i,j) : j \leq i \text{ e } \exists k \geq i \text{ tal que } a_{kj} \neq 0\}$$

De (I-12) e (I-14), segue-se que

$$(I\text{-}15) \qquad Tenv(A) = Env(A^{\#})$$

onde $A^{\#}$ é a transposta de A, em relação a sua diagonal menor.

A figura (I-2) exemplifica alguns desses conceitos.

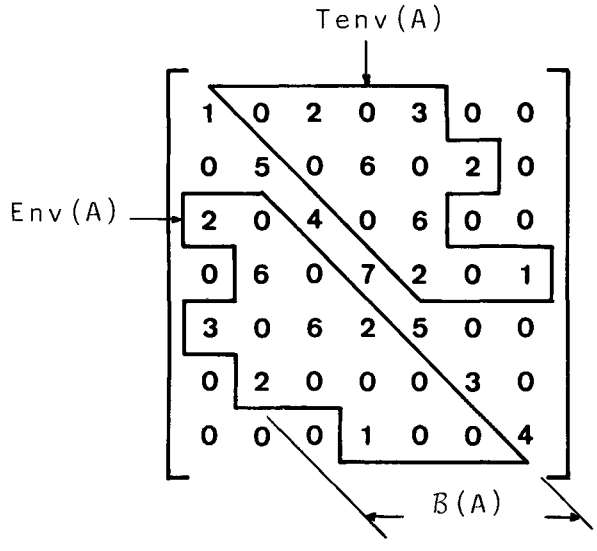


Figura I-2: Exemplo de uma matriz A de banda.
$P(A) = |Env(A)| = 15$ e $|Tenv(A)| = 13$.

I.2.2. Representação Computacional de Matrizes

Um metodo comum para o armazenamento de uma matriz de banda A simetrica, com largura de banda $\beta(A)$ é o denominado esquema de armazenamento diagonal. Consiste em armazenar os elementos do triângulo inferior de A, incluindo a diagonal, numa matriz retangular n x ($\beta(A)$ + 1) de maneira que os elementos da diagonal ocupem a ($\beta(A)$ + 1)-ésima coluna. Veja a figura (I-3).

$$\begin{bmatrix} a_{11} & a_{12} & 0 & a_{14} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{33} & 0 & a_{35} & a_{36} & 0 \\ a_{41} & 0 & 0 & a_{44} & a_{45} & 0 & 0 \\ 0 & 0 & a_{53} & a_{54} & a_{55} & 0 & a_{57} \\ 0 & 0 & a_{63} & 0 & 0 & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & a_{75} & a_{76} & a_{77} \end{bmatrix}$$

(a)

$$\begin{bmatrix} & & & a_{11} \\ & & a_{21} & a_{22} \\ & 0 & 0 & a_{33} \\ a_{41} & 0 & 0 & a_{44} \\ 0 & a_{53} & 0 & a_{55} \\ a_{63} & 0 & 0 & a_{66} \\ 0 & a_{75} & a_{76} & a_{77} \end{bmatrix}$$

(b)

a_{11}	a_{21}	a_{22}	a_{33}	a_{41}	0	0	a_{44}	a_{53}	a_{54}	a_{55}	a_{63}	0	0	a_{66}	a_{75}	a_{76}	a_{77}

1	3	4	8	11	15	18

(c)

a_{11}	a_{22}	a_{12}	a_{33}	a_{44}	0	0	a_{14}	a_{55}	a_{45}	a_{35}	a_{66}	0	0	a_{36}	a_{77}	a_{67}	a_{57}

1	2	4	5	9	12	16	19

(d)

Figura I-3: Representação de uma matriz (a) Matriz A = (a_{ij}) simetrica, (b) esquema de armazenamento diagonal, (c) esquema de armazenamento de Jennings e (d) esquema de armazenamento por coluna, derivado do esquema de Jennings.

Entretanto, em muitas situações, pode haver grande variação de $\beta_i(A)$, e, nesse caso, o esquema de armazenamento diagonal consideraria, ainda, muitos zeros. Tal variação de $\beta_i(A)$ pode ser explorada através de um esquema proposto por Jennings [15]: para cada linha i de A, os elementos correspondentes as colunas, de $\ell_i(A)$ até a diagonal, são armazenados em posições contíguas num vetor. Um outro vetor de n apontadores é requerido para indicar as posições dos elementos da diagonal. Nesse caso o espaço requerido é de $n + \sum_{i=1}^{n} \beta_i(A)$. Um exemplo é dado na figura (I-3c).

Um outro esquema de armazenamento, derivado do de Jennings, mas que armazena em posições contíguas os elementos de cada coluna é também mostrado na figura (I-3d) Nesse caso, o vetor de apontadores tem n + 1 elementos para apontar, também, para o elemento que se seguiria ao último, para indicar o final do vetor, e o espaço requerido é $n + \sum_{i=1}^{n} \beta_i(A) + 1$.

I.3. Grafos

I.3.1. Definições

Definição I.5.: Um grafo não dirigido G = (V,E) consiste de um conjunto finito e não vazio de vertices e de um conjunto E de arestas que são pares não ordenados de vertices distintos de V. O número de vertices de V é representado por n e o número de arestas por m.

Um subgrafo $\bar{G} = (\bar{V},\bar{E})$ de G é um grafo tal que $\bar{V} \subseteq V$ e $\bar{E} = \{\{x,y\} \in E : x \in \bar{V} \text{ e } y \in \bar{V}\}$.

Os vertices x e y são adjacentes se $\{x,y\} \in E$. O conjunto de adjacencias de $X \subset V$, denotado por adj(X), é o conjunto

(I-16) $adj(X) = \{v \in V : v \notin X \text{ e } \exists x \in X \text{ tal que } \{v,x\} \in E\}$.

Se X = {u} escreve-se adj(u), em vez de adj({u}).
o grau de um vertice x é a cardinalidade do conjunto adj(x), i.e., grau (x) = |adj(x)|.

Um grafo G é completo se todo par de vertices distintos define uma aresta de G. Para grafos completos tem-se, portanto, grau(x) = n - 1, para todo x.

Um caminho C(x,y) entre dois vertices distintos x e y de V é um conjunto ordenado de vertices distintos $\{v_0,v_1,\ldots,v_k\}$, onde $v_0 = x$, $v_k = y$, e $\{v_i,v_{i+1}\} \in E$, $0 \leq i < k$. k é o comprimento do caminho. Um ciclo é um caminho que começa e termina no mesmo vertice. Um grafo G é conexo se existir um caminho ligando cada par distinto de seus vertices, caso contrario G é desconexo e consiste de dois ou mais componentes conexos, que são subgrafos conexos maximais.

A distancia d(x,y) entre dois vertices distintos x e y em um grafo conexo G é o comprimento do menor caminho entre eles. A excentricidade de um vertice x é a quantidade

(I-17) $$e(x) = \max \{d(x,y) : y \in V\}.$$

O diâmetro d(G) de um grafo conexo G é a maior distancia entre quaisquer dois de seus vertices, i.e.,

(I-18) $$d(G) = \max \{e(x) : x \in V\}.$$

Um vertice x é um vertice periférico de G se e(x) = d(G). Um vertice x é um vertice pseudo-periférico, se

(I-19) $$\forall y \in V \ (d(x,y) = e(x) \Rightarrow e(y) = e(x)).$$

Se x é um vertice pseudo-periférico, e(x) é dito pseudo-diâmetro de G.

Definição I.6.: Uma estrutura de níveis N de um grafo conexo G = (V,E) é uma partição

(I-20) $$N(G) = \{N_1,N_2,\ldots,N_k\}$$

do conjunto de vertices, tal que

(i) $adj(N_1) \subseteq N_2$, $adj(N_k) \subseteq N_{k-1}$

(ii) $adj(N_i) \subseteq N_{i-1} \cup N_{i+1}$, $1 < i < k$

k é a profundidade de $N(G)$. A largura do nível i é o número $w(N_i) = |N_i|$ e a largura da estrutura de níveis N é o número $w(N) = \max \{w(N_i)\}$.

Uma estrutura de níveis com raiz em v é uma estrutura de níveis

$$N_v(G) = \{N_1(v), N_2(v), \ldots, N_{k(v)}(v)\} \qquad \text{(I-21)}$$

onde $N_1(v) = \{v\}$ e $N_i(v) = adj\left(\bigcup_{j=1}^{i-1} N_j(v)\right)$.

Uma árvore $T = (X,E)$ é um grafo conexo sem ciclos. Desde que T não possue ciclos, é fácil mostrar que $|E| = |X| - 1$ e que existe um único caminho entre cada par de vertices de X. Uma árvore com raiz em u é uma árvore $T_u = (X,E)$ em que o vertice u é distinguido, sendo denominado raiz de T. Como existe um único caminho entre cada par de vertices de T, existe um caminho entre u e cada vertice $x \neq u$ em X. Se esse caminho passa por y, então y é dito um ancestral de x, e x é um descendente de y. Ainda, se $y \in adj(x)$ y é o pai de x, e x é um filho de y.

Definição I.7.: Dada uma matriz $A = (a_{ij})$ simetrica, de ordem n, define-se um grafo $G(A) = (V,E)$ associado a matriz A, tal que

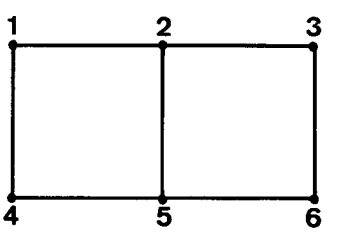
(i) $V = \{x_1, x_2, \ldots, x_n\}$

(ii) $\{x_i, x_j\} \in E \iff a_{ij} \neq 0$

Um exemplo de um grafo G(A) associado a uma matriz A é mostrado na figura (I-4).

$$\begin{bmatrix} 1 & 2 & & 8 & & \\ 2 & 8 & 2 & & 7 & \\ & 2 & 3 & & & 4 \\ 8 & & & 9 & 8 & \\ & 7 & & 8 & 2 & 3 \\ & & 4 & & 3 & 1 \end{bmatrix}$$

(a)



(b)

Figura I-4: (a) Matriz $A = (a_{ij})$, e (b) Grafo G(A) associado.

Definição I.8.: Seja G = (V,E) um grafo de n vertices. Uma numeração (ordenação, rotulação) dos vertices de V é uma função biunívoca $\phi : V \rightarrow \{1,2,\ldots,n\}$.

Algumas das definições relativas a matrizes são, então, formuladas em termos de grafos G(A) = (V,E).

Dado um grafo G = (V,E) e uma numeração $\phi(G)$, define-se:

(I-22) $$\ell_{\phi(x_i)}(G) = \min \{\phi(x_j) : x_j \in adj(x_i)\}$$

(I-23) $$\beta_{\phi(x_i)}(G) = \phi(x_i) - \ell_{\phi(x_i)}(G)$$

Definição I.9.: A largura de banda de um grafo G relativa a numeração $\phi(G)$ é o número

(I-24) $$B_{\phi}(G) = \max \{\beta_{\phi(x)}(G) : x \in V\}$$

A menor $B_{\phi}(G)$ sobre todas as numerações ϕ de G é dita largura de banda de G.

Definição I.10.: O perfil de um grafo G, relativo a numeração $\phi(G)$ é o número

(I-25) $$P_{\phi}(G) = \sum_{i=1}^{n} \beta_{\phi(x_i)}(G)$$

Os algoritmos apresentados nos capítulos seguintes constroem numerações $\phi(G)$, a partir de estruturas de níveis de G, e nesse caso, são usadas as notações: $B_{\phi}(N_v(G))$, $P_{\phi}(N_v(G))$, etc.

O lema seguinte mostra uma propriedade de estruturas de níveis de grafos conexos e sem ciclos que é utilizada na demonstração de outros resultados, no capítulo IV.

Lema I.3.: Dados um grafo conexo e sem ciclos G = (V,E), e uma estrutura de níveis $N_v(G) = \{N_1(v), N_2(v), \ldots, N_{k(v)}(v)\}$ com raiz em v. Sejam $u \in N_i(v)$, $w \in N_j(v)$, e $x \in N_p(v)$, tais que $x \in C(u,w)$, e

(I-26) $$\forall y \in C(u,w) \; (y \neq x \Rightarrow \exists q > p \text{ e } y \in N_q).$$

Então:

(i) $\forall y \in C(u,w)(d(u,w) = d(u,y) + d(y,w))$, e

(ii) $d(u,x) = p - i$ e $d(x,w) = p - j$.

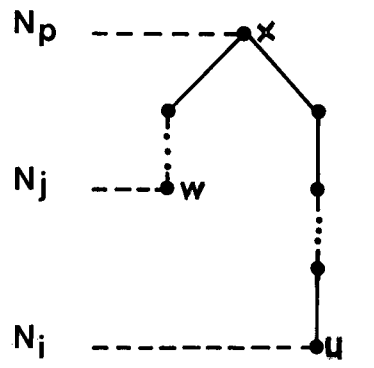


Figura I-5: Caminho $C(u,w)$ em $N_V(G)$

Demonstração:

(i) Como $y \in C(u,w)$, pode-se escrever

(I-27) $$C(u,w) = C(u,y) \cup C(y,w)$$

mas como G é sem ciclos existe um único caminho entre cada par de vertices e, de (I-9), tem-se

(I-28) $$d(u,w) = d(u,y) + d(y,w)$$

(ii) Seja $C(u,x) = (u,s_1,s_2,\ldots,s_r,x)$
Por definição de estrutura de níveis, tem-se

$x \in N_p$, $s_r \in N_{p+1}, s_{r-1} \in N_{p+2}, \ldots, s_1 \in N_{p+r}$, e $u \in N_{p+r+1}$

onde $p + r + 1 = i$.
Logo, $d(u,x) = p - i$.
Analogamente, demonstra-se que $d(x,u) = p - j$.

I.3.2. Representação Computacional de Grafos e de Estruturas de Níveis

Dado um grafo $G = (V,E)$, uma lista de adjacencias de x em V é uma lista de vertices contendo todos os vertices em adj(x). Uma estrutura de adjacencias de G é o conjunto.

das listas de adjacencias de todos os vertices x de V. Além de estruturas de adjacencias, existem diversas outras representações possíveis para grafos |27|, como por exemplo, as matrizes de adjacencias, que são matrizes binárias $A = (a_{ij})$, tais que

(I-29) $$a_{ij} = 1 \iff \{x_i, x_j\} \in V$$

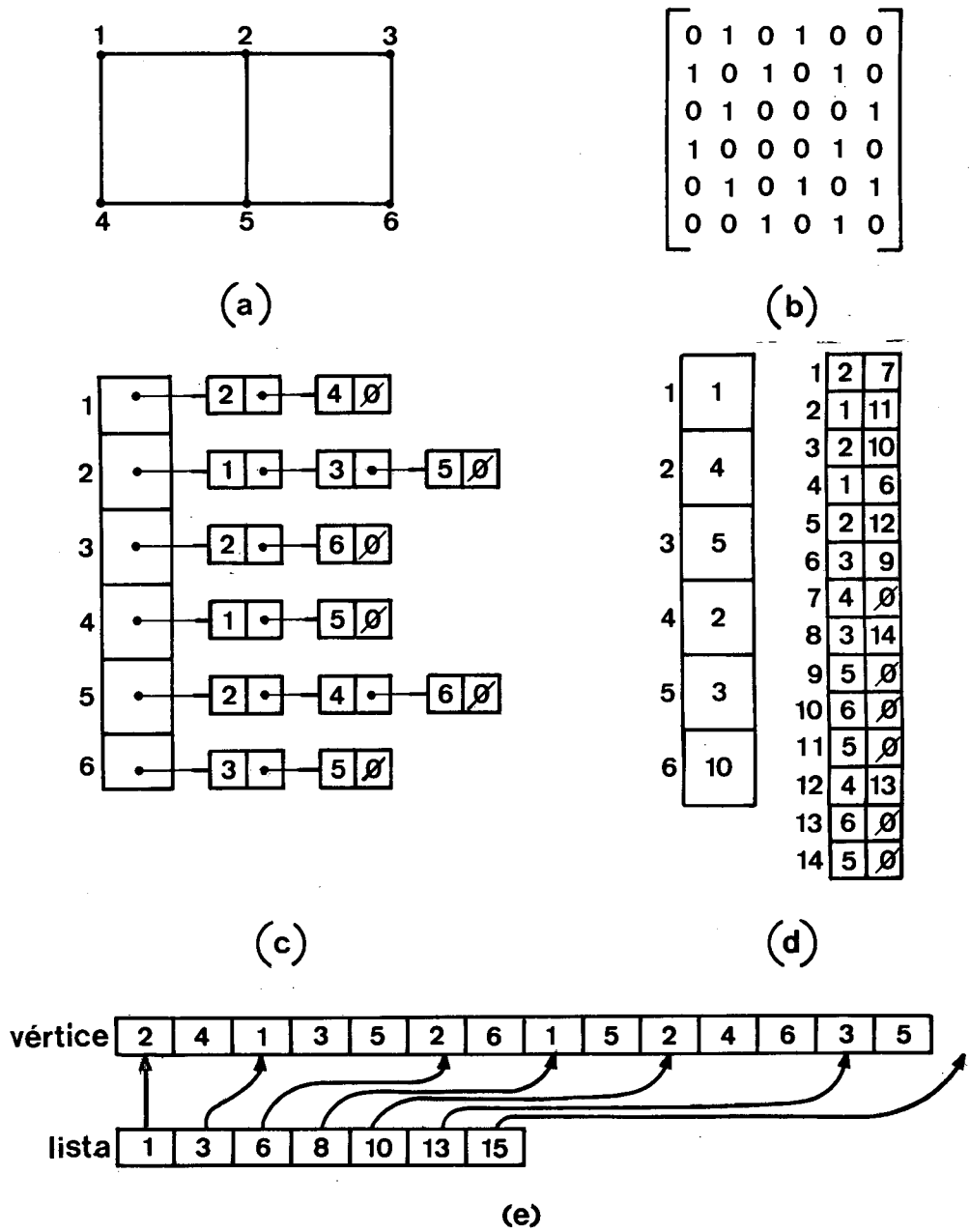


Figura I-6: Representação de um grafo (a) Grafo G = (V,E), (b) matriz de adjacencias de G, (c) estrutura de adjacencias de G, (d) estrutura de adjacencias de G com listas de adjacencias encadeadas e (e) estrutura de adjacencias de G com as listas de adjacencias armazenadas sequencialmente.

A figura (I-6) mostra um exemplo de representações de grafos: a parte (a) mostra um grafo G = (V,E), a parte (b) mostra sua matriz de adjacencias, a parte (c) mostra uma estrutura de adjacencias de G com as listas encadeadas, a parte (d) mostra uma estrutura de adjacencias com as adjacencias dos vertices armazenadas em um vetor, porém utilizando outro vetor (de apontadores) para compor as listas de adjacencias, e a parte (e) mostra uma estrutura de adjacencias com as listas de adjacencias armazenadas sequencialmente em um vetor.

Nos algoritmos descritos nesse trabalho é utilizada a representação mostrada na figura (I-6d) por ser a mais adequada as características dos mesmos. Para um grafo de n vertices e m arestas essa representação requer n + 4m elementos.

Uma estrutura de níveis pode ser representada com os vertices de cada nível armazenados sequencialmente em um vetor, com um outro vetor, de apontadores, para indicar o início de cada nível no vetor de vertices, veja a figura (I-7a). Em vez do vetor de apontadores, pode-se utilizar marcas ("tag") associadas ao vetor de vertices, isto é, armazenar o primeiro vertice de cada nível com o sinal negativo. Veja exemplo na figura (I-7b).

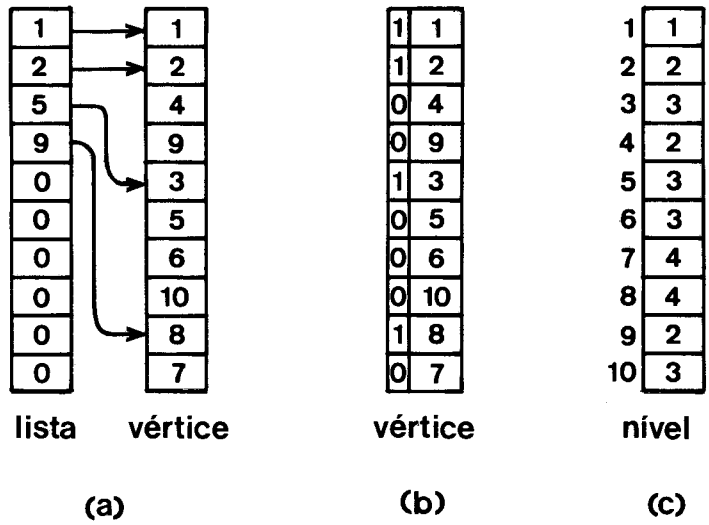


Figura I-7: Representação de estruturas de níveis (a) representação utilizando dois vetores, (b) representação utilizando um vetor com marcas e (c) representação com armazenamento dos ní-

Uma outra representação de estruturas de níveis é através de um vetor de níveis, de tal maneira que, se o i-ésimo elemento do vetor contem j, significa que o vertice x_i está no nível j da estrutura de níveis. Veja exemplo na figura (I-6c).

I.4. Caracterização de Problemas NP-Completos

Para caracterizar o que é um problema NP-Completo ([3],[18],[19]) é necessária uma noção mais precisa de "algoritmo de tempo polinomial", para tal podemos definir um algoritmo como uma "caixa preta" que, ao receber uma sequencia de caracteres de entrada produz uma sequencia de caracteres de saída. Graficamente, esta noção é dada na figura I-8.

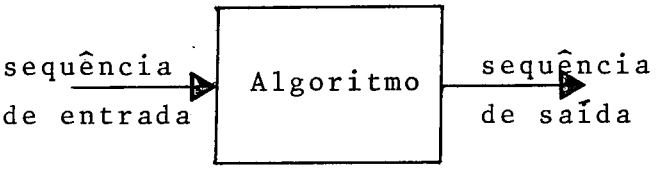


Figura I-8: Um algoritmo visto como uma "caixa preta".

A sequencia de entrada do problema e a sequencia de saída do algoritmo podem ser codificados como números binários, e o algoritmo pode ser pensado como uma sequencia de operações binárias elementares que utilizam uma memória de dígitos binários (bits), que pode ser arbitrariamente grande.

Desde que todos os tipos de dados de uma linguagem de alto-nível (tipo Algol, por exemplo) contem números finitos de bits, a tradução de algoritmos de linguagens de alto nível para o nível de bits aumenta o número de operações apenas polinomialmente, pois as operações de linguagens de alto nível requerem tempos polinomiais no comprimento de seus operandos. Assim, se um algoritmo requer $T(l)$ operações de alto nível para uma entrada de comprimento l, então requererá, no máximo, $P(T(l))$ operações com bits para tal entrada. Assim, o carater polinomial ou exponencial do tempo de execução de um

algoritmo é invariante, uma vez que $P(T(l))$ é limitado por um polinomio em l, se e, somente se, $T(l)$ for limitado por um polinomio em l. O mesmo argumento mostra que se as sequencias de entrada e de saída forem codificadas numa maneira "racional"*, o carater polinomial ou exponencial do comprimento da sequencia de saída, como uma função do comprimento da sequencia de entrada, é também invariante.

Definimos um algoritmo de tempo polinomial como sendo um algoritmo cujo tempo de execução (número de operações binárias elementares que ele executa), para uma sequencia de entrada de comprimento l, é limitado superiormente por algum polinomio $P(l)$. *P* é a classe de todos os problemas que podem ser resolvidos por um algoritmo desse tipo.

Para conceituar a classe *NP* de problemas é necessário introduzir a idéia de algoritmo não-determinístico.

Informalmente, definimos o estado de um algoritmo como sendo o endereço da instrução correntemente sendo executada, juntamente com os valores de todas as suas variáveis. Um algoritmo não-determinístico é um algoritmo tal que, para algum dado estado, existe mais de um próximo estado válido**, e o algoritmo passaria, simultaneamente, para todos os próximos estados válidos, continuando sua execução.

De uma maneira mais intuitiva, considere um algoritmo que vai efetuando suas operações sequencialmente até alcançar um ponto (estado) em que uma escolha deve ser feita entre várias alternativas possíveis. Um algoritmo determinístico teria de explorar uma alternativa simples e retornar mais tarde para explorar as alternativas restantes. Um algoritmo não-determinístico pode explorar todas as alternativas simul-

---

* Não há uma definição precisa de "racional", mas, em geral, não seria racional codificar a entrada de um problema em mais caracteres que o número mínimo exigido pela teoria da informação.

** Algoritmos não-determinísticos não são em nenhum sentido algoritmos probabilísticos ou aleatórios.

taneamente, essencialmente "criando" uma cópia de si mesmo para cada alternativa. Todas as suas cópias funcionam independentemente, sem nenhum tipo de comunicação entre elas. Essas cópias podem naturalmente "criar* outras cópias de si mesmas, e assim por diante. Se uma cópia determina que fez uma escolha incorreta, ela interrompe sua execução. Se qualquer cópia encontra uma solução, ela anuncia o sucesso e todas as cópias interrompem suas execuções.

Representaremos um algoritmo não-determinístico utilizando os três primitivos [19] choice, success e failure. choice (S) é uma função cujos valores são os elementos de um conjunto finito S, sua execução corresponde a "criação" de |S| cópias do algoritmo - uma para cada possível escolha de um elemento do conjunto S. success causa a interrupção da execução de todas as cópias existentes e indica o termino da execução com sucesso. failure causa a interrupção na execução apenas da cópia do algoritmo que alcançar esse primitivo.

Obviamente, não há limitação para o número de cópias de um algoritmo não-determinístico, que podem ser criadas para uma sequencia de entrada generica, não sendo tais algoritmos, portanto, viáveis na pratica; o não-determinísmo é uma abstração que possibilita simplificações na descrição de algoritmos permitindo evitar a programação de retrocessos na execução ("backtrack") para permitir a exploração de alternativas ainda não examinadas.

NP é a classe de todos os problemas que podem ser resolvidos, por algoritmos não-determinísticos, num tempo polinomial no comprimento da sequencia de entrada.

Problemas [19] NP-Difíceis e NP-Completos são dois conceitos fundamentais relativos a classe *NP*. Para defini-los vejamos o conceito de transformabilidade de problemas.

Um problema P_1 é transformável num problema P_2 se qualquer instancia de P_1 pode ser transformada, num tempo polinomial, em uma instancia de P_2, tal que, a solução de P_1 possa ser obtida da solução da instancia de P_2, num tempo polinomial. Esse conceito está graficamente mostrado na figura (I-9).

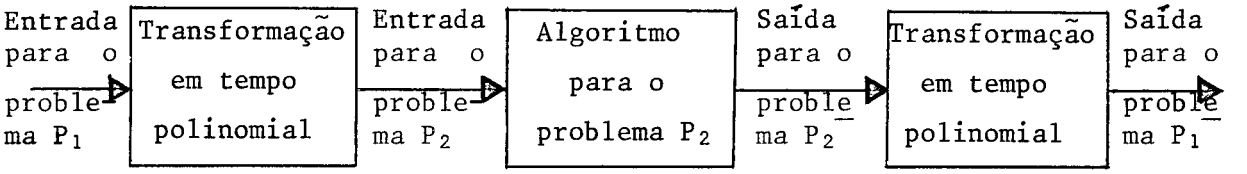


Figura I-9: Diagrama da transformabilidade do problema P_1 ao problema P_2.

Um problema é NP-Difícil se todo problema em *NP* é transformável a ele. Um problema NP-Difícil é NP-Completo se pertence a *NP*.

Para provar que um problema A é NP-Completo tem-se que construir um algoritmo não-determinístico que determine, num tempo polinomial, se uma qualquer instancia de A tem solução, como também, tem-se que provar que algum problema NP-Difícil é transformável a A.

A dificuldade está em estabelecer, como ponto de partida, que algum problema particular é NP-Difícil. Atualmente, já foi demonstrado que um grande número de problemas são NP-Difíceis. Em geral, o problema de satisfazibilidade ("satisfiability") ([3],[19]) da logica matematica é citado ([3],[19]) como ponto de partida.

Seja $S = \{x_i, \bar{x}_i : x_i = \text{true} \iff \bar{x}_i = \text{false},\ 1 \leq i \leq n\}$, as variáveis booleanas $x_i, \bar{x}_i$ são denominadas literais, e as operações booleanas "ou" e "e" são representadas pelos símbolos "V" e "Λ", respectivamente. Uma clausula é o "ou" de um conjunto de literais, e uma expressão booleana na forma normal conjuntiva é o "e" de um conjunto de clausulas. Pode-se mostrar que qualquer expressão booleana pode ser reescrita na forma normal conjuntiva [19].

Uma expressão booleana é dita satisfazível se existir alguma atribuição dos valores true e false as suas variáveis, de tal maneira que a expressão tenha o valor true.

O problema de satisfazibilidade (denotado por

SAT) é determinar se uma expressão na forma normal conjuntiva é satisfazível.

Teorema I.4.: O problema SAT é NP-Completo. Uma demonstração formal desse teorema, pode ser encontrada em [3], não sendo de interesse repeti-la aqui.

O corolário I.5. é usado no capítulo II, para mostrar que o problema de minimização de largura de banda é, também, NP-Completo.

3-SAT é o problema de satisfazibilidade em que a expressão booleana na forma normal conjuntiva tem no maximo 3 literais por clausula. Está demonstrado em [19].

Corolário I.5.: O problema 3-SAT é NP-Completo.

Demonstração: 3-SAT está em *NP*, pois o algoritmo seguinte é um algoritmo não-determinístico limitado polinomialmente que resolve 3-SAT.

E $(x_1,x_2,\ldots,x_n)$ representa uma expressão boolea na qualquer na forma normal conjuntiva com, no maximo, 3 literais por clausula.

```
begin
  for i <- 1 until n do x_i <- choice ({true,false});
  if E (x_1,x_2,...,x_n) then success;
                         else failure;
end;
```

Para mostrar que 3-SAT é NP-Difícil basta mostrar que o problema SAT é transformável no problema 3-SAT.

Dada uma expressão booleana $E(x_1,x_2,\ldots,x_n)$ na forma normal conjuntiva, podemos substituir cada clausula $(\alpha_1,\alpha_2,\ldots,\alpha_k)$ $k > 3$, pèla expressão

$$(\alpha_1 \vee \alpha_2 \vee \beta) \wedge (\alpha_3 \vee \ldots \vee \alpha_k \vee \bar{\beta})$$

onde β é uma nova variável. Repetindo esse processo até que nenhuma clausula tenha mais que três literais obtemos uma expressão booleana E'.

É fácil mostrar que E' é satisfazível se e so se a expressão E original for satisfazível. Além disso, E' pode

ser construida, a partir de E, num tempo polinomial no comprimento de E.

Portanto, 3-SAT é NP-Completo.

II. PROBLEMAS DE MINIMIZAÇÃO DE LARGURA DE BANDA E DE PERFIL

Nesse capítulo é demonstrado que o Problema de Minimização de Largura de Banda (PMB) é NP-Completo. Tal resultado é apresentado por Papadimitriou em [17], e é aqui transcrito.

O Problema de Minimização de Perfil, segundo Papadimitrou, está implicito no Teorema 1.5., em [20].

Para provar que o PMB é NP-Completo é necessário demonstrar, como resultados intermediários, que outros problemas são NP-Completos. Tais problemas, bem como o PMB, são formalizados abaixo.

Definição II.1.: O PMB é o seguinte problema:
Dado um grafo $G = (V,E)$ e um inteiro $b > 0$, é $B(G) \leq b$?

Definição II.2.: Exatamente 3-SAT é o problema de satisfazibilidade em que a expressão booleana na forma normal conjuntiva tem exatamente 3 literais por clausula.

Definição II.3.: O Problema de Arranjo Linear (PAL) é o seguinte: Dado um grafo G e uma função inteira ℓ de rotulos sobre as arestas, pode-se dispor os vertices em um arranjo linear, com distancias unitarias entre vertices consecutivos, de tal forma que nenhum par de vertices adjacentes tenham uma distancia maior que o rotulo da aresta que os une?

Um exemplo da solução do PAL para o grafo na figura (II-1a) é mostrado na figura (II-1b).

Definição II.4.: O PAL Restrito é um PAL onde os únicos rotulos permitidos às arestas são b e 2b - 1, para algum inteiro $b > 0$.

Observe que o PMB é também um caso especial do PAL em que o único rotulo permitido às arestas é $b > 0$.

Uma partição* $\{P,Q\}$ do conjunto $S = \{x_i, \bar{x}_i : 1 \leq i \leq n\}$ de n variáveis booleanas e suas negações em dois conjuntos P e Q é dita consistente se $\forall i \leq n$ $(x_i \in P \iff \bar{x}_i \notin P)$.

---

* Evidentemente, para que $\{P,Q\}$ seja uma partição de S é necessário que $P \cap Q = \phi$ e $S = P \cup Q$.

Uma partição consistente S = P ∪ Q induz a seguinte relação verdade t

(II-1) $$t\ [x_i] = \underline{true} \iff \bar{x}_i \in P.$$

Na seção seguinte é mostrado que os problemas Exatamente 3-SAT, LAP, LAP Restrito e PMB são NP-Completos.

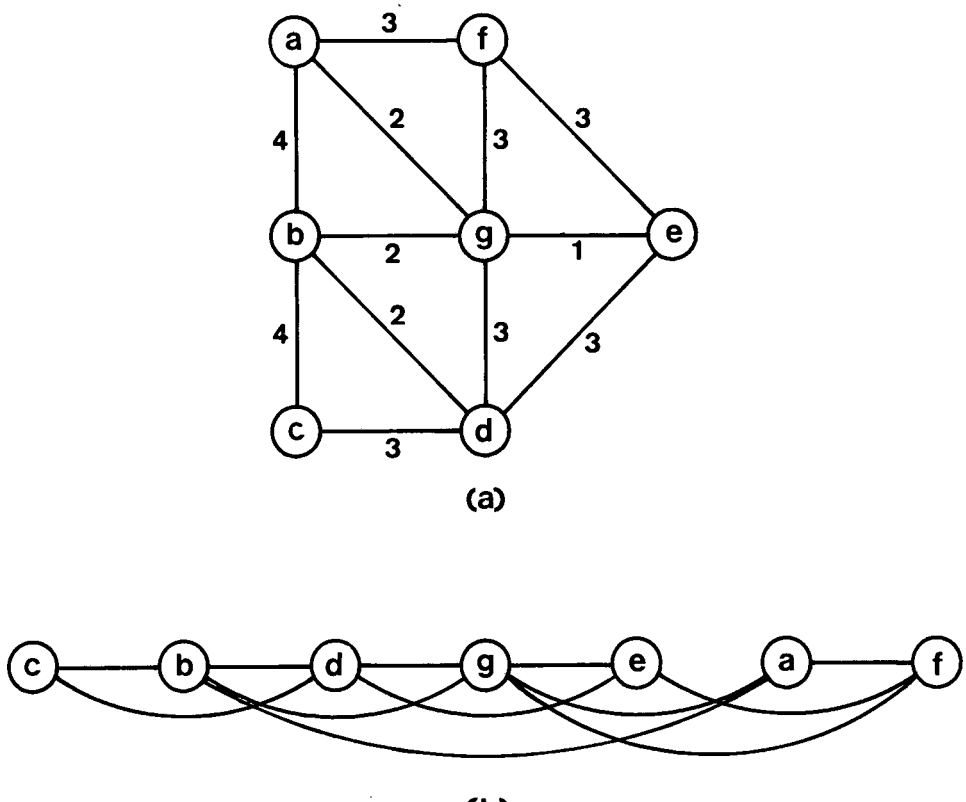


Figura II-1

II.2. Caracterização do PMB como NP-Completo

Lema II.1.: Os seguintes problemas estão em *NP*:

a) Exatamente 3-SAT

b) PAL

c) PAL Restrito, PMB

Demonstração:

a) Como o problema Exatamente 3-SAT é um caso especial do problema SAT [3], também está em *NP*.

b) O algoritmo não determinístico abaixo resolve PAL [G] num tempo polinomial, para qualquer grafo G = (V,E,ℓ)* e, portanto, o PAL está em *NP*.
Esse algoritmo utiliza uma tabela c_{x_i,x_j} assim definida:

$c_{x_i,x_j} = c_{x_j,x_i} = \ell(\{x_i,x_j\})$, se $\{x_i,x_j\} \in E$, e

$c_{x_i,x_j} = c_{x_j,x_i} = \infty$, caso contrario.

```
begin
  V <- {x1,x2,...,xn};
  v1 <- choice (V);
  V <- V - {v1};
  k <- 1;
  while V ≠ φ do
    begin
      k <- k + 1;
      vk <- choice (V);
      V <- V - vk;
      for i <- 1 until k - 1 do
        if k - i > c(vi,vk) then failure;
    end;
  success;
end;
```

c) Esses problemas são casos especiais do PAL.

Lema II.2.: O problema Exatamente 3-SAT é NP-Completo.

Demonstração: A demonstração é dada em [17] por uma redução do problema 3-SAT (que é NP-Completo [19]) ao problema Exatamente 3-SAT, num tempo polinomial.

Teorema II.3.: O PAL é NP-Completo.

Demonstração: Seja $B = F_1 \wedge F_2 \wedge \ldots \wedge F_r$ uma expressão booleana na forma normal conjuntiva, com variáveis $x_1,x_2,\ldots,x_n$, contendo cada clausula F_i exatamente três variáveis.

Construiremos um grafo G rotulado, tal que, B é satisfazível se, e somente se, PAL [G] tem solução.

---

* onde ℓ é a função de rotulos sobre as arestas.

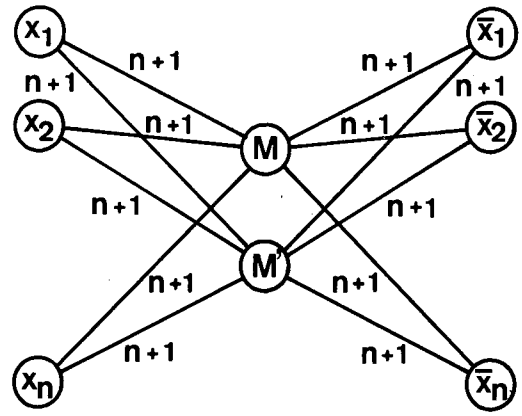


Figura II-2

Considere o grafo H na figura (II-2) o PAL [H] é satisfeito por qualquer arranjo dos seus vertices, tal que M e M' ocupam a (n + 1)-ésima e (n + 2)-ésima posições, respectivamente. Assim, qualquer solução do PAL [H] corresponde a uma partição de $S = \{x_i, \bar{x}_i : 1 \leq i \leq n\}$, em conjuntos P e Q de tamanho n, com M, M' servindo como delimitadores.

O grafo G contem n + r + 1 cópias de H (denotadas por $H_1, \ldots, H_{n+r+1}$) e n + r outros vertices A_i, i = 1,...,n+r. Cada A_i é ligado por arestas de comprimento (rotulo) n + 2 a M e M' de H_i e H_{i+1}. Note que, desse modo, o PAL [G] pode ser resolvido pelo arranjo mostrado na figura (II-3), onde uma solução de cada cópia H_i é seguida pelo vertice A_i e pela solução de H_{i+1}.

Agora ligamos os literais correspondentes em cópias consecutivas de H por arestas de comprimento 2n + 5 (figura (II-3)). Isso obriga a que em todas as cópias de H se tenha a mesma partição P e Q, embora seja possível alguma "mobilidade" dentro dos conjuntos. Denotamos essa partição única por S = P ∪ Q, assumindo que Q é o conjunto de literais em H_i, mais próximos de A_i.

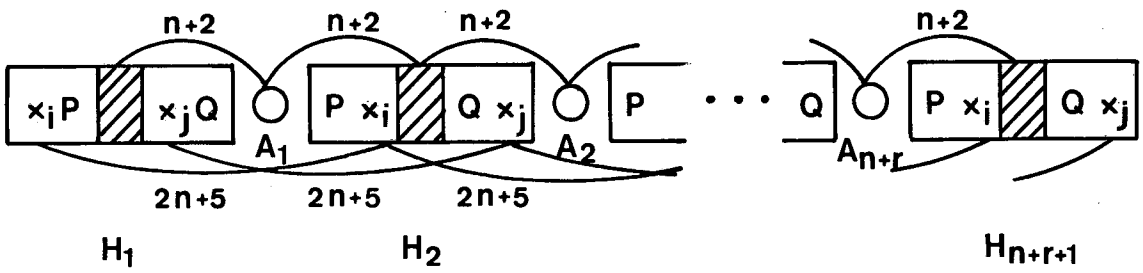


Figura II-3

Para garantir a consistência dessa partição ligamos A_i com as cópias de x_i e $\bar{x}_i$ em H_i por arestas de comprimento $n + 3$, para todo $i \leq n$. Suponha, então, que ambos x_i e $\bar{x}_i$ estejam em P, para algum i (partição inconsistente). Então, A_i está a uma distancia de, pelo menos, $n + 4$ de x_i ou $\bar{x}_i$ em H_i e, portanto, não seria uma solução aceitavel do PAL.

Nas próximas r cópias de H (de H_{n+1} a H_{n+r}) a partição P U Q é forçada a satisfazer B, ligando o vertice A_{n+i} com as cópias em H_{n+i} de todos os três literais que ocorrem em F_i por arestas de comprimento $n + 4$. Assuma que é dado o valor false a todo literal em P, e considere o vetor verdade t induzido por P U Q. Se t não satisfaz B, então para algum $j \leq r$ $F_j(t)$ = false, i.e., todos os três literais de F_j estão em P, assim, em H_{n+j} pelo menos um desses literais estaria a uma distancia de, no mínimo, $n + 5$ de A_{n+j}. Por outro lado, se t satisfaz B, pode ser achada uma partição tal que t é induzido por ela; nas $n + r$ cópias de H os literais podem ser arranjados (devido a mobilidade já mensionada) de maneira que o PAL [G] seja resolvido.

Assim, B é satisfazível se, e somente se, PAL [G] for resolvível.

Além do que, as construções empregadas nessa demonstração são de complexidade polinomial em tempo.

Corolário II.4.: O PAL Restrito é NP-Completo.

Demonstração: Uma variação da construção empregada na demonstração do teorema II.3. é aqui utilizada, estando detalhada em [17].

Em vez dos vertices M e M' na i-ésima cópia de H é usado um bloco de $m_i \geq 3$ vertices, para cada i. Cada vertice A_i é também substituido por um bloco de 3 vertices, para cada i.

E, analogamente, à demonstração anterior, são definidas arestas em G, de maneira que a partição P U Q seja consistente. Esse grafo G contém somente arestas de comprimento n + 4 e 2n + 7, e é tal que B é satisfazível se, e somente se, PAL [G] Restrito for resolvível.

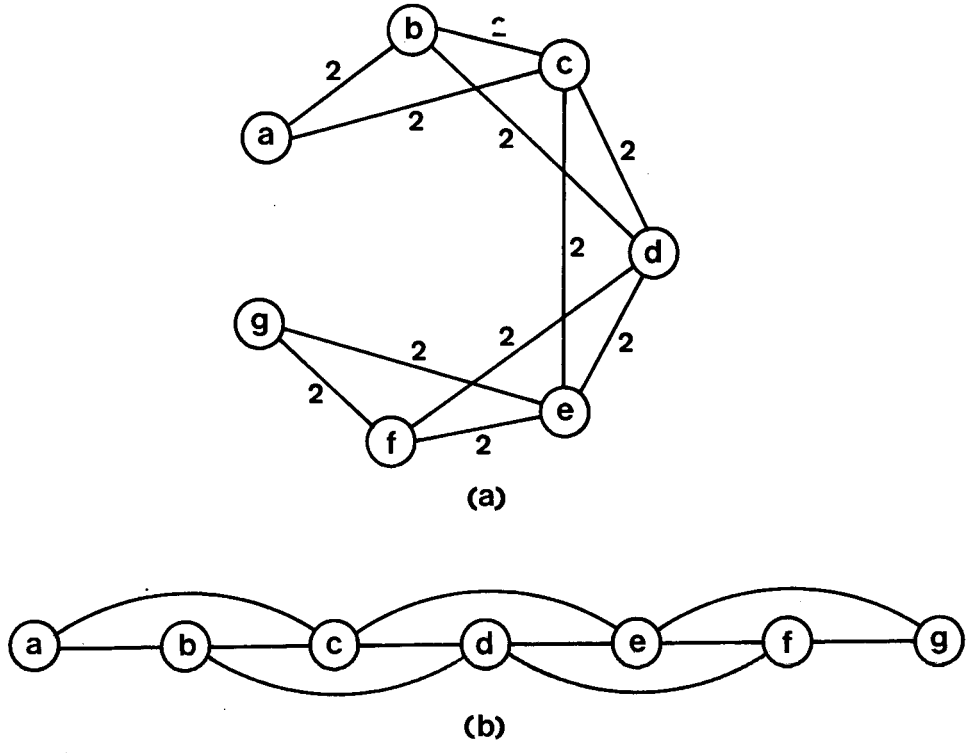


Figura II-4: (a) Exemplo de um grafo K_3^7, e (b) uma solução do PAL $[K_3^7]$.

Para a demonstração do lema II.5., definimos um grafo $K_p^q = (V,E,\ell)$, com $p \leq q$, tal que $V = \{v_1,v_2,\ldots,v_q\}$, $E = \{\{v_i,v_j\} : |i - j| < p\}$, e $\ell(\{i,j\}) = p - 1$, para todo $\{v_i,v_j\} \in E$. A demonstração do lema II.5. se baseia no seguinte fato: se um grafo K_p^q for um subgrafo de algum grafo G e, q for maior que o comprimento de qualquer aresta de G, em qualquer solução do PAL [G] K_p^q é colocado em uma extremidade do arranjo linear de seus vertices.

Da definição das arestas de K_p^q é evidente que, numa solução de PAL [G], nenhum vertice x não pertencente a K_p^q pode aparecer entre seus vertices. E por outro lado, sendo q maior que o comprimento de qualquer aresta de G, na solução de PAL [G], K_p^q não pode ocorrer entre vertices de G.

A figura II-4 mostra o grafo K_3^7 e uma solução do PAL $[K_3^7]$.

Lema II.5.: Qualquer PAL Restrito com $m > 0$ arestas de comprimento $2b - 1$ pode ser transformado, em tempo polinomial, num PAL Restrito com $m - 1$ arestas de comprimento $2b' - 1$, onde b' é o parâmetro do novo PAL Restrito.

Demonstração: Seja um grafo $G = (V,E,\ell)$ com $|V| = N$ e seja $\{x_i,x_j\} \in E$ com $\ell(\{x_i,x_j\}) = 2b - 1$. Seja $k = [N/b]$*.

Obtemos G' e G pela seguinte transformação:

1. Adicionar um vertice c e as arestas $\{x_i,c\}$ e $\{x_j,c\}$ com comprimento $b + 1$.
   Eliminar a aresta $\{x_i,x_j\}$.
2. Aumentar o comprimento de todas as arestas de G de b para $b + 1$, e de $2b - 1$ para $2b + 1$.
3. Adicionar duas cópias do grafo K_{b+1}^{2b+1} com vertices $\{v_1,v_2,\ldots,v_k\}$ e $\{v_1',v_2',\ldots,v_k'\}$, respectivamente.
4. Adicionar as cadeias $C = \{c = c_0,c_1,\ldots,c_k = v_1\}$ e $C' = \{c = c_0',c_1',\ldots,c_k' = v_1'\}$, com arestas $\{c_{i-1},c_i\}$ e $\{c_{i-1}',c_i'\}$ de comprimento $b + 1, i = 1,\ldots,k$.

Suponha que o PAL [G'] seja resolvível. Então, as duas cópias de K_{b+1}^{2b+1} estarão nas duas extremidades do

---

* $[N/b]$ indica o maior inteiro contido em $\frac{N}{b}$.

arranjo linear e, assim, os vertices das duas cadeias C e C' ocorrem ao longo de todo o arranjo. Consequentemente, em qualquer intervalo de comprimento b + 1 existe, pelo menos, um vertice dessas cadeias. Assim, pela simples remoção das duas cadeias, obtemos uma solução do PAL [G].

Suponha agora que PAL [G] seja resolvível. Podemos assumir que, no pior caso, x_i e x_j estão exatamente a uma distância 2b - 1 um do outro. Então, uma solução do PAL [G'] é obtida colocando-se o vertice c no meio entre x_i e x_j e dispondo os outros vertices de C e C' de forma a manter a distância b + 1 entre cada dois vertices adjacentes, atravessando assim todo o grafo. As cópias de K_{b+1}^{2b+1} são colocadas nas duas extremidades do arranjo.

Consequentemente, o PAL [G] é resolvível se e so se o PAL [G'] for resolvível. Mais ainda, o grafo G' pode ser construido num tempo polinomial e tem, exatamente, m - 1 arestas de comprimento 2b + 1.

Teorema II.6.: O PMB é NP-Completo.

Demonstração: Aplicando a construção do lema II.5. m vezes a qualquer PAL Restrito de m > 0 arestas de comprimento 2b - 1, obtem-se um PAL com todos os rotulos iguais a algum b' e, portanto, o PAL Restrito se reduz ao PMB. Resta provar, então, que essa construção é polinomial, o que está detalhado em [17].

III. ALGORITMOS AUXILIARES

III.1. Introdução

Vários dos algoritmos que são discutidos nos capítulos seguintes, ordenam subconjuntos de vertices pertencentes a listas de adjacencias de grafos G(A) = (V,E), em ordem crescente de grau.

Nos capítulos IV e V é mostrado que se as listas de adjacencias do grafo já estiverem com seus vertices em ordem crescente de grau, os passos de ordenação naqueles algoritmos poderão ser eliminados. Em alguns desses algoritmos é suficiente que o conjunto V de vertices esteja com seus vertices ordenados para que seus passos de ordenação possam ser eliminados.

Assim, são apresentados, nas seções III.2. e III.3., esquemas de ordenação de estruturas de adjacencias, e de conjuntos de vertices, respectivamente, que serão utilizados nos próximos capítulos.

Pelo metodo da raiz ("radix sort") [3] se consegue ordenar um conjunto de n números inteiros (conhecidos os seus valores limites) num tempo de O(k + n) [3], onde k é a diferença entre seus valores limites, e por isso, os algoritmos de ordenação aqui apresentados estão baseados nesse metodo.

Na seção III.4. é apresentado um esquema de geração de estruturas de níveis, que será, utilizado em outros capítulos. É, também, mostrado, que se a estrutura de adjacencias do grafo estiver previamente ordenada, a estrutura de níveis então gerada estará ordenada, no sentido definido no lema III.2..

III.2. Ordenação de Estruturas de Adjacencias

O esquema de ordenação de estruturas de adjacencias, aqui apresentado, utiliza uma ordem lexicográfica definida sobre um conjunto de pares ordenados (i,j) associados aos vertices de V, definidos a seguir.

Dado um grafo G(A) = (V,E), o conjunto S de pares ordenados (i,j) é associado a G(A), de tal forma que:

(III-1) $\quad$ (grau (x_i), j) ε S <=> $\{x_i, x_j\} \varepsilon$ E

A cada vertice x_i em V, são, então, associados os pares ordenados (grau (x_i), j) de S, tais que $\{x_i, x_j\}$ pertença a E. Assim, se um determinado vertice tem grau d, ele é associado a d pares ordenados e, consequentemente, aos n vertices de V (n = |V|) são associados 2m (m = |E|) pares ordenados.

Definição III.1.: Uma ordem lexicográfica parcial sobre S, é uma ordem "<", tal que, para todo (i_1,j_1) e (i_2,j_2) em S

$$(i_1,j_1) < (i_2,j_2) \iff \begin{cases} i_1 < i_2, \text{ ou} \\ i_1 = i_2 \text{ e } j_1 < j_2 \end{cases}$$

Definição III.2.: Uma ordenação lexicográfica f de S é uma correspondencia biunívoca f : S -> {1,2,...,2m} tal que

$$\forall\, a,b \in S \; (a < b \Rightarrow f(a) < f(b))$$

Definição III.3.: Uma estrutura de adjacencias de um grafo G = (V,E) é dita estar ordenada, se suas listas de adjacencias estiverem com seus vertices em ordem crescente de grau.

III.2.1. Esquema de Ordenação Lexicográfica

O algoritmo abaixo constroi o conjunto S de pares ordenados, gera uma ordenação lexicográfica f de S e, utilizando f, constroi a estrutura de adjacencias ordenada.

Os graus dos vertices de V (primeiro elemento dos pares ordenados) são armazenados num vetor de n elementos. Tal vetor é, também, fornecido como saída, uma vez que será, posteriormente, usado em outros algoritmos.

Para gerar f, são utilizadas n listas para guardar os 2m pares ordenados, de maneira que a i-ésima lista contenha todos os pares cujo primeiro elemento é i (associados aos vertices de grau i).

Os pares ordenados não são efetivamente transferidos para as listas, apenas um apontador é colocado na i-ésima lista. Assim, essas listas utilizam, apenas, um vetor de n elementos para seus topos, e um vetor de apontadores de 2m elementos.

Algoritmo A1: Ordenação lexicográfica de estruturas de adjacencias.

Entrada: Grafo G = (V,E) representado na forma de uma estrutura de adjacencias.

Saída: Estrutura de adjacencias de G ordenada e vetor contendo os graus dos vertices de V.

Metodo:

1. Calcular o grau de cada vertice, e completar a construção dos pares ordenados (grau (x_i),j), associados aos vertices $x_i \in V$, a partir da estrutura de adjacencias de G.

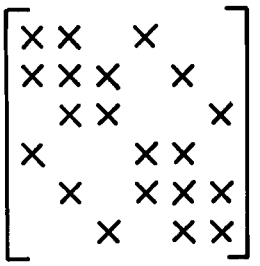
2. Colocar os 2m pares ordenados em n listas, de tal forma que o par (i,j) seja colocado na i-ésima lista, para todo i.

3. Construir a estrutura de adjacencias de G ordenada como se segue:
   Percorrer todas as n listas em ordem, da n-ésima para a primeira, e para cada par (i,j) assim encontrado, inserir o vertice x_k, associado a (i,j) no início da j-ésima lista de adjacencias.

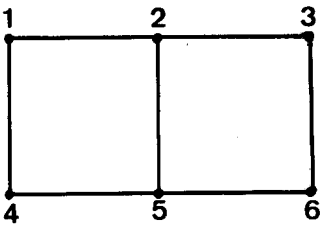
Exemplo III-1.: Considere o grafo G(A), na figura (III-1b). A figura (III-1c) e (d) mostram estruturas de adjacencias de G(A).

As figuras (III-2), (III-3) e (III-4) mostram a estrutura de dados utilizada nesse algoritmo, após a execução de cada um de seus passos. Os vetores LISTA e VERTICE-ELO se referem a estrutura de adjacencias do grafo, os vetores TOPO e Q se referem as n listas auxiliares usadas nos passos 2 e 3, e o vetor GRAU guarda o grau de cada vertice.

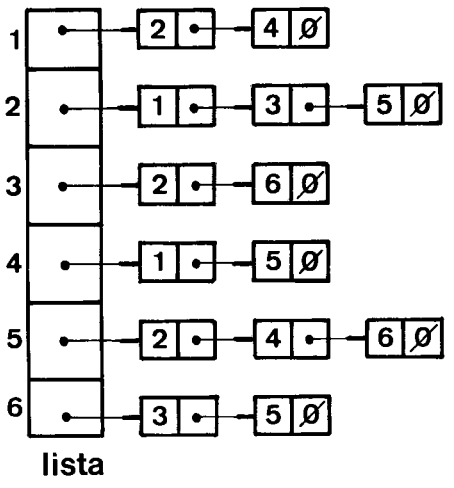
Tais figuras são mais facilmente entendidas se vistas juntamente com a representação tipo Algol desse algoritmo, dada na seção III.3.



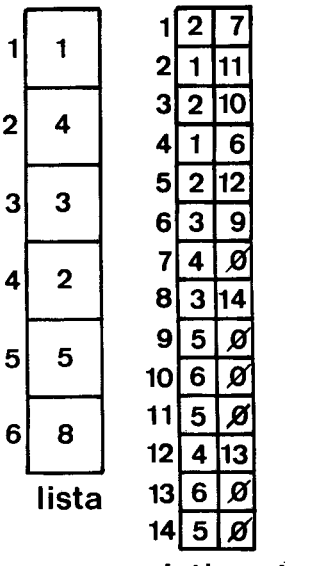
(a)



(b)



(c)



(d)

Figura III-1: Um grafo simples para ilustrar o funcionamento do algoritmo A1:

(a) Matriz $A = (a_{ij})$ simétrica,

(b) Grafo G(A) associado a matriz A,

(c) e (d) Estruturas de adjacencias de G.

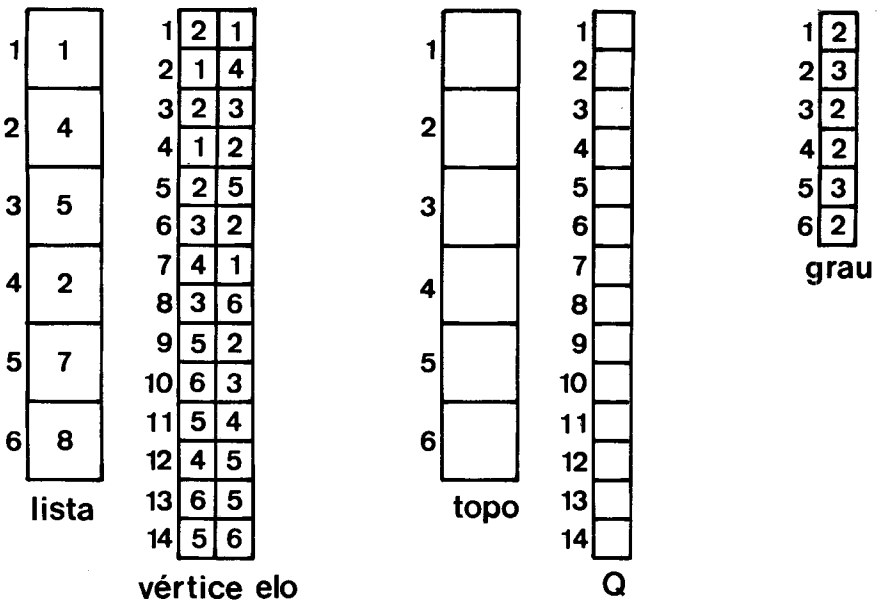


Figura III-2: Estrutrua de dados usada pelo algoritmo A1, após a execução do passo 1.

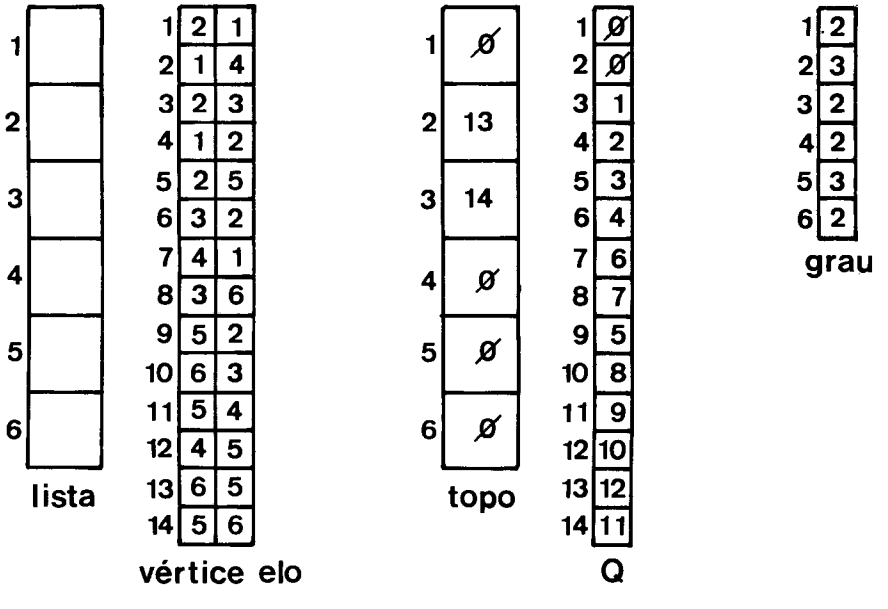


Figura III-3: Estrutura de dados usada pelo algoritmo A1, após a execução do passo 2.

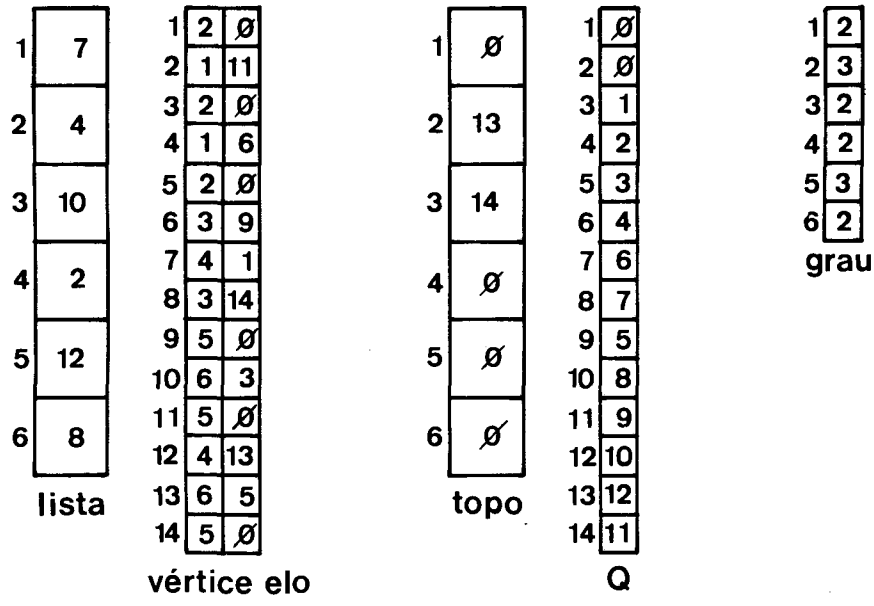


Figura III-4: Estrutura de dados usada pelo algoritmo A1, após a execução do passo 3, com a estrutura de adjacencias já ordenada.

Teorema 1: Dado um grafo G = (V,E), o algoritmo I constroi uma estrutura de adjacencias de G ordenada.

Demonstração:

No passo 1 do algoritmo, para todo x_i pertencente a j-ésima lista de adjacencias de G (i.e., $\forall x_i \in adj(x_j)$) é associado um par ordenado (grau (x_i),j).

No passo 2, todos os pares ordenados (i,j) associados a vertices x_k de grau i são armazenados na i-ésima lista, para todo i.

No passo 3, as n listas são percorridas em ordem decrescente de grau e, consequentemente, $\forall x_u, x_v \in adj(x_i)$, se grau (x_u) < grau(x_v), x_v é colocado na na i-ésima lista de adjacencias de G antes da colocação de x_u.

Como os vertices são sempre colocados no início das listas de adjacencias, tais listas ficam ordenadas por ordem crescente de grau.

III.2.2. Análise de Complexidade

Espaço de Memória

Conforme já foi visto, além da estrutura de adjacencias do grafo, com (n + 4m) elementos, são utilizados dois vetores de n elementos, e um vetor de 2m elementos, portanto são necessários 3(n + 2m) posições de memória.

A complexidade em espaço desse algoritmo é então de 0(n + m).

Tempo de Execução

No passo 1, a estrutura de adjacencias de G é percorrida uma vez, e para o cálculo dos graus dos vertices são efetuadas 2m adições.

Assim, para gerar os pares ordenados (i,j) é necessário um tempo de 0(n + m).

No passo 2, os 2m pares são colocados nas n listas num tempo de 0(m).

No passo 3, as n listas são percorridas num tempo de 0(n + m), uma vez que um teste é executado para identificar cada lista vazia. Um tempo de 0(m) é necessário para construir a estrutura de adjacencias ordenada e, então, um tempo de 0(n + m) é consumido no passo 3.

Logo, a complexidade em tempo de execução desse algoritmo é 0(n + m). Para matrizes esparsas, tais que m é de 0(n) as complexidades em tempo e espaço são, evidentemente, de 0(n).

III.2.3. Formulação Tipo Algol do Algoritmo

Estrutura de Dados Utilizada

O grafo G é representado na forma de uma estrutura de adjacencias (I-6) com listas encadeadas. O vetor LISTA contem os topos das listas de adjacencias, que são constituidas das celulas VERTICE-ELO armazenadas em dois vetores de 2m elementos, como indicado na figura (III-1).

O grau de cada vertice (primeiro elemento dos pares ordenados) é armazenado no vetor GRAU de n elementos. O segundo elemento de cada par é armazenado no campo ELO da estrutura de adjacencias de G, e o campo VERTICE possibilita a correspondencia de cada vertice x_i com os pares ordenados (grau (x_i),j) a ele associados.

Para as n listas, são usados o vetor TOPO de n elementos, representando os topos das pilhas, e o vetor Q de 2m apontadores. A cada par (i,j) apenas um apontador (elemento de Q) é colocado na i-ésima lista, uma vez que os pares ordenados não são efetivamente transferidos para as listas.

Descrição Tipo Algol do Algoritmo A1

```
begin
  comment 1. Construção dos pares ordenados.
  for I <- 1 until N do
    begin
      GRAU [I] <- 0;
      J <- LISTA [I];
      while J > 0 do
        begin
          L <- J;
          J <- ELO [L];
          ELO [L] <- I;
          GRAU [I] <- GRAU [I] + 1;
        end;
      LISTA [I] <- 0;
    end;
  comment 2. Colocação dos pares ordenados nas n listas.
  for I <- 1 until 2*M do
    begin
      J <- GRAU [VERTICE [I]];
      Q [I] <- TOPO [J];
      TOPO [J] <- I;
    end;
  comment 3. Ordenação da estrutura de adjacencias.
  for I <- N step -1 until 1 do
    begin
```

```
      while TOPO [I] > 0 do
        begin
          J <- TOPO [I];
          TOPO [I] <- Q [J];
          L <- ELO [J];
          ELO [J] <- LISTA [L];
          LISTA [L] <- J;
        end;
    end;
end;
```

III.3. Ordenação de Conjuntos de Vertices

Em alguns dos algoritmos discutidos nos capítulos seguintes, é necessário ordenar o conjunto V de vertices, em vez da estrutura de adjacencias do grafo.

Um esquema para classificação dos vertices de um grafo em ordem crescente de grau, pelo metodo da raiz, é mais simples que o de ordenação de estruturas de adjacencias, pois nesse caso, em vez dos pares ordenados tem-se que considerar apenas os respectivos graus.

III.3.1. Esquema de Ordenação de Vertices

Nesse esquema a estrutura de adjacencias é percorrida uma vez para gerar o vetor grau, exatamente como no algoritmo A1.

São utilizadas n listas para guardar os n vertices, de tal maneira que os vertices de grau i sejam colocados na i-ésima lista, para todo i. São utilizados um vetor para os topos das listas, e um vetor de apontadores, ambos com n elementos.

Algoritmo A2: Ordenação de vertices.

Entrada: Grafo G = (V,E) representado na forma de uma estrutura de adjacencias.

Saída: Vetor V com os vertices ordenados por grau.

Metodo:

1. Calcular o grau de cada vertice a partir da estrutura de adjacencias do grafo.
2. Colocar os n vertices em n listas, de tal forma que os vertices de grau i sejam colocados na i-ésima lista.
3. Construir o conjunto ordenado de vertices como se segue: Percorrer todas as n listas em ordem, da primeira para n-ésima, e colocar em V cada vertice assim encontrado.

Esse algoritmo além da estrutura de adjacencias do grafo utiliza os dois vetores de n elementos para as listas, e o vetor V para guardar os vertices ordenados, portanto, requer (4n + 6m) posições de memória, o que significa uma complexidade em espaço de memória de 0(n + m).

Os passos 2 e 3 requerem tempos de execução 0(n), mas como a estrutura de adjacencias tem que ser percorrida no passo 1, a complexidade global em tempo de execução é de 0(n + m).

III.3.2. Formulação Tipo Algol

Estrutura de Dados Utilizada

O grafo G é representado na forma de uma estrutura de adjacencias com listas encadeadas. O vetor LISTA contem os topos das listas de adjacencias, e as listas são compostas das celulas VERTICE-ELO armazenadas em dois vetores de 2m elementos como indicado na figura (III-1).

O grau de cada vertice é armazenado no vetor GRAU de n elementos.

Para as n listas são usados os vetores TOPO e Q, ambos de n elementos. Como os graus podem variar de 0 a n - 1 são admitidos serem esses os limites da dimensão de TOPO.

Descrição Tipo Algol do Algoritmo A2

```
begin
  comment 1. Calculo do vetor GRAU
  for I <- 1 until N do
```

```
    begin
      GRAU [I] <- 0;
      J <- LISTA [I];
      while J > 0 do
        begin
          GRAU [I] <- GRAU [I] + 1;
          J <- ELO [J];
        end;
    end;
  comment 2. Colocação dos vertices nas listas.
  for I <-1 until N do
    begin
      J <- GRAU [I];
      Q [I] <- TOPO [J];
      TOPO [J] <- I;
    end;
  comment 3. Ordenação do conjunto de vertices
  L <- 0;
  for I <- 0 until N - 1 while L < N do
    begin
      J <- TOPO [I];
      while J > 0 do
        begin
          L <- L + 1;
          V [L] <- J;
          J <- Q [J];
        end;
    end;
end;
```

III.4. Geração de Estruturas de Níveis

Um esquema de geração de estruturas de níveis está, também, incluido nesse capítulo por ser um procedimento básico, usado em vários dos algoritmos seguintes, com alterações mínimas de um para outro.

Esse algoritmo, além da própria estrutura de níveis fornece como saída, ainda, o pai de cada vertice relativo a estrutura de níveis gerada, definido a seguir.

Definição III.4.: Dada uma estrutura de níveis $N_v(G) = \{N_1(v), N_2(v), \ldots, N_{k(v)}(v)\}$, e um vertice x pertencente a $N_p(v)$, $p > 1$, o pai de x relativo a estrutura de níveis $N_v(G)$ é o primeiro vertice incluido em $N_{p-1}(v)$ adjacente a x. O pai de v relativo a $N_v(G)$ é $\emptyset$.

III.4.1. Esquema de Geração de Estruturas de Níveis

Esse algoritmo coloca o vertice inicial v (raiz) no nível N_1 da estrutura de níveis $N_v(G)$, sendo gerada. E, tomando cada vertice w já na estrutura de níveis (em sua ordem de inclusão), percorre a lista de adjacencias de w colocando em $N_v(G)$ todo vertice assim encontrado, ainda não incluido na estrutura de níveis.

O grafo G é representado na forma de uma estrutura de adjacencias, e a estrutura de níveis $N_v(G)$ é representada na forma de um vetor, sendo a mudança de nível indicada por uma marca ("tag"): o primeiro vertice de cada nível é representado com o sinal negativo.

Um outro vetor de n elementos é utilizado para guardar o pai de cada vertice, relativo a estrutura de níveis N_v. Esse vetor é usado de tal maneira que, se o vertice x de V ainda não tiver sido incluido em N_v, então pai (x) < 0.

Esse vetor será, posteriormente, utilizado por outros algoritmos.

Algoritmo A3: Geração de estruturas de níveis.

Entrada: Grafo G(A) = (V,E), representado na forma de uma estrutura de adjacencias, e um vertice v de V.

Saída: Estrutura de níveis $N_v(G) = \{N_1(v), N_2(v), \ldots, N_{k(v)}(v)\}$, de profundidade $k(v)$.

Metodo:

1. Colocar v no nível N_1 de $N_v(G)$.
   Fazer i <— 1 e j <— 1.

2. Fazer i <— i + 1.
   Construir o nível N_i ($i \geq 2$) como se segue:

Considerar cada vertice $\underline{w}$ em N_{i-1}, em sua ordem de inclusão, e incluir em N_i, todos os vertices na lista de adjacencias de $\underline{w}$, ainda não colocados em N_i, marcando $\underline{w}$ como pai de tais vertices.

3. Fazer $j \leftarrow j + |N_i|$, e repetir o passo 2 enquanto $j < n$.

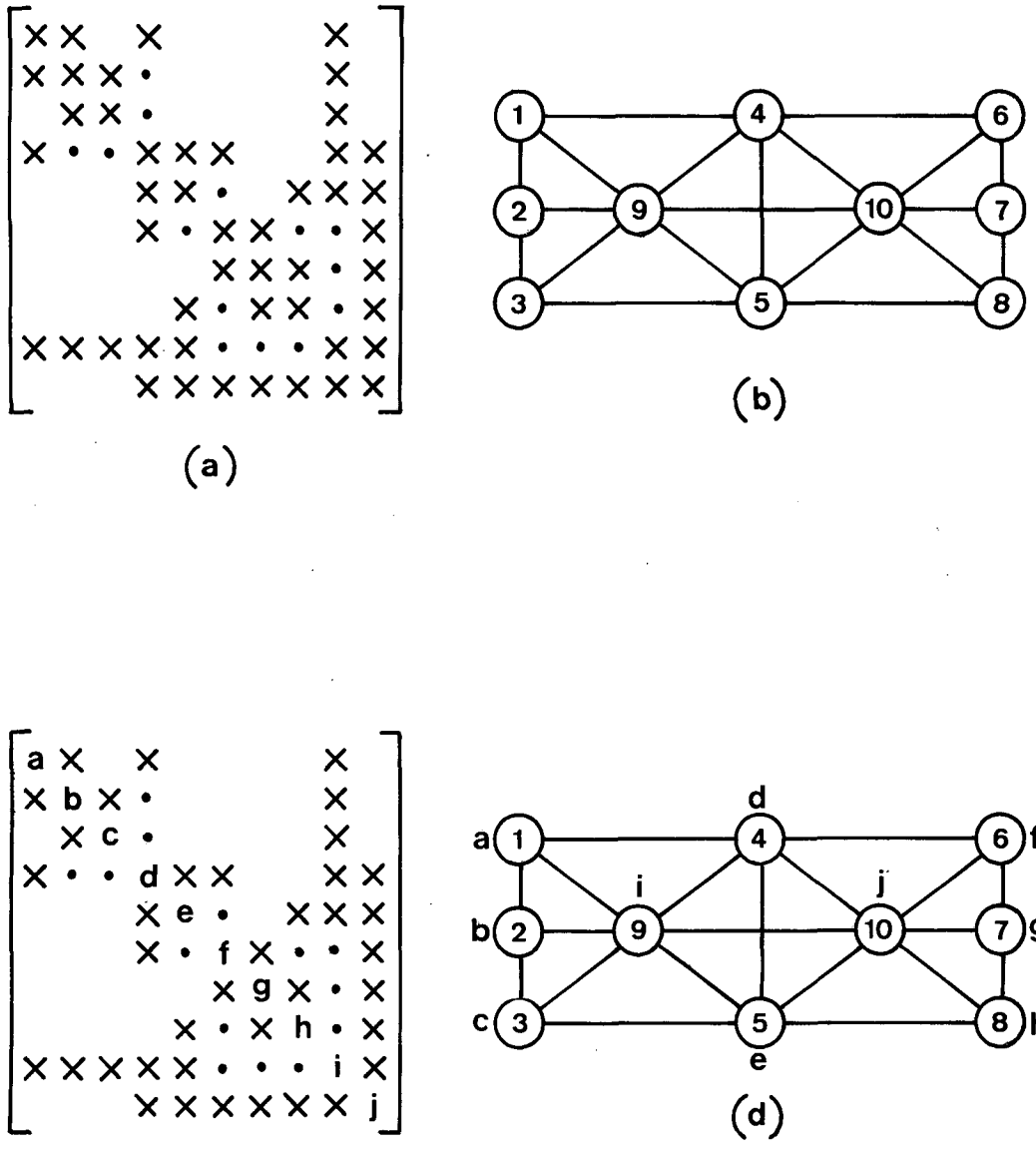


Figura III-5: Um grafo simples para ilustrar o funcionamento do algoritmo A3.

(a) Matriz A, com $B(A) = 8$ e $P(A) = 26$

(b) Grafo G(A) associado

(c) Matriz A com os rotulos alfabeticos na diagonal

(d) Grafo G(A) com os vertices rotulados

Exemplo III-2: Considere a matriz A, figura (III-5a) e o grafo G(A) associado, na figura (III-5b).

O símbolo X, na matriz A, denota a localização dos elementos não nulos, e o ponto (•) denota a localização dos elementos nulos, existentes dentro do envelope da matriz.

Os vertices de G(A) são representados por círculos, e os inteiros contidos nos círculos, se referem ao número da linha (e coluna) de A associada ao referido vertice de G(A). Tais inteiros constituem uma numeração dos vertices de G(A).

Em geral, um rotulo alfabetico é, ainda, associado a cada vertice de G(A). Tais rotulos são colocados nos elementos da diagonal principal de A, permitindo identificar, de imediato, a correspondencia entre os vertices e as linhas e colunas de A, vide figuras (III-5(c)) e (III-5(d)).

Para o grafo da figura (III-5(d)) o algoritmo A3 procede como se segue:

Passo 1: Suponha que a seja o vertice inicial, isto é, v = a. $N_1 = \{a\}$ é então gerado.

Passo 2: A lista de adjacencias de a é percorrida e o nível $N_2 = \{b,d,i\}$ é gerado com i = 2.

Passo 3: Faz j = 4 e volta ao passo 2.

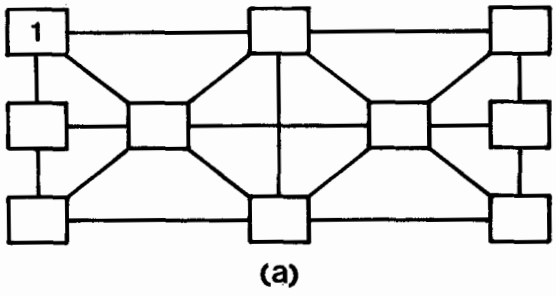
Passo 2: As listas de adjacencias de b, d e i são percorridas e o nível $N_3 = \{c,f,e,j\}$ é gerado com i = 3.

Passo 3: Faz j = 8 e volta ao passo 2.

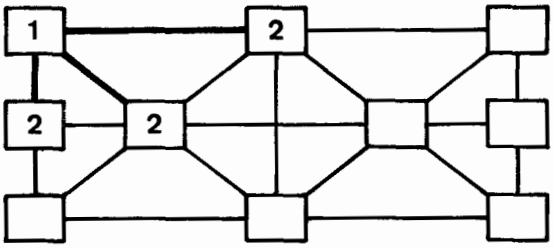
Passo 2: As listas de adjacencias de c, f, e e j são percorridas e o nível $N_4 = \{g,h\}$ é gerado com i = 4.

Passo 3: Faz j = 10, e o algoritmo para.

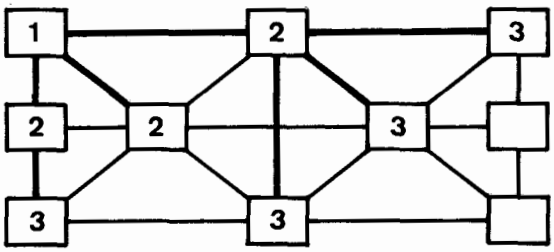
Na figura (III-6) o funcionamento desse algoritmo é mostrado com a estrutura de níveis sendo representada, graficamente, na forma do grafo G. Nesse caso, os vertices são representados por retangulos e os inteiros contidos nos retangulos indicam os números dos níveis da estrutura $N_v(G)$ onde tais vertices são colocados.



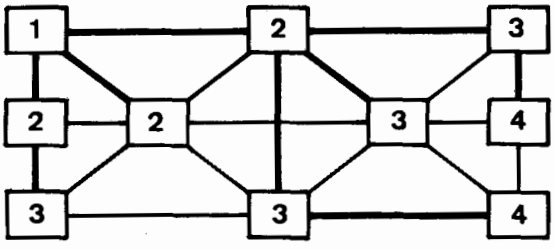
(a)



(b)



(c)



(d)

Figura III-6: Representação gráfica do funcionamento do algoritmo A3, mostrando, passo a passo, a geração da estrutura de níveis $N_a(G)$, com raiz no vertice a.

As arestas mais largas são as arestas "percorridas" na geração da estrutura de níveis e mostram que, efetivamente, o processo de geração de uma estrutura de níveis é equivalente ao processo de determinação de uma árvore geradora de G por busca em largura ("breadth-first").

O vetor pai criado no passo 2 é usado, também, para marcar os vertices ainda não incluidos na estrutura de níveis $N_v(G)$, de tal forma que: pai $(x_i) < 0 \iff x_i$ ainda não tiver sido colocado em N_v.

Por outro lado, ao incluir um vertice u em N_v, faz-se pai (u) = w, onde w é o vertice cuja lista de adjacencias esteja sendo percorrida no momento da inclusão de u. Isto é, pai (u) aponta para o ancestral de u na árvore geradora correspondente a estrutura de níveis N_v.

Lema III.2.: Se a estrutura de adjacencias de um grafo G = (V,E) estiver previamente ordenada, então uma estrutura de níveis qualquer $N_v(G) = \{N_1(v), N_2(v), \ldots, N_{k(v)}(v)\}$, gerada pelo algoritmo A3, é tal que, para todo nível $N_p(v)$, p > 1:

$$\forall u,w \in N_p \ (\text{pai}(u) = \text{pai}(w) \text{ e } \text{grau}(u) < \text{grau}(w) \Rightarrow u \text{ é incluido em } N_p(v) \text{ antes de } w),$$

onde pai(x) é o pai de x relativo a $N_v(G)$.

Demonstração: Se a estrutura de adjacencias do grafo estiver ordenada, ao percorrer, no passo 2, a lista de adjacencias de um vertice w qualquer, suas adjacencias serão incluidas em $N_v(G)$ por ordem crescente de grau.

III.4.2. Análise de Complexidade

Espaço de Memória

A estrutura de adjacencias de G ocupa, conforme já foi visto, (n + 4m) elementos. A estrutura de níveis $N_v(G)$ e o vetor dos pais relativos a N_v ocupam dois vetores de n elementos.

Portanto, o espaço de memória requerido é (3n + 4m), ou seja, de O(n + m).

Tempo de Execução

Os passos 1 e 3 requerem um tempo de 0(c).

No passo 2, no pior caso, serão percorridas todas as listas de adjacencias do grafo, até que os n vertices de V sejam incluidos em $_{V}(G)$. Logo, um tempo de 0(n + m) é requerido no passo 2.

Portanto, a complexidade em tempo de execução desse algoritmo é de 0(n + m).

Para matrizes esparsas, tais que m é 0(n), as complexidades em espaço e tempo serão, evidentemente, de 0(n).

III.4.3. Formulação Tipo Algol

Estrutura de Dados Utilizada

O grafo G é representado na forma de uma estrutura de adjacencias com listas encadeadas.

A estrutura de níveis $_{V}(G)$ é armazenada no vetor ESTNIVEL de n elementos. Sendo o primeiro vertice de cada nível armazenado com o sinal negativo.

O pai de cada vertice, relativo a estrutura de níveis $_{V}$, é armazenado no vetor PAI de n elementos.

Descrição Tipo Algol do Algoritmo A3

```
begin
  II ← 0;
  TOPO ← 1;
  ULTIMO ← 1;
  KV ← 1;
  comment 1. Colocar V no nível 1 de ESTNIVEL
  ESTNIVEL [1] ← V;
  PAI [*] ←- 1;
  PAI [V] ← 0;
  while II < TOPO & TOPO < N do
    begin
      comment 2. Construção do nível (KV + 1) de ESTNIVEL
      II ← II + 1;
      X ← ESTNIVEL [II]
      I ← LISTA [X]
      if II = ULTIMO then
```

```
        begin
          ESTNIVEL [II] ← - X;
          KV ← KV + 1;
          ULTIMO ← TOPO + 1;
        end;

      while I ≠ 0 do
        begin
          Y ← VERTICE [I]
          if PAI [Y] < 0 then
            begin
              TOPO ← TOPO + 1;
              ESTNIVEL [TOPO] ← Y;
              PAI [Y] ← X;
            end,
          I ← PROXELO [I];
        end;
    end;
    ESTNIVEL [ULTIMO] ← - ESTNIVEL [ULTIMO];
end;
```

IV. ALGORITMO DE CUTHILL-McKEE

IV.1. Introdução

O Algoritmo de Cuthill-McKee (CM) [4], surgiu na literatura em 1969. Desde então, tem sido considerado como metodo padrão para a solução do problema de redução de largura de banda de matrizes simetricas esparsas, em função da experiência acumulada por vários autores em problemas de elementos finitos.

Tem sido citado, frequentemente, na literatura ([5],[6],[7],[8],[9],[14]) e posteriores modificações lhe foram sendo acrescentadas, como a sugerida por Alan George [5], em 1971, que deu origem ao esquema conhecido como Algoritmo de Cuthill-McKee Reverso (RCM). A diferença entre os dois é que o esquema RCM possui um passo adicional em que a numeração obtida pelo esquema CM é invertida.

No decorrer desse capítulo é mostrado que o algoritmo RCM produz uma numeração que gera a mesma largura de banda que se obtem através do algoritmo CM, mas com um perfil que pode ser significativamente menor. Wai-Hung Liu e Andrew H. Sherman [7] demonstraram que este perfil nunca é maior que o obtido pelo metodo CM original.

O proposito inicial de Elizabeth Cuthill em [4] era, dada uma matriz simetrica A e um grafo $G(A) = (V,E)$ associado, definir um esquema que gerasse uma numeração dos vertices de V (ou, em outras palavras, que gerasse uma permutação da matriz A) que, usada como ponto de partida em metodos iterativos, como o de Rosen [10], levasse a obtenção de uma numeração correspondente a uma largura de banda mínima ou, pelo menos, próxima da mínima.

Tais metodos iterativos se mostraram pouco eficazes, tanto por consumir um grande tempo de execução como pela pequena (ou nenhuma) redução adicional da largura de banda conseguida por eles, a partir da numeração produzida pelo algoritmo CM, e foram abandonados. Desde então o algoritmo R(CM) passou a ser a estrategia de numeração mais utilizada.

Esse algoritmo está descrito como um processo de numeração dos vertices da estrutura de grafo G(A) = (V,E). Uma vez determinada uma numeração $\phi(G)$ dos vertices de V, pode-se definir uma matriz de permutação $P = (p_{ij})$, tal que

(IV-1) $\quad \forall j \leq n \ (\phi(x_j) = i \Rightarrow p_{ij} = 1 \text{ e } p_{ik} = 0, \ \forall \ k \neq j)$

Lema IV.1.: Dados uma matriz $A = (a_{ij})$, n x n, e o grafo associado G(A) = (V,E), sejam $\phi(G)$ uma numeração dos vertices de V, e $P = (p_{ij})$ uma matriz de permutação, n x n, conforme definida na equação (IV-1). Então a matriz $PAP^t = (a'_{ij})$ é tal que:

(i) $\forall i \leq n \ (\phi(x_i) = r \Rightarrow a'_{rr} = a_{ii})$

(ii) $\forall i,j \leq n \ (\phi(x_i) = r \text{ e } \phi(x_j) = s \Rightarrow a'_{rs} = a_{ij} \text{ e } a'_{sr} = a_{ji})$

Demonstração: A demonstração é decorrência imediata do lema I.1. e da expressão (IV-1).

Embora esse esquema se aplique a grafos com qualquer número de componentes conexos, o grafo G(A) será considerado conexo, ou, equivalentemente, a matriz A será suposta irredutível.

Obviamente, se o grafo tiver mais de um componente conexo, o algoritmo poderá ser, independentemente, empregado em cada componente, de forma que os vertices de cada um deles sejam rotulados consecutivamente.

Na secção IV.2. é discutido um esquema simplificado baseado no metodo de Cuthill-McKee. Nesse esquema (I) o vertice inicial (a ser rotulado com o número 1) é escolhido arbitrariamente. Esse algoritmo I foi baseado nas formas do esquema CM descritas em [4] e [7].

Na secção IV.3. é discutido o problema da escolha do vertice inicial, sendo apresentado o algoritmo CM(II) em sua forma mais completa, conforme descrito em [8], porém com as simplificações decorrentes da inclusão de um passo inicial de ordenação da estrutura de adjacencias do grafo.

Na secção IV.4. é discutido o Algoritmo de Cuthill-McKee Reverso (III) que consiste em se acrescentar ao algoritmo CM um passo final de inversão da numeração $\phi(G)$ obtida.

IV.2. Esquema Simplificado

Nessa seção o algoritmo CM é apresentado numa forma simplificada, não se discutindo o problema da escolha do vertice inicial, que é aqui determinado arbitrariamente.

Na analise de complexidade desse algoritmo é mostrada as vantagens de se incluir no esquema um passo de ordenação da estrutura de níveis antes do passo de numeração dos vertices de V.

IV.2.1. Esquema CM Simplificado

Esse algoritmo, escolhe um vertice inicial v em V, gera uma estrutura de níveis (algoritmo A3) $N_v(G) = \{N_1(v), N_2(v), \ldots N_{k(v)}\}$ e numera os vertices em cada nível (tomados do nível 1 para o nível k(v)), consecutivamente, conforme descrito adiante.

O grafo G é representado na forma de uma estrutura de adjacencias.

A estrutura de níveis $N_v(G)$ produzida pelo algoritmo A3 é armazenada em um vetor de n elementos, sendo a mudança de nível indicada por uma marca ("tag"): o primeiro vertice de cada nível é representado com o sinal negativo.

Dois outros vetores guardam o pai de cada vertice relativo a estrutura de níveis $N_v(G)$, e a numeração $\phi(G)$ final.

Algoritmo I: Esquema simplificado de numeração baseado no metodo de Cuthill-McKee.

Entrada: Grafo G = (V,E), representado na forma de uma estrutura de adjacencias.

Saída: Uma numeração $\phi(G)$ dos vertices de V.

Metodo:

1. Escolher* arbitrariamente um vertice v em V.

2. Gerar uma estrutura de níveis $N_v(G) = \{N_1, N_2, \ldots, N_{k(v)}\}$ calculando o vetor pai relativo a N_v, pelo algoritmo A3.

3. Numeração dos vertices de V.

   (i) Rotular o vertice v com 1.
   Fazer i <— 1.

   (ii) Fazer i <— i + 1.
   Numerar os vertices no nível N_i (i > 1), como se segue:
   Considerar cada vertice w em N_{i-1}, na ordem em que foram rotulados e rotular os vertices em N_i, cujo pai é w, consecutivamente, em ordem crescente de grau.

   (iii) Repetir o passo 3ii até que todos os vertices em $N_{k(v)}$ tenham sido rotulados.

Como observado em [4], a geração de uma numeração para os vertices de V, pelo esquema acima, corresponde a produção de uma árvore geradora pelo metodo de busca em largura ("breadth-first").

Um exemplo simples, descrevendo o funcionamento desse algoritmo, é mostrado abaixo.

Exemplo IV-1: Considere a matriz A, figura (III-5a) e o grafo G(A) associado, na figura (III-5b).

Para esse grafo o algoritmo I procede como se segue:

Passo 1: Suponha que o vertice a tenha sido escolhido como vertice inicial. Assim, v = a.

---

* Esse vertice deve, em geral, estar localizado numa extremidade do grafo e, se possível, ter poucos vertices adjacentes a ele. Porém, nessa forma simplificada do algoritmo tais problemas não serão considerados.

Passo 2: A estrutura de níveis de G, com raiz em a é então gerada: $N_a(G) = \{\{a\},\{b,d,i\},\{c,f,e,j\},\{g,h\}\}$.
$k(a) = 4$ e $w(N_a) = 4$.

Passo 3i: $\delta(a) = 1$.

Passo 3ii: $\delta(b) = 2$, $\delta(d) = 3$ e $\delta(i) = 4$.

Passo 3ii: $\delta(c) = 5$, $\delta(f) = 6$, $\delta(e) = 7$ e $\delta(j) = 8$.

Passo 3ii: $\delta(g) = 9$ e $\delta(h) = 10$.

Passo 3iii: O algoritmo para, uma vez que todos os vertices de $N_{k(v)}(v)$ foram rotulados.

A figura (IV-1) mostra a estrutura de níveis $N_a(G)$ gerada no passo 2 (algoritmo A3), representada na forma do grafo G.

A figura (IV-2) mostra, passo a passo, a numeração dos vertices de V.

A figura (IV-3) apresenta a matriz PAP^t, denotada por A_c, obtida de A pela aplicação do algoritmo I. Como pode ser notado, com a reordenação das linhas e colunas de A, a largura de banda foi significativamente reduzida de 8 para 5, embora o perfil tenha aumentado de 26 para 27.

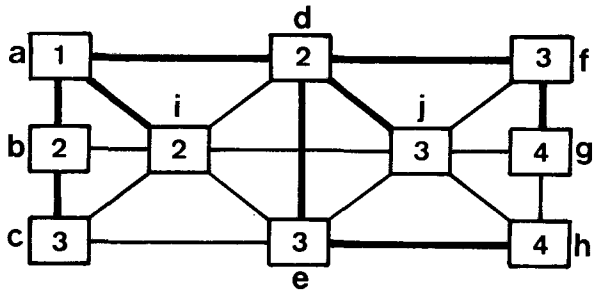


Figura IV-1: Representação da estrutura de níveis $N_a(G)$ gerada pelo algoritmo A3.

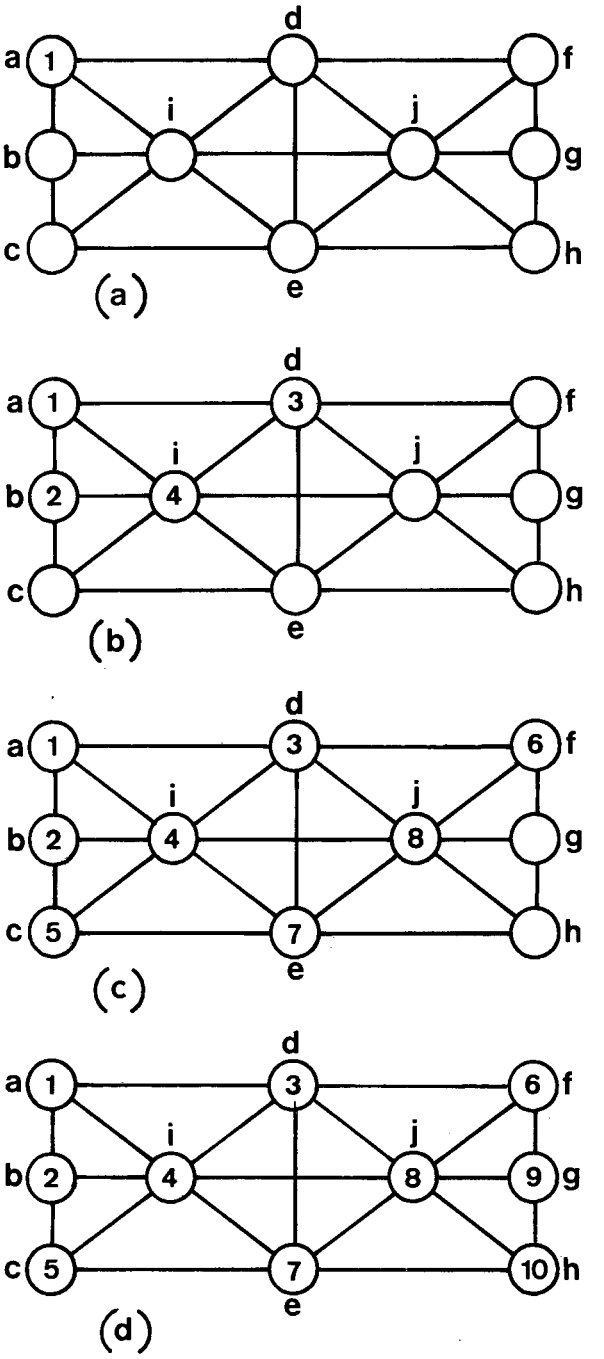


Figura IV-2: Numeração dos vertices de V: (a), (b), (c) e (d) mostram a numeração dos vertices nos níveis 1, 2, 3 e 4 de $N_a(G)$, respectivamente.

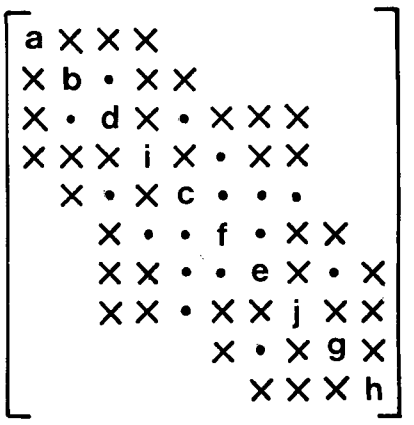


Figura IV-3: Matriz A_c obtida pela numeração CM dos vertices de G(A).

Uma importante propriedade da matriz A_c obtida pela aplicação do algoritmo CM é formalizada no lema IV.2. abaixo, sendo posteriormente utilizada na demonstração de resultados relativos ao esquema RCM.

Lema IV.2.: Seja A_c a matriz obtida pela reordenação das linhas e colunas de A, segundo a numeração $\zeta(G)$. A_c é tal que:

(i) $i \leq j \Rightarrow \ell_i(A_c) \leq \ell_j(A_c)$, e

(ii) $i > 1 \Rightarrow \ell_i(A_c) < i$

Demonstração: Seja $V_c = \{v_1, v_2, \ldots, v_n\}$, o conjunto ordenado de vertices de $G(A_c)$.

(i) $i < j \Rightarrow v_i$ precede v_j em V_c, e então

$\ell_i(A_c) \equiv \min \{k : v_k \in adj(v_i)\} \leq \min \{k : v_k \in adj(v_j)\} \equiv \ell_j(A_c)$ pois $\ell_j(A_c) < \ell_i(A_c)$ contraria a natureza do algoritmo CM (passo 3 do algoritmo I) que numera primeiro os vertices adjacentes aos vertices de menores rotulos.

(ii) Se existir algum $i > 1$, tal que $\ell_i = i$, então de (i) resulta que $\forall j > i : \ell_i = i \leq \ell_j$, e a matriz A_c seria redutível, o que contraria a hipotese.

Logo, $i > 1 \Rightarrow \ell_i(A_c) < i$.

A propriedade (i) do lema IV.2. é denominada propriedade de envelope monótono.

IV.2.2. Análise de Complexidade

No passo 3ii, se, para cada w em V, for feita uma ordenação dos vertices cujo pai é w, teríamos 0(n) ordenações, cada uma das quais requerendo, pelo metodo da raiz, um tempo de execução de 0(n) conforme foi visto no capítulo III.

Então, antes do passo de numeração dos vertices, deve ser feita uma ordenação de toda a estrutura de níveis baseada na ordem lexicográfica seguinte, o que requer apenas um tempo de 0(n).

A cada vertice x de V é associado um par ordenado (grau(x),pai(x)), sendo denotado por S o conjunto dos n pares ordenados assim construidos.

Define-se, então, a seguinte relação de ordem.

Definição IV.1.: Uma ordem lexicografica parcial sobre S, é, uma ordem R, tal que

(i) $\forall i_1,i_2,j$ $\begin{cases} j \neq 0 \Rightarrow (i_1,0)\ R\ (i_2,j) \\ i_1 < i_2 \Rightarrow (i_1,j)\ R\ (i_2,j) \end{cases}$

(ii) dados (i_1,j_1) e (i_2,j_2) associados, respectivamente aos vertices x e y e, tais que $j_1,j_2 \neq 0$, seja (i_3,j_3) associado ao pai(x) e (i_4,j_4) associado ao pai(y), então:

$j_1 \neq j_2$ e $(i_3,j_3)\ R\ (i_4,j_4) \Rightarrow (i_1,j_1)\ R\ (i_2,j_2)$

O passo 3 do algoritmo I é, então, implementado como se segue:

3'. Numeração dos vertices de V:

(i) Associar a cada vertice x de V o par ordenado (grau(x),pai(x)) e ordenar os vertices de V segundo a ordem lexicografica R definida acima, armazenando o resultado no vetor ord*.

---

* Evidentemente, como pai(v) = 0 tem-se ord(1) = v

(ii) Definir a numeração $\phi(G)$ como se segue:

for i <- 1 until n do ϕ[ord[i]] <- i;

Após a ordenação (passo 3'i) os pais dos vertices não são mais necessários e esse mesmo espaço de memória pode ser usado para armazenar os vertices já ordenados (vetor ord).

Espaço de Memória

Além dos (n + 4m) elementos requeridos pela estrutura de adjacencias de G, são utilizados ainda sete vetores de n elementos:

- para guardar os pais dos vertices de V relativos a N_V.
- para guardar os graus dos vertices de V.
- para guardar a numeração $\phi(G)$, e
- quatro vetores adicionais durante o processo de ordenação dos vertices.

Portanto, o espaço de memória requerido é de (8n + 4m) que é de O(n + m).

Tempo de Execução

O passo 1 consome um tempo de O(c), pois o vertice inicial é escolhido arbitrariamente.

O passo 2 consome um tempo de O(n + m), como já foi visto na análise do algoritmo A3.

O passo 3i requer um tempo de O(n + m) para calcular o grau de cada vertice, e um tempo de O(n) para dispor os vertices de V segundo a ordem R (definição IV.1.).

Essa ordenação (metodo da raiz) requer a criação de n listas, de tal forma que os vertices de grau i sejam colocados na i-ésima lista.

Em seguida os vertices nessas listas são transferidos para outras n listas de maneira que os vertices de pai x_j sejam colocados na j-ésima lista, para todo x_j em V. Essas n listas são, então, percorridas gerando-se um vetor com os vertices de V ordenados.

O passo 3ii requer um tempo de O(n), pois basta percorrer o vetor com os vertices já ordenados, gerando a

numeração $\phi(G)$.

Portanto, a complexidade global, em tempo de execução desse algoritmo é de $O(n + m)$.

IV.2.3. Formulação Tipo Algol

Estrutura de Dados Utilizada

O grafo G é representado na forma de uma estrutura de adjacencias, com listas encadeadas.

O vetor PAI guarda o pai de cada vertice de V relativo a $N_v(G)$.

O vetor GRAU guarda o grau de cada vertice de V.

O vetor ORD (que poderia ser o próprio vetor PAI) guarda os vertices dispostos segundo a ordem R definida em IV.1.

Para as n listas são usados dois vetores TOPO1 e TOPO2 para guardar os topos das listas e os vetores Q1 e Q2 para as listas propriamente ditas.

Finalmente, a numeração $\phi(G)$ é armazenada no vetor F.

Descrição Tipo Algol do Algoritmo III

```
begin
  comment 1. Escolha do vertice inicial.
  V <- x1;
  comment 2. Geração da estrutura de níveis N_V(G).
  executar o algoritmo A3, gerando em PAI os pais dos vertices
  relativos a N_V(G).
  comment 3i. Ordenação da estrutura de níveis.
  construir o vetor GRAU com o grau de cada vertice de V (vide
  algoritmo A2).
  for I <- 1 until N do
    begin
      TOPO1 [I] <- ∅;
      TOPO2 [I] <- -1;
    end;
```

```
  for I <- 1 until1 N do
    begin
      J <- GRAU [I];
      Q1 [I] <- TOPO1 [J];
      TOPO1 [J] <- I;
    end;
  for I <- N step -1 until1 1 do
    begin
      L <- TOPO1 [I];
      while L > 0 do
        begin
          J <- PAI [L];
          Q2 [L] <- TOPO2 [J];
          TOPO2 [J] <- L;
          L <- Q1 [L];
        end;
    end;
  J <- ∅;
  I,II <- ∅:
  while I < N do
    begin
      L <- TOPO2 [J];
      while L > 0 do
        begin
          I <- I + 1;
          ORD [I] <- L;
          L <- Q2 [L];
        end;
      II <- II + 1;
      J <- ORD [II];
    end;
  comment 3ii. Numeração dos vertices de V.
  for I <- 1 until1 N do F[ORD[I]] <- I;
end;
```

IV.3. Problema da Escolha do Vertice Inicial

Dados um grafo G = (V,E) e uma estrutura de níveis $N_v(G)$, a largura de banda, relativa a uma numeração

$\delta(G)$, produzida por um esquema que rotula os vertices em cada nível de N_v com inteiros consecutivos (como o algoritmo CM), depende da largura de N_v, conforme pode ser deduzido do teorema IV.3. seguinte.

Evidentemente, um limite inferior para a medida da largura de banda, relativa a qualquer numeração δ dos vertices de V, é dada pelo menor inteiro maior ou igual a $d_{max}/2$, onde d_{max} é o grau máximo dos vertices de V.

Teorema IV.3.: Sejam um grafo G(A) = (V,E), uma estrutura de níveis $N_v(G)$ de profundidade k(v), e uma numeração $\delta : \{1,2,\ldots,n\} \rightarrow V$ tal que

(IV-2) $\forall i,j \leq n \; (i < j \Rightarrow \forall x_r \in N_i(v), \forall x_s \in N_j(v)(\delta(x_r) < \delta(x_s))$

Então:

(i) $B_\delta(G) \leq 2w(N_v) - 1$

(ii) $P_\delta(G) \leq \sum_{p=2}^{k(v)} w(N_p) \; [w(N_{p-1}) + (w(N_p) - 1)/2]$

(iii) $B_\delta(G) \geq w(N_v)$

(iv) $P_\delta(G) \geq \sum_{p=2}^{k(v)} w(N_p) \; [w(N_p) + 1]/2$

Demonstração: Seja N_p o p-ésimo nível de $N_v(G)$, $p > 1$.

(i) Sejam x_r o vertice de menor rotulo em N_{p-1}, e x_s o vertice de maior rotulo em N_p.
No pior caso, tem-se $x_s \in adj(x_r)$, pois nesse caso:
$\delta(x_s) - \delta(x_r) = w(N_{p-1}) + w(N_p) - 1$.
Assim, em geral, tem-se
$\forall x_i \in N_{p-1}, \forall x_j \in N_p \; (\delta(x_j) - \delta(x_i) \leq w(N_{p-1}) + w(N_p) - 1)$
Como $B_\delta(G) = \max \{|\delta(x_j) - \delta(x_i)| : x_j \in adj(x_i)\}$, tem-se

(IV-3) $B_\delta(G) \leq \max \{w(N_{p-1}) + w(N_p) - 1 : 1 < p \leq k(v)\} \leq$

$\leq 2w(N_v) - 1$

(ii) Seja x_r o vertice de menor rotulo em N_{p-1}.
No pior caso, tem-se $x_s \in adj(x_r)$, para todo x_s em N_p, pois nesse caso:

$$\ell_{\delta(x_s)} = w(N_{p-1}) + |\{x_t \in N_p : \delta(x_t) \leq \delta(x_s)\}| - 1.$$

Assim, em geral, tem-se

$$\forall x_i \in N_{p-1}, \forall x_j \in N_p \ (\ell_{\delta(x_j)} \leq w(N_{p-1}) + |\{x_t \in N_p : \delta(x_t) < \delta(x_j)\}| - 1.$$

A contribuição de cada nível N_p ao perfil $P_\delta(G)$ é, então, menor ou igual a

$$\sum_{j=1}^{w(N_p)} w(N_{p-1}) + j - 1 = w(N_p)\,[w(N_{p-1}) + w(N_p) - 1)/2] \qquad \text{(IV-4)}$$

E, finalmente, aplicando a equação (IV-4) na definição de $P_\delta(G)$, tem-se

$$P_\delta(G) \leq \sum_{p=2}^{k(v)} w(N_p)\,[w(N_{p-1}) + (w(N_p) - 1)/2] \qquad \text{(IV-5)}$$

(iii) Seja x_s o vertice de maior rotulo em N_p, e x_r o vertice de maior rotulo em N_{p-1}.
No melhor caso, x_r é o único vertice de N_{p-1} adjacente a x_s, pois, nesse caso:

$$\delta(x_s) - \delta(x_r) = w(N_p)$$

Assim, em geral, tem-se

$$\forall x_i \in N_{p-1}, \forall x_j \in N_p \ (\delta(x_j) - \delta(x_i) \geq w(N_p))$$

Como, pelo menos para algum $p > 1$, $w(N_p) = w(N_v)$, tem-se

$$B_\delta(G) \geq w(N_v) \qquad \text{(IV-6)}$$

(iv) Seja x_r o vertice de maior rotulo em N_{p-1}.
No melhor caso, tem-se $x_s \in adj(x_r)$, para todo $x_s \in N_p$, pois nesse caso:

$$\ell_{\delta(x_s)} = |\{x_t \in N_p : \delta(x_t) \leq \delta(x_s)\}| \qquad \text{(IV-7)}$$

A contribuição de cada nível N_p ao perfil $P_\delta(G)$, é então maior ou igual a:

$$(IV\text{-}8) \qquad \sum_{j=1}^{w(N_p)} j = w(N_p)\ (w(N_p) + 1)/2$$

E, finalmente, aplicando a equação (IV-8) na definição de $P_{\delta}(G)$, tem-se

$$(IV\text{-}9) \qquad P_{\delta}(G) \geq \sum_{p=2}^{k(v)} w(N_p)\ [w(N_p) + 1]/2$$

Como pode ser observado pela demonstração desse teorema, as propriedades (i) e (ii) são válidas para qualquer estrutura de níveis com ou sem raiz ($w(N_1) \geq 1$). A propriedade (iii) é válida também para as estruturas de níveis sem raiz ($w(N_1) > 1$) em que $\exists p > 1$ tal que $w(N_p) = w(N)$. Para estruturas de níveis sem raiz a propriedade (iv) seria

$$(IV\text{-}10) \qquad P_{\delta}(G) > \sum_{p=2}^{k(v)} w(N_p)\ [w(N_p) + 1]/2$$

As propriedades (i) e (iii) do teorema IV.3. indicam que quanto menor for a largura da estrutura de níveis menor pode ser a largura de banda obtida. Cuthill em 1969 já concluira isso e em [4] procura definir criterios (empíricos) para a escolha do vertice inicial de forma a encontrar uma estrutura de níveis mais estreita.

A idéia basica discutida em [4] é que as estruturas de níveis mais estreitas estão, em geral, entre aquelas com raizes em vertices de grau baixo.

Cuthill determinou experimentalmente a formula

$$(IV\text{-}11) \qquad d_{min} \leq d \leq d_{min} + \alpha\, d_{max}$$

onde d_{min} e d_{max} são os graus mínimo e máximo, respectivamente, dos vertices de V e $\alpha \simeq 0.5$. E seu algoritmo seria, então, executado para cada vertice em V de grau d, sendo escolhida como numeração final aquela que produzisse a menor largura de banda.

Mais tarde, conforme citado em [8] chegou-se a formula

$$(IV\text{-}12) \qquad d \leq \max \{\min \{(d_{max} + d_{min})/2,\ d_{med} - 1\},\ d_{min}\}$$

passando o algoritmo CM a produzir estruturas de níveis para todo vertice em V de grau d.

d_{med} indica o grau mediano de G, isto é, o grau tal que a metade dos vertices de V tem grau menor ou igual a d_{med}.

$p = \min \{(d_{max} + d_{min})/2, d_{med} - 1\}$ procura garantir que sejam escolhidos menos da metade dos vertices de V e $\max \{p, d_{min}\}$ garante que, pelo menos, um vertice é escolhido.

Para grafos em que $d_{max} = d_{min}$, evidentemente, seriam geradas estruturas de níveis com raiz em cada um dos vertices de V.

IV.3.1. Esquema de Cuthill-McKee

Pelo teorema III.3., se a estrutura de adjacencias do grafo estiver previamente ordenada, a estrutura de níveis produzida pelo algoritmo A3 estará ordenada, evidentemente satisfazendo a relação de ordem R (definição IV.1.).

Portanto, o algoritmo CM, apresentado nessa seção, possui um passo inicial em que o algoritmo A1 é executado para ordenar a estrutura de adjacencias do grafo, e para cada vertice u de grau d o algoritmo I é executado para gerar a numeração $\delta(G)$ relativa a estrutura de níveis com raiz em u. O algoritmo I é considerado aqui sem o passo 3'i, sendo o vetor ORD o próprio vetor ESTNIVEL gerado pelo algoritmo A3, porém sem as marcas de mudança de nível. O vetor PAI gerado pelo algoritmo A3 é usado para calcular a largura de banda.

São utilizados, a estrutura de adjacencias do grafo, dois vetores para armazenar a última estrutura de níveis gerada e seu vetor pai. Dois vetores para armazenar duas numerações $\delta(G)$ relativas a estruturas de níveis, além do vetor de grau calculado pelo algoritmo A1.

Algoritmo II: Algoritmo CM

Entrada: Grafo G = (V,E), representado na forma de uma estrutura de adjacencias.

Saída: Uma numeração $\phi(G)$ dos vertices de V.

Metodo:

1. Ordenar a estrutura de adjacencias de G, pelo algoritmo A1.
2. Determinar* o conjunto S dos vertices $\underline{x}$ de V, tais que: grau (x) $\leq$ max {min {$(d_{max} + d_{min})/2$, $d_{med} - 1$}, d_{min}}
3. Executar o algoritmo I para um vertice $\underline{v}$ de S. Retirar $\underline{v}$ de S. Calcular a largura de banda $B_{\phi}(N_v(G))$.
4. Executar o algoritmo I para um vertice $\underline{u}$ de S. Retirar $\underline{u}$ de S. Calcular a largura de banda $B_{\phi}(N_u(G))$.
5. Se $B_{\phi}(N_u(G)) < B_{\phi}(N_v(G))$, então fazer v <— u. Repetir o passo 4.
6. O algoritmo para quando S for vazio.

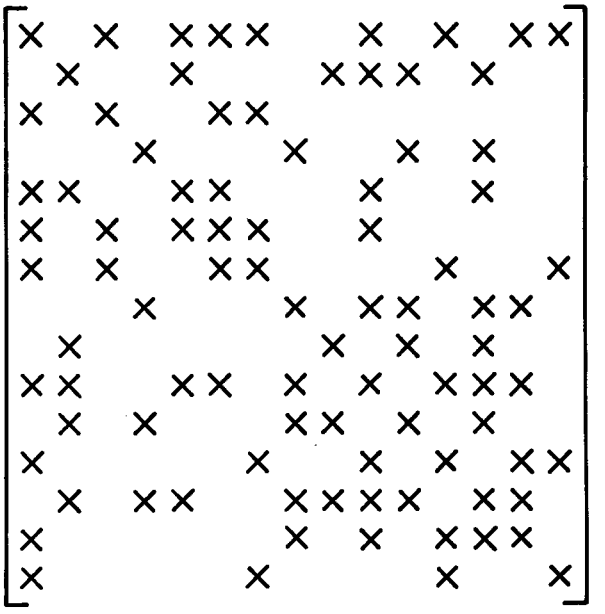


Figura IV-4: Um exemplo de matriz para ilustrar o funcionamento do algoritmo II (CM).

---

* Algumas implementações |24| desse algoritmo permitem, também, que S seja fornecido como parâmetro.

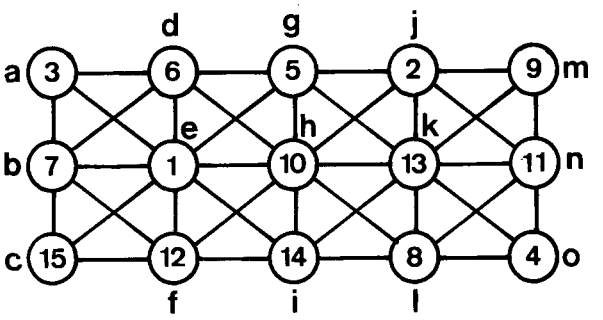


Figura IV-5: Grafo G(A) associado a matriz A da figura (IV-4).

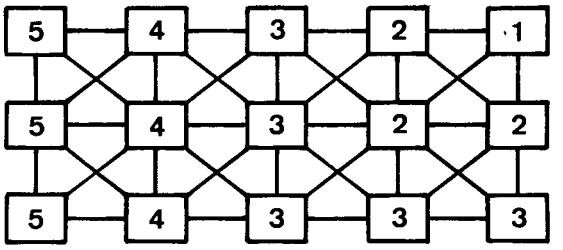


Figura IV-6: Estrutura de níveis com raiz em m, gerada pelo algoritmo A3.

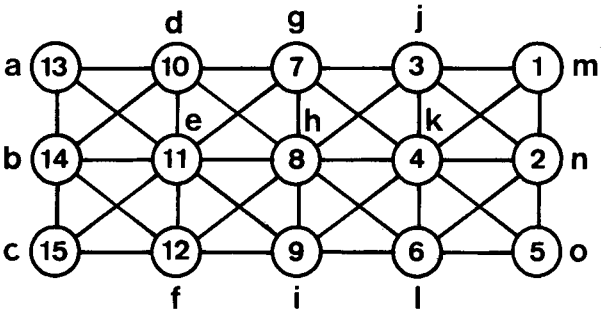


Figura IV-7: Numeração $\phi(N_m(G))$ dos vertices de G produzida pelo algoritmo II.

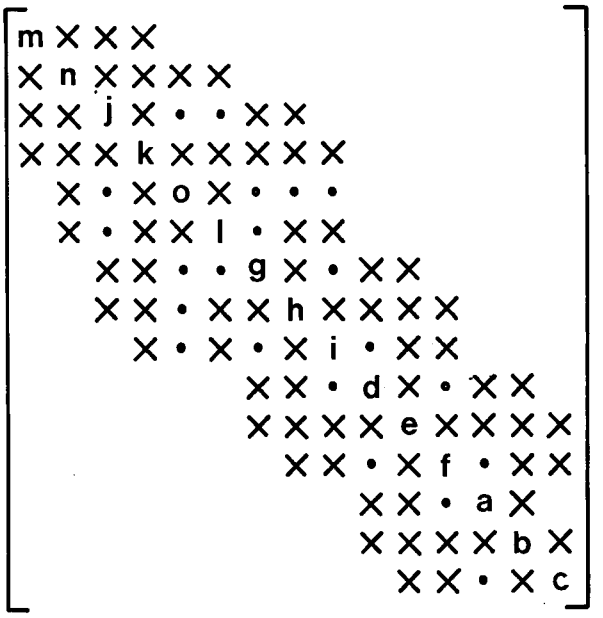


Figura IV-8: Matriz A_c obtida pela aplicação do algoritmo II ao grafo G(A) da figura (IV-5). $B(A_c) = 5$ e $P(A_c) = 49$.

Exemplo IV-2: Considere a matriz A, figura (IV-4) e o grafo G(A) associado, na figura (IV-5).
Nesse caso: $d_{max} = 8$, $d_{min} = 3$ e $d_{med} = 5$ e, portanto, $S = \{a, c, m, o\}$.

Como as estruturas de níveis com raizes nos vertices em S são todas semelhantes, devido a forma do grafo, apenas a estrutura de níveis com raiz em m está representada na figura (IV-6). E, desde que as larguras de banda obtidas são todas iguais, o algoritmo IV para, por exemplo, com $\delta(N_m(G))$ que está mostrada na figura (IV-7).

A figura (IV-8) mostra a matriz A_c, obtida de A pela aplicação do algoritmo II. Como pode ser notado, a largura de banda obtida é 5, e o perfil 49.

IV.3.2. Análise de Complexidade

Espaço de Memória

Além das (n + 4m) posições de memória requeridas pela estrutura de adjacencias do grafo são necessários,

ainda, cinco vetores de n elementos:

- para armazenar a última estrutura de níveis $N_u(G)$ gerada, e o pai de cada vertice relativo a $N_u(G)$,
- para armazenar as numerações $\phi(G)$, relativas a duas estruturas de níveis, e
- para armazenar os graus dos vertices de V.

O algoritmo A1 (passo 1) requer ainda (n + 2m) posições de memória para as n pilhas a serem utilizadas na ordenação da estrutura de adjacencias.

O algoritmo I (passo 3 e 4), desde que seu passo 3'i foi eliminado, não requer espaço adicional, e o espaço total necessário é, então, de (7n + 6m), que é de O(n + m).

Tempo de Execução

Os passos 1, 3 e 4, como já foi visto anteriormente, requerem tempos de O(n + m).

O passo 2 requer um tempo de O(n), pois o vetor contendo os graus dos vertices de V é calculado no passo anterior.

Como o algoritmo I é executado para cada vertice em S e |S| é de O(n), a complexidade global, em tempo de execução desse algoritmo é de O(n(n + m)).

IV.3.3. Formulação Tipo Algol

Estrutura de Dados

Para armazenar as estruturas de níveis é utilizado o vetor ESTNIVEL [N].

Analogamente, a matriz F [2,N] contém as numerações dos vertices de V, relativas a duas estruturas de níveis, e o vetor BANDA [2] contém as respectivas larguras de banda.

A matriz F é usada como área auxiliar no processo de determinação de d_{med}, e o conjunto S não ocupa espaço adicional de memória.

Os vetores GRAU e PAI construidos nos algoritmos A1 e A3 respectivamente, são também usados.

Descrição Tipo Algol do Algoritmo II

```
begin
  comment 1. Ordenação da estrutura de adjacencias.
  Executar o algoritmo A1.
  comment 2. Determinação do grau máximo dos vertices de S.
  DMAX,DMIN <- GRAU [1];
  F [1,*] <- ∅;
  for I <- 2 until N do
    begin
      J <- GRAU [I];
      if DMAX < J
        then DMAX <- J;
        else if DMIN > J then DMIN <- J;
      F [1,J] <- F [1,J] + 1;
    end;
  for I <- 1 step 1 while F [1,I] < N/2 do
    F [1,I + 1] <- F [1,I] + F [1,I + 1];
  DMED <- I;
  D <- MAX (MIN ((DMAX + DMIN)/2, DMED - 1), DMIN);
  for I <- 1 until N while GRAU [I] > D do;
  comment 3. Geração da estrutura de níveis N_V(G).
  V <- 1;
  U <- 2;
  Executar o algoritmo I, gerando em ESTNIVEL a estrutura de
  níveis com raiz no vertice x_I.
  BANDA [V] <- ∅;
  for J <- 2 until N do
    BANDA [V] <- MAX (BANDA [V], F [V,J] - F [V,PAI [J]]);
  for I <- I + 1 until N do
    begin
      if GRAU [I] <= D then
        begin
          comment 4. Geração da estrutura de níveis N_u(G).
          executar o algoritmo I, gerando em ESTNIVEL a estru-
          tura de níveis com raiz no vertice x_I.
          BANDA [U] <- ∅;
          for J <- 2 until N do
            BANDA [U] <- MAX (BANDA [U], F [U,J] - F [U,PAI [J]])
```

```
            comment 5. Comparação das larguras de banda.
            if BANDA [U] < BANDA [V] then
              begin
                V <- U;
                U <- 3 - V;
              end;
          end;
      end;
end;
```

IV.4. Esquema Reverso de Cuthill-McKee (RCM)

Alan George, em [5], sugere que a numeração δ obtida pelo esquema CM seja invertida, da seguinte maneira:

(IV-13) $\quad \delta(x_i) = n - \delta(x_i) + 1 \quad , \quad 1 \leq i \leq n$

Recentemente, Sherman e Liu [7] demonstraram que a largura de banda obtida pela numeração CM é a mesma que a obtida pela numeração RCM, e que o perfil obtido pela segunda numeração nunca é maior que o obtido pela primeira, podendo, inclusive, ser significativamente menor.

Devido a sua importancia as principais conclusões de [7] são aqui apresentadas, na seção IV.4.2.

IV.4.1. Esquema RCM

Para se obter uma numeração RCM (IV-13) basta acrescentar ao algoritmo II um passo final de inversão da numeração obtida.

Algoritmo III: Algoritmo RCM.

Entrada: Grafo G(A) = (V,E), representado na forma de uma estrutura de adjacencias.

Saída: Uma numeração $\delta(G)$ dos vertices de V.

Metodo:

. Executar o algoritmo II, produzindo uma numeração $\delta(G)$ dos vertices de V.

2. Inverter a numeração $\phi(G)$, fazendo:
   $\phi(x_i) \leftarrow n - \phi(x_i) + 1 \quad , \quad 1 \leq i \leq n.$

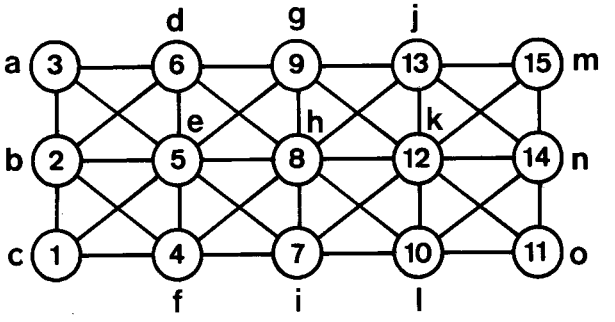


Figura IV-9: Numeração $\phi(N_m(G))$ produzida pelo algoritmo RCM, para o grafo da figura (IV-5).

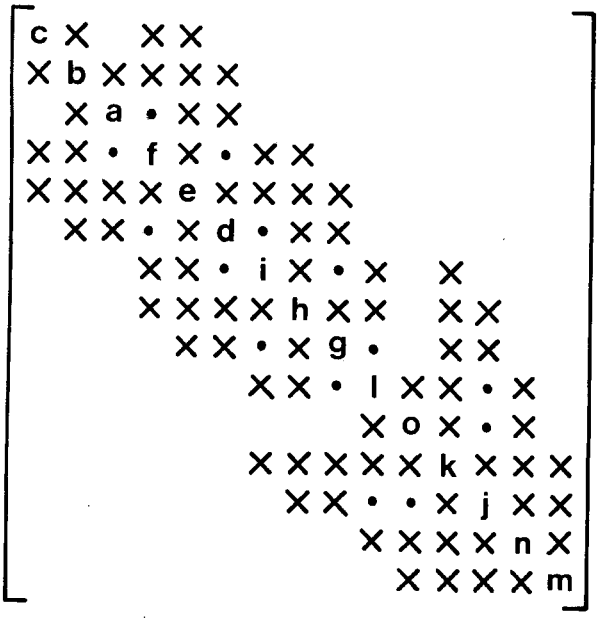


Figura IV-10: Matriz A_r obtida pela aplicação do algoritmo II ao grafo G(A) da figura (IV-5).
$B(A_r) = 5$ e $P(A_r) = 45$.

Exemplo IV-3: Considere o grafo G(A) na figura (IV-5). A figura (IV-7) mostra a numeração obtida no passo A (numeração CM) do algoritmo III e na figura (IV-9) essa numeração é invertida (passo B).

A matriz obtida pela aplicação do algoritmo RCM ao grafo G(A) é denotada por A_r estando mostrada na figura (IV-10). Como pode ser observado, o perfil é menor que o obtido pela numeração CM (figura (IV.8)).

Obviamente, esse algoritmo não requer nenhum espaço de memória adicional, além do requerido pelo algoritmo II, portanto, o espaço necessário é de $0(n + m)$.

A complexidade em tempo de execução do passo B é de $0(n)$ e, consequentemente, o algoritmo III requer um tempo de $0(n\ (n + m))$.

IV.4.2. Comparação dos Esquemas CM e RCM

Dado um grafo G(A) associado a uma matriz A simetrica, denota-se por A_c e A_r as matrizes obtidas de A a partir das numerações CM e RCM, respectivamente. $G(A_c)$ e $G(A_r)$ são seus grafos associados.

Os resultados abaixo, extraidos de [7], mostram que o perfil da matriz A_r nunca é maior que o perfil da matriz A_c, sendo menor quando o grafo $G(A_c)$ possuir a propriedade $\tilde{p}$ (definição IV.2. seguinte).

Lema IV.4.: $\text{Env}\ (A_r) = \text{Tenv}\ (A_c)$

Demonstração: Imediata, a partir das definições de Env (A) e Tenv (A), dadas no capítulo I.

Teorema IV.5.: $\text{Tenv}\ (A_c) \subseteq \text{Env}\ (A_c)$

Demonstração: Assuma que para algum $i \geq j$,

(IV-14) $\quad \{x_i, x_j\} \in \text{Tenv}\ (A_c)$ e $\{x_i, x_j\} \notin \text{Env}\ (A_c)$

Então:

(IV-15) $\quad \{x_i, x_j\} \in \text{Tenv}\ (A_c) \Rightarrow \exists\ k \geq i$ tal que $x_k \in \text{adj}\ (x_j)$, e

(IV-16) $\quad \{x_i, x_j\} \notin \text{Env}\,(A_c) \Rightarrow j < \ell_i\ (A_c) \Rightarrow \forall h \leq j\ \ x_i \notin \text{adj}(x_h)$

De (IV-15) e (IV-16), resulta que $\ell_k (A_c) < \ell_h (A_c)$, o que contraria o lema IV.2., pois $i \leq k$.

Corolario IV.6: Env $(A_r) \subseteq$ Env (A_c).

Demonstração: Do lema IV.4. e do teorema IV.5., tem-se Env (A_r) = Tenv $(A_c) \subseteq$ Env (A_c)

Corolario IV.7.: $P(A_r) \leq P(A_c)$

Demonstração: Imediata a partir do corolário IV.6.

Com o objetivo de caracterizar a classe dos grafos G(A), tais que $P(A_r) \leq P(A_c)$, define-se a seguinte propriedade $\tilde{p}$.

Definição IV.2.: Dado um grafo $G(A_c)$, associado a uma numeração CM, diz-se que $G(A_c)$ tem a propriedade $\tilde{p}$ se existirem vertices x_i, x_j e x_k, $i < j < k$, tais que $x_k \in$ adj (x_i) e para todo $\ell \geq k$, $x_\ell \notin$ adj (x_j).

O grafo $G(A_c)$ na figura (IV-7) possue a propriedade $\tilde{p}$, pois $4 < 5 < 7$, $x_7 = g$, $x_5 = o$, $x_4 = k$ e $x_7 \in$ adj (x_4) e $\forall \ell \geq 7$, $x_\ell \notin$ adj (x_5).

Teorema IV.8.: $Env(A_r) \subsetneq Env(A_r)$ se e somente se $G(A_c)$ possuir a propriedade $\tilde{p}$.

Demonstração: $G(A_c)$ tem a propriedade $\tilde{p}$ se, e somente se, existirem $i < j < k$ tais que

$\{x_k, x_i\} \in Env(A_c)$, $\{x_k, x_j\} \in Env(A_c)$ e $\{x_k, x_j\} \notin Tenv(A_c)$, que, pelo teorema IV.5. e lema IV.4., equivale a $Env(A_r) \subsetneq Env(A_c)$.

Corolário IV.9: Se $G(A_c)$ tem a propriedade $\tilde{p}$, então $P(A_r) < P(A_c)$

Demonstração: Nesse caso, pelo teorema IV.8, $Env(A_r) \subsetneq Env(A_c) \Rightarrow P(A_r) < P(A_c)$.

V. ALGORITMO DE GIBBS, POOLE E STOCKMEYER

V.1. Introdução

Nesse capítulo é discutida a estrategia de numeração desenvolvida por GIBBS, POOLE e STOCKMEYER [8], que é aqui referida como Algoritmo GPS.

Essa estrategia se compõe dos três esquemas seguintes:

(i) Esquema de determinação de vertices pseudo-periféricos que produz duas estruturas de níveis com raizes nos vertices extremidade de um pseudo-diâmetro. Esse esquema é discutido na seção V.2..

(ii) Esquema de composição de estruturas de níveis que compõe as estruturas de níveis geradas em (i) em uma nova estrutura de níveis. É discutido na seção V.3..

(iii) Esquema de numeração de vertices que rotula os vertices em V, a partir da estrutura de níveis gerada em (ii) e, das adjacencias do grafo. É discutido na seção V.4..

Nesses três esquemas, as estruturas de níveis são armazenadas na forma de um vetor NIVEL em que:

(V-1) $\quad NIVEL\ [i] = j \iff x_i \in N_j,$

onde N_j é o j-ésimo nível de referida estrutura.

Nas utilizações do algoritmo A3, de geração de estruturas de níveis, é suposto que tal algoritmo produza o vetor NIVEL em vez do vetor PAI. Tal modificação é trivial.

V.2. Determinação de Vertices Pseudo-periféricos

Como foi visto no capítulo anterior, a eficiência de metodos de ordenação depende, criticamente, da escolha adequada do vertice inicial para o processo de geração de estruturas de níveis.

A experiência tem levado vários autores [5], [8] a constatação de que estruturas de níveis de menor largura estão, geralmente, entre as de maior profundidade. Evidentemen-

te, um número maior de níveis corresponde a um menor número médio de vertices por nível, o que não implica, necessariamente, em menor largura. Entretanto, isso geralmente ocorre em problemas de elementos finitos, devido as características (esparsidade, regularidade) de suas malhas.

Tal constatação induz ao uso de vertices periféricos como raizes das estruturas de níveis. Infelizmente não se conhece um algoritmo eficiente para determinação de vertices periféricos de um grafo geral.

O esquema discutido nessa seção determina um par de vertices pseudo-periféricos (não necessariamente periféricos). Tal esquema, se baseia no lema V.1. abaixo, buscando, iterativamente, vertices u e v, tais que

(V-2) $\qquad e(u) = d(u,v) = e(v)$

Lema V.1.: Dado um grafo G(A) = (V,E) e uma estrutura de níveis $N_u(G)$ com raiz em u e profundidade k(u), então

$$w \in N_{k(u)} \Rightarrow e(w) \geq e(u)$$

Demonstração: Por definição,

(V-3) $\qquad \forall x \in V \ (e(w) \geq d(x,w))$

Por outro lado

(V-4) $\qquad e(u) = k(u) - 1 = d(u,w)$

pois, $w \in N_{k(u)}$ e, obviamente, os vertices mais distantes de u estão em $N_{k(u)}$.
De (V-3) e (V-4) resulta, para x = u:

(V-5) $\qquad e(w) \geq e(u)$

V.2.1. Esquema de Determinação de Vertices Pseudo-periféricos

Dado um grafo G(A) = (V,E) como entrada, é produzida, nesse esquema, uma estrutura de níveis $N_v(G) = \{N_1(v), N_2(v), \ldots, N_{k(v)}(v)\}$ com raiz em um vertice v qualquer, de grau mínimo. Em seguida, são geradas estruturas

de níveis com raizes nos vertices de $N_{k(v)}(v)$, até que seja produzida uma estrutura de níveis $N_r(G)$, tal que $e(r) > e(v)$, isto é, até que $k(r) > k(v)$. E, iterativamente, passam a ser examinados, então, os vertices em $N_{k(r)}(r)$. O algoritmo para quando

(V-6) $\quad \forall w \in N_{k(r)} \; (e(w) = e(r))$

São utilizados, além da estrutura de adjacencias do grafo, três vetores de n elementos, para armazenar estruturas de níveis. Dois outros vetores de n elementos armazenam os graus dos vertices e o conjunto $N_{k(v)}(v)$, com os vertices ordenados por ordem crescente de grau.

Algoritmo IV: Determinação de vertices pseudo-periféricos.

Entrada: Grafo $G(A) = (V,E)$, representado na forma de uma estrutura de adjacencias.

Saída: Duas estruturas de níveis N_v e N_u tais que v e u são vertices pseudo-periféricos de G.

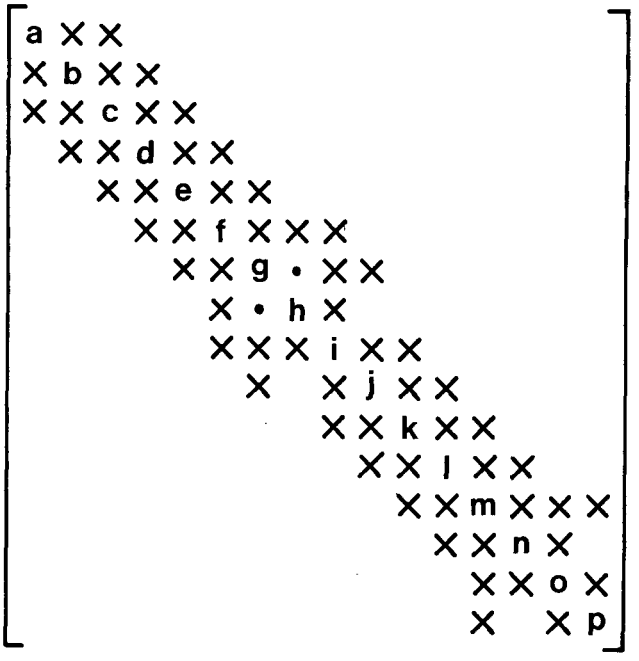
Metodo:

1. Escolher um vertice inicial v de grau mínimo.
2. Gerar, pelo algoritmo A3, a estrutura de níveis $N_v(G) = \{N_1(v), N_2(v), \ldots, N_{k(v)}(v)\}$, com raiz em v e profundidade $k(v)$.
   Calcular a largura $w(N_v)$.
3. Colocar em S os vertices de $N_{k(v)}(v)$ (para os quais não tenha sido gerada estrutura de níveis), em ordem crescente de grau.
4. Gerar, pelo algoritmo A3, a estrutura de níveis $N_r(G)$ com raiz num vertice r de menor grau em S e profundidade $k(r)$.
   Calcular a largura $w(N_r)$, e retirar r de S.
5. Se $k(r) > k(v)$, então fazer $v \leftarrow r$ e ir para o passo 3. Caso contrario* ir para o passo 4.

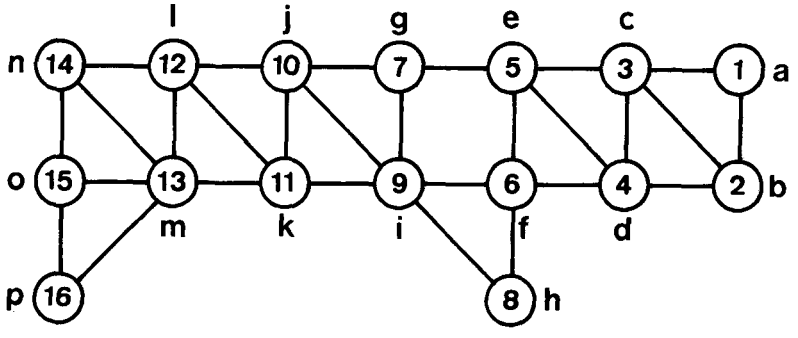
---

* Nesse ponto, as larguras das estruturas de níveis com raiz em vertices de S devem ir sendo comparadas, guardando-se em $N_u(G)$ a de menor largura dentre as já geradas.

6. O algoritmo para quando, no passo 4, S for vazio.
   Os vertices u e v são vertices pseudo-periféricos, sendo u o vertice de $N_{k(v)}(v)$ cuja estrutura de níveis tem menor largura.



(a)



(b)

Figura V-1: Um grafo simples para ilustrar o funcionamento do algoritmo IV. A parte (b) mostra o grafo G = (V,E) associado a matriz A = (a_{ij}) em (a).

Exemplo V-1: Considere a matriz A na figura (V-1a) e o grafo associado, na figura (V-1b). Para esse grafo o algoritmo IV procede como se segue:

Passo 1: É calculado o grau de cada vertice, e escolhido um vertice de grau mínimo.
Suponha v = h.

Passo 2: A estrutura de níveis com raiz em v é então gerada.
$N_v(G)$ = {{h},{f,i},{d,e,g,j,k},{b,c,l,m},{a,n,o,p}}
k(v) = 5 e $w(N_v(G))$ = 5.
Passo 3: S = {p,a,n,o}.
Passo 4: A estrutura de níveis com raiz em r = p é então gerada.
$N_r(G)$ = {{p},{m,o},{k,l,n},{i,j},{f,g,h},{d,e},{b,c},{a}}
k(r) = 8 e $w(N_r(G))$ = 3.
Passo 5: Como k(r) > k(v), v <— p.
Passo 3: S = {a}
Passo 4: A estrutura de níveis com raiz em r = a é então gerada.
$N_r(G)$ = {{a},{b,c},{d,e},{f,g},{h,i,j},{k,l},{m,n},{o,p}}
k(r) = 8 e $w(N_r)$ = 3.
Passo 5: u = a.
Pasos 6: Os vertices p e a são os vertices pseudo-periféricos determinados.

Considere, como um segundo exemplo, a matriz $A = (a_{ij})$ na figura (IV-4) e o grafo G(A) na figura (IV-5) — esse grafo é, também, utilizado para mostrar o funcionamento dos dois outros esquemas que compõem o algoritmo GPS.

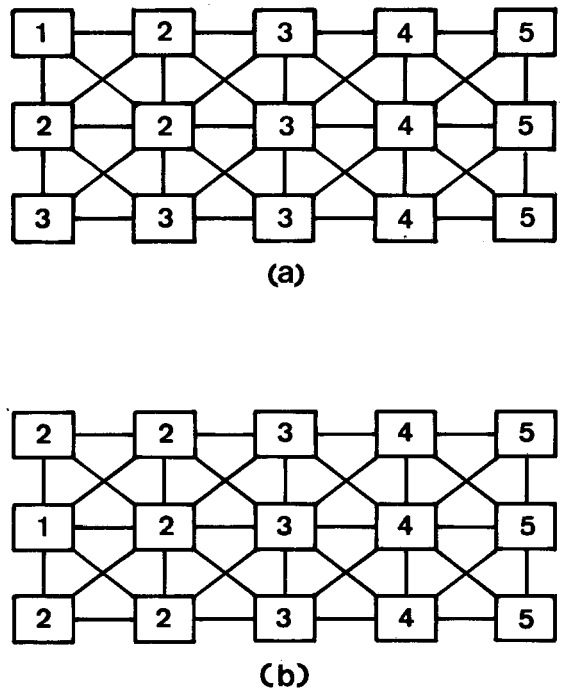


Figura V-2: Estruturas de níveis geradas no passo 4 do algoritmo IV, para o grafo G da figura (IV-5).

Suponha que no passo 1 seja escolhido v = m. Então, no passo 2, é gerada a estrutura de níveis $N_m(G)$, já mostrada na figura (IV-6), e S = {a,b,c}.

A figura (V-2) mostra as estruturas de níveis $N_a(G)$ e $N_b(G)$ que são construidas no passo 4*.

Desde que, k(m) = k(a) = k(b) = k(c) = 5, e $w(N_a) = w(N_b) = w(N_c) = 5$, o algoritmo IV para, por exemplo, com v = m e u = b.

Quando o grafo G(A) não possuir ciclos, isto é, for uma árvore, então o algoritmo IV determina, efetivamente, vertices periféricos, conforme teorema V.3. seguinte.

Lema V.2.: Sejam G = (V,E) um grafo conexo e sem ciclos e $N(G)$ uma estrutura de níveis com raiz em v e profundidade k(v). Então, todo diâmetro de G tem pelo menos uma extremidade em $N_{k(v)}$.

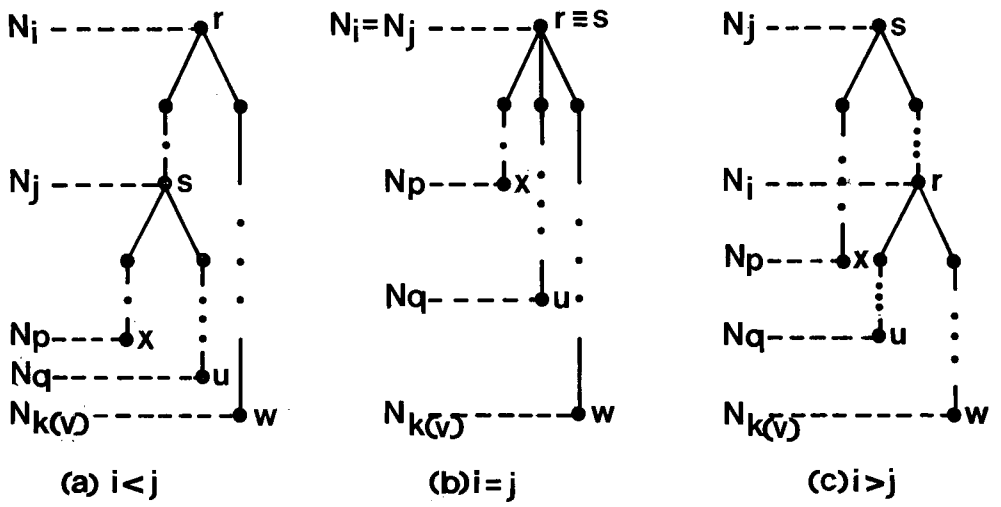


Figura V-3: Casos discutidos na demonstração do lema V.2.

---

* $N_c(G)$ é idêntica a $N_a(G)$.

Demonstração: Suponha que existam vertices u e x, $u \in N_q$ e $x \in N_p$, $q, p < k(v)$ tais que $d(u,x) = d(G)$. Seja $w \in N_{k(v)}$. Seja $r \in N_i$ o ancestral de maior nível dos vertices u e w, e $s \in N_j$ o ancestral de maior nível de u e x.
Pelo lema I.3:

(V-7) $d(w,r) = k(v) - i > q - i = d(u,r)$

(V-8) $d(w,s) = d(w,r) + d(r,s)$, onde $d(r,s) \equiv 0$, se $i = j$

(V-9) $d(u,x) = d(u,s) + d(s,x)$

De (V-7) e (V-8), resulta

(V-10) $d(w,s) > d(u,r) + d(r,s)$

Considere os diversos casos possíveis, conforme figura (V-3):

. $i < j$
De (V-10) resulta, pelo lema I.3.:

(V-11) $d(w,s) > d(u,r) + d(r,s) = q - i + j - i > q - j = d(u,s)$

. $i \geq j$
De (V-10) resulta, pelo lema I.3:

(V-12) $d(w,s) > d(u,r) + d(r,s) = q - i + i - j = q - j = d(u,s)$

De (V-9), (V-11) e (V-12) resulta, finalmente

(V-13) $d(u,x) = d(u,s) + d(s,x) < d(w,s) + d(s,x) = d(w,x)$

que contraria a hipótese de que $d(u,x) = d(G)$. Portanto, todo diâmetro tem pelo menos uma extremidade em $N_{k(v)}$.

Teorema V.3.: Sejam $G = (V,E)$ um grafo conexo e sem ciclos e, $N_v(G)$ uma estrutura de níveis com raiz em v e profundidade $k(v)$. Então, todo vertice em $N_{k(v)}$ é um vertice periferico de G.

Demonstração: Basta considerar a demonstração do lema V.2 com $q = k(v)$.
Neste caso, as equações (V-7), (V-10), tornam-se

(V-7)' $d(w,r) = k(v) - i = d(u,r)$

(V-10)' $d(w,s) = d(u,r) + d(r,s)$

Considere os diversos casos possíveis (figura (V-3)):

. $i < j$. De (V-10)' tem-se:

(V-11)' $d(w,s) = d(u,r) + d(r,s) = q - i + j - i > q - j = d(u,s)$

E de (V-9) e (V-11)', resulta

(V-14) $d(u,x) = d(u,s) + d(s,x) < d(w,s) + d(s,x) = d(w,x)$

que contraria a hipótese de $d(u,x) = d(G)$.

. $i \geq j$. De (V-10)' tem-se:

(V-12)' $d(w,s) = d(u,r) + d(r,s) = q - i + i - j = q - j = d(u,s)$

e usando (V-9) tem-se:

(V-15) $d(w,x) = d(w,s) + d(s,x) = d(u,s) + d(s,x) = d(u,x) = d(G)$

Logo w também é um vertice periférico.

V.2.2. Análise de Complexidade

A determinação dos graus dos vertices de V é necessária para a ordenação dos conjuntos S construidos no passo 3. Teríamos um tempo de $0(n + m)$ para a determinação de um vetor de graus, e um tempo de $0(n)$ para a ordenação de cada conjunto S pelo metodo da raiz, conforme mostrado no capítulo III.

Assim, a análise de complexidade desse algoritmo está feita com base numa forma modificada do algoritmo em que os passos 1 e 3 são substituidos pelos abaixo, para evitar a ordenação de cada conjunto S:

1'. Ordenar os vertices de V por ordem crescente de grau, pelo algoritmo A2, e escolher como vertice inicial o primeiro vertice desse conjunto.

3'. Gerar o conjunto S da seguinte forma:
Percorrer o conjunto V ordenado e ir colocando em S os vertices no último nível de $N_v(G)$, para os quais não tenha sido gerada estrutura de níveis.

Embora a complexidade em tempo do passo 3'. seja a mesma do passo 3., que é 0(n), seu número de operações é, evidentemente, menor.

Espaço de Memória

Além dos (n + 4m) elementos requeridos pela estrutura de adjacencias de G, são utilizados, ainda, três vetores de n elementos para guardar as estruturas de níveis, e um vetor de n elementos para guardar os vertices de V ordenados por grau.

O conjunto S não precisa ser, efetivamente, armazenado, pois o próximo vertice de menor grau em $N_{k(v)}(v)$ pode ser calculado no momento da execução do passo 4.

Portanto, são necessários (5n + 4m) posições de memória (além do espaço adicional utilizado pelos algoritmos A2 e A3). Assim, esse algoritmo requer 0(n + m) posições de memória.

Tempo de Execução

O passo 1', 2 e 4, como já foi visto (algoritmos A2 e A3) requerem tempo de 0(n + m).

A escolha do próximo vertice de menor grau em $N_{k(v)}(v)$, de acordo com o passo 3' requer tempo de 0(n).

Os passos 5 e 6 requerem tempos constantes, de 0(c).

No pior caso, esse algoritmo gera 0(n) estruturas de níveis antes de parar, pois $|N_{k(v)}(v)|$ pode ser de 0(n). Portanto, a complexidade global em tempo de execução é de 0(n (n + m)).

É interessante notar que, no pior caso, esse algoritmo tem a mesma complexidade em tempo (0 (n (n + m))) de um algoritmo para calcular, efetivamente, o diâmetro do grafo - gerando estruturas de níveis com raiz em todos os vertices e determinando a maior profundidade obtida.

Devido a importancia desse algoritmo, o qual é novamente utilizado no capítulo VI, são discutidos a seguir alguns de seus piores e melhores casos.

Determinação de Piores e Melhores Casos

1. Grafos Completos

Para grafos completos tem-se o número de arestas (m) da ordem de n^2.

No passo 2 é gerada uma estrutura de níveis de profundidade 2, com n - 1 vertices no segundo nível, num tempo de $0(n^2)$.

Os passos 4 e 5 são alternadamente executados para todos os vertices em S, sendo geradas mais n - 1 estruturas de níveis.

Assim, para um grafo completo é requerido um tempo de $0(n^3)$.

2. Grafos Conexos Sem Ciclos (árvores)

Neste caso, m = n - 1 e o passo 1 consome um tempo de 0(n).

O passo 2 requer um tempo de 0(n) para gerar a estrutura de níveis $N_v(G)$, de profundidade k(v).

Os passos 3 e 4 também requerem tempos de 0(n).

Dependendo do vertice v inicial podem ocorrer duas situações:

(i) Se v for um vertice periférico, então são geradas (passo 4) estruturas de níveis para todos os vertices em $N_{k(v)}(v)$, as quais tem a mesma profundidade. Assim, são geradas 0(n) estruturas de níveis, uma vez que a cardinalidade de $N_{k(v)}$ é, no pior caso, de 0(n).

(ii) Se v não for um vertice periférico, o teorema V.3. garante que todo vertice em $N_{k(v)}$ é um vertice periférico, portanto, o passo 4 é executado para o primeiro vertice de $N_{k(v)}$ gerando uma estrutura de níveis $N_r(G)$ de profundidade igual a d(G). E como r é um vertice periférico cai-se no caso (i).

Da situação (i) acima resulta que para um grafo conexo sem ciclos é requerido um tempo de $0(n^2)$.

É interessante notar que, se for utilizado um algoritmo especifico para determinação de diâmetros de árvores, um vertice periférico seria obtido num tempo de O(n), desde que, pelo teorema V.3. seria necessário gerar, no máximo, duas estruturas de níveis.

Evidentemente, uma árvore cujo grau máximo (d_{max}) é igual a 2, tem diâmetro igual a n - 1, e nesse caso o algoritmo IV gera duas, ou três, estruturas de níveis, dependendo do vertice inicial v ser ou não um vertice periférico. Portanto, seria consumido um tempo de O(n).

V.2.3. Formulação Tipo Algol

Estrutura de Dados

No passo inicial o conjunto de vertices do grafo é ordenado por grau pelo algoritmo A2, ficando tais vertices armazenados no vetor VG [N].

Três estruturas de níveis devem estar na memória ao mesmo tempo, sendo armazenadas na matriz NIVEL [3,N], de tal forma que

NIVEL [V,I] = J

se o vertice I estiver no nível J da estrutura de níveis V.

O conjunto S não é utilizado, e W, K, WK são vetores de 3 elementos que guardam, a largura, a profundidade e a largura do último nível, respectivamente, de cada uma das estruturas de níveis na memória.

Um vetor MARCA [N] é usado para marcar os vertices para os quais já foi gerada estrutura de níveis.

Descrição Tipo Algol do Algoritmo IV

```
begin;
  V <- 1;
  R <- 2;
  U <- 3;
  MARCA [*] <- ∅;
  comment 1. Ordenação dos vertices do grafo.
  Executar o algoritmo A2 guardando no vetor VG os vertices de
```

```
G em ordem crescente de grau.
comment 2. Geração da estrutura de níveis N_v(G) com raiz num
vertice de grau mínimo.
Executar o algoritmo A3, gerando em NIVEL [V,*] a estrutura
de níveis com raiz em VG [1], profundidade K [V] e largura
W [V] e com largura do último nível igual a WK [V].
I <- ∅;
W [U] <- N;
MARCA [1] <- 1;
while I < N do
  begin
    comment 4. Escolha do vertice r de menor grau em N_k(v)
    e geração da estrutura de níveis N_r (G), onde r é um ver
    tice de menor grau em N_k(v) para o qual não tenha sido
    gerada estrutura de níveis.
    for I <- I + 1 until N while MARCA [I] = 1 |
      NIVEL [V,VG [I]] ≠      do;
    if I ≤ N then
      begin
        Executar o algoritmo A3, gerando em NIVEL [R,*] a es
        trutura de níveis com raiz no vertice VG [I], profun
        didade K [R], largura W [R] e com largura do último
        nível igual a WK [R].
        MARCA [I] <- 1;
        comment 5. Comparação das profundidades das estrutu-
        ras de níveis.
        if K [R] > K [V] then
          begin
            L <- V;
            V <- R;
            R <- L;
            I <- ∅;
            W [U] <- N;
          end;
        else if W [U] > W [R] then
          begin
            L <- U;
            U <- R;
            R <- L;
          end;
      end;
```

```
    end;
end;
```

V.3. Composição de Estruturas de Níveis

No capítulo IV foi visto (teorema IV.3.) que, para um grafo G(A) = (V,E), a largura de banda relativa a uma numeração $\delta(G)$, que satisfaça a propriedade (IV-2), nunca é menor que a largura da estrutura de níveis com raiz usada no passo de numeração do algoritmo.

Entretanto a largura de banda mínima de um grafo pode ser consideravelmente menor que a largura de qualquer de suas estruturas de níveis com raiz; isso ocorre, por exemplo, para o grafo da figura (IV-5) cuja estrutura de níveis mais estreita tem largura 5, e cuja largura de banda relativa a numeração produzida pelo algoritmo GPS é 4, conforme mostra a figura (V-9), e que é, efetivamente, a largura de banda mínima para esse grafo, pois $d_{max} = 8$, e

(V-16) $\quad \forall \delta : V \rightarrow \{1,2,\ldots,n\} \; (B_{\delta}(G) \geq d_{max}/2)$

O esquema de composição de estruturas de níveis, discutido nessa seção, tenta gerar uma estrutura de níveis de menor largura, a partir das estruturas de níveis, com raizes em vertices pseudo-periféricos, e obtidas através do algoritmo IV. A estrutura de níveis resultante pode ter mais de um vertice no primeiro nível, e qualquer de seus níveis pode conter vertices que não são adjacentes a nenhum dos vertices nos níveis anteriores, ainda assim porém, em concordancia com a definição de estruturas de níveis (sem raiz).

V.3.1. Esquema de Composição de Estruturas de Níveis

Esse algoritmo associa, a cada vertice x em V, um par ordenado de níveis (i,j), determinado pelos níveis $N_i(v)$ e $N_{k+1-j}(u)$ que contem x, das estruturas de níveis geradas pelo algoritmo IV. Coloca no nível E_i da estrutura de níveis $N(G) = \{E_1, E_2, \ldots, E_k\}$ todo vertice associado a um par da forma (i,i). Determina os componentes conexos do subgrafo resultante de G pela remoção de todo vertice x (e de toda aresta in

cidente a x) tais que x está associado a um par de níveis da forma (i,i), e inclue em $N(G)$ os vertices de cada um desses componentes utilizando o elemento do par de níveis que produza uma estrutura de níveis $N(G)$ de menor largura.

São utilizados, além da estrutura de adjacencias (para a determinação dos componentes conexos), três vetores de n elementos para armazenar as estruturas de níveis, e três outros vetores auxiliares de n elementos usados durante o processo de escolha do elemento do par de níveis a ser utilizado para cada componente. No processo de determinação dos componentes conexos são usados dois outros vetores de n elementos, para guardar os vertices, e o número de vertices, de cada componente.

Algoritmo V: Composição das estruturas de níveis geradas pelo algoritmo IV.

Entrada: Grafo G(A) = (V,E), representado na forma de uma estrutura de adjacencias, as duas estruturas de níveis $N_u(G)$ e $N_v(G)$ geradas pelo algoritmo IV, de profundidade k = k(u) = k(v), e larguras $w(N_u)$ e $w(N_v)$, grau(u) e grau(v).

Saída: Estrutura de níveis $N(G) = \{E_1, E_2, \ldots, E_k\}$ derivada de $N_u(G)$ e $N_v(G)$.

Metodo:

1. Seja $\bar{v}$ o vertice de menor grau entre u e v, e $\bar{u}$ o outro vertice.
   Associar a cada vertice x, em V, um par ordenado (i,j) de níveis*, determinado pelos vertices $N_i(\bar{v})$ e $N_{k+1-j}(\bar{u})$, que contem x.

2. Colocar no nível E_i de $N(G)$ todo vertice x associado a um par da forma (i,i), e remover de G x e todo o conjunto de arestas incidentes a x.
   Seja $\bar{G} = (\bar{V},\bar{E})$ o grafo resultante.
   Se $\bar{V} = \emptyset$, então parar.

---

* Note que o par (1,1) é associado ao vertice $\bar{v}$ e o par (k,k) é associado ao vertice $\bar{u}$.

3. Determinar os componentes conexos $C_1, C_2, \ldots C_t$ de $\bar{G}$, ordenados de tal forma que
$|V(C_1)| \geq |V(C_2)| \geq \ldots \geq |V(C_t)|$

4. Calcular o vetor $s = (s_1, s_2, \ldots, s_k)$, onde $s_i = |E_i|$. Fazer $\ell \leftarrow 1$.

5. Colocar os vertices do componente C_ℓ em N, nos níveis indicados pelo elemento do par ordenado de níveis, escolhido como se segue:

   (i) Calcular os vetores $p = (p_1, p_2, \ldots, p_k)$ e $q = (q_1, q_2, \ldots, q_k)$, onde:
   $p_i = s_i + |\{x \in C_\ell$: o par (i,j) está associado a x$\}|$
   $q_j = s_j + |\{x \in C_\ell$: o par (i,j) está associado a x$\}|$

   (ii) Determinar $p_o = \max \{p_i : p_i - s_i > 0\}$, e $q_o = \max \{q_i : q_i - s_i > 0\}$.

   . Se $p_o < q_o$, então escolher o primeiro elemento dos pares e fazer $s \leftarrow p$.

   . Se $p_o > q_o$, então escolher o segundo elemento dos pares associados e fazer $s \leftarrow q$.

   . Caso contrario se $w(N_v) \leq w(N_u)$, então escolher o primeiro nível do par associado e fazer $s \leftarrow p$, senão escolher o segundo nível do par associado e fazer $s \leftarrow q$.

6. Repetir o passo 5, com $\ell \leftarrow \ell + 1$, enquanto $\ell \leq t$.

Exemplo V-2: Considere o grafo G na figura (IV-5) e as estruturas de níveis $N_m(G)$ e $N_b(G)$ na figura (V-2).

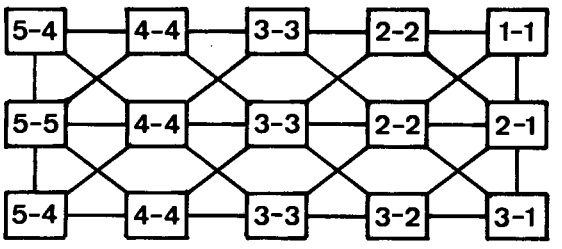


Figura V-4: Pares ordenados de níveis associados aos vertices de V, no passo 1 do algoritmo V.

Os pares ordenados de níveis (i,j) associados aos vertices de V são mostrados na figura (V-4).

Os vertices associados aos pares do tipo (i,i) são colocados no nível E_i da estrutura $N(G)$ sendo gerada e os componentes conexos resultantes são C_1, C_2, C_3, conforme indicados na figura (V-5).

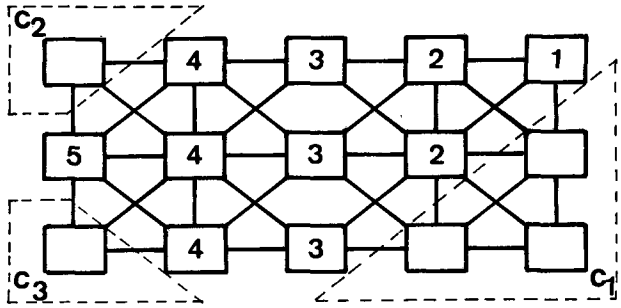


Figura V-5: Estrutura de níveis $N(G)$ com os vertices associados aos pares do tipo (i,i), e componentes conexos determinados no passo 3.

Para o componente C_1 tem-se $r = (1,2,3,3,1)$, $p = (1,3,5,3,1)$ e $s = (3,3,3,3,1)$. Segue-se que $p_o = 5$ e $s_o = 3$, e o segundo elemento dos pares de níveis é, então, usado para C_1, conforme mostra a figura (V-6).

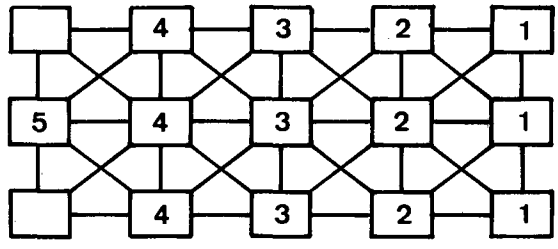


Figura V-6: Estrutura de níveis $N(G)$ após a inclusão dos vertices do componente conexo C_1.

Após a inclusão dos vertices relativos aos componentes C_2 e C_3, obtem-se a estrutura de níveis $N(G)$, na figura (V-7).

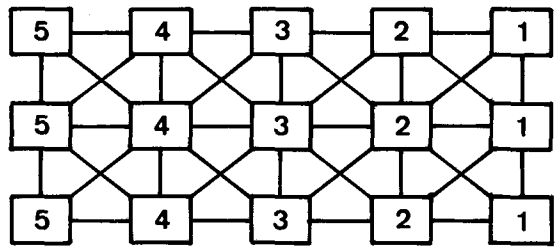


Figura V-7: Estrutura de níveis $N(G)$ gerada pelo algoritmo V, para o grafo G(A) da figura (IV-5).

V.3.2. Análise de Complexidade

Espaço de Memória

Além dos (n + 4m) elementos requeridos pela estrutura de adjacencias de G, e dos três vetores de n elementos usados para armazenar as estruturas de níveis, são necessários os três vetores de n elementos s, p e q. Portanto, (7n + 4m) elementos.

Para determinar os componentes conexos e dispô-los em ordem decrescente de número de vertices é necessário um espaço adicional de O(n).

Assim, esse algoritmo requer um espaço de memória de O(n + m).

Tempo de Execução

Como foi dito no início desse capítulo, as estruturas de níveis estão representadas na forma de vetores contendo os níveis de cada vertice. Portanto, no passo 1 apenas uma das estruturas de níveis (com raiz no vertice de maior grau) é percorrida, invertendo-se a numeração de seus níveis (j = k + 1 - j), o que requer um tempo de O(n).

O passo 2 também requer tempo de O(n), pois as estruturas de níveis são percorridas e os vertices incluidos em $N(G)$ são "marcados", para serem identificados no processo de determinação dos componentes conexos.

O passo 3 requer um tempo de $0(n + m)$, pois a estrutura de adjacencias do grafo é percorrida para determinar as adjacencias de cada vertice não incluido em $N(G)$.

O passo 4 requer um tempo de $0(n)$ para calcular o vetor s, e o calculo de p e q (passo 5i) requer um tempo de $0(n)$ (proporcional ao número de vertices de cada componente). O restante do passo 5 requer, também, um tempo de $0(n)$, e como existem t componentes, o passo 5 necessita, para completar a construção de N, de um tempo de $0(tn)$.

Portanto, o tempo de execução requerido é de $0(tn + m)$, mas, no pior caso, t pode ser de $0(n)$ implicando numa complexidade em tempo de execução de $0(n^2)$.

V.3.3. Formulação Tipo Algol

Estrutura de Dados

A matriz NIVEL [3,N] usada no algoritmo IV é aqui utilizada para guardar as estruturas de níveis N_v, N_u e N.

O vetor S [N] guarda o número de vertices por nível em N, após a inclusão de cada componente conexo de $\bar{G}$. P [N] e Q [N] são os outros vetores auxiliares usados no processo de escolha do elemento do par de níveis a ser usado para cada componente.

Um vetor binário FALTA [N] é usado para marcar os vertices ainda não incluidos em N, e é considerado que os vetores COMP [N] com os vertices de cada componente conexo e TCOMP [N] com os números de vertices dos componentes são gerados no processo de determinação e ordenação desses componentes, que, evidentemente, utiliza a estrutura de adjacencias do grafo.

O vetor D [3] e W [3], gerados pelo algoritmo IV, contem, respectivamente, os graus de u, v, r e as larguras de $N_u(G)$, $N_v(G)$ e $N_r(G)$. Os valores das variáveis U, V e R são os valores finais calculados no algoritmo IV.

Descrição Tipo Algol do Algoritmo V

```
begin
  comment 1.  Associação dos pares ordenados.
  if D [V] > D [U] then
    begin
      L <- V;
      V <- U;
      U <- L;
    end;
  L <- N + 1;
  for I <- 1 until N do NIVEL [U,I] <- L - NIVEL [U,I];
  comment 2. Colocação em N dos vertices associados a pares do
  tipo (i,i).
  for I <- 1 until N do
    begin
      L <- NIVEL [V,I];
      if L = NIVEL [U,I] then
        begin
          NIVEL [R,I] <- L;
          FALTA [I] <- ∅;
        end;
      else FALTA [I] <- 1;
    end;
  comment 3. Determinação dos componentes conexos de Ḡ.
  Utilizando o vetor FALTA, determinar os T componentes cone-
  xos de Ḡ, e dispô-los em ordem decrescente de número de ver-
  tices, no vetor COMP.  O vetor TCOMP contem o número de ver-
  tices de cada componente.
  comment 4. Calculo do vetor S.
  S [*] <- ∅;
  for I <- 1 until N do
    begin
      LL <- NIVEL [R,I];
      S [LL] <- S [LL] + 1;
    end;
  J <- ∅;
  L <- ∅;
  while  L < T do
```

```
begin
  L <- L + 1;
  comment 5. Inclusão do componente c_L.
  JJ <- J;
  P [*] <- Ø;
  Q [*] <- Ø;
  comment 5i. Calculo dos vetores p e q.
  for I <- 1 until TCOMP [L] do
    begin
      J <- J + 1;
      LL <- NIVEL [V,COMP [J]];
      P [LL] <- P [LL] + 1;
      LL <- NIVEL [U,COMP [J]];
      Q [LL] <- Q [LL] + 1;
    end;
  comment 5ii. Determinação do elemento do par de níveis a
  ser usado.
  calcular os valores de PØ e QØ.
  if PØ < QØ then
    begin
      II <- V;
      S <- S + P;
    end;
  else if PØ > QØ then
    begin
      II <- U;
      S <- S + Q;
    end;
  else if W [V] ≤ W [U] then
    begin
      II <-  V;
      S <- S + P;
    end;
  else begin
    II <- U;
    S <- S + Q;
    end;
  J <- JJ;
```

```
        for I <- 1 until TCOMP [L] do
          begin
            J <- J + 1;
            LL <- COMP [J];
            NIVEL [R,LL] <- NIVEL [II,LL];
          end;
      end;
end;
```

V.4. Numeração dos Vertices do Grafo

Esse esquema de numeração dos vertices do grafo, rotula-os a partir da estrutura de níveis gerada pelo algoritmo V e da estrutura de adjacencias de G. É análogo ao esquema RCM no sentido de que atribui inteiros consecutivos aos vertices, nível por nível. Possui algumas diferenças (passo 2) decorrente do fato das estruturas de níveis obtidas pelo algoritmo V serem mais gerais que as estruturas de níveis com raiz usadas no algoritmo RCM.

Um passo final de inversão da numeração obtida (como no algoritmo RCM) é executado, ou não, em função de ter sido usado o primeiro ou o segundo elemento dos pares ordenados de níveis, para o componente C_1.

São utilizados a estrutura de adjacencias do grafo, a estrutura de níveis $N(G)$ representada na forma de um vetor de níveis, e um vetor para guardar a numeração $\phi(G)$.

Algoritmo VI: Numeração dos vertices a partir da estrutura de níveis produzida pelo algoritmo V.

Entrada: Grafo G(A) = (V,E), representado na forma de uma estrutura de adjacencias, estrutura de níveis $N(G) = \{E_1, E_2, \ldots, E_k\}$, gerada pelo algoritmo V, e número do elemento do par de níveis usado para o componente C_1.

Saída: Uma numeração ϕ dos vertices de V.

Metodo:

1. Rotular o vertice v com o número 1.
   Fazer i <- 1.

2. Numerar os outros vertices no nível E_i ($i \geq 1$), se existirem, como se segue:

   (i) Considerar cada vertice $\underline{w}$ em E_i, na ordem em que foram rotulados e numerar os vertices em E_i, adjacentes a $\underline{w}$ e ainda não rotulados, consecutivamente, em ordem crescente de grau.

   (ii) Se existir algum vertice em E_i ainda não rotulado, numerar um dos vertices de menor grau e repetir o passo 2i.

3. Fazer $i \leftarrow i + 1$.
   Se $i \leq k$, então considerar cada vertice $\underline{w}$ em E_{i-1}, na ordem em que foram rotulados e numerar os vertices em E_i adjacentes a w, ainda não rotulados, consecutivamente, em ordem crescente de grau.
   Ir para o passo 2i.

4. Caso contrario, a numeração $\phi(G)$, assim obtida, deve ser invertida, trocando i por n - i + 1, para todo i, se o algoritmo V selecionou os primeiros elementos dos pares de níveis para o componente c_1.

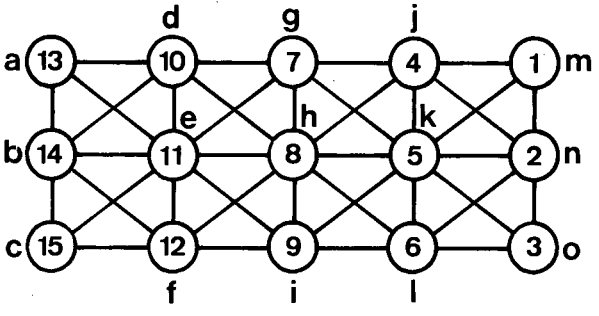


Figura V-8: Aplicação do algoritmo VI a estrutura de níveis $N(G)$ na figura (V-5).

Exemplo V-3: Considere o grafo G(A) na figura (IV-5), e a estrutura de níveis $N(G)$, na figura (V-7). O algoritmo VI, rotula os vertices de G(A) (conforme mostrado na figura (V-8)) procedendo como se segue:

Passo 1: O vertice m é rotulado com o número 1.

Passo 2i: O vertice n, adjacente a m, é rotulado com o número 2, e o vertice o, adjacente a n, é rotulado com o 3.

Passo 3: Os vertices j e k, adjacente a m são rotulados com 4 e 5, respectivamente, e o vertice l é rotulado com 6.

Passo 3: Os vertices g e h (adjacentes a j) são rotulados com 7 e 8, respectivamente, e o vertice i é rotulado com 9.

Passo 3: Analogamente, os vertices d, e e f são rotulados com 10, 11 e 12, respectivamente.

Passo 3: Os vertices a e b são rotulados com 13 e 14, e o vertice c com 15.

Passo 4: Como o algoritmo V selecionou os segundos elementos dos pares de níveis, para o componente conexo c_1, não há inversão da numeração assim obtida.

A figura (V-9) mostra a matriz A_g obtida a partir da numeração $\phi(G)$, na figura (V-8), produzida pelo algoritmo VI.

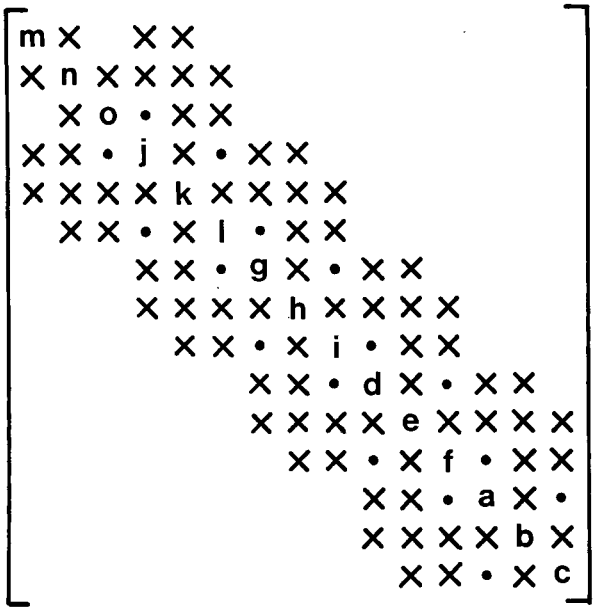


Figura V-9: Matriz A_g obtida pela aplicação do algoritmo VI ao grafo G(A) da figura (IV-5).
$B(A_g) = 4$ e $P(A_g) = 46$.

V.4.2. Análise de Complexidade

Obviamente, o passo 2 desse algoritmo é decorrencia do tipo de estrutura de níveis gerada pelo algoritmo V, e trata da numeração dos vertices dos componentes conexos (vide seção V.3.) que foram incluidos em $N(G)$ usando o segundo elemento dos pares ordenados de níveis. Tais vertices são divididos em dois grupos: (i) vertices adjacentes a vertices no mesmo nível já numerados, e (ii) demais vertices.

Para numerar os vertices do grupo (i) é necessário consultar a estrutura de adjacencias do grafo, uma vez que são utilizadas arestas que não definem relação de "parentesco" (relação pai) relativa a nenhuma das estruturas de níveis $N_v(G)$ ou $N_u(G)$ usadas pelo algoritmo V.

A numeração dos vertices do grupo (ii) exige a classificação de subconjuntos de vertices em ordem crescente de grau, para se determinar um dos vertices não rotulados de menor grau, no nível sendo numerado.

Se forem feitas ordenações de vertices para cada nível de $N(G)$, a complexidade em tempo de execução será de $O(n^2)$, uma vez que $O(n)$ ordenações seriam executadas.

Em consequência disso, nessa análise de complexidade é considerada uma versão modificada do algoritmo, que utiliza os vetores MAXROT com o maior rotulo válido para cada nível e o vetor ROTULO com o menor rotulo ainda disponível para cada nível. Desse modo, vertices de diferentes níveis podem ser rotulados ao mesmo tempo.

A estrutura de adjacencias é ordenada preliminarmente, e os vertices de V são classificados segundo a ordem lexicográfica "<" (vide definição III.1) definida a partir do conjunto S de pares ordenados (grau (x), nível (x)), para todo x em V, e guardados no vetor ORD.

Uma fila é ainda utilizada para guardar os vertices sendo numerados, indicando, assim, a ordem em que as listas de adjacencias devem ser percorridas.

Os três passos iniciais do algoritmo são, então modificados para a forma seguinte:

1'. Ordenar a estrutura de adjacencias de G, pelo algoritmo A1.

2'. Ordenar a estrutura de níveis N(G) da seguinte maneira: Associar a cada vertice x de V um par ordenado (grau (x), nível (x)), e dispô-los em ORD, segundo a ordem lexicográfica "<".

3'. Numeração dos vertices de V.

(i) Rotular o vertice v com 1.
Colocar v em FILA.

(ii) Enquanto FILA $\neq \emptyset$ fazer:
Retirar w da FILA.
Percorrer a lista de adjacencias de w e para cada vertice x assim encontrado:

. Se x não tiver sido ainda numerado, e se o nível de w não for maior que o de x, numerar x com o próximo rotulo disponível para o nível de x, e colocar x na FILA. Repetir o passo 3ii.

(iii) Percorrer a estrutura de níveis ordenada (vetor ORD) e numerar o primeiro vertice x ainda não rotulado com o próximo rotulo disponível para seu nível, colocar x na FILA. Repetir o passo 3ii.

Espaço de Memória

São utilizados:

. Estrutura de adjacencias do grafo.

. Estrutura de níveis representada na forma de um vetor de níveis de n elementos.

. Vertices de V dispostos segundo a ordem parcial "<".

. Vetores ROTULO e MAXROT

. Vetor de n elementos representando a fila auxiliar usada no passo 3', e

. Vetor contendo a numeração dos vertices de V.

Assim, o espaço de memória necessário é

(7n + 4m), além do espaço adicional requerido nos passos 1' e 2' que é de 0(n + m).

Portanto, o espaço de memória requerido é de 0(n + m).

Tempo de Execução

O passo 1 requer um tempo de 0(n+ m).

O passo 2, uma vez que o vetor com os graus dos vertices foi determinado no passo 1, requer um tempo de 0(n), conforme no passo 3 do algoritmo I.

No passo 3 a estrutura de adjacencias é percorrida uma vez, o que requer um tempo de 0(n + m). No passo 3iii a estrutura de níveis é também percorrida uma vez, o que requer um tempo de 0(n).

O passo 4 requer um tempo de 0(n).

Portanto, a complexidade global em tempo de execução é de 0(n + m).

V.4.3. Formulação Tipo Algol

Estrutura de Dados Utilizada

O grafo G é representado na forma de uma estrutura de adjacencias, com listas encadeadas.

A estrutura de níveis é representada no vetor NIVEL que guarda o nível de cada vertice em $N(G)$.

O vetor GRAU guarda o grau de cada vertice de V.

O vetor ORD guarda os vertices em $N(G)$ ordenados no passo 2'.

O vetor ROTULO guarda os rotulos disponíveis para cada nível de N, e o vetor MAXROT guarda o maior rotulo válido para cada nível.

A fila é armazenada em um vetor FILA, e a numeração $\phi(G)$ no vetor F.

A variável ELC1 indica o elemento do par ordenado de níveis usado pelo algoritmo V.

Descrição Tipo Algol do Algoritmo VI

```
begin
  comment 1. Ordenação da estrutura de adjacencias.
  Executar o algoritmo A1.
  comment 2. Ordenação da estrutura de níveis. Analogamente
  ao passo 3i do algoritmo I, ordenar (no vetor ORD) a estrutu
  ra de níveis N(G) segundo uma ordem lexicográfica definida a
  partir dos pares ordenados (grau (x_i), nível (x_i)), associa-
  dos aos vertices x_i ε V, 1 ≤ i ≤ n.
  Os vetores ROTULO e MAXROT, devem ser inicializados durante
  essa ordenação.
  comment 3. Numeração dos vertices de V.
  F [V] <- 1;
  I <- 1;
  FILA <= V;
  while I ≤ K do
    begin
      while FILA ≠ ∅ do
        begin
          W <= FILA;
          percorrer a lista de adjacencias de W (ordenada), e
          para cada vertice X assim encontrado fazer:
          if F [X] ≠ 0 then
            if NIVEL [W] ≤ NIVEL [X] then
              begin
                L <- ROTULO [NIVEL [X]];
                F [X] <- L;
                ROTULO  [NIVEL [X]] <- L + 1;
                FILA <= X;
              end;
        end;
      comment 3(iii). Determinação de um vertice de menor grau
      ainda não rotulado.
      for I <- I until K while ROTULO [I] > MAXROT [I] do;
      if I ≤ K then
        begin
```

```
            for J <- MAXROT [I - 1] + 1 step 1
            while FILA = ∅ do
              begin
                if F [ORD [J]] = ∅ then
                  begin
                    X <- ORD [J];
                    L <- ROTULO [NIVEL [X]];
                    F [X] <- L;
                    ROTULO [NIVEL [X]] <- L + 1;
                    FILA <= X;
                  end;
              end;
          end;
      end;
    comment 4. Inversão da numeração obtida no passo 3.
    if ELC1 = 1 then
      for I <- 1 until N do F [I] <- N + 1 - F [I];
end;
```

VI. DISCUSSÃO DOS ALGORITMOS APRESENTADOS

VI.1. Introdução

Os algoritmos RCM e GPS são discutidos e comparados na seção VI.2., onde algumas desvantagens de um e de outro são destacadas. Um esquema que combina as vantagens do algoritmo de determinação de vertices pseudo-periféricos com uma numeração tipo RCM é aventado.

Um tal esquema é apresentado na seção VI.3., sendo discutidas modificações visando reduzir o tempo de execução do mesmo. Uma nova modificação é incorporada a esse esquema, de forma a reduzir a cardinalidade dos conjuntos S, desse modo, diminuindo o seu tempo de execução.

VI.2. Comparação dos Algoritmos RCM e GPS

Como foi visto no capítulo IV, o maior inconveniente do algoritmo RCM é seu passo A.2, onde o processo de escolha de vertices iniciais para a geração de estruturas de níveis é exaustivo. Caso todos os vertices do grafo tenham o mesmo grau seria gerada uma estrutura de níveis para cada um dos vertices do grafo.

Outro inconveniente desse algoritmo é que, para cada estrutura de níveis $N_v(G)$ gerada, é necessário calcular a correspondente numeração $\phi(G)$ e sua largura de banda $B_\phi(G)$, para permitir a escolha da numeração que corresponde a menor largura de banda obtida.

Um terceiro inconveniente, ou mais propriamente uma limitação desse esquema, é (cf. teorema IV.3.) a impossibilidade de alguma numeração $\phi(G)$ resultar em uma largura de banda menor que a largura da estrutura de níveis $N_v(G)$ usada, muito embora a largura de banda mínima de um grafo possa ser, consideravelmente, menor que a largura de qualquer de suas estruturas de níveis.

Nos últimos anos, tem se tornado cada vez mais frequente ([5], [7], [9], [24]), em implementações do metodo dos elementos finitos, a utilização de estruturas de dados na

forma de envelope (em vez de na forma de banda), para o armazenamento de matrizes esparsas. Com isso, a citada limitação do esquema RCM torna-se discutível, pois os problemas de redução de largura de banda e de perfil são, efetivamente, independentes. E, pelo teorema IV.3., o perfil relativo a uma numeração $\delta(G)$ depende das larguras de cada um dos níveis da estrutura de níveis utilizada, e não da largura dessa estrutura de níveis.

Portanto, efetivamente, o algoritmo RCM apresenta os dois primeiros inconvenientes referidos.

Para auxiliar na análise do algoritmo GPS, bem como, na sua comparação com o algoritmo RCM, está transcrita abaixo (tabela VI-1) os resultados de 19 testes realizados, por GIBBS at al. [8], com os algoritmos GPS e RCM. Foram acrescentadas as colunas de B_{RCM}/B_{GPS} e P_{RCM}/P_{GPS} para possibilitar uma observação mais clara das variações de largura de banda e de perfil. Os índices RCM e GPS indicam que as respectivas medidas se referem a tais algoritmos; T_{RCM}/T_{GPS} indica a variação nos seus tempos de execução.

Tais testes se referem a 19 matrizes esparsas, que foram acumuladas em um período de vários anos por E. H. Cuthill e G. C. Everstine, algumas das quais aparecem em [23]. Essas matrizes ocorrem na resolução de vários problemas de equações diferenciais e variacionais em cálculo de estruturas, quando se utiliza o metodo dos elementos finitos. Os 19 problemas tratados incluem aplicações diversas, e neles foram utilizados elementos bidimensionais triangulares e quadrilateros.

Cada um dos três esquemas que constituem o algoritmo GPS são aqui discutidos separadamente.

O esquema IV, de determinação de vertices pseudo-periféricos, é um metodo heurístico de determinação de estruturas de níveis de maior profundidade. Para uma grande classe de grafos (que inclue todas as árvores) tal profundidade é, efetivamente, o diâmetro do grafo. Conforme citado em [8], em todos os testes na tabela (VI-1) foram determinadas estruturas de níveis com profundidades iguais aos respectivos

Caso	n	$\mathcal{B}$	$\mathcal{B}_{RCM}$	$\mathcal{B}_{GPS}$	$\mathcal{B}_{RCM}/\mathcal{B}_{GPS}$	P	P_{RCM}	P_{GPS}	P_{RCM}/P_{GPS}	T_{RCM}/T_{GPS}
1	68	45	5	7	0,71	598	236	269	0,88	11,06
2	90	85	9	7	1,28	1.020	575	579	0,99	3,90
3	92	80	14	13	1,08	2.127	739	736	1,00	6,42
4	130	126	19	18	1,06	3.615	1.562	1.588	0,98	2,13
5	159	19	11	12	0,92	1.046	983	971	1,01	2,02
6	174	16	14	13	1,08	1.569	1.615	1.466	1,10	4,23
7	185	168	30	29	1,03	7.534	3.664	3.610	1,01	4,54
8	220	166	13	12	1,08	8.532	1.809	1.868	0,97	33,76
9	263	262	19	19	1,00	2.681	2.337	2.346	1,00	10,55
10	263	30	13	14	0,93	2.040	2.925	2.001	1,01	6,35
11	310	302	14	14	1,00	23.357	2.725	2.726	1,00	8,17
12	312	262	33	37	0,89	18.076	5.812	5.548	1,05	5,75
13	346	216	43	46	0,93	16.435	7.180	7.650	0,94	6,41
14	360	344	33	34	0,97	29.790	6.001	6.364	0,94	6,00
15	436	173	34	33	1,03	7.913	8.181	7.844	1,04	3,81
16	512	399	28	29	0,97	35.837	4.838	4.669	1,04	1,90
17	555	480	110	91	1,21	56.322	29.904	28.976	1,03	8,55
18	861	833	79	71	1,11	100.560	45.961	45.525	1,01	13,18
19	918	840	46	49	0,94	124.607	21.479	20.369	1,05	9,97
Totais		4,846	567	548	1,03	443.659	147.624	145.105	1,02	7,83

Tabela VI-1

diâmetros, em apenas duas iterações (duas execuções do passo 3).

Embora a complexidade em tempo de execução desse esquema, para um grafo geral, seja de $O(n\ (n + m))$, é inegavel a sua superioridade sobre um processo exaustivo de determinação de estruturas de níveis de maiores profundidades, como o usado no algoritmo RCM (passo A.2).

O algoritmo V tenta construir uma estrutura de níveis $N(G)$ de menor largura, a partir das duas estruturas de níveis com raiz, $N_v(G)$ e $N_u(G)$, geradas pelo algoritmo IV. Na execução do algoritmo V, a única característica considerada, dos componentes conexos (passo 3), é a cardinalidade de seus conjuntos de vertices, pois determina a ordem em que tais componentes são manipulados para a inclusão de seus vertices em $N(G)$.

Desde que, para a inclusão de um componente conexo, não são consideradas propriedades dos componentes conexos ainda não incluidos em $N(G)$, a estrutura de níveis gerada pode, evidentemente, ser mais larga que as duas estruturas de níveis fornecidas como entrada ao algoritmo - o que pode significar uma largura de banda maior que a obtida pela aplicação do algoritmo RCM, diretamente, em uma das duas estruturas de níveis de entrada. Isso, efetivamente, ocorre nos casos 1,5,10,12,13,14,16 e 19, na tabela (VI-1).

O algoritmo VI produz uma numeração dos vertices de um grafo $G = (V,E)$, de uma maneira análoga a utilizada pelo algoritmo RCM, a partir da estrutura de níveis $N(G)$ gerada pelo algoritmo V e, das adjacencias do grafo. Porém, nesse caso, a numeração obtida é invertida (passo 4), apenas, quando, no algoritmo V, houver sido utilizado o primeiro elemento dos pares de níveis associados, para a inclusão dos vertices do componente C_1 em $N(G)$. Ao contrario do algoritmo RCM, que sempre inverte (passo B) a numeração obtida no passo A.

A utilização do primeiro elemento dos pares de níveis associados, para um componente conexo C_i, significa que os vertices de tal componente são incluidos em $N(G)$ exata

mente como aparecem em $N_{\bar{v}}(G)$*, ao passo que a utilização do segundo elemento dos pares de níveis associados, significa que os vertices de tal componente são incluidos em $N(G)$ na forma inversa (os números dos níveis de $N_{\bar{u}}(G)$ são invertidos) daquela em que aparecem em $N_{\bar{u}}(G)$.

Desse modo, de acordo com seu passo 4, o algoritmo GPS presupõe o componente C_1 predominante sobre os outros componentes conexos do grafo, considerando os resultados do corolário IV.6., apenas em relação ao componente C_1. Isto é, admite como valida a seguinte afirmativa:

(VI-1) $\quad \text{Env}\ (C_r) \subseteq \text{Env}\ (C_d) \Rightarrow \text{Env}\ (A_r) \subseteq \text{Env}\ (A_d)$

onde C_d e A_d são as matrizes obtidas, respectivamente, através das numerações ϕ produzidas no passo 3 do algoritmo VI, para os grafos C_1 e G; e, C_r e A_r são as matrizes obtidas invertendo-se, respectivamente, as numerações $\phi(C_1)$ e $\phi(G)$.

Para o grafo G da figura (IV-5), o componente conexo C_1 (figura (V-3)) é, realmente, predominante sobre os outros dois, pois $|C_1| = 3$, $|C_2| = 1$ e $|C_3| = 1$, e o passo 4 do algoritmo VI não foi, então, executado porque, no algoritmo V, foram utilizados os seguintes elementos dos pares de níveis associados para a inclusão do componente C_1. A implicação (VI-1), porém, não é sempre verdadeira.

Nos testes 1,2,4,8,9,11,13 e 14, da tabela (VI-1), o perfil obtido é maior quando se utiliza o algoritmo GPS.

Do que foi exposto a respeito dos algoritmos RCM e GPS, conclui-se que:

(i) O algoritmo GPS possibilita, em alguns casos, a obtenção de uma largura de banda e/ou perfil menores que os obtidos pelo algoritmo RCM, não se podendo, a priori, afirmar, entretanto, se isso ocorre para determinado grafo. Há casos, inclusive, em que o melhor resultado seria obtido através do algoritmo RCM.

---

* $\bar{v}$ é o vertice de menor grau dentre u e v, e $\bar{u}$ é o vertice de maior grau dentre u e v.

(ii) O algoritmo IV de determinação de vertices pseudo-periféricos é, efetivamente, o aperfeiçoamento mais importante do algoritmo GPS, sendo o único responsável pelos menores tempos de execução obtidos nos testes da tabela (VI-1)*.

Um esquema de numeração tipo RCM, que utiliza o algoritmo IV para gerar a estrutura de níveis teria, ainda, complexidade em tempo de execução de $O(n\ (n + m))$**, por causa do algoritmo IV. Porém, a numeração RCM consumiria um tempo de $O(n)$, uma vez que o vetor grau teria sido gerado no passo 1 do algoritmo IV.

Um tal esquema é apresentado na seção VI.3., e tanto esse como o esquema GPS, são viáveis como algoritmos de redução de largura de banda (ou perfil) de matrizes esparsas. O segundo esquema pode gerar uma menor (em alguns casos, maior) largura de banda (ou perfil), enquanto o primeiro possui um passo de numeração de menor complexidade que os algoritmos V e VI.

VI.3. Combinação dos Algoritmos III e IV

Considerando que a finalidade de um algoritmo de ordenação de vertices é a determinação de uma numeração que produza uma menor largura de banda, torna-se mais importante gerar estruturas de níveis mais estreitas do que, efetivamente, encontrar vertices pseudo-periféricos (que, não necessariamente, levam a estruturas de níveis mais estreitas).

Como decorrência desse fato, em implementações do algoritmo IV ([24],[9]) estão incluidos procedimentos que rejeitam as estruturas de níveis mais largas (interrompendo sua geração), tão logo elas sejam detetadas, reduzindo assim, o tempo de execução desse algoritmo. A consequência, dessa

---

* Nos 19 testes citados o algoritmo RCM gera de 10 a 20 vezes mais estruturas de níveis que o algoritmo GPS num tempo, na média, 7,8 vezes maior.

** $O(k\ (n + m))$, onde k é o número de estruturas de níveis geradas, o que, no pior caso, é de $O(n)$.

rejeição de estruturas de níveis, é que o algoritmo IV pode determinar vertices que não sejam, efetivamente, pseudo-periféricos, além de afetar (melhorando ou piorando, dependendo do grafo) a qualidade da numeração produzida em termos ou de largura de banda ou de perfil, sendo garantido, entretanto, que o algoritmo para, com uma estrutura de níveis de menor largura, dentre as geradas por ele.

Alan George [9] incorporou, ainda, em sua implementação desse algoritmo, uma outra modificação que, em alguns casos, pode reduzir significativamente seu tempo de execução. Para alguns problemas, o conjunto $N_{k(v)}(v)$ pode ser muito grande, e a modificação introduzida consiste em colocar no conjunto S (passo 3), apenas, vertices representativos desse conjunto $N_{k(v)}(v)$, isto é, apenas um dos vertices de menor grau de cada componente conexo do subgrafo
$C\ (N_{k(v)}) = (N_{k(v)}, E\ (N_{k(v)}))$.

A determinação dos componentes conexos de $C\ (N_{k(v)})$ requer um tempo de execução de $0\ (n + m)$, representando um aumento no tempo de execução, unicamente, se $E\ (N_{k(v)}) = \emptyset$.

Torna-se difícil uma avaliação dessa modificação, para um grafo geral, devido a natureza iterativa do algoritmo que gera sucessivos conjuntos S. Pode-se afirmar, entretanto, que estruturas de níveis com raizes em vertices de um mesmo componente conexo, com cerca de $0(k(v))$ vertices, podem ser completamente diferentes entre si. E isso parece indicar, que a modificação sugerida por Alan George pode afetar em muito a qualidade da numeração obtida.

VI.3.1. Esquema Combinado de Numeração

No esquema de numeração apresentado aqui, está incorporado o procedimento de rejeição de estruturas de níveis mais largas, tão logo elas sejam detetadas.

Além desse, foi incorporado um novo procedimento que permite reduzir a cardinalidade dos conjuntos S e consiste em se colocar em S, apenas, um dos vertices de menor

grau em $N_{k(v)}(v)$ cujo pai relativo a $N_v(G)$ é x, para cada x em $N_{k(v)-1}(v)$.

Esse último procedimento pode ser menos efetivo que o proposto por George, porém, tem as seguintes vantagens:

- Como será visto mais adiante, não implica em nenhum processamento adicional, exceto pela ordenação inicial da estrutura de adjacencias, e
- dificilmente a qualidade da numeração obtida seria prejudicada, devido a semelhança entre as estruturas de níveis com raizes em vertices de mesmo pai.

No passo 1, em vez de ordenar o conjunto V de vertices, ordena-se a estrutura de adjacencias do grafo, para permitir que, no passo 3, os vertices em $N_{k(v)}(v)$ de mesmo pai sejam acessados, diretamente, por ordem crescente de grau. A ordenação da estrutura de adjacencias simplifica, também, o passo de numeração dos vertices.

Esse algoritmo utiliza uma matriz ESTNIVEL [2,N] para armazenar as estruturas de níveis $N_v(G)$ e $N_r(G)$, calculadas pelo algoritmo A3.

O algoritmo A3 deve possuir um parâmetro que indique a largura de nível máxima permitida para uma estrutura de nível, interrompendo-se a geração de estruturas de níveis mais largas.

São utilizados, ainda, o vetor pai, a estrutura de adjacencias e o vetor de numeração δ. O conjunto S não precisa ser, efetivamente, armazenado, de acordo com a análise de complexidade do algoritmo IV.

Algoritmo VII: Esquema de numeração derivado dos algoritmos GPS e RCM.

Entrada: Grafo G(A) = (V,E), representado na forma de uma estrutura de adjacencias.

Saída: Uma numeração δ dos vertices de V.

Metodo:

1. Ordenar a estrutura de adjacencias do grafo pelo algoritmo A1.
   Escolher um vertice v de grau mínimo.

2. Gerar a estrutura de níveis (algoritmo A3)
   $N_v(G) = \{N_1(v), N_2(v), \ldots, N_{k(v)}(v)\}$, com raiz em v e profundidade $k(v)$, calculando sua largura $w(N_v)$.

3. Construir o conjunto S da seguinte forma:
   Percorrer o conjunto $N_{k(v)}(v)$, incluindo em S o primeiro* dos vertices de mesmo pai encontrado, para cada possível vertice pai.

4. Para cada vertice r em S gerar**, pelo algoritmo A3, a estrutura de níveis $N_r(G) = \{N_1(r), N_2(r), \ldots, N_{k(r)}(r)\}$ de largura $w(N_r)$, e retirar r de S.
   Se $k(r) > k(v)$, fazer v <— r e ir para o passo 3.

5. Quando, no passo 4, S for vazio, numerar os vertices de V como se segue:
   for i <— 1 until n do ⌊ [ESTNIVEL [v,i]] <— i;

Exemplo VI-1: Para o grafo da figura (IV-5), se o vertice m for escolhido como vertice inicial, de acordo com a figura (IV-6), o conjunto S construido no passo 3 teria os vertices a e c, não sendo gerada a estrutura de níveis com raiz em b.

De acordo com as análises de complexidade dos algoritmos I, II e IV, o espaço de memória requerido por esse algoritmo é de $0(n + m)$, e a complexidade global em tempo de execução é de $0(n(n + m))$.

---

* Se já tiver sido gerada uma estrutura de níveis com raiz nesse vertice, nenhum vertice de mesmo pai é colocado em S.

** A geração de uma estrutura de níveis $N_r(G)$ deve ser interrompida no momento em que for detetado que $w(N_r) > w(N_v)$.

VI.3.2. Formulação Tipo Algol

Estrutura de Dados

No passo 1, a estrutura de adjacencias é ordenada e o vetor GRAU, então calculado, não é novamente utilizado.

Duas estruturas de níveis devem estar na memória ao mesmo tempo, sendo armazenadas na matriz ESTNIVEL [2, N]. Os pais dos vertices, relativos as duas estruturas de níveis são armazenados na matriz PAI [2, N].

O conjunto S não é, efetivamente, armazenado e K, W e WK são vetores de dois elementos que guardam a profundidade, a largura e, a largura do último nível, respectivamente, de cada uma das estruturas de níveis na memória.

É suposto que a interrupção na geração de estruturas de níveis mais largas seja feita pelo algoritmo A3.

Descrição Tipo Algol do Algoritmo VII

```
begin
  V <- 1;
  R <- 2;
  comment 1. Ordenação da estrutura de adjacencias do grafo.
  Executar o algoritmo A1, escolhendo um vertice x de grau mí-
  nimo.
  comment 2. Geração da estrutura de níveis com raiz em x.
  Executar o algoritmo* A3, gerando em ESTNIVEL [V,*] a estru-
  tura de níveis com raiz em x, profundidade K [V], largura
  W [V] e, largura do último nível igual a WK [V]. O vetor
  PAI [V] gerado pelo algoritmo A3, é também utilizado.
  LMAX <- W [V];
  CALICE <- ∅;
  I <- N - WK [V] + 1;
  for I <- I until N do
    begin
```

---

* O algoritmo A3 deve interromper a geração da estrutura de níveis, tão logo detete ter ela largura maior que LMAX. Nesse caso, W [R] indica a maior largura de nível gerada.

```
      comment 3. Escolha do próximo vértice de S.
      if CALICE ≠ PAI [V,ESTNIVEL[V,I]] then
        begin
          CALICE <- PAI[V,ESTNIVEL[V,I]];
          if não foi gerada estrutura de níveis com raiz em
            ESTNIVEL[V,I] then
            begin
              comment 4. Geração da estrutura de níveis com
              raiz no vértice ESTNIVEL[V,I].
              Executar o algoritmo A3, gerando em ESTNIVEL[R,*]
              a estrutura de níveis com raiz em ESTNIVEL[V,I], profundi
              dade K[R]. O vetor PAI[R,*] é também gerado pelo
              algoritmo A3.
              if W[R] ≤ LMAX then
                if K[R] > K[V] then
                  begin
                    V <- R;
                    R <- 3 - V;
                    CALICE <- ∅;
                  end;
            end;
        end;
    end;
  comment 5. Numeração dos vértices do grafo
  if W[R] < W[V] then V <- R;
  for I <- 1 until N do F[ESTNIVEL[V,I]] <- I;
end;
```

VII. CONCLUSÕES

Um esquema de numeração RCM que utiliza o algoritmo IV para geração de estruturas de níveis, bem como, o esquema GPS se mostraram viáveis como algoritmos de redução de largura de banda (ou perfil) de matrizes esparsas simetricas.

Matrizes provenientes da aplicação do metodo de elementos finitos são, frequentemente, esparsas com m de $0(n)$. Para esse tipo de matriz aqueles algoritmos, no pior caso, consomem tempos de $0(n^2)$. E, ainda, quando para a obtenção da numeração forem geradas apenas k estruturas de níveis ($k << n$), o tempo de execução será, evidentemente, de $0(n)$.

Os custos provenientes desses processos de numeração podem ser pouco significativos, quando muitos problemas tendo a mesma estrutura zero x não-zero tiverem que ser resolvidos, ou em problemas dependentes do tempo, em que a mesma matriz de coeficientes é usada com vários vetores b (termo independente) distintos.

Uma caracterização de casos típicos de elementos finitos seria importante para permitir uma análise de caso medio desses algoritmos, bem como, para permitir o estudo de novas heurísticas específicas para determinados tipos de malhas. O desenvolvimento de bons esquemas de ordenação para casos especiais de malhas de elementos finitos, e o estudo teórico de heurísticas parecem ser áreas frutíferas para futuras pesquisas. A modificação sugerida por Alan George [9] de determinação dos componentes conexos dos vertices em $N_{k(v)}$ poderia então ser estudada para certos casos típicos.

O objetivo maior do uso de esquemas de numeração, que é reduzir o número de operações no processo de resolução do sistema de equações (Cholesky, Gauss, etc.), pode ser alcançado com base em outros enfoques diferentes.

Endre Tarjan at al. apresentam em [23], um metodo interessante que consiste em determinar diretamente uma ordenação que produza um preenchimento reduzido (não mínimo) da matriz durante o processo de eliminação de Gauss.

Um método de particionamento de matrizes é discutido em ([9],[25]). Utiliza o esquema de determinação de vértices pseudo-periféricos, e o esquema RCM para numerar as partições.

Outros métodos são ainda discutidos em [9] e [26].

Os problemas de minimização de largura de banda e de perfil são apresentados como NP-Completos em [17]. O problema de minimização de preenchimento de matrizes não simétricas é mostrado ser NP-Completo em [28], e em [23] se conjectura que o mesmo problema, porém para matrizes simétricas seja, também, NP-Completo.

Conjecturamos que os problemas de minimização de largura de banda e de perfil para matrizes, cujos grafos são árvores, sejam também NP-Completos, a despeito de não termos obtido nenhum resultado concreto nessa direção.

BIBLIOGRAFIA

1 RAUPP, Marco A. - Análise Numérica e Equações Diferenciais. Primeira Escola Brasileira de Matematica Aplicada, Laboratório de Cálculo - CBPF, Rio de Janeiro, 1978. 181 p.

2 WILKINSON, J. H. - The Algebraic Eigenvalue Problem. Clarendon Press, London, 1965.

3 AHO, A. V.; HOPCROFT J. E. e ULLMAN J. D. - The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Mass., 1974. 470 p.

4 CUTHILL, Elizabeth e McKEE, J. - Reducing the Bandwidth of Sparse Symmetric Matrices. ACM Proc. of 24th National Conference, Association of Computing Machinery, New York, 157-172, 1969.

5 GEORGE, J. Alan - Computer Implementation of the Finite Element Method. Tech. Rep. STAN-CS-71-208, Computer Science Dept., Stanford Univ., California, 1971.

6 CUTHILL, Elizabeth - Several Strategies for Reducing the Bandwidth of Matrices. Sparse Matrices and their Applications, Willoughby, eds., Plenum Press, New York, 157-166, 1972.

7 LIU, Wai-Hung e SHERMAN, Andrew H. - Comparative Analysis of the Cuthill-McKee and the Reverse Cuthill-McKee Ordering Algorithms for Sparse Matrices. SIAM J. Numer. Analysis, 13(2): 198-213, 1976.

8 GIBBS, Norman E.; POOLE JR., William G. e STOCKMEYER, Paul K. - An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix. SIAM J. Numer. Analysis, 13(2): 236-250, 1976.

9 GEORGE, J. Alan - Solution of Linear Systems of Equations: Direct Methods for Finite Element Problems. Sparse Matrix Techniques em Lecture Notes in Mathematics, 572: 53-101, 1976.

10 ROSEN, Richard - Matrix Bandwidth Minimization. ACM Proc.

of 23th National Conference, Association of Computing Machinery, New York, 585-595, 1968.

11 CHENG, K. Y. - Minimizing the Bandwidth of Sparse, Symmetric Matrices. Computing 11: 103-110, 1973.

12 CHENG, K. Y. - Note on Minimizing the Bandwidth of Sparse, Symmetric Matrices. Computing 11: 27-30, 1973.

13 GEORGE, J. Alan e LIU, Wai-Hung - A Note on Fill for Sparse Matrices, SIAM J. Numer. Analysis, 12(4): 452-455, 1975.

14 JENNINGS, A. - Matrix Computation for Engineers and Scientists. Queen's University, Belfast, John Wiley & Sons.

15 JENNINGS, A. - A Compact Storage Scheme for the Solution of Simultaneous Equations. Computer Journal 9: 281-285, 1966.

16 ZAMBARDINO, R. A. - Decomposition of Positive Definite Symmetric Band Matrices. Computer Journal 13: 421-422, 1970.

17 PAPADIMITRIOU, Ch. H. - The NP-Completeness of the Bandwidth Minimization Problem. Computing, 16: 263-270, 1976.

18 TARJAN, Robert Endre - Complexity of Combinatorial Algorithms. SIAM Review 20(3): 457-491, 1978.

19 REINGOLD, Edward M.; NIEVERGELT Jurg e DEO Narsingh - Combinatorial Algorithms Theory and Practice. Prentice-Hall, Inc., New Jersey, 1977. 433 p.

20 GAREY, M. R.; JOHNSON, D. S. e STOCKMEYER, L. Some Simplified NP-Complete Problems. ACM Proc. of 6th Annual Symposium on Theory of Computing, 47-63, 1974.

21 BATHE, K. e WILSON, E. L. - Numerical Methods in Finite Element Analysis. Prentice-Hall, Inc., New Jersey, 1976. 528 p.

22 DEO, Narsingh - Graph Theory with Applications to Engineering and Computer Science. Prentice-Hall,Inc.,

New Jersey, 1974. 478 p.

23 ROSE, Donald J.; TARJAN, Robert Endre e LUEKER, George S.- Algorithms Aspects of Vertex Elimination on Graphs, SIAM J. Comp. 5(2): 266-283, 1976.

24 CRANE JR., H. L.; GIBBS, N. E.; POOLE, W. G. e STOCKMEYER, P. K. - Matrix Bandwidth and Profile Minimization, Report nº 75-9, ICASE Report (1975).

25 GEORGE, Alan e LIU, Joseph W. H. - Algorithms for Matrix Partitioning and teh Numerical Solution of Finite Element Systems. SIAM J. Numer. Analysis, 15(2): 297-327, 1978.

26 GEORGE, Alan e McINTYRE, David R. - On the Application of the Minimun Degree Algorithm to Finite Element Systems. SIAM J. Numer. Analysis, 15(1): 90-112, 1978.

27 POMPO, Heloisa R., Representação de Grafos em Computador, Tese de Mestrado, COPPE/UFRJ, 1979.

28 ROSE, D. e TARJAN, R. - Algorithmic Aspects of Vertex Elimination on Directed Graphs. SIAM J. Comp., to appear.