APLICAÇÃO DE GRAFOS CORDAIS A SISTEMAS LINEARES ESPARSOS

Moisés Teles Carvalho Junior

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

Aprovada por:

Prof. Nelson Maculan Filho, D.Sc

Prof. Márcia Helena Costa Fampa, D.Sc.

Prof. Fábio Protti, D.Sc.

Prof. Luiz Satoru Ochi, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
NOVEMBRO DE 2002

CARVALHO JUNIOR, MOISÉS TELES

Aplicação de Grafos Cordais a Sistemas Lineares Esparsos [Rio de Janeiro] 2002

VII, 90 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2002)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Eliminação de Gauss

2 - Grafos Cordais

3 - Programação Linear

I. COPPE/UFRJ II. Título (série)

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

APLICAÇÃO DE GRAFOS CORDAIS A SISTEMAS LINEARES ESPARSOS

Moisés Teles Carvalho Junior

Novembro/2002

Orientadores: Nelson Maculan Filho
Marcia Helena Costa Fampa

Programa: Engenharia de Sistemas e Computação

Apresentamos um método para tornar um grafo conexo em cordal, através da inserção de novas arestas no grafo. A principal motivação deste problema é resolver sistemas lineares esparsos de grande porte através do método de eliminação de Gauss. O método de eliminação de Gauss, quando aplicado a esses sistemas esparsos, provoca, em geral, preenchimento na matriz associada. Consideraremos o problema supondo a matriz quadrada, simétrica e definida positiva, e a representaremos por um grafo. Nosso objetivo é tornar esse grafo cordal com a inserção do menor número de arestas possível. Com relação à resolução do sistema linear através do método de eliminação de Gauss, isso é equivalente a encontrar uma ordem de pivoteamento na matriz que acarretará num menor preenchimento com elementos não-nulos. Apresentaremos um modelo de programação linear inteira mista para o problema do preenchimento mínimo em um grafo conexo para torná-lo cordal e por fim, apresentaremos alguns resultados numéricos obtidos através do *software* XPRESS-MP.

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

CHORDAL GRAPHS WITH APPLICATIONS TO SPARSE LINEAR SYSTEMS

Moisés Teles Carvalho Junior

November/2002

Advisors: Nelson Maculan Filho
Marcia Helena Costa Fampa

Department: Computing and Systems Engineering

We present a method to transform a connected graph into a chordal graph, by adding edges into the graph. The main motivation of this fact is to solve large sparse linear systems using Gauss elimination method. Generally, when this method is applied to sparse systems it causes the associated matrix to be fulfilled. We consider a problem with square, symmetric, positive definite matrix problem and we represent it by a graph. Our aim is to make this graph became a chordal graph, by adding the minimum amount of edges. Solving the linear system using Gauss elimination is equivalent to find the pivoting order in the matrix that results in a smaller fulfillment with nonzero elements. We present a linear mixed integer programming model for the problem of minimum fulfillment in a connected graphs in order to transform it into a chordal graph. Finally, we present some numerical results, obtained with the software XPRESS-MP.

Agradecimentos

A Deus.

A minha mãe Maria, pela dedicação, incentivo e companheirismo durante esse período de trabalho e estudos. Também pela imensa compreensão, incompreensível, nos momentos mais difíceis.

A todos os meus parentes, próximos ou distantes, que torceram pelo sucesso desse trabalho.

Ao meu ilustre orientador Nelson Maculan Filho, por sua atenção, incentivo e principalmente pelo exemplo, durante o período do curso de mestrado, e também pela confiança por ter aceito minha orientação.

A minha professora Isabel Lugão Rios gostaria de agradecer de forma muito especial, não somente por ter me orientado como monitor na graduação, mas também pela ajuda e confiança ao me recomendar para o mestrado. Gostaria também de agradecer pela paciência, compreensão e sobretudo pela amizade, que foram de extrema importância para mim. Será sempre um exemplo de postura a ser seguido.

À professora Renata Raposo Del-Vecchio, pela conduta de profissional e também amizade, meu agradecimento e homenagem.

A todos os meus professores de graduação que contribuíram para minha formação.

A todos os amigos. Muito obrigado. Em especial aos novos, e grandes, amigos conquistados durante o curso: TIBÉRIUS DE OLIVEIRA E BONATES, Cláudio Prata Santiago, Henrique Limaverde Cabral de Lima, Élder Magalhães Macambira e Leonardo Gomes Holanda, componentes da "ban-

cada cearense". Muito do que aprendi, foi com a ajuda deles.

Aos amigos Alexandre Soares Alves, Selma Foligne Crespio, Shane Aparecida e Ana Mirtes "Rosinha" Fouro, companheiros de inúmeros almoços e horas de estudos. Mesmo que hoje distantes, são, e serão, sempre lembrados como grandes amigos. Sempre tinham uma palavra de apoio nos momentos difíceis.

A todos os colegas de estudos, funcionários e professores do Programa. De forma especial, às secretárias Cláudia Prata e Solange, à Mercedes, à Sônia, à Sueli e à Lúcia, que sempre demostraram extrema boa vontade quando precisei.

Aos professores Claudson Ferreira Bornstein pela sugestão do tema, e colaboração com este trabalho, e a Márcia Cerioli, sua atenção por alguns momentos valeram por dias de estudos.

As minhas grandes amigas Denise Candal Reis Fernandes, minha "mãe paraguaia", e a Patrícia Regina de Abreu Lopes. Sempre que precisei estavam dispostas a me ajudar, tanto na realização deste trabalho, quanto nos problemas pessoais, que não foram poucos. Algumas amizades são boas, mas passageiras, outras são duradouras e necessárias, as amizades de Denise e Patrícia são extremamente necessárias.

Ao meu "irmãozinho" Wallace Alves Salgueiro Júnior. Mesmo não participando diretamente, sua amizade foi fundamental para a conclusão deste curso. E será fundamental para minha vida.

A Adriane de Quevedo Cardozo, que mesmo sem perceber, despertou em mim uma pessoa que mesmo eu não conhecia. Um Dia Conheci Uma Menina, como Ventos Do Sul que sopram nos Pampas.

A minha também orientadora Marcia Helena Costa Fampa, que aceitou o desafio da minha orientação. Sua conduta, amizade e compreensão foram de

inestimável valia. Gostaria de agradecer também sua generosidade comigo, que mesmo sabendo das dificuldades, não desanimou, e não me deixou desanimar. Poucos teriam tal coragem.

A COPPE/UFRJ, pela oportunidade de desenvolver este trabalho.

A Capes pela bolsa de estudos concedida, durante o período do curso de mestrado.

Aos Engenheiros do Hawaii.

Conteúdo

Capítulo 1

Introdução

1.1 Motivação - Resolução de sistemas lineares

Vários problemas em diversas áreas, como por exemplo cálculo estrutural em engenharia, pesquisa operacional, na solução de problemas de otimização, como escalonamento de máquinas em uma fábrica, são modelados por sistemas lineares.

Para a resolução desses sistemas lineares existem alguns métodos, diretos, como eliminação de Gauss e Gauss-Jordan, ou iterativos, como Gradiente Conjugado e método de Gauss-Seidel, entre outros.

Um dos mais utilizados e eficientes métodos conhecidos é o método de Eliminação de Gauss, onde tornamos a matriz associada ao sistema, triangular superior através de escalonamento, e a partir dessa matriz triangular, calculamos o valor de cada componente utilizando o resultado das componentes anteriores.

Em muitas aplicações práticas, esses sistemas são grandes e esparsos, ou seja, existem muitos elementos nulos na matriz associada.

Sendo assim, para armazenarmos a matriz no computador nos utilizamos da sua esparsidade, obtendo assim uma economia em espaço de memória, ou seja, armazenamos apenas os elementos não-nulos da matriz associada ao sistema linear.

Com isso, é desejável que, uma vez armazenada a matriz, não tenhamos, durante o processo de eliminação de Gauss, que aumentar o número de elementos não-nulos armazenados no computador.

1.2 Objetivo

O método de Eliminação de Gauss consiste na escolha de um pivô, elemento $a_{i,j}$ da matriz A, e a partir dele tornamos nulos todos os elementos de sua coluna j, que estão abaixo dele na coluna.

Isto é feito através de operações elementares, soma ou subtração, da linha do pivô com múltiplos das demais linhas da matriz.

Este método pode acarretar em um preenchimento da matriz com elementos não-nulos, de acordo com a ordem escolhida dos pivôs, e este preenchimento ocorrerá nas linhas, excetuando-se a linha do pivô.

Cada ordenação dos pivôs do sistema linear poderá criar diferentes preenchimentos e nosso obejtivo é minimizar o número de elementos nulos que se tornam não-nulos na matriz durante a resolução do sistema linear, ou seja, definir uma ordem de pivoteamento da matriz que minimize o preenchimento.

Para isso, iremos utilizar uma relação entre a matriz associada ao sis-

tema linear e grafos. No restante deste trabalho suporemos que a matriz é quadrada, simétrica e definida positiva, evitando assim que tenhamos pivôs nulos.

Essa relação entre matrizes e grafos, se dará da seguinte forma: toda matriz quadrada $A \in \Re^{n \times n}$, simétrica, de um sistema linear $Ax = b$ pode ser vista como um grafo não-direcionado $G = (V, E)$.

Cada nó de G está associado a uma coluna, e uma linha, de A, e existe uma aresta (i, j) no grafo G se e somente se o elemento $a_{i,j}$ da matriz A é diferente de zero.

Nesse contexto, o problema de escolher uma ordem de pivoteamento no método de eliminação de Gauss pode ser visto como um problema de preenchimento de um grafo até torná-lo cordal, pois esta classe específica de grafos nos permite encontrar uma ordem de pivoteamento na matriz onde não ocorre preenchimento.

Nosso objetivo é encontrar o preenchimento mínimo de um grafo para torná-lo cordal. Encontrar esse preenchimento mínimo é um problema NP-completo [23].

Para resolvermos o problema de preenchimento de um grafo, utilizaremos algumas estruturas de grafos, como árvore de eliminação, grafos de interseção e subárvores de uma árvore de eliminação.

Finalmente, o modelaremos como um problema de programação linear inteira mista, e os resultados computacionais para o modelo serão obtidos através do uso do *software* XPRESS-MP.

1.3 Organização do Texto

No capítulo 2, faremos uma revisão do método de Eliminação de Gauss, mostrando cada passo deste método, e ainda a relação entre as matrizes associadas a esses sistemas com grafos.

No capítulo seguinte, introduziremos alguns conceitos sobre grafos, e mais especificamente sobre grafos cordais, e além disso, apresentaremos também conceitos sobre grafos de interseção, que serão necessários para a introdução de árvores de eliminação e subárvores de uma árvore de eliminação.

No capítulo 4, apresentaremos os métodos aproximados existentes mais utilizados para o problema de minimizar o preenchimento de um grafo para torná-lo cordal.

Ilustraremos a metodologia proposta através de figuras e mostraremos o modelo de programação linear inteira mista propriamente dito no capítulo cinco.

No capítulo 6, consideraremos os métodos de solução do problema de programação linear e apresentaremos resultados numéricos, obtidos através do pacote para resolução de problemas de programação linear XPRESS-MP.

Por fim, concluiremos o nosso trabalho, e discutiremos algumas idéias sobre o problema e possíveis direcionamentos para trabalhos futuros.

Capítulo 2

Método de Eliminação de Gauss

Neste capítulo, apresentaremos detalhadamente o método de Eliminação de Gauss para resolução de sistemas lineares.

Em seguida, mostraremos a dificuldade de aplicação deste método a sistemas cujas matrizes associadas são de grande porte e esparsas.

Finalmente, definiremos um grafo G associado a matriz dos coeficientes de um dado sistema, com a finalidade de apresentar uma técnica para contornar essa dificuldade.

2.1 Eliminação de Gauss e escalonamento de matrizes

O método de Eliminação de Gauss consiste em transformar o sistema linear original num sistema linear equivalente com a matriz dos coeficientes triangular superior, pois estes são de resolução imediata, como pode ser visto

em [13].

Seja $Ax = b$ um sistema linear, onde $A \in M(m \times n)$, $x \in \Re^n$ e $b \in \Re^m$. Para resolver tal sistema, devemos tornar a matriz de coeficientes A escalonada, ou triangular superior.

Essa matriz será chamada de matriz escalonada quando o primeiro elemento não-nulo de cada uma de suas linhas está à esquerda do primeiro elemento não-nulo de cada uma das linhas seguintes e, além disso, as linhas nulas, caso existam, estão abaixo das demais.

Com esta definição, podemos dizer que as linhas não-nulas de uma matriz escalonada são vetores linearmente independentes, ou seja, uma matriz escalonada $r \times n$ tem posto r se suas linhas forem todas diferentes de zero.

Exemplo 1: As matrizes abaixo são escalonadas:

$$A = \begin{bmatrix} 1 & 3 & 7 & 2 \\ 0 & 2 & 5 & 1 \\ 0 & 0 & 0 & 3 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 1 & 2 & 3 & 1 \\ 0 & 0 & 4 & 5 & 2 \\ 0 & 0 & 0 & 6 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Ambas têm posto 3.

Dados os vetores $v_1, \ldots, v_m \in \Re^n$, vamos alterá-los passo a passo de tal modo que, em cada etapa, os vetores obtidos geram o mesmo subespaço que os da etapa anterior e, no final, os vetores resultantes formam as linhas de uma matriz escalonada.

Os vetores não-nulos dentre eles formarão uma base do subespaço gerado pelos vetores originalmente dados.

Para alterarmos os vetores, utilizaremos as chamadas operações ele-

mentares, que levam os vetores $v_1, \ldots, v_m \in \Re^n$ em vetores $v'_1, \ldots, v'_m \in \Re^n$ que geram o mesmo subespaço: $S(v'_1, \ldots, v'_m) = S(v_1, \ldots, v_m)$.

Essas operações elementares são descritas abaixo:

1- Trocar a posição de dois vetores v_i, v_j ($i < j$) na lista dada. Esta operação é esquematizada como

$$(v_1, \ldots, v_i, \ldots, v_j, \ldots, v_m) \mapsto (v_1, \ldots, v_j, \ldots, v_i, \ldots, v_m).$$

2- Multiplicar um dos vetores por uma constante C não-nula. Esquematizamos essa operação assim:

$$(v_1, \ldots, v_j, \ldots, v_m) \mapsto (v_1, \ldots, Cv_j, \ldots v_m).$$

3- Somar a um dos vetores um múltiplo de outro vetor da lista, ou seja, substituir v_j por $v'_j = v_j + \alpha v_i$, $i \neq j$.

Esta operação é justificada por: sejam $V = (v_1, \ldots, v_m)$ e $V' = (v_1, \ldots, v'_j, \ldots, v_m)$, com v'_j como definido acima. Temos que $S(V') \subset S(V)$. Além disso, como $v_j = v'_j - \alpha v_i$, temos que $S(V) \subset S(V')$. Logo V e V' geram o mesmo subespaço, ou seja, $S(V) = S(V')$.

Em termos de matriz cujas linhas são os vetores dados, estas operações elementares se exprimem assim:

1- Trocar a posição de duas linhas;

2- Multiplicar uma linha por uma constante não-nula.

3- Somar a uma linha um múltiplo de outra linha.

Assim, o subespaço gerado pelas linhas de uma matriz não se altera quando essas operações elementares são aplicadas a essa matriz.

A seguir, descreveremos o processo de eliminação (ou escalonamento), o qual, mediante aplicações sucessivas das operações elementares às linhas de uma matriz, produz uma matriz escalonada. O procedimento é o seguinte:

A) Se $a_{1,1} \neq 0$, o processo começa deixando a primeira linha intacta e somando a cada linha L_i, com $i \geq 2$, a primeira linha multiplicada por $-\frac{a_{i1}}{a_{1,1}}$.

Com isto se obtém uma matriz cuja primeira coluna é ($a_{1,1}$, 0, ... , 0).

B) Se $a_{1,1} = 0$, uma troca de linhas fornece uma matriz com $a_{1,1} \neq 0$, desde que a primeira coluna não seja nula. Se, porém, todos os elementos da primeira coluna são iguais a zero, passa-se para a segunda coluna ou, mais geralmente, para a coluna mais próxima, à direita da primeira, onde haja algum elemento não-nulo e opera-se como antes, de modo a obter uma matriz cuja primeira coluna não-nula começa com um elemento não-nulo mas todos os demais são iguais a zero. A partir daí não se mexe mais na primeira linha. Recomeça-se o processo, trabalhando com as linhas a partir da segunda, até obter uma matriz escalonada.

Exemplo 2: Sejam os vetores

$v_1 = (1, 2, 3, 4)$,

$v_2 = (5, 6, 7, 8)$ e

$v_3 = (9, 10, 11, 12)$

em $\Re^4$.

Indicamos abaixo a sequência de operações elementares efetuadas sobre a matriz cujas linhas são estes vetores, conduzindo a uma matriz escalonada

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

A partir da matriz A, efetuaremos $L_2 - 5L_1$ e também $L_3 - 9L_1$:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & -8 & -16 & -24 \end{bmatrix}$$

Seguindo o escalonamento, agora faremos as seguintes operações: $L_3 - 2L_2$:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Como a matriz escalonada final tem duas linhas diferentes de zero, os três vetores dados geram um subespaço de dimensão dois em $\Re^4$ e $z_1 = (1, 2, 3, 4)$, $z_2 = (0, -4, -8, -12)$ formam uma base desse subespaço.

A notação $L_i + \alpha L_j$ adotada no exemplo anterior, e nos exemplos que se seguem, significa que a matriz foi obtida da matriz anterior somando-se à i-ésima linha, o múltiplo αL_j da j-ésima linha. Analogamente usaremos a notação $L_i \leftrightarrow L_j$ para indicar a troca da linha i pela linha j.

Exemplo 3:

Consideremos os vetores $v_1 = (0, 1, 2, 3)$, $v_2 = (2, 1, 3, 0)$, $v_3 = (3, 4, 2, 0)$ e $v_4 = (4, 2, 0, 1)$ em $\Re^4$. Indicamos abaixo a sequência de operações elementares efetuadas sobre a matriz que tem esses vetores como linhas a fim de obter uma matriz escalonada

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 1 & 3 & 0 \\ 3 & 4 & 2 & 0 \\ 4 & 2 & 0 & 1 \end{bmatrix}$$

Trocaremos de posição as linhas L_2 e L_1, $L_2 \leftrightarrow L_1$

$$\begin{bmatrix} 2 & 1 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 3 & 4 & 2 & 0 \\ 4 & 2 & 0 & 1 \end{bmatrix}$$

No próximo passo, faremos $L_3 - \frac{3}{2}L_1$, e também $L_4 - 2L_1$

$$\begin{bmatrix} 2 & 1 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & \frac{5}{2} & -\frac{5}{2} & 0 \\ 0 & 0 & -6 & 1 \end{bmatrix}$$

Agora efetuaremos $L_3 - \frac{5}{2}L_2$, temos então

$$\begin{bmatrix} 2 & 1 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & -\frac{15}{2} & -\frac{15}{2} \\ 0 & 0 & -6 & 1 \end{bmatrix}$$

E finalmente, $L_4 - \frac{4}{5}L_3$, assim

$$\begin{bmatrix} 2 & 1 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & -\frac{15}{2} & -\frac{15}{2} \\ 0 & 0 & 0 & 7 \end{bmatrix}$$

Podemos ver que os vetores $z_1 = (\ 2,\ 1,\ 3,\ 0\ )$, $z_2 = (\ 0,\ 1,\ 2,\ 3\ )$, $z_3 = (\ 0,\ 0,\ -\frac{15}{2},\ -\frac{15}{2}\ )$ e $z_4 = (\ 0,\ 0,\ 0,\ 7\ )$ formam uma base de $\Re^4$.

2.2 Resolução de sistemas lineares

O método de eliminação, embora simples, é uma maneira eficaz de resolver um sistema de m equações lineares, com n incógnitas, apresentado sob a forma matricial $Ax = b$, onde $A \in M(m \times n)$, $x \in \Re^n$ e $b \in \Re^m$.

O sistema $Ax = b$ possui solução se, e somente se, o vetor b, pertence à imagem da transformação linear $\mathcal{A} : \Re^n \to \Re^m$ cuja matriz (nas bases canônicas de $\Re^n$ e $\Re^m$) é A.

Dito de outra maneira, o sistema $Ax = b$ possui solução se, e somente se, o vetor b pertence ao subespaço gerado pelas colunas de A. Isto equivale a dizer que a matriz aumentada $[A; b] \in M(m \times (n + 1))$ tem o mesmo posto que a matriz A do sistema.

Uma transformação linear $\mathcal{A} : E \to F$ é dita injetiva se $\forall x, y \in E$, se $x \neq y$ então $\mathcal{A}(x) \neq \mathcal{A}(y)$ em F.

Uma afirmação mais completa é a seguinte: o sistema $Ax = b$ não possui solução quando $b \notin Im(\mathcal{A})$, possui uma única solução quando $b \in Im(\mathcal{A})$ e $\mathcal{A}$ é injetiva, e possui infinitas soluções quando $b \in Im(\mathcal{A})$ e $\mathcal{A}$ não é injetiva.

Em termos matriciais, o sistema $Ax = b$, com $A \in M(m \times n)$, $x \in \Re^n$ e $b \in \Re^m$, admite as seguintes alternativas:

1) Não possui solução quando o posto da matriz aumentada $[A; b]$ é maior do que o posto de A;

2) Possui uma única solução quando a matriz A e a matriz aumentada $[A; b]$ têm o mesmo posto, igual ao número n de incógnitas;

3) Possui infinitas soluções quando se tem posto $[A; b]$ = posto A = $r < n$. Neste caso, o conjunto das soluções é uma variedade afim de dimensão

$n - r$.

Na prática, essa teoria não propicia reconhecer em qual dos casos se enquadra um sistema dado. Isto se faz através do escalonamento da matriz aumentada do sistema. No caso em que o sistema tem uma única solução, esta pode ser encontrada pelo método de Eliminação de Gauss.

O processo de eliminação se baseia na observação de que ao efetuar uma operação elementar sobre as linhas da matriz aumentada $[A; b]$ obtém-se uma matriz $[A'; b']$ que é a matriz aumentada de um sistema $A'x = b'$, equivalente ao sistema original $Ax = b$.

No final do processo, obtém-se um sistema $A'x = b'$, equivalente ao sistema proposto $Ax = b$, no qual a matriz $[A'; b']$ é escalonada. (Isto é o mesmo que dizer que A' é escalonada). O sistema $A'x = b'$ é facilmente resolvido de baixo para cima: acha-se primeiro o valor da última incógnita, substituindo-a por esse valor na equação anterior e assim por diante.

Exemplo 1: Consideremos o sistema

$$\begin{array}{rcrcrcrcl} & & y & + & 2z & + & 3t & = & 1 \\ 2x & + & y & + & 3z & & & = & 1 \\ 3x & + & 4y & + & 2z & & & = & 1 \\ 4x & + & 2y & & & + & t & = & 1 \end{array}$$

O escalonamento da matriz aumentada é feito abaixo:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 1 \\ 2 & 1 & 3 & 0 & 1 \\ 3 & 4 & 2 & 0 & 1 \\ 4 & 2 & 0 & 1 & 1 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 2 & 1 & 3 & 0 & 1 \\ 0 & 1 & 2 & 3 & 1 \\ 3 & 4 & 2 & 0 & 1 \\ 4 & 2 & 0 & 1 & 1 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 2 & 1 & 3 & 0 & 1 \\ 0 & 1 & 2 & 3 & 1 \\ 0 & \frac{5}{2} & -\frac{5}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & -6 & 1 & -1 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 2 & 1 & 3 & 0 & 1 \\ 0 & 1 & 2 & 3 & 1 \\ 0 & 0 & -\frac{15}{2} & -\frac{15}{2} & -3 \\ 0 & 0 & 0 & 7 & \frac{7}{5} \end{bmatrix}.$$

Obtém-se assim a matriz aumentada do sistema

$$\begin{array}{rcrcrcl} 2x & + & y & + & 3z & & = & 1 \\ & & y & + & 2z & + & 3t & = & 1 \\ & & & & -\frac{15}{2}z & - & \frac{15}{2}t & = & -3 \\ & & & & & & 7t & = & \frac{7}{5}. \end{array}$$

Resolvendo esse sistema de baixo para cima, tem-se: $t = \frac{1}{5}$, $z = \frac{1}{5}$, $y = 0$, $x = \frac{1}{5}$. Esta é a única solução do sistema dado. Como a matriz do sistema tem posto 4, a solução existiria e seria única, fosse qual fosse o vetor b.

Exemplo 2:

Seja o sistema

$$\begin{array}{rcrcrcr} x & + & 2y & - & 3z & = & 4 \\ 2x & + & 3y & + & 4z & = & 5 \\ 4x & + & 7y & - & 2z & = & 12. \end{array}$$

O escalonamento da sua matriz aumentada é o seguinte:

$$\begin{bmatrix} 1 & 2 & -3 & 4 \\ 2 & 3 & 4 & 5 \\ 4 & 7 & -2 & 12 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & -3 & 4 \\ 0 & -1 & 10 & -3 \\ 0 & -1 & 10 & -4 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & -3 & 4 \\ 0 & -1 & 10 & -3 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Vemos portanto que o sistema dado é equivalente a:

$$\begin{array}{rcrcrcr} x & + & 2y & - & 3z & = & 4 \\ & - & y & + & 10z & = & -3 \\ 0x & + & 0y & + & 0z & = & -1 \end{array}$$

O qual é impossível. O sistema dado não tem solução.

Exemplo 3: Seja o sistema

$$\begin{array}{rcrcrcrcr} x & + & 2y & + & 3z & + & 4t & = & 1 \\ 5x & + & 6y & + & 7z & + & 8t & = & 2 \\ 9x & + & 10y & + & 11z & + & 12t & = & 3 \end{array}$$

O escalomento da sua matriz aumentada segue o esquema:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 1 \\ 5 & 6 & 7 & 8 & 2 \\ 9 & 10 & 11 & 12 & 3 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 1 \\ 0 & -4 & -8 & -12 & -3 \\ 0 & -8 & -16 & -24 & -6 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 1 \\ 0 & -4 & -8 & -12 & -3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

A última matriz obtida é aumentada do sistema:

$$\begin{array}{rcrcrcrcr} x & + & 2y & + & 3z & + & 4t & = & 1 \\ & - & 4y & - & 8z & - & 12t & = & -3 \end{array}$$

ou

$$\begin{array}{rcrcrcrcr} x & + & 2y & = & - & 3z & - & 4t & + & 1 \\ & - & 4y & = & & 8z & + & 12t & - & 3 \end{array}$$

Este sistema pode ser resolvido de baixo para cima (esquecendo que z e t são incógnitas) e nos dá a solução:

$$y = -2z - 3t + \frac{3}{4}, \qquad x = z + 2t - \frac{1}{2} \tag{2.1}$$

O sistema dado possui portanto uma infinidade de soluções, que podem ser obtidas atribuindo-se valores arbitrários a z e a t e calculando x e y em função delas por meio destas duas últimas igualdades.

2.3 Esparsidade da matriz A

Inúmeros sistemas lineares, que surgem de problemas práticos como discretização de equações diferenciais por método dos elementos finitos ou método de diferenças finitas e descrição de redes de potência, são de grande porte com matriz dos coeficientes esparsa. Para estes casos, são adotados esquemas especiais para armazenamento da matriz A, que tiram proveito de sua esparsidade.

O método de eliminação de Gauss, quando aplicado a um sistema linear esparso, pode provocar preenchimento na matriz A (veja [21]), isto é, durante o processo de eliminação poderão surgir elementos não-nulos em posições $a_{i,j}$ que originalmente eram nulas.

Seja, por exemplo, a seguinte matriz:

$$\begin{bmatrix} 2 & 2 & 3 & 4 & 1 \\ 2 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Após a primeira etapa do processo de eliminação teremos:

$$\begin{bmatrix} 2 & 2 & 3 & 4 & 1 \\ 0 & -1 & -3 & -3 & -1 \\ 0 & -2 & -1 & -4 & -1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & -2 & -3 & -4 & 1 \end{bmatrix}.$$

As posições $a_{2,3}$, $a_{2,5}$, $a_{3,2}$, $a_{3,4}$, $a_{3,5}$, $a_{5,2}$, $a_{5,3}$ e $a_{5,4}$, que originalmente eram nulas, agora são não-nulas.

Portanto, se a matriz A for esparsa e de grande porte, uma desvantagem do método de eliminação de Gauss para a resolução do sistema linear $Ax = b$ é o preenchimento na matriz.

O objetivo do nosso trabalho é apresentar uma técnica onde será possível definir uma ordem de pivoteamento com a qual se reduzirá, ou mesmo se extinguirá, o preenchimento da matriz A com elementos não-nulos.

Para a melhor compreensão desta técnica, definiremos na próxima seção um grafo associado a matriz dos coeficientes do sistema linear a resolver. Em seguida veremos como as propriedades deste grafo podem sugerir a escolha do pivô a cada iteração do método.

2.4 Eliminação de Gauss e Grafos

A partir desta seção estaremos considerando sistemas lineares $Ax = b$, onde A é uma matriz quadrada, simétrica e definida positiva. Neste caso, o sistema tem uma única solução que pode ser obtida pelo método de Eliminação de Gauss.

Definiremos para uma dada matriz A ($n \times n$) simétrica, o grafo G (V, E), tal que $\mid V \mid = n$, e o vértice V_i corresponde a i-ésima linha (e coluna) de A. A aresta (V_i, V_j) $\in E$ se e somente se $a_{i,j} \neq 0$.

Nos exemplos a seguir, vamos ilustrar o grafo associado a matriz dos coeficientes de sistemas lineares em cada etapa do método de Eliminação de Gauss.

Exemplo 1: Dada a matriz A de um sistema linear $Ax = b$ abaixo,

aplicaremos as operações elementares em suas linhas para torná-la escalonada, e simultaneamente, apresentaremos a figura do grafo.

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 13 & -2 & -12 & 0 & 0 \\ -2 & 6 & 0 & 2 & 0 \\ -12 & 0 & 25 & 6 & 0 \\ 0 & 2 & 6 & 5 & -1 \\ 0 & 0 & 0 & -1 & 5 \end{bmatrix} \end{array}$$

A seguir, temos o grafo associado a matriz A do sistema linear $Ax = b$, onde podemos notar que, como A é uma matriz simétrica, e cada linha e coluna é equivalente a um vértice do grafo, cada elemento não-nulo da matriz equivale a uma aresta entre os vértices.

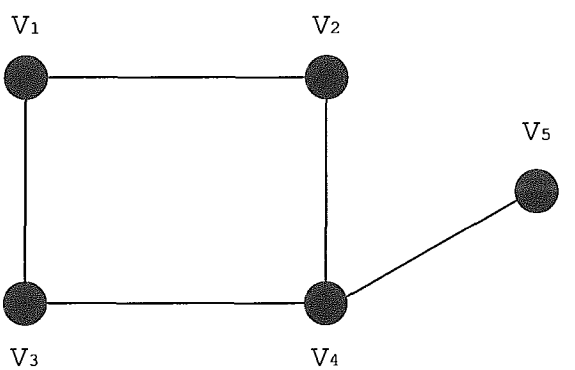


Figura 2.1: Grafo associado a matriz A

Como temos o elemento $a_{1,1} \neq 0$, iniciaremos o escalonamento uti-

lizando o elemento $a_{1,1}$ como pivô, então efetuaremos $L_2 = \frac{13}{2}L_2 + L_1$ e $L_3 = \frac{13}{12}L_3 + L_1$.

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 13 & -2 & -12 & 0 & 0 \\ 0 & 37 & -12 & 13 & 0 \\ 0 & -2 & \frac{181}{12} & \frac{13}{2} & 0 \\ 0 & 2 & 6 & 5 & -1 \\ 0 & 0 & 0 & -1 & 5 \end{bmatrix} \end{array}$$

Considerando somente as alterações no triangulo inferior da matriz, podemos perceber na figura 2.2, que ao pivotearmos pelo elemento $a_{1,1}$ equivalente a linha do vértice V_1 e ao efetuarmos as operações para escaloná-la, no grafo isso equivale a eliminarmos este vértice V_1, e além disso, podemos ver também que uma nova aresta foi criada entre os vértices V_2 e V_3.

Esse preenchimento ocorre devido ao aparecimento de um elemento não-nulo onde anteriormente era elemento nulo na matriz, elemento $a_{2,3}$.

Faremos agora as seguintes operações: $L_3 = \frac{37}{2}L_3 + L_2$, e também $L_4 = -\frac{37}{2}L_4 + L_2$. Temos como resultado a matriz:

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 13 & -2 & -12 & 0 & 0 \\ 0 & 37 & -12 & 13 & 0 \\ 0 & 0 & \frac{6409}{24} & \frac{533}{4} & 0 \\ 0 & 0 & -123 & -\frac{159}{2} & \frac{37}{2} \\ 0 & 0 & 0 & -1 & 5 \end{bmatrix} \end{array}$$

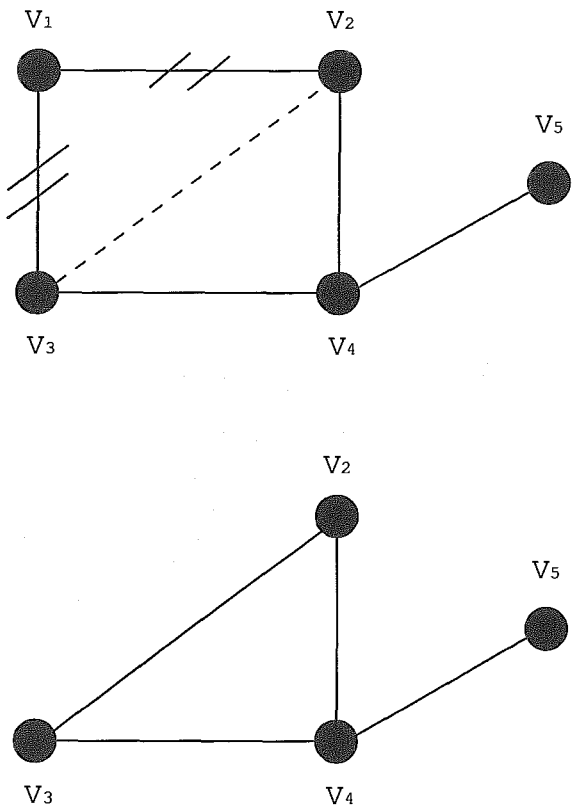


Figura 2.2: Eliminação do vértice V_1 e criação de uma aresta entre V_2 e V_3

Essas operações na matriz produzem alterações no grafo, como mostramos na figura 2.3. Como a eliminação do vértice V_2.

No próximo passo do escalonamento, efetuaremos a operação $L_4 = \frac{6409}{2952}L_4 + L_3$:

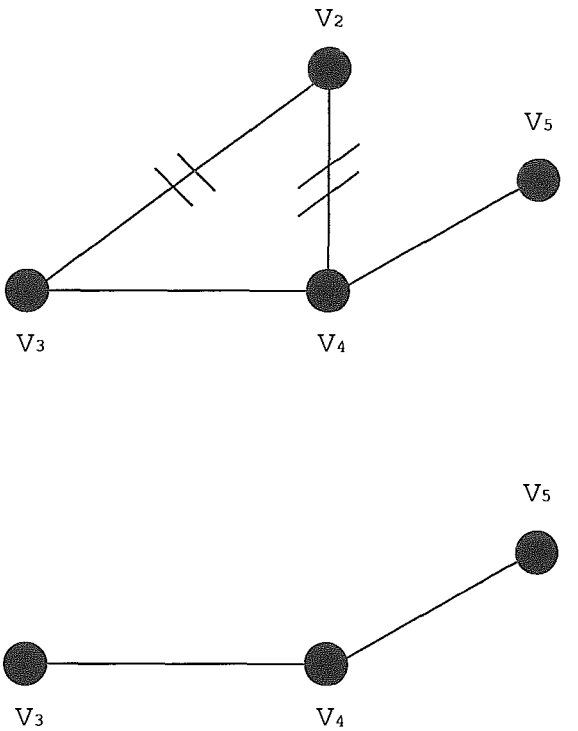


Figura 2.3: Eliminação do vértice V_2

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 13 & -2 & -12 & 0 & 0 \\ 0 & 37 & -12 & 13 & 0 \\ 0 & 0 & \frac{6409}{24} & \frac{533}{4} & 0 \\ 0 & 0 & 0 & -\frac{232323}{5904} & \frac{237133}{5904} \\ 0 & 0 & 0 & -1 & 5 \end{bmatrix} \end{array}$$

Mostramos na figura 2.4 que, após essa operação, elimina-se o vértice V_3 no grafo.

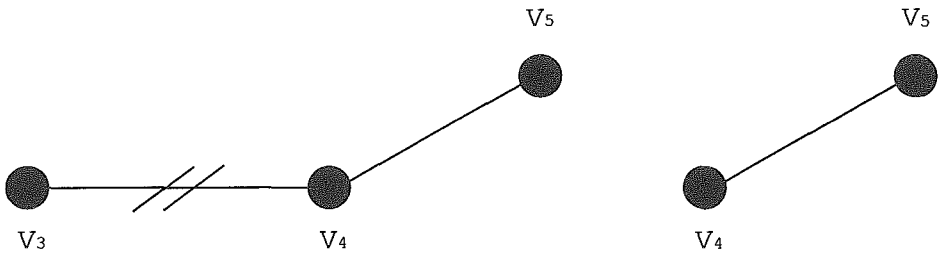


Figura 2.4: Eliminação do vértice V_3

Finalmente, para terminar com o escalonamento, faremos a operação $L_5 = -\frac{232323}{5904}L_5 + L_4$, obtendo assim a matriz escalonada:

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 13 & -2 & -12 & 0 & 0 \\ 0 & 37 & -12 & 13 & 0 \\ 0 & 0 & \frac{6409}{24} & \frac{533}{4} & 0 \\ 0 & 0 & 0 & -\frac{232323}{5904} & \frac{237133}{5904} \\ 0 & 0 & 0 & 0 & -\frac{462241}{2952} \end{bmatrix} \end{array}$$

Após a operação na matriz, que a torna escalonada, o grafo tem seu vértice V_4 eliminado, como mostra a figura 2.5, e conseqüentemente a última aresta eliminada, aresta V_4V_5.

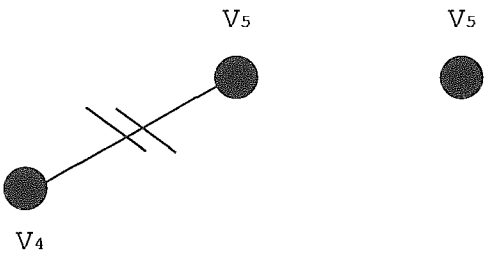


Figura 2.5: Eliminação do vértice V_4

Durante o processo de escalonamento de uma matriz simétrica, podemos constatar que, no grafo correspondente, quando tomamos como pivô o elemento correspondente a um determinado vértice, este é eliminado do grafo, ou seja, o vértice V_1 é eliminado ao tomarmos como pivô o elemento $a_{1,1}$.

Exemplo 2: A partir da matriz M, apresentaremos o comportamento do grafo equivalente durante o processo de eliminação de Gauss.

$$M = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 9 & 0 & 0 & 5 & 2 \\ 0 & 8 & 0 & 0 & 2 \\ 0 & 0 & 8 & -4 & -2 \\ 5 & 0 & -4 & 13 & 4 \\ 2 & 2 & -2 & 4 & 2 \end{bmatrix} \end{array}$$

Como primeiro passo, faremos as operações elementares entre as linhas L_1, L_4 e L_5. Assim, calcularemos $L_4 = -\frac{9}{5}L_4 + L_1$ e também $L_5 = -\frac{9}{2}L_5 + L_1$. Fazendo isso, temos a seguinte matriz resultante:

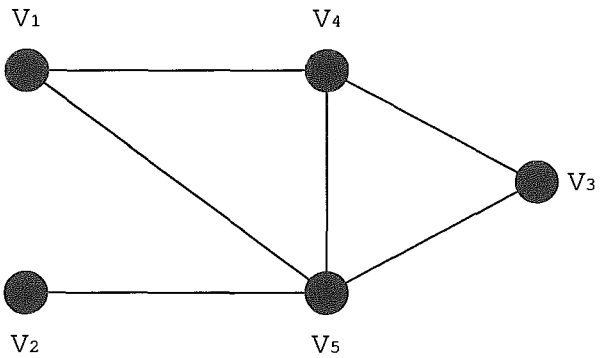


Figura 2.6: Grafo da matriz M

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 9 & 0 & 0 & 5 & 2 \\ 0 & 8 & 0 & 0 & 2 \\ 0 & 0 & 8 & -4 & -2 \\ 0 & 0 & \frac{36}{5} & -\frac{92}{5} & -\frac{26}{5} \\ 0 & -9 & 9 & -13 & -7 \end{bmatrix} \end{array}$$

No grafo associado podemos perceber que estas operações eliminam as arestas V_1V_4 e V_1V_5, e conseqüentemente o vértice V_1.

Como o próximo passo, efetuaremos a operação entre as linhas L_2 e L_5. Assim, faremos $L_5 = \frac{8}{9}L_5 + L_2$, dando como a matriz resultante:

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 9 & 0 & 0 & 5 & 2 \\ 0 & 8 & 0 & 0 & 2 \\ 0 & 0 & 8 & -4 & -2 \\ 0 & 0 & \frac{36}{5} & -\frac{92}{5} & -\frac{26}{5} \\ 0 & 0 & 8 & -\frac{104}{9} & -\frac{38}{9} \end{bmatrix} \end{array}$$

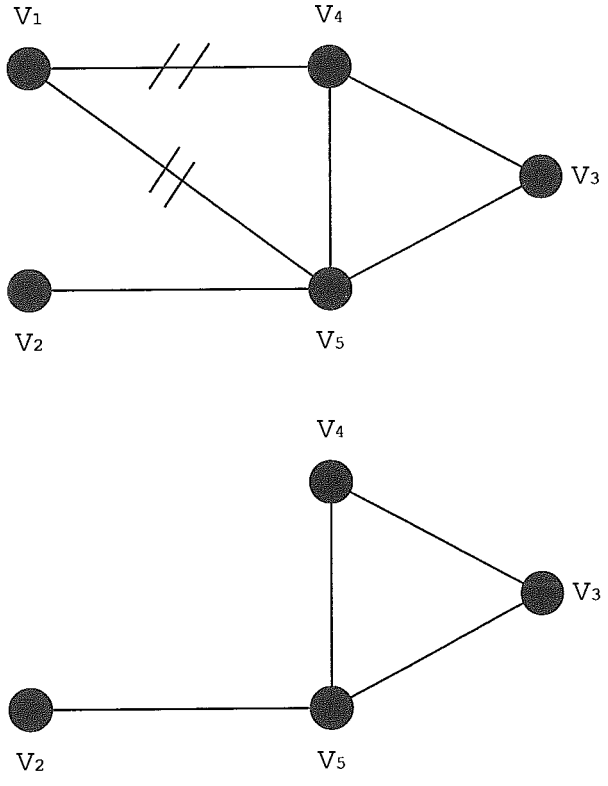


Figura 2.7: Eliminação do vértice V_1

No grafo, as alterações efetuadas na matriz tem como consequência a eliminação da aresta V_2V_5, e o vértice V_2, tendo como grafo resultante:

Efetuaremos agora $L_4 = -\frac{10}{9}L_4 + L_3$ e $L_5 = -L_5 + L_3$, e assim temos a matriz:

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 9 & 0 & 0 & 5 & 2 \\ 0 & 8 & 0 & 0 & 2 \\ 0 & 0 & 8 & -4 & -2 \\ 0 & 0 & 0 & \frac{148}{9} & \frac{34}{9} \\ 0 & 0 & 0 & \frac{68}{9} & \frac{20}{9} \end{bmatrix} \end{array}$$

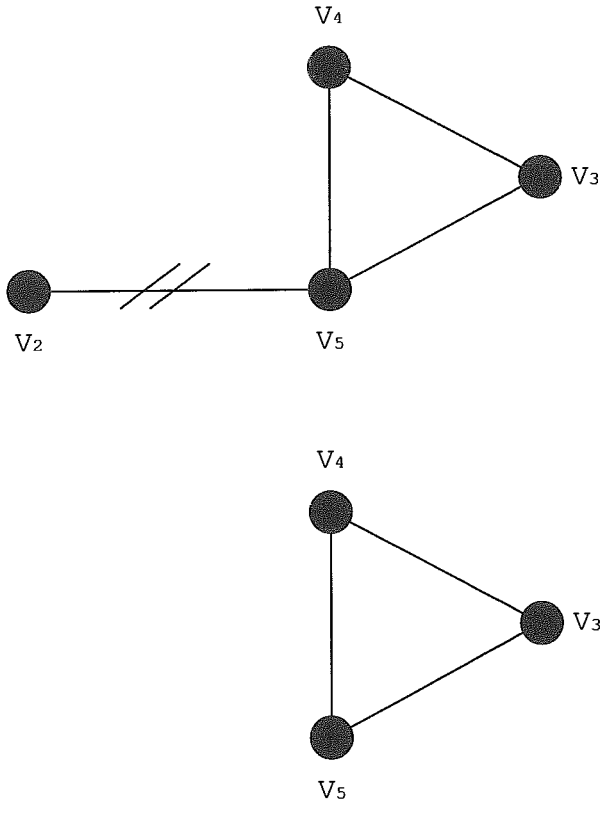


Figura 2.8: Eliminação do vértice V_2

Como consequência, temos no grafo a eliminação das arestas V_3V_4 e V_3V_5, e o vértice V_3.

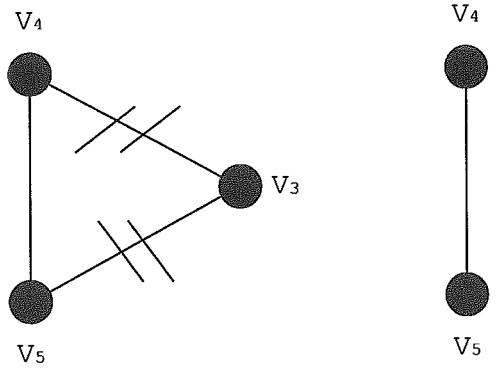


Figura 2.9: Eliminação do vértice V_3

Finalmente, para tornar a matriz escalonada, efetuaremos $L_5 = -\frac{37}{17}L_5 + L_4$, o que resulta na matriz escalonada:

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 9 & 0 & 0 & 5 & 2 \\ 0 & 8 & 0 & 0 & 2 \\ 0 & 0 & 8 & -4 & -2 \\ 0 & 0 & 0 & \frac{148}{9} & \frac{34}{9} \\ 0 & 0 & 0 & 0 & -\frac{54}{51} \end{bmatrix} \end{array}$$

Na figura do grafo, podemos perceber que esta última operação resultou na eliminação da aresta V_4V_5 e do vértice V_4.

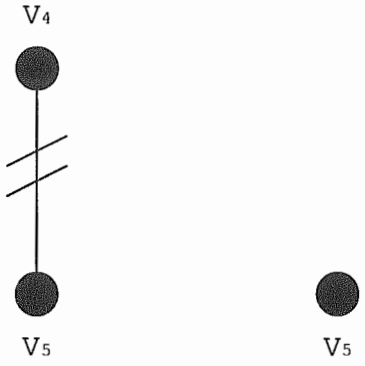


Figura 2.10: Eliminação do vértice V_4

Podemos perceber no **exemplo 1** que aconteceu preenchimento com elementos não-nulos na matriz, e conseqüentemente no grafo associado, e no **exemplo 2** esse preenchimento não aconteceu, ou seja, também pode não haver preenchimento na matriz durante o método de eliminação de Gauss.

Para que não ocorra esse preenchimento, é necessário a escolha de uma ordem de pivoteamento associada a uma classe específica de grafos chamada de **Grafos Cordais**, que será descrita com mais detalhes no próximo capítulo.

Capítulo 3

Grafos Cordais

Na primeira seção deste capítulo, apresentaremos algumas definições sobre grafos (veja [22]), mais especificamente sobre grafos cordais, que serão abordados de forma mais direta na segunda seção.

Na seção seguinte apresentaremos conceitos sobre grafos de interseção, que serão necessários para a caracterização de grafos cordais como grafos de interseção, onde mostraremos a construção de árvores de eliminação e por fim, das subárvores de uma árvore de eliminação.

3.1 Conceitos básicos sobre grafos

Um Grafo G é um conjunto finito não vazio $V(G)$ e um conjunto $E(G)$ de pares não ordenados de elementos distintos de $V(G)$. Os elementos de $V(G)$ são ditos vértices de G, enquanto os elementos de $E(G)$ são as arestas de G.

Uma aresta e pertencente a $E(G)$ é denotada pelo par de vértices (v, w)

que a forma. Neste caso v e w são vértices adjacentes e são as extremidades de e. Diz-se que a aresta e é incidente a v e w. Duas arestas são adjacentes quando possuem alguma extremidade comum. Denotaremos $n = |V(G)|$ e $m = |E(G)|$.

Os grafos geralmente são visualizados através de uma representação geométrica, na qual cada vértice é representado por um ponto distinto, e cada aresta é representada através de uma linha que une os pontos correspondentes a suas extremidades. Em geral confundiremos o grafo com sua representação geométrica e serão denominados indistintamente grafos.

Dado um grafo G, e um vértice $v \in V(G)$, o grafo $G \setminus \{v\}$ é obtido a partir de G retirando-se o vértice v de seu conjunto de vértices, e todas as arestas de $E(G)$ incidentes a v. Dada uma aresta e, o grafo $G \setminus \{e\}$ é o grafo obtido retirando-se a aresta e de $E(G)$.

Uma extensão dos grafos são os chamados grafos direcionados ou dígrafos nos quais o conjunto $E(G)$ é formado por pares ordenados.

Um subgrafo G' de um grafo G é um subconjunto V' de seu conjunto de vértices V e um subconjunto $E' \subseteq E$ de arestas incidentes apenas aos vértices de V'. Quando E' contém todas as arestas de E cujas extremidades estão contidas em V' então G' é o subgrafo induzido em G por V'.

Uma sequência de vértices $v(0)$, ... , $v(i)$ tal que $(v(j), v(j+1)) \in E(G)$ é um caminho entre $v(0)$ e $v(i)$. Se os vértices $v(j)$ são distintos, então trata-se de um caminho simples.

O valor i é o comprimento do caminho. A distância $d(v, w)$ entre dois vértices v, $w \in G$ é o comprimento de um caminho mínimo entre v e w em G.

Quando não existe um caminho entre v e w, então $d(v, w)$ é considerada infinita. Denotamos por $P(n)$ o grafo formado por um único caminho simples de comprimento n.

Um ciclo é um caminho $v(1), ..., v(i), v(1)$. Se $v(1), ..., v(i)$ é um caminho simples, então o ciclo também é dito simples. Todos os ciclos que consideramos são simples, a não ser que o contrário seja dito de forma expressa. Um grafo que não possui ciclos é dito acíclico. G é conexo se existe um caminho entre qualquer par de seus vértices.

Um grafo é uma árvore quando é acíclico e conexo. Uma árvore enraizada é uma árvore na qual foi escolhido um vértice como especial. Este vértice é denominado como raiz da árvore.

Uma árvore geradora de um grafo G é um subgrafo acíclico de G no qual existe exatamente um caminho simples entre cada par de vértices de G. Um subgrafo conexo de uma árvore é dito subárvore.

Um conjunto S é maximal em relação a uma determinada propriedade P se S satisfaz P, e todo conjunto S' que contém propriamente S não satisfaz P. Uma componente conexa de um grafo G é um subgrafo maximal conexo de G.

O grafo de interseção de uma família $\mathcal{F}$ de conjuntos é o grafo obtido tomando um vértice correspondente a cada conjunto da família, e incluindo uma aresta entre dois destes vértices se e somente se os conjuntos correspondentes se intersectam.

Um grafo de intervalo é o grafo de interseção de intervalos de um conjunto linearmente ordenado.

Um grafo é completo quando contém uma aresta entre cada par de seus vértices. O grafo completo com n vértices é designado por k_n.

Um conjunto de vértices V contido em $V(G)$ que induz um subgrafo completo é dito clique. Uma clique que não está propriamente contida em nenhuma outra clique constitui uma clique maximal de G.

Na figura 3.1, mostramos um exemplo de clique em um subgrafo de um grafo G, os vértices B, C e D formam uma clique, pois induzem a um

subgrafo completo de G, já os vértices A, B, D e E não induzem a um subgrafo completo, logo não formam uma clique.

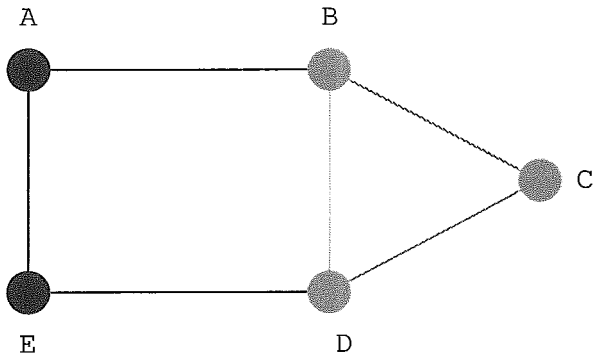


Figura 3.1: Exemplo de clique em um grafo

O grafo clique de G, denotado por $K(G)$, é o grafo de interseção das cliques maximais de G. Quando conveniente, denotaremos o grafo G por $K^0(G)$, e em geral denotaremos por $K^i(G)$ o grafo $K(K^{i-1}(G))$. Nos referiremos a K como o operador clique.

Dizemos que uma família $\mathcal{F}$ de grafos é fechada sob o operador clique se, para todo grafo G pertence a $\mathcal{F}$, tem-se $K(G) \in \mathcal{F}$ e, além disso, existe um grafo $H \in \mathcal{F}$ tal que $G = K(H)$.

Um conjunto independente de vértices é um conjunto de vértices não-adjacentes dois a dois.

O conjunto $Adj(v)$ denota os vértices adjacentes a v, e pertencente a $V(G)$, e é dito vizinhança de v. A vizinhança fechada de v, denotada por $N(v)$, é definida como $Adj(v) \cup \{v\}$.

O grau de v, representado por $d(v)$, é o número de vértices adjacentes a v, ou seja, $d(v) = |Adj(v)|$.

Um grafo é cordal quando qualquer ciclo com pelo menos quatro vértices

possui uma corda, isto é, existe uma relação de adjacência entre um par de vértices não consecutivos no ciclo.

Dirac (1961), e depois Lekkerkerker e Boland (1962) mostraram que todo grafo cordal possui um vértice v, cuja vizinhança induz a um subgrafo completo [4, 12]. Um vértice que apresenta esta característica é dito simplicial.

Na Figura 3.2, o vértice C é simplicial, já que todos os vértices adjacentes a ele no grafo, são vizinhos entre si, ou seja, formam uma clique.

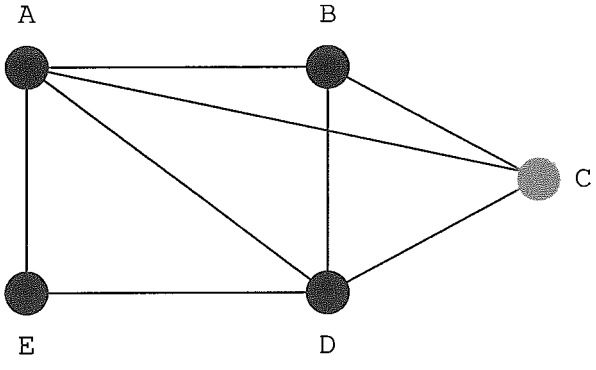


Figura 3.2: Exemplo de vértice simplicial

3.2 Grafos cordais

Uma classe de grafos muito estudada são os grafos cordais, pois importantes aplicações estão relacionadas a esta classe, como a possibilidade de se encontrar uma ordem de pivoteamento na resolução de sistemas lineares tal que não ocorra preenchimento na matriz correspondente ao grafo, ou seja, não haverá mudança de elementos nulos para não-nulos em $A_{n\times n}$.

Ser cordal é também uma propriedade hereditária, ou seja, todo subgrafo

G' induzido de um grafo G é também cordal.

Grafos cordais são conhecidos também como grafos triangulados, grafos de eliminação perfeita, grafos de circuito rígido, e grafos monótono transitivos.

Na figura 3.3 apresentamos um exemplo de grafo não-cordal, e logo abaixo, um exemplo de um grafo cordal, que possui uma corda, evitando assim ciclos maiores do que três.

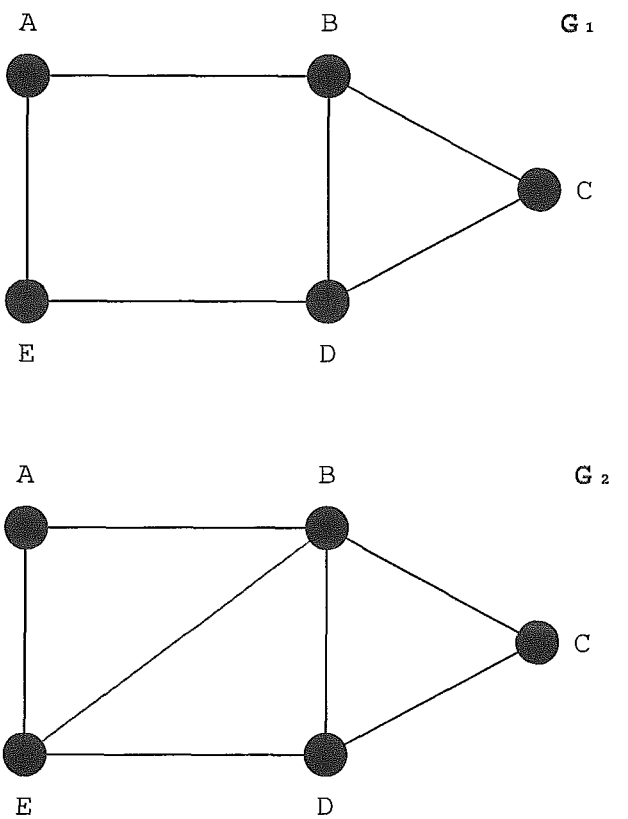


Figura 3.3: Exemplo de grafo não-cordal G_1 e grafo cordal G_2

Definição 1 *Ordem de eliminação perfeita é uma ordem de eliminação W dos vértices do grafo $G = (V, E)$, tal que $W = \{ w_1, w_2, \ldots, w_n \}$, onde cada vértice $w_i \in V$, é simplicial em $G \setminus \{ w_1, w_2, \ldots, w_{i-1} \}$.*

Definição 2 *Um subconjunto S contido em V é um separador de vértices para vértices a e b não-adjacentes, ou também chamado de a_b separador, se a remoção de S do grafo G separa a e b em componentes conexas distintas.*

Se nenhum subconjunto próprio de S é um *a_b separador*, então S é um separador de vértices minimal para a e b.

Lema 1 *Todo grafo cordal $G = (V, E)$ possui um vértice simplicial. Além disso, se o grafo G não é uma clique, então este grafo possui ao menos dois vértices simpliciais não-adjacentes.*

Prova:

Se o grafo G é completo, então todo vértice de G é simplicial, logo o lema é verdadeiro.

Assumimos que o grafo G não é completo, e que G possui dois vértices não-adjacentes a e b e que o lema é verdadeiro para todo grafo cordal com menos vértices do que o grafo G.

Seja S um separador de vértices minimal para a e b com $G(A)$ e $G(B)$ sendo as componentes conexas de $G(V - S)$ contendo respectivamente a e b.

Por indução, ou o subgrafo $G(A+S)$ possui dois vértices simpliciais não-adjacentes, e como S induz a um subgrafo completo (teorema 1, que será descrito posteriormente), um dos vértices deve estar em $G(A)$, ou $G(A+S)$ é um subgrafo completo e qualquer vértice é simplicial em $G(A + S)$. E além disso, como $Adj(A)$ está contido em $A + S$, um vértice simplicial de $G(A + S)$ em A é simplicial em todo grafo G.

Logo, por similaridade B contém um vértice simplicial de G, provando o lema.

■

Uma caracterização de grafos cordais pode ser dada pelo seguinte teorema [8]:

Teorema 1 *Seja G um grafo cordal. São equivalentes as seguintes afirmações:*

a) G é cordal.

b) G possui uma ordem de eliminação perfeita. Além disso, qualquer vértice simplicial pode iniciar uma ordem de eliminação perfeita.

c) Todo separador de vértices minimal induz a um subgrafo completo de G.

Prova:

$c) \Rightarrow a)$

Seja $\{ a, x, b, y_1, y_2, \ldots, y_k, a \}$ $(k \geq 4)$ um ciclo simples de $G = (V, E)$. Todo *a_b separador minimal* deve conter vértices x e y_i para algum i, logo, $(x, y_i) \in E$, que é uma corda do ciclo.

$a) \Rightarrow c)$

Suponha que S é um *a_b separador minimal* com $G(A)$ e $G(B)$ sendo as componentes conexas de $G(V - S)$ contendo a e b, respectivamente.

Como S é minimal, cada $x \in S$ é adjacente a algum vértice em $G(A)$ e algum vértice em $G(B)$. Logo, para qualquer par $\{ x, y \} \in S$ existem caminhos:

$$\{ x,\ a_1,\ a_2,\ \ldots,\ a_r,\ y \} \text{ e}$$

$$\{ y,\ b_1,\ b_2,\ \ldots,\ b_t,\ x \},$$

onde cada $a_i \in A$ e $b_i \in B$, tal que esses caminhos são escolhidos para ser o de tamanho menor possível. Isto segue que:

$$\{ x,\ a_1,\ \ldots,\ a_r,\ y,\ b_1,\ \ldots,\ b_t,\ x \}$$

é um ciclo simples cujo tamanho é ao menos quatro, implicando que este deve ter uma corda. Mas $(a_i, b_j) \notin E$ pela definição de vértice separador, e $(a_i, a_j) \notin E$ e $(b_i, b_j) \notin E$ pela minimalidade de r e t. Assim, a única possibilidade de corda é $(x, y) \in E$. Observe que desta prova segue também que $r = t = 1$, implicando que $\forall x,\ y \in S$, existem vértices em A e em B que são adjacentes tanto a x quanto a y.

$a) \Rightarrow b)$

De acordo com o lema, se G é cordal, então este tem um vértice simplicial, chamemos de x este vértice. Como $G\ (V - x)$ é cordal e menor do que G, este tem por indução, um esquema de eliminação perfeito que, quando une-se ao índice de x, forma um esquema de eliminação perfeito para G.

$b) \Rightarrow a)$

Seja C um ciclo simples de G e seja x o vértice de C com o menor índice no esquema de eliminação perfeito. Como $mod\ (Adj\ (x) \cap C) \geq 2$, a eventual simplicialidade de x garante uma corda em C.

■

Portanto podemos afirmar que grafos cordais são chamados de grafos de eliminação perfeita porque é uma classe de grafos que possui uma ordenação de eliminação perfeita de seus vértices.

3.3 Grafos de interseção

Um importante conceito sobre grafos são os chamados grafos de interseção [17], onde grafo de interseção é definido abaixo:

Definição 3 *Seja uma família*

$$\mathcal{F} = \{ S_1, S_2, \ldots, S_n \}$$

de conjuntos finitos (ou estruturas) de $\mathcal{F}$, *com* $V(G) = \mathcal{F}$ *e tal que* $\{ i, j \} = (i,j) \in E(G)$ *se e somente se* $S_i \cap S_j \neq \emptyset$, $i \neq j$.

Como o grafo não é direcionado, $\{ i, j \} = \{ j, i \}$, isto é, $(i,j) = (j,i)$, onde $|V(G)| = n$, ou seja, um vértice para cada conjunto S_i.

Consideramos uma família pois podemos ter conjuntos repetidos. Dado uma família $\mathcal{F}$, um grafo G é caracterizado de interseção se existe uma família $\mathcal{F}$, onde $\mathcal{F}$ é uma representação (ou modelo de interseção) de G. Uma caracterização para grafos de interseção é dado pelo teorema de Marczewsky (1945) [16]:

Teorema 2 *Todo grafo é um grafo de interseção.*

Prova:

Dado um grafo $G = (V, E)$ tal que:

$$E(G) = \{ e_1, e_2, \ldots, e_m \}.$$

Definimos para cada i, $1 \leq i \leq n$, onde $n = |V\ (G)|$,
$S_i = \{\ e_j \mid e_j\ incide\ em\ i\ \}$, $V\ (G) = \{\ 1,\ 2,\ \ldots\ ,\ n\ \}$ e existe (i,j) se e somente se $S_i \cap S_j \neq \emptyset$.

Falta mostrar que:

$$\mathcal{F} = \{\ S_1,\ S_2,\ \ldots\ ,\ S_n\ \}$$

é uma representação de G.

Sejam i e j vértices adjacentes em G.

Logo, $e_k = (i,j) \in G$. Por construção, $e_k \in S_i$ e $e_k \in S_j$ então $ek \subseteq S_i \cap S_j$ e $S_i \cap S_j \neq \emptyset$.

Sejam i e j, $i \neq j$ tal que $S_i \cap S_j \neq \emptyset$.

Existe uma aresta $e_k \in S_i \cap S_j$. Por construção, e_k é incidente a i e e_k é incidente a j. Como $i \neq j$, $e_k = (i,j)$ e i e j são vértices adjacentes em G.

■

3.4 Caracterização de grafos cordais como grafos de interseção

Nesta seção apresentaremos a relação entre grafos cordais e grafos de interseção, iniciando-se pela formação de uma árvore de eliminação dos vértices do grafo cordal.

Esta árvore de eliminação será baseada em uma ordem de eliminação de seus vértices, e além disso, mostraremos também a formação de subárvores da árvore de eliminação.

Cada subárvore estará associada a um vértice da árvore de eliminação e juntas comporão uma família de estruturas, que corresponderá ao grafo de interseção, que será cordal.

3.4.1 Árvore de Eliminação

A partir de uma ordem de eliminação dos vértices de um grafo G, pode ser formada uma árvore de eliminação T, onde cada vértice de T é equivalente a um vértice de G.

Para se construir essa árvore de eliminação, seguiremos o seguinte procedimento:

Rotularemos os vértices de G, que nos dará a ordem de eliminação dos vértices, de acordo com o seguinte critério (veja [2]): todo vértice simplicial, que não tenha como adjacente outro simplicial, recebe rótulo 1. Caso tenhamos dois ou mais vértices simpliciais adjacentes entre si, apenas um deles receberá rótulo 1.

Assim, os vértices simpliciais rotulados com 1 formam um conjunto independente. O mesmo ocorrerá com todos os conjuntos de vértices com rótulos iguais.

Eliminamos estes vértices do grafo e pelo **Lema 1**, como o grafo é cordal, existirá ao menos um vértice simplicial no novo grafo, e caso este não seja uma clique, existirão ao menos dois.

Ao eliminarmos esses vértices de rótulo 1, todos os vértices simpliciais não-adjacentes a outros simpliciais, neste novo grafo, recebem rótulo 2. Caso tenhamos neste novo grafo, dois ou mais simpliciais adjacentes, apenas um receberá rótulo 2, ou seja, os vértices simpliciais neste novo grafo rotulados com 2 formarão um conjunto indenpendente.

Continuamos com esse procedimento até que todos os vértices do grafo estejam rotulados.

Ao rotularmos o maior número de vértices com o mesma numeração, estamos paralelizando a eliminação, o que nos permite em um passo, a eliminação de mais de um vértice.

Num grafo completo K_n, ou mesmo um subgrafo clique G' de G, não é possível a paralelização na ordem de eliminação, pois este não possui um conjunto independente de tamanho maior que um.

Cada vértice de uma clique terá uma rotulação diferente, uma vez que todos são simpliciais e adjacentes. Sua eliminação se dará de forma sequencial.

Para construirmos uma árvore de eliminação T, tomaremos como raiz r da árvore, o vértice de maior rótulo do grafo $G(V, E)$, onde $|V| = N$.

A seguir, tomaremos como descendentes imediatos da raiz de T, os vértices de maior rótulo de cada componente conexa C_i do grafo $G \setminus \{r\}$.

Caso tenhamos uma única componente conexa C, então tomaremos o vértice de maior rótulo como o único descendente da raiz.

Cada vértice x_i, descendente de r na árvore, será raiz da subárvore $T[x_i]$. Assim, repetiremos o processo acima para cada componente conexa D_j do grafo $C_i \setminus \{x_i\}$, ou seja, os descendentes imediatos de cada x_i serão os vértices, em cada componente conexa de D_j, de maior rótulo.

Seguiremos com o procedimento até que todos os vértices tenham sido incluídos na árvore.

Segue deste procedimento, que será ascendente imediato de um vértice v, o primeiro vértice adjacente em G que foi ordenado após v na ordem de eliminação, ou seja, o vizinho que possuir o menor rótulo após v. Maiores detalhes veja em [2].

Assim, todo vértice a ser incluído na árvore de eliminação T, possuirá uma rotulação maior na ordem de eliminação do que os outros vértices da sua componente conexa ainda não incluídos.

Cada par de vértices u e v na árvore T, possuem a seguinte caracterísitica: se u e v são vértices adjacentes em G, então ou v é descendente de u em T ou u é descendente de v em T.

Ao se formar essa estrutura, a árvore de eliminação terá como folhas sempre um vértice simplicial, pois este terá, na ramificação, o menor rótulo.

Vértices não-adjacentes em G poderão ter rótulos iguais. Estes vértices que possuem rótulos iguais poderão ser eliminados ao mesmo tempo, ou ainda, não necessariamente haverá uma ordenação única para eliminar as folhas da árvore de eliminação.

A eliminação dos vértices da árvore se dará a partir da sua ramificação, os vértices descendentes serão eliminados primeiro que seus ascendentes, ou seja, a eliminação se dará pelas folhas.

Na figura 3.4, temos um grafo cordal com seus vértices A, B, C, D e E rotulados em uma ordem de eliminação perfeita e sua respectiva árvore de eliminação. Podemos observar ainda que, mesmo sendo adjacente a B no grafo, o vértice A não é descendente imediato de B.

Para grafos não-cordais, é necessário que se tenha em princípio, uma ordem de eliminação para os vértices do grafo G.

Essa ordenação poderá ser dada pelo seguinte procedimento (veja [15, 19]): Inicialmente, considere todos os vértices do grafo não marcados. Selecione um vértice v não-marcado, e adicione arestas a G até que todos os vizinhos não-marcados de v sejam adjacentes entre si (ou seja, esses vizinhos formem uma clique), então marque v . Repita esse procedimento até que todos os vértices do grafo estejam marcados.

Esse procedimento preenche o grafo G com novas arestas e o torna cordal.

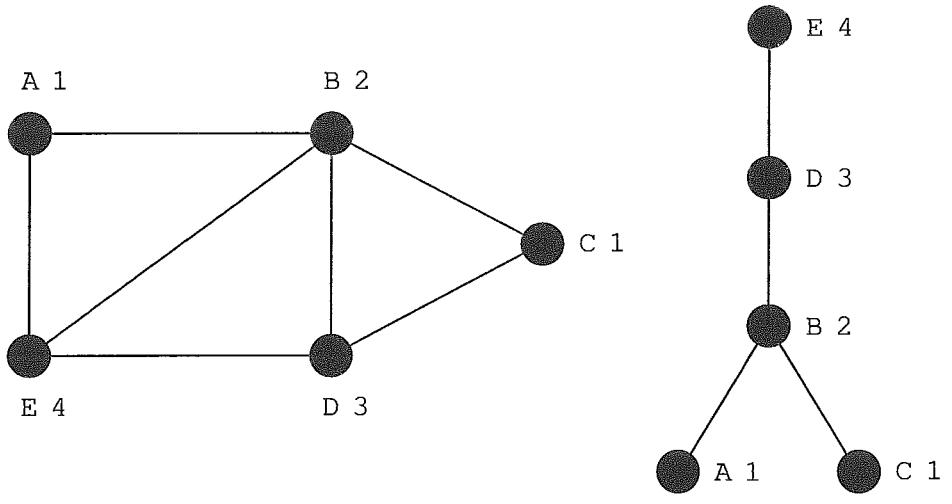


Figura 3.4: Árvore de eliminação de um grafo cordal

Denotamos esse grafo preenchido como G^*, e o chamamos de *filled graph*, ou ainda, *chordal completion* de G, e as novas arestas incluídas no grafo são chamadas de arestas de preenchimento, *fill edges*, ou somente *fill*.

A ordem em que são selecionados os vértices para efetuarmos o textitchordal completion de G, é dada por alguma heurística de ordenação, possivelmente sequencial.

Como o grafo G^* é cordal, podemos rotular seus vértices da mesma forma que rotulamos os vértices de um grafo cordal, podendo assim paralelizar essa ordem de eliminação.

Essa rotulação também pode ser dada por alguma heurística como por exemplo *Nested Dissection* ou Grau Mínimo, que serão vistas posteriormente.

A construção da árvore de eliminação correspondente será feita da mesma maneira que foi descrita para grafos cordais.

3.4.2 Subárvores da Árvore de Eliminação

Como vimos na subseção anterior, a partir de uma ordem de eliminação, pode-se construir uma árvore de eliminação T considerando a ordenação dos vértices do grafo G, tomando-se como raiz o vértice de maior rótulo na ordem de eliminação.

Os vértices descendentes da raiz r dessa árvore de eliminação possuem rótulo inferior ao da raiz, e assim até os vértices de menor rótulo, que serão as folhas da árvore de eliminação.

Com a construção da árvore de eliminação, podemos então construir subárvores da árvore de eliminação como um modelo de interseção que manterá a estrutura da árvore.

Seja T uma árvore de eliminação de um grafo G. A partir de T podemos construir subárvores T_i que representarão uma família de estruturas que corresponderão a um grafo de interseção.

Cada uma das subárvores representará um vértice de G, logo existirão n subárvores e a interseção dessa família de estruturas, subárvores $\{ T_v \}$, formarão um grafo de interseção que será cordal, como será mostrado posteriormente no teorema 3.

A subárvore T_i terá como raiz o vértice i, e será formada pela união de todos os caminhos de i até v na árvore de eliminação, iniciando-se por i a todos os vértices v, adjacentes a i no grafo G original, e descendentes de i na árvore T.

Essa estrutura será formada a partir da raiz até as folhas, e assim, a subárvore da raiz irá intersectar as subárvores dos seus descendentes que representam vértices adjacentes ao vértice equivalente a raiz em G.

As subárvores dos descendentes imediatos de r irão intersectar as

subárvores de todos os seus descendentes, netos de r, que representam vértices adjacentes a eles no grafo original, e assim até as folhas de T.

Como a construção das subárvores é realizada de cima para baixo, as folhas da árvore de eliminação não terão descendentes, então para cada um desses vértices representados, será criada, para efeito de visualização, uma subárvore que será intersectada pelas subárvores que representam vértices adjacentes em G.

Os vértices são introduzidos na árvore de eliminação de modo que os vértices adjacentes no grafo G, são ascendentes ou descendentes na árvore.

Não necessariamente esses vértices são ascendentes ou descendentes imediatos na árvore, então na construção das subárvores essas relações de adjacências serão mantidas através das interseções.

Assim, para cada par de vértices $\{ u, v \}$ adjacentes no grafo G, as subárvores T_v e T_u se intersectam.

Na construção de uma subárvore, ao se intersectar as subárvores que representam vértices adjacentes no grafo G, pela formação da árvore de eliminação, haverá subárvores que se intersectarão, mas que no grafo G não havia aresta entre os vértices.

Isto precisamente é o preenchimento que ocorrerá no grafo, com a introdução de novas arestas que o tornará cordal.

Como queremos que o grafo tenha o menor preenchimento possível, temos que ter o menor número de arestas em cada subárvore, pois cada aresta na família de subárvores $\{ T_v \}$ da árvore de eliminação representa uma aresta em $G*$ (grafo preenchido ou cordal).

Uma melhor caracterização para subárvores pode ser dada pelo seguinte teorema [8]:

Teorema 3 *Seja G = (V,E) um grafo não direcionado. As seguintes*

afirmações são equivalentes:

a) G é um grafo cordal.

b) G é um grafo de interseção de uma família de subárvores de uma árvore.

c) Existe uma árvore $T = (K, E)$ cujo conjunto de vértices K é o conjunto de cliques maximais de G tal que cada subgrafo induzido $T_{k_v}(v \in V)$ é conexo (e uma subárvore) , onde K_v é uma clique maximal que contém v.

Prova:

$c) \Rightarrow b)$

Assumimos que existe uma árvore $T = (K, E')$ satisfazendo c). Seja $\{ v, w \} \in V$. Agora $(v, w) \in A$; $\{ v, w \} \in A$ para alguma clique $A \in K$, $K_v \cap K_w \neq \emptyset$, $T(K_v) \cap T(K_w) \neq \emptyset$.

Assim G é um grafo de interseção de uma família de subárvores $\{ T(K_v) \mid v \in V \}$ de T.

$b) \Rightarrow a)$

Seja $\{ T_v \}(v \in V)$ uma família de subárvores de uma árvore T, tal que $(v, w) \in E$, se $T_v \cap T_w \neq \emptyset$.

Suponha G contendo um ciclo sem corda:

$$\{ v_0, v_1, \ldots, v_{k-1}, v_0 \} \text{ com } k \geq 4$$

correspondente a sequência de subárvores $\{ T_0, T_1, \ldots, T_{k-1}, T_0 \}$ da árvore T, isto é, $T_i \cap T_j \neq \emptyset$ se e somente se i e j diferem por no máximo 1 módulo K. Toda aritmética será feita em módulo K.

Escolha um ponto $a(i)$ de $T_i \cap T_{i+1}, (i = 0, 1, \ldots, k-1)$. Seja $b(i)$ o

último ponto em comum no caminho (único) simples de $a\ (i)$ até $a\ (i-1)$ e $a\ (i)$ para $a\ (i+1)$.

Esses caminhos ficam em T_i e T_{i+1}, respectivamente, tal que $b\ (i)$ também fica em $T_i \cap T_{i+1}$.

Seja um caminho simples conectando $b\ (i)$ e $b\ (i+1)$.

Temos então $P_i \subseteq T_i$, $P_i \cap P_j = \emptyset$, para i e j diferindo por mais do que 1 módulo K.

Além disso, $P_i \cap P_{i+1} = b\ (i)$ para $\{\ i = 0,\ 1,\ \ldots\ ,\ k-1\ \}$. Assim, $\cap_i P_i$ é um ciclo simples em T, contradizendo a definição de árvore.

$a) \Rightarrow c)$

Provaremos a implicação do teorema por indução no tamanho de G.

Assumimos que o teorema é verdadeiro para todo grafo tendo menos vértices do que G.

Se G é completo, então é um vértice simples e o resultado é trivial.

Se G é desconexo com componentes $\{\ G_1,\ \ldots\ ,\ G_k\ \}$, então por indução existe uma árvore correspondente T_i satisfazendo c) para cada G_i. Nós conectamos um ponto de T_i com um ponto de T_{i+1}, $\{\ i = 1,\ \ldots\ ,\ k-1\ \}$, para obter uma árvore satisfazendo c) para G.

Assumimos que G é conexo mas não completo. Escolha um vértice simplicial a de G e $A = \{a\} \cup Adj\ (a)$.

Temos que A é uma clique maximal de G. Seja $U = \{\ u \in A \mid adj\ (u) \subset A\ \}$ e $Y = A - U$. Note que os conjuntos U, Y e $V - A$ são não vazios já que G é conexo mas não completo.

Considere o subgrafo induzido $G' = G_{v-u}$, que é cordal e tem menos vértices do que G.

Por indução, seja T' uma árvore cujo conjunto de vértices K' é o conjunto de máximas cliques de G' tal que para cada vértice $v \in V - U$ o conjunto $K'_v = \{ X \in K' \mid v \in X \}$ induz um subgrafo conexo (subárvore) de T'. Observe que: ambos, $K = K' + A - Y$ ou $K = K' + A$ dependem se Y é ou não clique maximal de G'.

Seja B uma clique maximal de G' contendo Y.

Caso 1. Se $B = Y$, então obtemos T a partir de T' renomeando B e A.

Caso 2. Se B é diferente de Y, então obtemos T a partir de T' conectando o novo vértice de A para B.

Em ambos os casos, $K_u = A$, $\forall u \in U$ e $K_v = K'_v$, $\forall v \in V - A$, cada um dos quais induz uma subárvore de T.

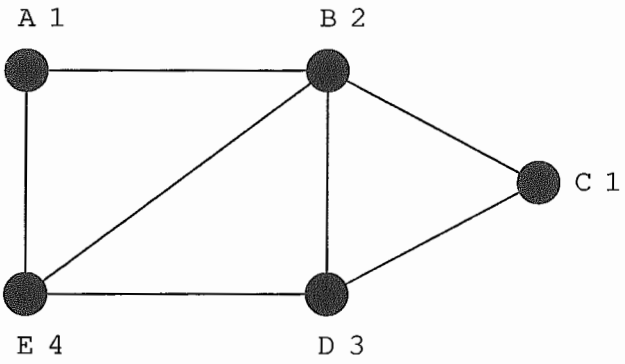
Precisamos observar somente sobre os conjuntos K_y ($y \in Y$). No caso 1, $K_y = K'_y + A - B$, que induz a mesma subárvore como K'_v já que somente nomes são trocados.

No caso 2, $K_y = K'_y + A$, que induz uma subárvore.

Assim, temos construído a árvore T e provado o teorema.

■

Na figura 3.5, temos um exemplo de um grafo cordal, com seus vértices rotulados em uma ordem de eliminação perfeita, e além disso, temos também sua respectiva árvore de eliminação e as subárvores da sua árvore de eliminação.



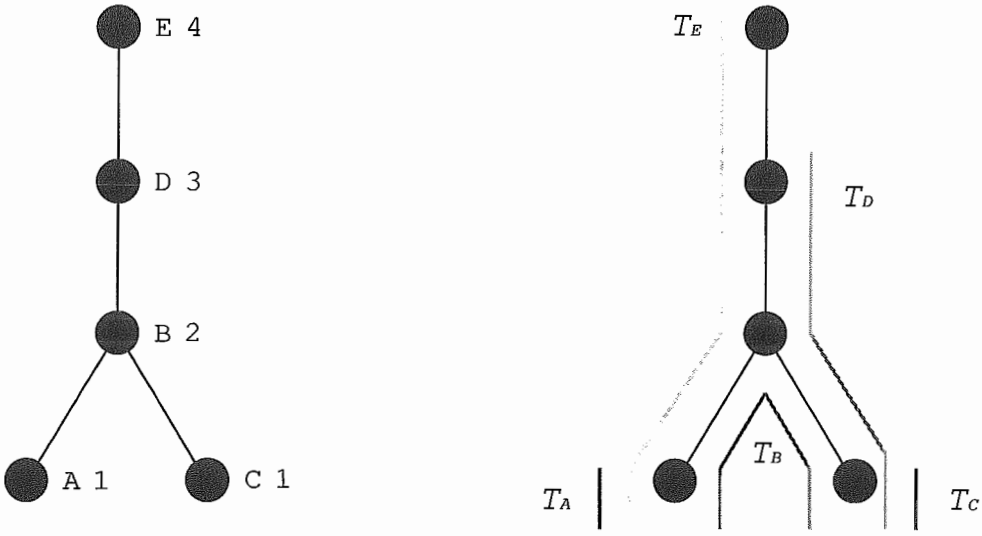


Figura 3.5: Subárvores de uma árvore de eliminação de um grafo

Capítulo 4

Métodos Existentes de Solução Aproximada

Ao longo dos anos, vários métodos de ordenação para vértices de um grafo, que minimize o preenchimento para torná-lo cordal foram estudados.

Neste capítulo apresentaremos algumas das heurísticas mais utilizadas na escolha de uma ordem de eliminação de vértices que minimize o preenchimento de um grafo.

Apresentaremos primeiro a heurística de grau mínimo, que é baseada no grau dos vértices, em que são escolhidos os vértices de menor grau no grafo para serem eliminados primeiro.

Depois mostraremos a heurística *Nested Dissection*, que se baseia na escolha de separadores no grafo, fazendo-se assim divisões no grafo e ordenando os vértices pertencentes a esses separadores.

Também apresentaremos um método híbrido, ou seja, um método que utiliza tanto da heurística de grau mínimo, quanto da *Nested Dissection*, que consiste em dividir o grafo até um certo tamanho, *Nested Dissection*, e então aplica-se grau mínimo nos subgrafos restantes.

4.1 Heurística de grau mínimo

A escolha de uma ordem de eliminação de vértices para grafos não-cordais, em que ocorra o menor preenchimento possível têm sido muito pesquisada, e alguns estudos foram desenvolvidos, como a heurística de grau mínimo.

A heurística de grau mínimo dá origem a um algoritmo conhecido como algoritmo de ordenação de grau mínimo [1, 10, 20, 1]. Este algoritmo trata do problema de identificar uma ordenação para os pivôs em uma matriz A em $\Re^{n\times n}$, quadrada, simétrica e definida positiva, de uma maneira mais direta no sentido de minimizar o preenchimento da matriz, ou de um grafo correspondente.

Este tratamento direto corresponde a escolher, em cada passo da eliminação de Gauss, um vértice v do grafo correspondente que tenha o menor grau para ser eliminado, e adiciona-se arestas em G para tornar os vértices adjacentes à v vizinhos entre si, ou seja, $Adj(v)$ irá induzir a um subgrafo completo, após a eliminação de v.

Alguns problemas são detectados na ordenação dos vértices, como a possibilidade de se ter muitos vértices com grau mínimo, isto é, ter mais de um vértice de grau mínimo a ser escolhido para ser eliminado. Esse problema, geralmente é resolvido de forma arbitrária.

Para um grafo com estrutura de árvore, essa heurística produz uma ordenação em que não ocorre nenhum preenchimento, ou seja, produz uma ordem de eliminação perfeita, já que para cada passo da eliminação é sempre escolhido uma das folhas da árvore que possui grau 1, até que termine todos os vértices do grafo.

Para classes de grafos mais gerais, a ordenação de grau mínimo nos vértices do grafo G pode não ser uma ordem de preenchimento mínimo na

matriz, ou seja, a heurística de grau mínimo, para encontrar uma ordenação dos vértices de um grafo, não é ótima.

No grafo da figura 4.1, pode-se ver que E é o vértice de grau mínimo, $d(E) = 2$, mas ao eliminarmos E teremos um preenchimento no grafo, sendo criada uma aresta entre os vértices D e F, mesmo sendo o grafo cordal, que possui uma ordem de eliminação perfeita, onde não ocorre preenchimento no grafo.

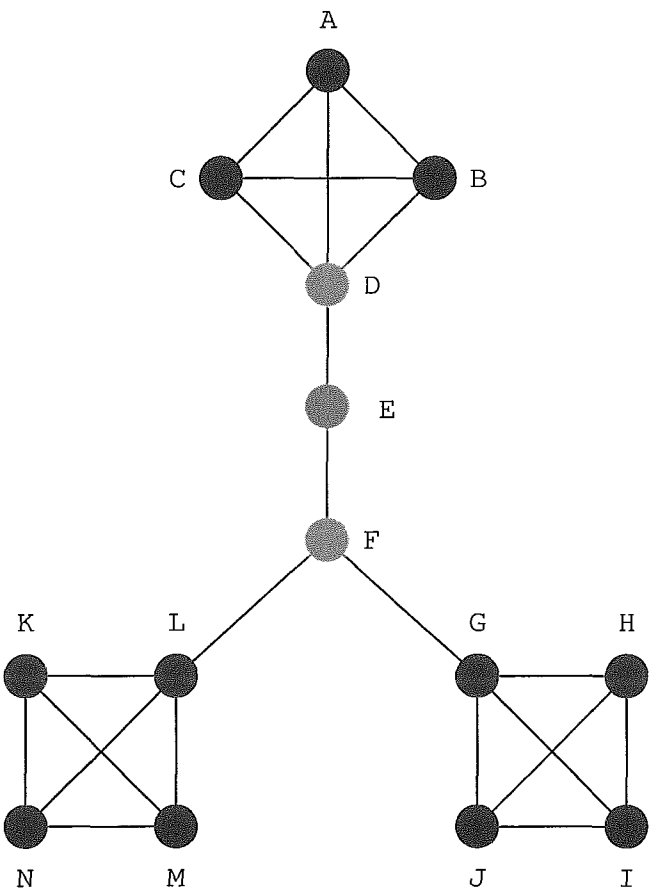


Figura 4.1: Exemplo de que a heurística de grau mínimo não é ótima

Outro problema que ocorre com o algoritmo de ordenação de grau mínimo é o fato de que um vértice de grau baixo, pode ter seu grau aumentado quando um, ou mais de seus vizinhos são eliminados, já que pode haver um preenchimento ao se tornar os vizinhos dos vértices já eliminados

adjacentes entre si.

Ao escolher a ordem para a eliminação dos vértices, em cada passo da eliminação, é recalculado o grau da cada vértice do grafo, então esse vértice passa a ter um grau atualizado.

Recalcular o grau dos vértices do grafo na eliminação de Gauss é uma das principais razões de consumo de tempo do algoritmo de ordenação de grau mínimo.

Algumas melhorias foram desenvolvidas para esse algoritmo como a eliminação concentrada [7], estudos esses feitos por George e McIntyre. Foi observado que na eliminação de um vértice v de grau mínimo, frequentemente existe um subconjunto Y de vértices adjacentes a v que podem ser eliminados imediatamente após a v.

O teorema a seguir contém a base teórica desta observação:

Teorema 4 *Se v é selecionado como o vértice de grau mínimo no grafo G, então o subconjunto $Y = \{ z \in Adj\ (v) \mid grau\ (G_v(z)) = grau\ (v) - 1 \}$ pode ser selecionado junto, e em qualquer ordem, no algoritmo de grau mínimo.*

Este teorema nos permite evitar algumas transformações no grafo e passos de atualizações de grau, já que providencia um conjunto de vértices de grau mínimo que podem ser selecionados junto com o vértice v inicial.

Em vez de fazer-se uma transformação de G para $G_v = G \setminus \{ v \}$ e atualizar os graus dos vértices, pode-se fazer uma eliminação dos vértices em Y e o vértice v simultaneamente.

Eliminando-se o vértice v com o subconjunto Y, não é necessário recalcular o grau de todos os vértices de Y no passo seguinte da ordem de eliminação, e com isso diminui-se o tempo gasto no algoritmo, e além disso, também não é necessário tornar os vizinhos de v que pertençam a Y adjacentes entre si.

4.2 *Nested Dissection*

Uma outra heurística para a escolha de uma ordem de pivoteamento é conhecida como ordenação *Nested Dissection* [5, 6, 9, 10, 14, 20]. Esta ordenação foi proposta por George [5] como uma alternativa para a heurística de grau mínimo.

A idéia básica desta heurística é a de identificar um conjunto de colunas S, cuja remoção divide a matriz em duas partes, X e Y, cujos valores não-nulos, estão em linhas e colunas disjuntas.

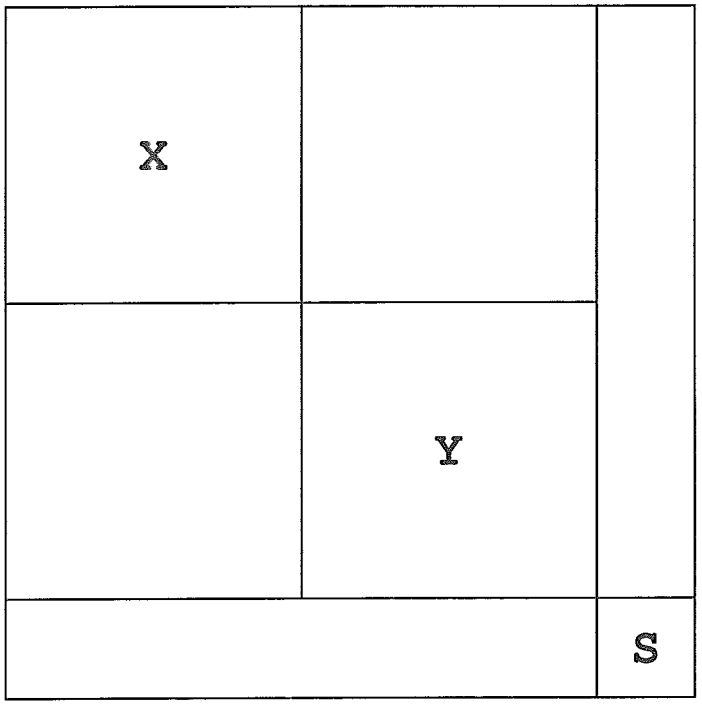


Figura 4.2: Característica da matriz após *Nested Dissection*

Ao se ordenar o conjunto S após X e Y, não ocorrerá nenhum preenchimento na matriz nos blocos fora da diagonal das submatrizes consistindo de X e Y. Ao se encontrar o conjunto S, X e Y podem ser reordenados pela aplicação da heurística *Nested Dissection* recursivamente, ou por qualquer outro método de ordenação, como por exemplo a heurística de grau mínimo.

Em termos de grafos, *Nested Dissection* pode ser visto como a definição de um separador S, e X e Y são as duas partes separadas no grafo G por S.

Para uma efetiva estratégia de *Nested Dissection*, é desejado uma escolha de um separador pequeno e também um equilíbrio entre o tamanho das partes separadas X e Y. Nem X, nem Y devem ser muito pequenos. Com um equilíbrio entre X e Y, podemos ter um melhor separador para o grafo.

A ordenação dos vértices do grafo G se dá na ordem de definição dos separadores, ou seja, ao se definir os vértices do separador, estes terão rótulos maiores que os demais vértices que pertençam a separadores ainda não definidos, caso os separadores possuam mais de um vértice, a ordenação destes vértices é feita de forma arbitrária.

Em comparação com a heurística de grau mínimo, *Nested Dissection* pode ser visto como uma ordenação global dos vértices do grafo, enquanto a heurística de grau mínimo é uma ordenação apenas local. Isto faz com que a heurística *Nested Dissection* seja mais atraente para análises teóricas. George [5] provou que *Nested Dissection* produz ordenações ótimas assintoticamente para problemas de grades regulares.

Para problemas pequenos, a heurística *Nested Dissection* se mostra não muito eficiente em comparação com a heurística de grau mínimo, mas para problemas maiores *Nested Dissection* fornece resultados satisfatórios.

4.3 Método híbrido

Para uma melhora tanto no tempo do algoritmo quanto na qualidade da ordenação, é usado um híbrido da heurística de grau mínimo e da heurística de *Nested Dissection.*

Este método híbrido [9, 10] é iniciado com um *Nested Dissection* padrão incompleto no grafo de $A \in \Re^{n \times n}$, em que se faz vários níveis de *Nested Dissection* até que todos os subgrafos sejam menores do que um certo tamanho, não maiores do que a trigésima segunda parte do número de vértices do grafo G original.

Após aplicar-se até cinco níveis da heurística *Nested Dissection* no grafo G, poderemos ter ainda subgrafos do grafo G, então será utilizada finalmente a heurística de grau mínimo nesses subgrafos menores.

Isto permite que se obtenha os benefícios da heurística de *Nested Dissection* para níveis maiores, onde muito do trabalho de fatoração é feito, e também obtemos as vantagens da heurísitica de grau mínimo para problemas pequenos.

É também feito uma ordenação nos separadores, obtidos com o *Nested Dissection*, utilizando grau mínimo. A intuição por trás desse método híbrido é que o *Nested Dissection* faz uma hipótese implícita de que uma divisão recursiva do problema é uma melhor aproximação para a ordenação.

Essa divisão recursiva permite que a heurística de grau mínimo reordene os vértices dos separadores removidos nessa hipótese.

Capítulo 5

Um Modelo de Programação Linear Mista Para o Problema de Completar um Grafo Para Torná-lo Cordal

5.1 Introdução ao modelo de programação linear mista

Neste capítulo, apresentaremos um modelo de programação linear mista para o problema de completar um grafo conexo para torná-lo cordal, utilizando o menor número possível de arestas.

Para construir este modelo, serão seguidas as estruturas definidas nos capítulos anteriores, como árvores de eliminação e subárvores de árvores de eliminação.

A partir de um grafo G, devemos então construir uma árvore de elimi-

nação, em que os vértices estejam dispostos de forma que esta tenha uma ordem de eliminação dos vértices em que ocorra o menor preenchimento de arestas possível no grafo G, de forma a torná-lo cordal.

Apresentaremos antes do modelo do problema de programação linear, um exemplo passo a passo do processo de tornar um grafo cordal através dessas estruturas.

Seja G o grafo conexo abaixo:

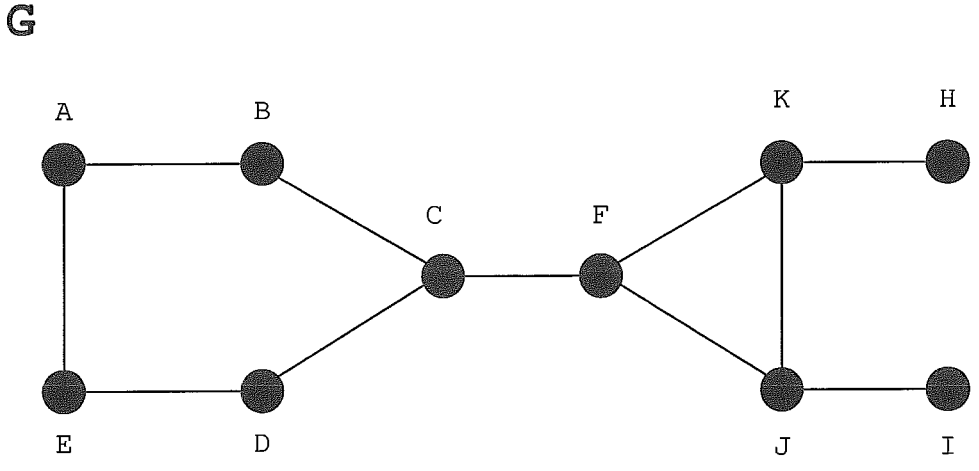


Figura 5.1: Exemplo de grafo conexo não-cordal

A partir do grafo G acima, devemos construir uma árvore de eliminação de seus vértices.

Para construírmos a árvore de eliminação do grafo G, teremos que ter uma ordem de eliminação de seus vértices.

Dada a rotulação a seguir, obtida através do método explicado na

subseção (3.4.1), construiremos a árvore de eliminação dos vértices de G.

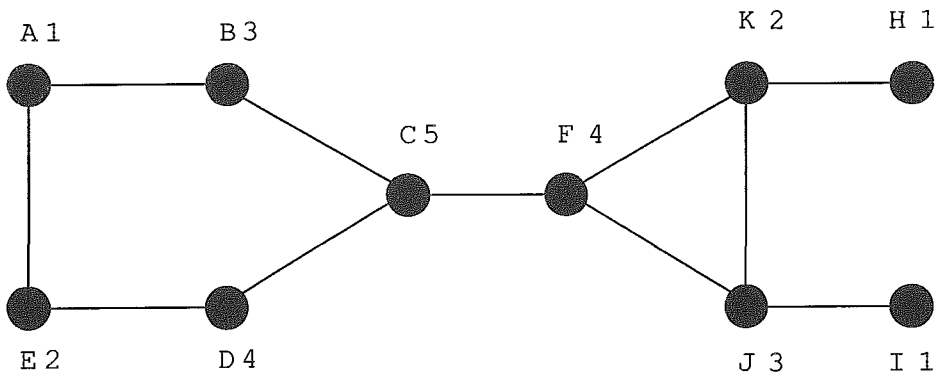


Figura 5.2: Definição de uma ordem de eliminação

A árvore de eliminação será construída a partir do maior rótulo, que será a raiz da árvore de eliminação, vértice C com rótulo 5, até os vértices com os menores rótulos, que serão as folhas, vértices A, H e I, com rótulos 1.

Com a raiz definida como o vértice C, teremos como descendentes imediatos de C, vértices com os maiores rótulos, de cada uma das duas componentes conexas de $G \setminus \{ C \}$, ainda não incluídos na árvore de eliminação, rótulos esses menores que o de C, como os vértice D e F, que possuem rótulo 4.

Como possuem o mesmo rótulo, os vértices D e F devem estar em ramificações diferentes da árvore de eliminação, como ilustrado na figura 5.3.

Tendo definido os vértices descendentes da raiz, teremos que definir os vértices descendentes de D e F na árvore de eliminação. Estes vértices serão os de maior rótulo de suas componentes conexas que ainda não foram introduzidos na árvore, vértices B e J, que possuem rótulo 3, como mostra

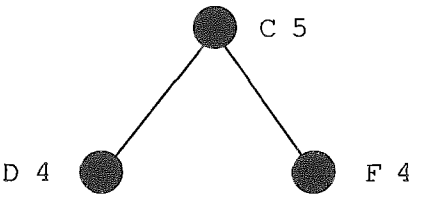


Figura 5.3: Definição dos vértices descendentes da raiz C

a figura 5.4.

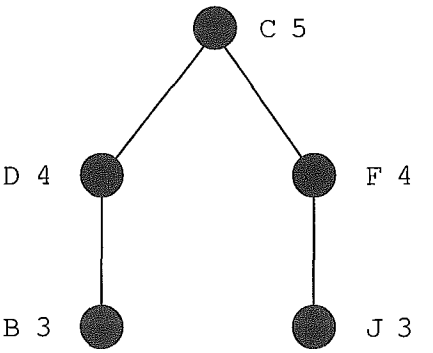


Figura 5.4: Definição dos vértices descendentes de D e F

Como a inclusão do vértice J na árvore acarreta na divisão de uma das componentes conexas em duas, os vértices de maior rótulo em cada uma das componentes serão incluídos descendentes de J na árvore de eliminação, ou seja, mesmo K e I não possuindo rótulos iguais, estes pertencerão a ramificações diferentes, mas com o mesmo vértice ascendente J.

Para o descendente de B, usaremos o mesmo processo usado anteriormente, ou seja, o vértice de maior rótulo da outra componente conexa que ainda não havia sido incluído na árvore de eliminação, será introduzido descendente imediato do vértice B, como na figura 5.5.

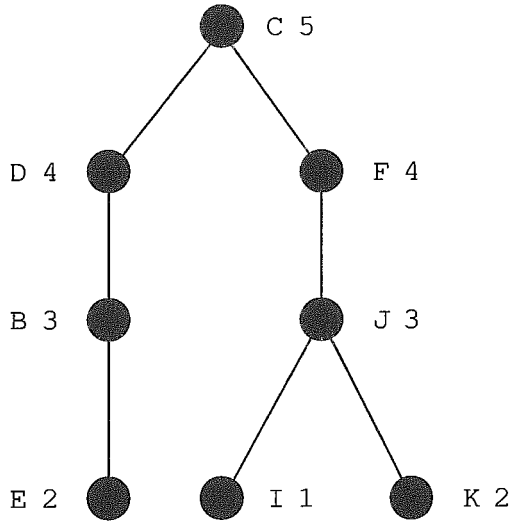


Figura 5.5: Definição dos vértices descendentes de B e J

Seguindo o procedimento, o vértice A, que possui rótulo 1 e é adjacente a E no grafo, será incluído descendente imediato do vértice E, como mostra a figura 5.6.

Assim, temos montada a árvore de eliminação T do grafo G, devemos então montar as subárvores da árvore de eliminação, que formarão uma família de estruturas correspondentes a um grafo de interseção.

Para cada vértice v na árvore de eliminação, haverá uma subárvore T_v correspondente, e para definirmos cada uma dessas subárvores, utilizaremos o procedimento descrito na subseção 3.4.2.

Cada subárvore T_v, terá raiz v, e será formada pela união de todos os caminhos entre o vértice v e os seus descendentes em T que pertencem à $Adj(\ v\ )$ no grafo original G.

Construiremos então a subárvore T_C, equivalente ao vértice raiz C. Para isso usaremos o conjunto $Adj(\ C\ )$ correspondente ao grafo G.

Como $Adj(\ C\ ) = \{\ B,\ D,\ F\ \}$, então a união dos caminhos deverá ter como destino os vértices B, D e F, uma vez que estes vértices são

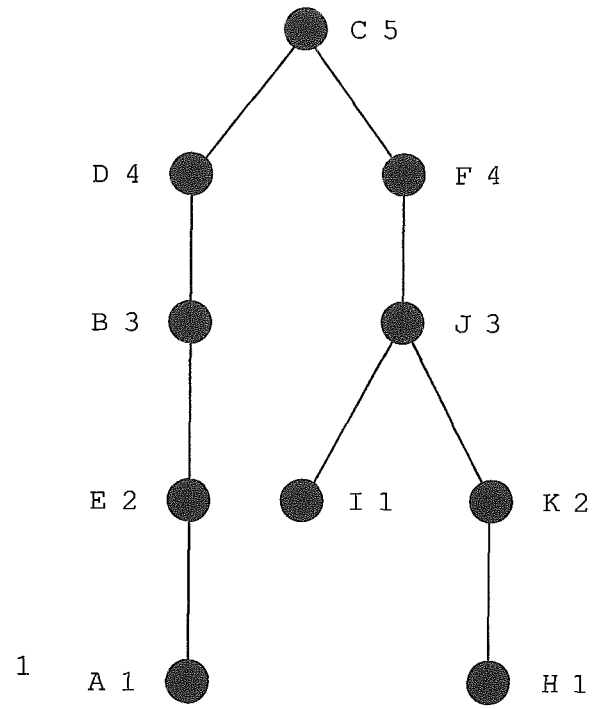


Figura 5.6: Definição dos vértices descendentes de E e K

descendentes de C na árvore, como mostra a figura 5.7.

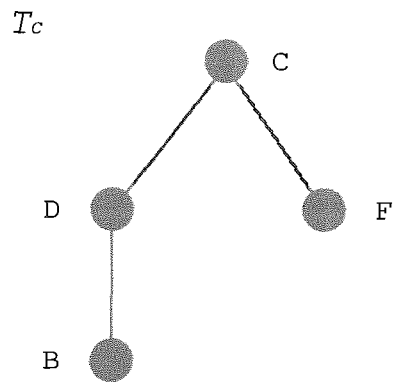


Figura 5.7: Definição da subárvore T_C

Após a construção da subárvore da raiz T_C, construiremos as subávores de seus descendentes, começando pelo vértice D.

O vértice D tem como vizinhos os vértices C e E no grafo G, ou seja,

$Adj(\ D\ ) = \{\ C,\ E\ \}$, então a união dos caminhos deverá ter como destino apenas o vértice E, pois o vértice C é ascendente na árvore de eliminação.

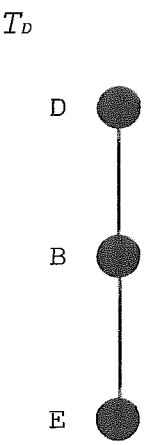


Figura 5.8: Definição da subárvore T_D

Pode-se perceber que a subárvore T_D contém o vértice B, mas no grafo G os vértices D e B não são adjacentes. Como veremos posteriormente, este fato estará associado a um preenchimento que ocorrerá e tornará o grafo G cordal.

Construiremos, a seguir, a subárvore do outro vizinho de C, a subárvore equivalente ao vértice F.

Como F tem como vizinhos os vértices C, K e J no grafo G, a união dos caminhos deverá ter como destino os vértices K e J, uma vez que C é a raiz de T, ou seja, é ascendente na árvore de eliminação, como mostra a figura 5.9.

Construiremos agora a subárvore T_B, equivalente ao vértice B.

No grafo G, o vértice B tem como vizinhos os vértices A e C, então a união dos caminhos terá como destino o vértice A, uma vez que C é a raiz da árvore de eliminação. Figura 5.10.

Na subárvore T_J, o a união dos caminhos terá como destino os vértices F, K e I, que pertencem a $Adj(\ J\ )$. Como mostra a figura 5.11.

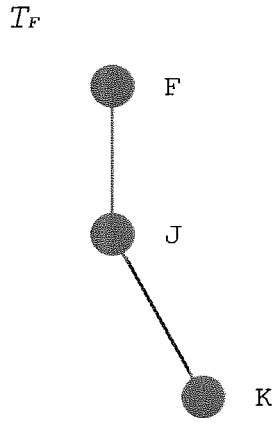


Figura 5.9: Definição da subárvore T_F

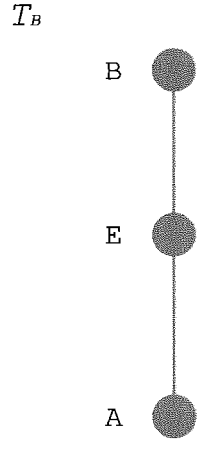


Figura 5.10: Definição da subárvore T_B

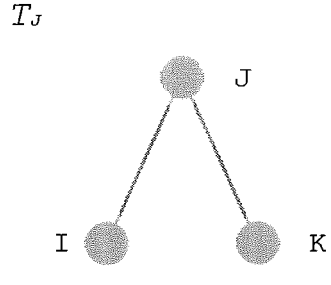


Figura 5.11: Definição da subárvore T_J

A união dos caminhos para a construção de T_E, terá como destino, os vértices pertencentes a $Adj(\ E\ )$, ou seja, os vértices A e D. Como D está acima na árvore de eliminação, terá como destino apenas a A. Como na figura 5.12.

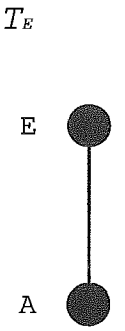


Figura 5.12: Definição da subárvore T_E

Para construir T_K, a união dos caminhos terá como destino o vértice H, uma vez que, mesmo pertencendo a $Adj(\ K\ )$, F e J são ascendentes de K na árvore de eliminação. Figura 5.13.

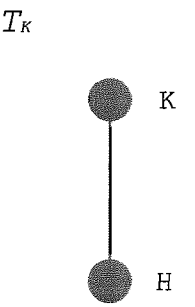


Figura 5.13: Definição da subárvore T_K

As folhas da árvore de eliminação não possuem descendentes, assim a união dos caminhos será apenas os próprios vértices, logo suas subárvores serão apenas as próprias folhas da árvore.

Para efeito da construção das interseções, consideraremos como as subárvores das folhas um traço, como mostra a figura 5.14.

Após construírmos as subárvores da árvore de eliminação, podemos então

T_A, T_H, e T_I

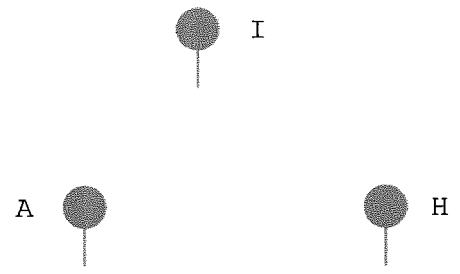


Figura 5.14: Definição da subárvore T_A, T_H e T_I

montar o grafo de interseção composto das interseções das subárvores, e assim, teremos o grafo $G*$ preenchido, devido as novas arestas que serão incluídas com as interseções. Como mostram as figuras 5.15 a 5.22.

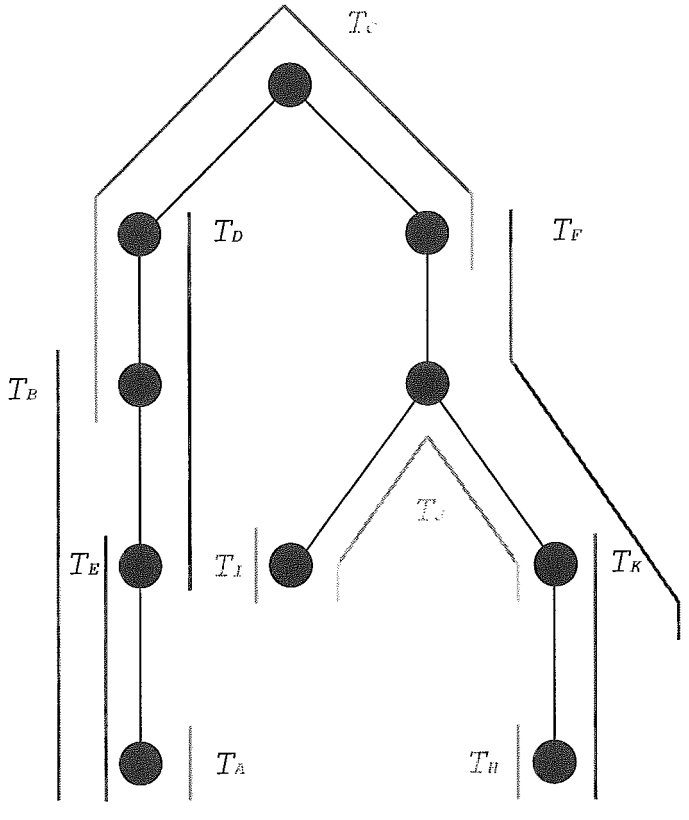


Figura 5.15: Subárvores da árvore de eliminação do grafo G

A união das subárvores da árvore de eliminação formam uma família de estruturas que correspondem a um grafo de interseção, onde cada subárvore representa um vértice, e cada interseção forma uma aresta no grafo $G*$, como definido na seção 3.3, sobre grafos de interseção.

A partir da união das subárvores, podemos então construir o grafo $G*$ (preenchido), ou seja, podemos construir um grafo $G*$ cordal.

Tomando cada interseção das subárvores como arestas, construiremos o grafo $G*$, agora cordal. Como a subárvore T_C intersecta as subárvores T_D, T_B e T_F, então existirá uma aresta entre C e cada um dos vértices D, B e F. Como mostra a figura 5.16.

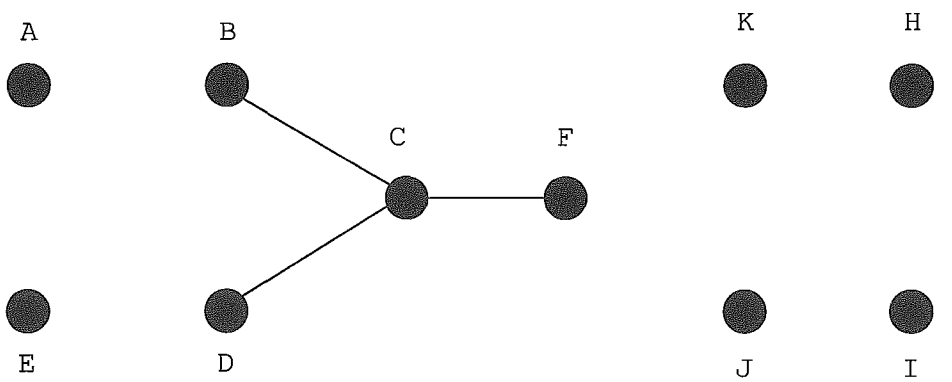


Figura 5.16: Vértices adjacentes ao vértice C no grafo $G*$

As interseções da subárvore T_F com as subárvores T_C, T_J e T_K acarretará em arestas entre o vértice F e os vértices C, J e K. Como já existe aresta entre os vértices F e C, não será necessário incluí-la. Figura 5.17.

A subárvore T_D intersecta as subárvores T_C, T_B e T_E, assim haverá uma aresta entre D e cada um dos vértices C, B e E, como para C já existe uma aresta, basta incluir uma aresta entre D e os vértices B e E. Como na figura 5.18.

A partir dessa figura, podemos perceber uma aresta (D, B) que não existia no grafo G original, ou seja, a interseção entre as subárvores T_B e T_D acarretou em um preenchimento no grafo, que o tornará cordal.

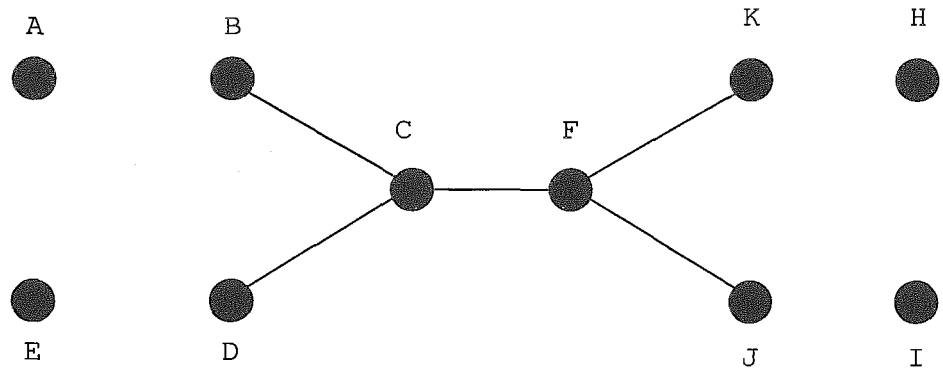


Figura 5.17: Vértices adjacentes ao vértice F no grafo $G*$

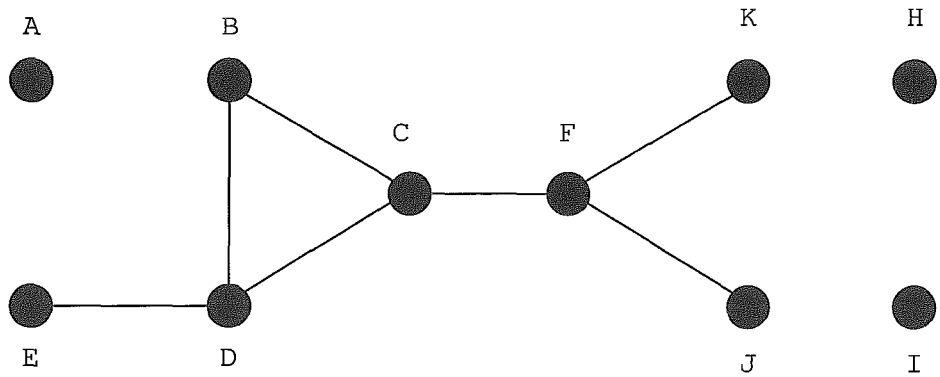


Figura 5.18: Vértices adjacentes ao vértice D no grafo $G*$

Para a subárvore T_J, temos que esta intersecta as subárvores T_F, T_K e T_I, não será necessário inserir uma aresta entre F e J, somente será incluída arestas entre o vértice J e os vértices K e I. como na figura 5.19.

A subárvore T_B intersecta as subárvores T_C, T_D, T_E e T_A, como já visto, as subárvores representadas por vértices que estão acima na árvore de eliminação, já possuem arestas entre eles, logo incluiremos arestas entre os vértices B e os vértices E e A. figura 5.20.

Para a subárvore T_K, temos que esta intersecta as subárvores T_F, T_J e T_H, logo construiremos arestas apenas para o vértice H, como visto na figura 5.21.

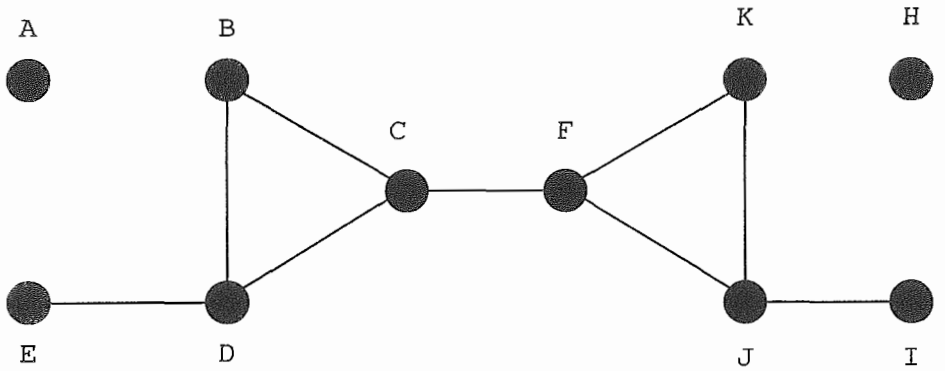


Figura 5.19: Vértices adjacentes ao vértice J no grafo $G*$

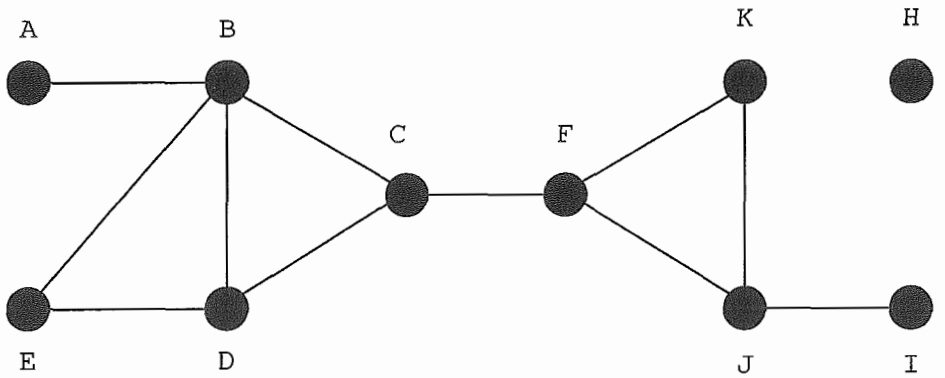


Figura 5.20: Vértices adjacentes ao vértice B no grafo $G*$

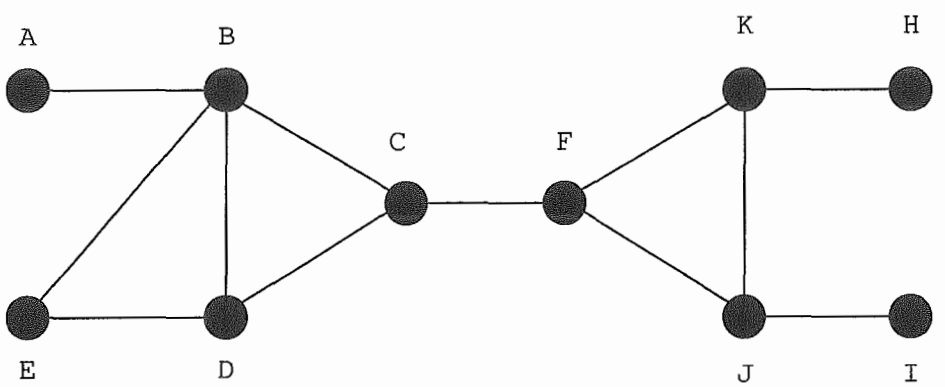


Figura 5.21: Vértices adjacentes ao vértice K no grafo $G*$

Na subárvore T_E, as subárvores que intersectam são T_D, T_B, e T_A, como T_D e T_B estão acima na árvore de eliminação, será inserida apenas uma

aresta entre E e A. Figura 5.22.

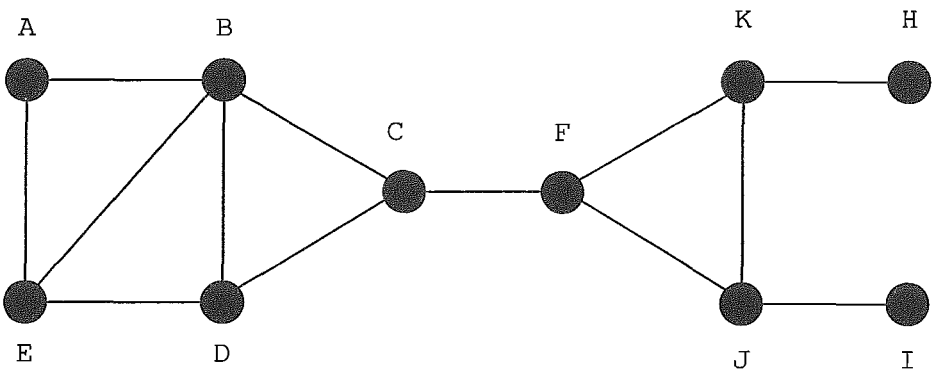


Figura 5.22: Vértices adjacentes ao vértice E no grafo $G*$

Finalmente, como as subárvores das folhas não possuem descendentes, não será necessário incluir mais nenhuma aresta, e com isso temos o nosso grafo preenchido $G*$.

Nesse novo grafo $G*$ preenchido, temos as novas arestas (E, B) e (B, D), que não existiam no grafo G, mas que pertencem ao conjunto de arestas do grafo $G*$, onde $G*$ é cordal.

Na seção seguinte, apresentaremos a formulação do modelo de programação linear mista que minimiza o número de arestas incluídas num dado grafo G para torná-lo cordal.

5.2 Modelo do problema de programação linear mista

Denotaremos, para construção deste modelo, por V o conjunto dos vértices do grafo G, onde $|V| = n$. Definiremos também um subconjunto $\bar{V}$ dos vértices de G, $\bar{V} \subset V$, em que $\bar{V}$ possui todos os vértices de V excetuando a raiz r, em outras palavras, $\bar{V} = V \setminus \{r\}$.

A seguir, definiremos as variáveis do problema, e mostraremos o modelo do problema de programação linear mista.

A variável binária $y_{i,j}$ representa a existência ou não de arestas na árvore de eliminação T entre os vértices i e j, ou seja,

$$y_{i,j} = 1, \text{ se } (i,j) \in T$$

$$y_{i,j} = 0, \text{ se } (i,j) \notin T.$$

Introduziremos uma outra variável binária $z_{i,j}$ definida para todo i, e todo j; $j > i$ que comporá as arestas da árvore. Esta variável, como veremos posteriormente, evitará que aconteça de termos a aresta $y_{i,j}$ e a aresta $y_{j,i}$ ao mesmo tempo na árvore T.

$$z_{i,j} \in \{0,\ 1\}.$$

Com o intuito de construir as subárvores, definiremos um fluxo unitário que parte invariavelmente da raiz de T para cada vértice $k \in T$ ($k \neq r$), ou seja, cada vértice k será destino de um fluxo unitário. A variável $f^k_{i,j}$ representa o fluxo que passa na aresta (i,j) com destino ao vértice k.

$$f^k_{i,j} \geq 0.$$

Após a construção da árvore de eliminação, temos então que construir as subárvores da árvore de eliminação. Com este objetivo, definimos uma variável $q^k_{i,j}$ que representa a existência ou não da aresta (i,j) na subárvore T_k, onde k é o vértice destino do fluxo $f^k_{i,j}$.

$$q^k_{i,j} = 1, \text{ se } (i,j) \in T_k$$

$$q^k_{i,j} = 0, \text{ se } (i,j) \notin T_k.$$

Após definirmos as variáveis do problema, apresentaremos então o modelo do programa de programação linear mista:

(P) :

$$min \quad \sum q_{i,j}^k, \qquad \forall i,\ \forall j,\ \forall k \tag{5.1}$$

$$sujeito\ a: \quad \sum_{i=1,\ j=i+1}^{n} z_{i,j} = n-1 \tag{5.2}$$

$$z_{i,j} \geq y_{i,j} + y_{j,i} \qquad \forall i;\ j > i \tag{5.3}$$

$$f_{i,j}^k \leq y_{i,j}, \qquad k \in \bar{V};\ \forall i;\ \forall j \tag{5.4}$$

$$\sum_{j=1;\ j\neq i}^{n} f_{j,i}^k - \sum_{j=1;\ j\neq i}^{n} f_{i,j}^k = 0, \quad k \in \bar{V};\ i \in \bar{V} \setminus \{k\} \tag{5.5}$$

$$\sum_{k=1}^{n} f_{j,k}^i + \sum_{k=1}^{n} f_{i,k}^j = 1, \quad \forall i;\ j > i;\ j \in Adj(i) \tag{5.6}$$

$$y_{i,r} = 0, \qquad \forall i \tag{5.7}$$

$$\sum_{i=1}^{n} f_{r,i}^k = 1, \qquad k \in \bar{V} \tag{5.8}$$

$$\sum_{i=1}^{n} y_{i,j} = 1, \qquad j \in \bar{V} \tag{5.9}$$

$$\sum_{j=1}^{n} f_{j,r}^k = 0, \qquad k \in \bar{V} \tag{5.10}$$

$$\sum_{j=1}^{n} f_{k,j}^k = 0, \qquad k \in \bar{V} \tag{5.11}$$

$$f_{r,j}^k = 0, \qquad j \in \bar{V};\ k \in \bar{V};\ j \notin Adj(r) \tag{5.12}$$

$$f_{i,j}^r = 0, \qquad \forall i,\ \forall j \tag{5.13}$$

$$q_{i,j}^k \geq f_{i,j}^w - f_{i,j}^k, \quad \forall i,\ \forall j,\ \forall k,\ \forall w,\ k \in Adj(w) \tag{5.14}$$

$$y_{i,j} \in \{0,\ 1\} \tag{5.15}$$

$$z_{i,j} \in \{0,\ 1\} \tag{5.16}$$

$$f_{i,j}^k \geq 0 \tag{5.17}$$

$$q_{i,j}^k \in \{0,\ 1\} \tag{5.18}$$

Considerando que o grafo G tem n vértices, a árvore de eliminação terá $n-1$ arestas, como determina a restrição (5.2), ou seja:

$$\sum_{i,j=1}^{n} z_{i,j} = n - 1.$$

Para evitar na construção da árvore de eliminação que a variável $y_{i,j}$ e a variável $y_{j,i}$ tenham valor unitário ao mesmo tempo, e assim a aresta (i,j) e a aresta (j,i) estarem ao mesmo tempo na árvore, temos a restrição (5.3), então para todo i e todo $j > i$:

$$z_{i,j} \geq y_{i,j} + y_{j,i}.$$

Definimos a variável $f_{i,j}^{k}$ como o fluxo que parte da raiz r com destino ao vértice k passando pela aresta (i,j). Garantimos que não haverá fluxo passando por uma aresta que não pertence à árvore de eliminação através da restrição (5.4), logo:

$$f_{i,j}^{k} \leq y_{i,j}, \quad \forall k.$$

Ou seja, se a aresta (i,j) não pertence a árvore então:

$$y_{i,j} = 0 \quad \text{e} \quad f_{i,j}^{k} = 0, \quad \forall k.$$

Não ocorre perda do fluxo que parte da raiz com destino a um determinado vértice k durante o percurso, ou seja, todo fluxo que entra em um vértice i, tal que $i \neq k$, é igual ao que sai deste mesmo vértice i, como visto na restrição (5.5), então para todo vértice i diferente da raiz r e do destino k, temos:

$$\sum_{j=1;\, j\neq i}^{n} f_{j,i}^{k} - \sum_{j=1;\, j\neq i}^{n} f_{i,j}^{k} = 0.$$

Esses fluxos determinam que arestas do grafo pertencem a cada subárvore da árvore de eliminação

A seguinte restrição garantirá, como mencionado anteriormente na seção 3.4.2, que na árvore de eliminação os vértices i e j adjacentes em G, sejam, ou i ascendente de j, ou j ascendente de i, relação que se mantém nas subárvores, restrição (5.6):

Se $(i,j) \in E(G)$ para $j > i$, então

$$\sum_{k=1}^{n} f_{j,k}^{i} + \sum_{k=1}^{n} f_{i,k}^{j} = 1.$$

Seja r a raiz da árvore de eliminação.

Como visto na restrição (5.7), teremos que r não será destino de nenhum vértice, isto é, para todo vértice $i \in \bar{V}$:

$$y_{i,r} = 0.$$

O fluxo que parte da raiz será sempre unitário para cada vértice destino k, restrição (5.8), logo:

$$\sum_{i} f_{r,i}^{k} = 1.$$

Definindo que todo vértice j, para $j \neq r$, terá exatamente um ascendente imediato, temos para $j \in \bar{V}$, (5.9):

$$\sum_{i=1}^{n} y_{i,j} = 1.$$

Como o fluxo é criado partindo da raiz com destino a um determinado vértice k, e a raiz não é destino de nenhum fluxo, temos então a restrição (5.10), logo para $k \in \bar{V}$:

$$\sum_{j=1}^{n} f_{j,r}^{k} = 0.$$

Cada fluxo criado tem um determinado vértice como destino, logo o fluxo não passará por esse vértice destino, como visto na restrição (5.11), então para $k \in \bar{V}$:

$$\sum_{j=1}^{n} f_{k,j}^{k} = 0.$$

Como o fluxo parte da raiz, e deverá passar por arestas da árvore de eliminação, então a restrição (5.12) faz com que o fluxo saia apenas para vértices que são adjacentes a raiz, logo, se $j \notin Adj(r)$, $j \in \bar{V}$ e para $k \in \bar{V}$:

$$f_{r,j}^{k} = 0.$$

Todo fluxo tem um vértice da árvore de eliminação como destino, e parte da raiz, logo não haverá fluxo para a raiz da árvore, como visto na restrição (5.13), então para todo i e para todo j:

$$f_{i,j}^{r} = 0.$$

A restrição (5.14) define exatamente as arestas que pertencerão as subárvores. Sendo k um vértice adjacente a w em G, a maior diferença, em cada aresta (i, j), entre os fluxos para w e k, dará o número de arestas da subárvore T_k, pois esta definirá as arestas que estarão exatamente após o vértice k, e que pertencerão a subárvore T_k. Então, $\forall\, w \in Adj(k)$, temos a restrição:

$$q_{i,j}^{k} \geq f_{i,j}^{w} - f_{i,j}^{k}.$$

Nesta restrição, como está se formando a subárvore T_k correspondente ao vértice k, será tomada a maior diferença entre os fluxos, em cada aresta,

que partem da raiz da árvore de eliminação para todo vértice adjacente a k no grafo, que seja descendente de k na árvore, e o fluxo que parte da raiz até o vértice k correspondente a essa subárvore T_k.

Dessa forma, resta somente o fluxo que passa por k com destino a seus vértices adjacentes no grafo G.

Assim teremos definido a maior diferença entre os fluxos que passam por k e têm como destino todos os vértices adjacentes a k no grafo G, com isso poderemos ter fluxos que passem por vértices que não são adjacentes a este vértice k no grafo original.

A aresta (i, j) na subárvore T_k existirá se a diferença entre os fluxos que passam por essa aresta for positivo, ou seja, se existir um fluxo com destino a algum vértice adjacente a k no grafo, e descendente de k na árvore passando por (i, j).

Como essas subárvores da árvore de eliminação formam uma família de estruturas que correspondem a um grafo de interseção, e estas arestas serão formadas por esses fluxos, então poderá haver interseção de subárvores correspondentes a vértices não-adjacentes no grafo original G.

Esta interseção de subárvores correspondentes a vértices não-adjacentes no grafo, indica o preenchimento com novas arestas, e tornará o grafo cordal, pois todo grafo de interseção de uma família de subárvores de uma árvore é um grafo cordal (Teorema 3).

Ao se minimizar o somatório da variável $q_{i,j}^k$, que determina as arestas que pertencerão as subárvores, e assim minimizar a quantidade de interseções nas subárvores, temos então a menor quantidade possível de interseções nas subárvores, e consequentemente, o menor preenchimento no grafo original G.

Capítulo 6

Relaxação Linear - Resultados Numéricos

O problema de completar um grafo para torná-lo cordal é um problema NP-completo [23]. Encontrar a solução exata para problemas aplicados é em geral, impraticável.

Desta forma heurísticas são propostas na literatura e oferecem soluções viávies para este problema, ou seja, oferecem limites superiores para a solução ótima. Uma maneira de avaliar estes limites superiores, é compará-los com limites inferiores para a solução.

Neste capítulo, temos como meta principal a obtenção de limites inferiores para o problema de completar um grafo para torná-lo cordal.

Estes limites serão calculados através da solução do problema de programação linear obtido ao se relaxar as restrições de integralidade das variáveis.

Sendo assim, o modelo de programação linear a ser resolvido é exatamente o modelo da seção (5.2), considerando-se no entanto que as variáveis binárias possam assumir qualquer valor no intervalo [0, 1].

Os métodos utilizados para resolver este modelo de programação linear serão o método simplex e o método de barreira (algoritmo de pontos interiores), ambos diponíveis no pacote de programas XPRESS-MP. Maiores informações ver [3, 18].

O método simplex é um dos métodos mais conhecidos de resolução de problemas de programação linear aplicados e tem se mostrado bastante eficiente. Em 1972, no entanto, Klee e Minty apresentaram um problema teórico com n restrições e $2n$ variáveis para o qual o método executa $2^n - 1$ iterações para encontrar a solução ótima do problema.

Um método cujo número de operações aritméticas requeridas por ele para solucionar um problema é limitado por um polinômio no tamanho do problema, é dito um método eficiente, ou seja, ele tem complexidade polinomial.

Um algoritmo com complexidade polinomial foi publicado por Karmarkar em 1984 para resolver um programa de programação linear, com boa performance quando aplicado a problemas práticos. Essa publicação originou um novo campo de pesquisa chamado de métodos de pontos interiores.

O método de pontos interiores caminha por dentro da região viável do problema, ao contrário do método simplex, que percorre a região viável pelos vértices, até que se encontre a solução ótima, aproveitando-se da estrutura combinatória do problema.

6.1 Resultados numéricos - XPRESS-MP

Nesta seção apresentaremos os resultados computacionais obtidos em nossos testes, utilizando o pacote de programas para resolver problemas de programacão linear XPRESS-MP. Fizemos os testes para grafos do tipo

grade.

A tabela 6.1 nos mostra resultados para grades quadradas 2x2, 3x3, 4x4, 5x5, 6x6, 7x7, além de grades 2x4, 2x5, 3x4, 4x5, 5x6, 6x7, entre outras.

Além do valor da função objetivo do modelo de programação linear relaxado, a tabela também mostra os resultados das heurísticas *Nested Dissection* e Grau Mínimo, e mostra também a quantidade de variáveis que cada instância produz, na modalagem do problema.

A coluna FO(NA) significa a função objetivo, que é correspondente ao número total de arestas da grade, ou seja, na grade 2x2, o resultado da função objetivo foi igual a 5, como a grade 2x2 possui 4 arestas, logo para torná-la um grafo cordal, foi necessária a introdução de uma aresta.

A coluna ND, nos mostra a quantidade de arestas totais (arestas originais + arestas de *fill*) necessárias, segundo a heurística *Nested Dissection*, para tornar o grafo cordal, e representa um limite superior do problema.

A coluna AMD, mostra também a quantidade de arestas totais (arestas originais + *fill*), necessárias para tornar o grafo cordal, segundo a heurística de grau mínimo, e também respresenta um limite superior do problema.

As colunas VB e VC, indicam o número de variáveis binárias e variáveis contínuas, respectivamente.

Na tabela 6.2, mostramos quantas iterações foram necessárias para se resolver o problema nos métodos simplex e barreiras, e o tempo consumido em cada método.

A coluna TMP(PI) é correspondente ao tempo em segundos que foi necessário para se resolver o problema através do pacote XPRESS-MP, se utilizando do algoritmo de pontos interiores (barreiras), já a coluna IT(PI) é o número de iterações requeridas no algoritmo de pontos interiores.

As colunas TMP(SPX) e IT(SPX) são, respectivamente, correspondentes

GRADE	FO(NA)	ND	AMD	VB	VC
G_2x2	5,000	5,000	5,000	6	144
G_2x3	9,000	9,000	9,000	15	468
G_2x4	13,000	13,000	13,000	28	1088
G_2x5	17,000	18,000	18,000	45	2100
G_2x6	21,000	23,000	23,000	66	3600
G_2x7	25,000	28,000	28,000	91	5684
G_2x8	29,000	33,000	33,000	120	8448
G_2x9	33,000	38,000	38,000	153	11988
G_2x10	37,000	43,000	43,000	190	16400
G_3x3	17,000	17,000	17,000	36	1539
G_3x4	24,318	26,000	26,000	66	3600
G_3x5	31,891	35,000	35,000	105	6975
G_3x6	39,427	45,000	45,000	153	11988
G_3x7	46,960	54,000	54,000	210	18963
G_3x8	54,488	64,000	64,000	276	28224
G_3x9	62,012	73,000	73,000	351	40095
G_3x10	69,531	83,000	83,000	435	54900
G_4x4	35,274	42,000	42,000	120	8448
G_4x5	47,214	59,000	59,000	190	16400
G_4x6	57,190	73,000	73,000	276	28224
G_4x7	68,195	90,000	90,000	378	44688
G_4x8	79,204	103,000	103,000	496	66560
G_4x9	90,211	120,000	120,000	630	94608
G_4x10	101,216	133,000	133,000	780	129600
G_5x5	61,210	81,000	81,000	300	31875
G_5x6	74,172	102,000	102,000	435	54900
G_6x6	92,200	131,000	131,000	630	94608
G_6x7	108,168	159,000	159,000	861	149940
G_7x7	129,186	198,000	198,000	1176	237699
G_7x8	148,160	234,000	234,000	1540	354368

Figura 6.1: Grades, resultado relaxado, heurística *Nested Dissection*, heurística Grau Mínimo, variáveis binárias e variáveis contínuas

ao tempo em segundos e ao número de iterações se utlizando do algoritmo simplex.

GRADE	TMP(PI)	IT(PI)	TMP(SPX)	IT(SPX)
G_2x2	0,030	8	0,020	47
G_2x3	0,110	94	0,140	323
G_2x4	0,520	337	0,780	951
G_2x5	1,650	737	4,350	2267
G_2x6	5,040	1355	18,060	4414
G_2x7	12,360	2344	69,130	8331
G_2x8	27,850	3729	205,630	14421
G_2x9	54,130	5482	407,200	19398
G_2x10	105,580	7839	1000,430	32504
G_3x3	0,940	169	2,380	1733
G_3x4	5,020	1146	35,250	6669
G_3x5	20,520	2567	204,280	15371
G_3x6	66,900	5163	860,540	32343
G_3x7	180,280	8911	2709,210	60746
G_3x8	420,440	79092	7129,760	102046
G_3x9	943,260	99485	25400,000	267272
G_3x10	1824,270	180516	—	—
G_4x4	30,130	3184	373,500	21513
G_4x5	129,880	52468	2081,530	53439
G_4x6	440,160	54648	11580,400	165672
G_4x7	1483,720	59188	50822,240	460986
G_4x8	2690,730	118532	—	—
G_4x9	5584,560	127087	—	—
G_4x10	10551,900	76459	—	—
G_5x5	691,830	128381	20487,130	235577
G_5x6	2226,090	31555	94517,430	704643
G_6x6	7626,910	54617	—	—
G_6x7	13322,220	98188	—	—
G_7x7	58340,990	156239	—	—
G_7x8	89259,600	257421	—	—

Figura 6.2: Tempo e número de iterações nos modos pontos interiores e simplex.

Podemos perceber pela avaliação da tabela 6.2, que nem todos os resultados da função objetivo foram números inteiros, isso se deve ao fato de estarmos resolvendo o problema de programação relaxado, sendo assim, os resultados são, na verdade, um limite inferior à solução do problema.

Para grades maiores, o tempo consumido foi muito grande, invariavelmente maior do que três dias, e portanto foram desconsiderados, sendo que algumas o próprio sistema não suportou.

Comparativamente, percebemos também que os resultados obtidos quando utilizado o algoritmo de pontos interiores, quase sempre foram mais velozes em relação ao tempo de execução.

Quanto ao número de iterações, os resultados obtidos com pontos interiores foram invariavelmente menores do que com o algoritmo simplex.

Alguns problemas não foram terminados ao utilizarmos o algoritmo simplex, isto é, para determinadas grades só foi possível utilizarmos o algoritmo de pontos interiores.

Capítulo 7

Conclusões

Identificar uma ordem de pivoteamento para a matriz A que cause o menor preenchimento possível com elementos não-nulos em A, é correspondente a encontrar o preenchimento mínimo de um grafo associado a essa matriz afim de torná-lo cordal. Desde que A seja quadrada, simétrica e definida positiva.

Pelos resultados obtidos, e apresentados no capítulo anterior, podemos afirmar que ainda existe campo para pesquisa e oportunidades para serem exploradas no problema de encontrar o preenchimento mínimo para tornar cordal um grafo.

O problema de minimizar o preenchimento de um grafo para torná-lo cordal, como já foi dito anteriormente, é um problema NP-completo, o que nos impõe a condição de buscarmos limites inferiores para a solução do problema através da resolução do modelo de programação linear de forma relaxada.

Na literatura, as heurísticas existentes, como *Nested Dissection* e Grau

Mínimo por exemplo, em geral não nos fornecem com exatidão resultados ótimos para o problema (solução exata), mas para determinadas instâncias fornecem resultados satisfatórios.

Muitas variações dessas heurísticas, especialmente os algoritmos híbridos [9, 10], que se utilizam das vantagens de cada um dos algoritmos existentes, tem mostrado resultados considerados muito bons, melhores que os resultados obtidos através das heurísticas.

Em nosso trabalho, buscamos uma formulação para o problema utilizando conceitos de árvore de eliminação e família de subárvores de uma árvore, que nos garante (Teorema 3) que o grafo de interseção resultante será cordal.

Em nossos testes, nos concentramos em avaliar o comportamento do modelo em grafos do tipo grades. Encontrar um limite inferior para o preenchimento de uma grade, afim de torná-la um grafo cordal tem sido assunto de grande interesse na literatura [6, 11, 7, 9].

É interessante ressaltar que o número de iterações ao se usar o algoritmo de pontos interiores, foi invariavelmente menor do que quando usado o algoritmo simplex, e ainda o algoritmo de pontos interiores mostrou, além de uma maior velocidade, comparado com o tempo no algoritmo simplex, um equilíbrio no consumo de tempo em relação a quantidade de variáveis de cada grade, ou seja, problemas com o mesmo número de variáveis consumiram tempos semelhantes.

Percebemos que o limite inferior obtido pelo problema de programação linear relaxado teve um *gap* muito grande comparado com os resultados das heurísticas, ou seja, diferiram bastante das soluções das heurísticas *Nested Dissection* e Grau Mínimo (limites superiores), principalmente para as grades maiores.

Este fato ocorreu a partir da grade quadrada 4x4, e nas outras a partir da 3x9. Tivemos uma diferença em relação aos resultados apresentados pelas

heurísticas, e tal diferença cresce quanto maior a dimensão da grade.

Isto nos permite afirmar que a busca de cortes na região viável do problema de programação linear inteira mista pode ser um campo de novas pesquisas na área, afim de que com esses cortes, tenhamos uma melhora nos limites inferiores, e assim aproximando os resultados da solução ótima do problema.

Bibliografia

[1] Berman, P., Schnitger, G. "On the Perfomance of the Minimum Degree Ordering for Gaussian Elimination". *SIAM J. Matrix Anal. Appl.*, v. 11, n. 1, pp. 83–88, 1990.

[2] Bornstein, C. F. *Parallelizing and De-parallelizing Elimination Orders.* Tese de doutorado, School of Computer Science - Carnegie Mellon University, Pittsburgh, PA 15213, 1998.

[3] Christofides, N. E. *Combinatorial Optimization.* Wiley, Chichester, England, 1979.

[4] Dirac, G. A. "On Rigid Circuit Graphs". *Abh. Math. Sem. Univ. Hamburg*, v. 25, pp. 71–76, 1961.

[5] George, A. *Nested Dissection of a Regular Finite Element Mesh.* Technical report, STAN-CS-208, Stanford University, 1971.

[6] George, A. "Nested Dissection of a Regular Finite Element Mesh". *SIAM J. Numer Anal.*, v. 10, n. 2, pp. 345–363, 1973.

[7] George, A., Liu, J. W. H. "The Evolution of the Minimum Degree Ordering Algorithm". *SIAM Review*, v. 31, n. 1, pp. 1–19, 1989.

[8] Golumbic, M. C. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, USA, 1980.

[9] Hendrickson, B., Rothberg, E. "Effective Sparse Matrix Ordering: Just Around the BEND". In *Proceedings of Eight SIAM Conf. Parallel Processing for Scientific Computing*, 1997.

[10] Hendrickson, B., Rothberg, E. "Improve the Runtime and Quality of Nested Dissection Ordering". *SIAM J. Sci. Stat. Comput.*, v. 20, n. 2, pp. 468–489, 1998.

[11] Hoffman, A. J., Martin, M. S., Rose, D. J. "Complexity Bounds for Regular Finite Difference and Finite Element Grids". *SIAM J. Numer. Anal.*, v. 10, n. 2, 1973.

[12] Lekkerkerker, C. G., Boland, J. C. "Representation of a Finite Graph by a Set of Intervals on the Real Line". *Fund. Math.*, v. 51, pp. 45–64, 1962.

[13] Lima, E. L. *Álgebra Linear.* Insituto de Matemática Pura e Aplicada, Rio de Janeiro, 1998.

[14] Lipton, R. J., Rose, D. J., Tarjan, R. E. "Generalized Nested Dissection". *SIAM J. Numer. Anal.*, v. 10, n. 2, pp. 346–358, 1979.

[15] Manne, F. *An Algorithm for Compution an Elimination Tree of Minimum Height for a Tree.* Working paper CS91 - 59, University of Bergen, Norway, 1991.

[16] Marczevsky, E. "Sur Deux Propriétés des Classes D'ensembles". *Fund. Math.*, v. 33, pp. 303–307, 1945.

[17] McKee, T. A., McMorris, F. R. *Intesection Graph Theory.* SIAM, Philadelphia, 1999.

[18] Papadimitriou, C. H. *Combinatorial Optimization: Algorithms and Complexity.* Englewoods Cliffes, Prentice Hall, 1982.

[19] Rose, D. J., Tarjan, R. E. "Algorithmic Aspect of Vertex Elimination on Directed Graphs". *SIAM J. Appl. Math.*, v. 34, pp. 176–197, 1978.

[20] Rothberg, E., Hendrickson, B. "Sparse Matrix Ordering Methods for Interior Point Linear Programming". *INFORMS J. Comput.*, v. 10, n. 1, pp. 107–113, 1998.

[21] Ruggiero, M. A. Gomes, Lopes, V. L. Rocha. *Cálculo Numérico: Aspetos Teóricos e Computacionais.* Departamento de Matemática Aplicada - IMECC - UNICAMP - Makron Books, São Paulo, 1996.

[22] Szwarficter, J. L. *Grafos e Algoritmos Computacionais.* Editora de Campus, Rio de Janeiro, 1983.

[23] Yannakakis, M. "Computing the Minimum Fill-in is NP-complete". *SIAM-Journal of Algebraic and Discrete Methods*, v. 2, pp. 77–79, 1981.