Aplicação de técnicas de fatoração de matrizes esparsas para inferência em redes bayesianas

Ernesto Coutinho Colla

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Área de Concentração: Ciências da Computação
Orientador: Prof. Dr. Julio Michael Stern

São Paulo, 31 Outubro de 2007

Aplicação de técnicas de fatoração de matrizes esparsas para inferência em redes bayesianas

Este exemplar corresponde à redação
final da dissertação/tese devidamente corrigida
e defendida por Ernesto Coutinho Colla
e aprovada pela Comissão Julgadora.

Banca Examinadora:

- Prof. Dr. Julio Michael Stern (orientador) - IME-USP.
- Prof. Dr. Carlos Alberto de Bragança Pereira - IME-USP.
- Prof. Dr. Fabio Gagliardi Cozman - POLI-USP.

Resumo

O objetivo deste trabalho foi desenvolver uma aplicação computacional que demonstre como técnicas de álgebra linear computacional aplicadas a fatoração de matrizes esparsas podem ser utilizadas para construir um algoritmo eficiente e paralelizável para inferência em Redes Bayesianas. Para atingir este objetivo o algoritmo implementado separa o processo de inferência em duas fases, a primeira fase simbólica e uma segunda fase numérica. Como será demonstrado, o processamento numérico da segunda fase pode ser otimizado e paralelizado utilizando estruturas de dados estáticas previamente alocadas e definidas na primeira fase. Esta separação viabilizou-se pela análise de algoritmos de fatoração de matrizes esparsas e algoritmos para inferência em Redes Bayesianas a partir de um arcabouço combinatório unificado. As estruturas combinatórias geradas na fase simbólica e comum aos dois processos são a chave para a implementação computacionalmente eficiente de um algoritmo capaz de lidar com grandes modelos.

Palavras-chave: Inferência em redes bayesianas, Fatoração de matrizes esparsas, Árvores de eliminação

Abstract

The main goal of this work is to show how sparse factorization methods could be used to build a efficient and parallel algorithm for inference in Bayesian networks. It was achieved with a complete separation between a first symbolic phase and a second numerical phase. Using static data structures previously allocated in the first phase the second one could be fully vectorized and parallelized. It is done examining the decoupling operations for sparse matrix models and inference algorithm in Bayesian networks from a unified combinatorial framework. This combinatorial structure is the key for implementing efficient computational algorithms capable of handling large models.

Keywords: Inference in bayesian networks, Sparce matrix factorizations, Elimination trees.

Sumário

Lista de Abreviaturas

BEL	Busca em Largura.
ENN	Elemento não Nulo.
ENNs	Elementos não Nulo.
GDA	Grafo Direcionado Acíclico.

Lista de Símbolos

V_G	Conjunto de vértices de um grafo
Γ_G	Função de filiação de um grafo
$G = (V_G, \Gamma_G)$	Grafo definido como conjunto de vértices e função de filiação
$H = (H_G, \Gamma_H)$	Árvore definida como conjunto de vértices e função de filiação
B_G	Matriz booleana de adjacência do Grafo G
F	Grafo preenchido ao longo do processo de eliminação dos nós de um grafo
X_i	Uma variável aleatória
$\mathbf{X}$	Conjunto de variáveis aleatórias
$\mathbf{X}_q$	Conjunto de variáveis questionadas no procedimento de inferência
$\mathbf{X}_R$	Conjunto de variáveis requisitadas no procedimento de inferência
$\mathbf{X}_E$	Conjunto de variáveis observadas no procedimento de inferência
E	Evidência composta por um conjunto de variáveis observadas
$\underline{e}_i$	*finding* utilizado para representar a informação a respeito do estado de uma variável
$pa(X_i)$	Conjuntos de vértices (ou variáveis) que são pais da variável X_i
$ch(X_i)$	Conjuntos de vértices (ou variáveis) que são filhos da variável X_i
$sp(X_i)$	Conjuntos de vértices (ou variáveis) que são esposos da variável X_i
$mb(X_i)$	Conjuntos de vértices que compões o *Markov Blanket* da variável X_i
$P(X\|Y)$	Probabilidade de uma variável X condicionada a uma variável Y
$\phi(X_i)$	Potencial da variável X_i
$dom\phi_i$	Domínio do potencial da variável X_i
$\Phi = \{\phi_1, ..., \phi_n\}$	Potenciais de um conjunto de variáveis $\mathbf{X} = \{X_1, ..., X_n\}$ com domínios $dom\phi_i$

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O objetivo deste trabalho é desenvolver uma aplicação computacional que demonstre como técnicas de álgebra linear computacional aplicadas a fatoração de matrizes esparsas podem ser utilizadas para construir um algoritmo eficiente e paralelizável para inferência em Redes Bayesianas. Este objetivo é atingido com a completa separação do processo de inferência em duas fases, a primeira fase simbólica e uma segunda fase numérica. Como será demonstrado, o processamento numérico da segunda fase pode ser otimizado e paralelizado utilizando estruturas de dados estáticas previamente alocadas e definidas na primeira fase. Esta separação é possível através da análise de algoritmos de fatoração de matrizes esparsas e algoritmos para inferência em Redes Bayesianas a partir de um arcabouço combinatório comum. As estruturas combinatórias da fase simbólica comum aos dois processos são a chave para a implementação computacionalmente eficiente de um algoritmo capaz de lidar com grandes modelos.

Rede Bayesiana é um exemplo de modelo gráfico probabilístico que pode ser utilizado para representar de forma gráfica e compacta a distribuição conjunta de probabilidades de variáveis aleatórias. Tais modelos probabilísticos são o resultado da união entre a teoria de probabilidade e a teoria de grafos, e mostram-se convenientes para lidar com problemas que envolvem incerteza e complexidade. Provavelmente por este motivo estão cada vez mais ocupando um papel importante em aplicações de suporte a decisão e *machine learning*.

Modelos gráficos probabilísticos, e mais especificamente Redes Bayesianas, são utilizados em aplicações de diagnóstico, prognóstico e planejamento do tratamento médico [1], [18], diagnósticos

em falhas de equipamentos e desenvolvimento de software conforme descrito por [7], [10]. Na área de finanças, modelos desta natureza foram aplicados para análise de portfólio de investimentos [3] e em sistemas de análise de crédito como por exemplo Chang *et al* [13], Abramowicz [22], Baesens *et al.* [2].

A idéia fundamental dos modelos gráficos é a noção de modularidade, a qual permite construir, ou modelar, sistemas complexos combinando partes mais simples. Neste contexto, a teoria de probabilidade é utilizada para combinar as partes elementares, garantir a consistência do modelo como uma todo e prover uma interface entre o modelo e os dados empíricos.

No caso específico de Redes Bayesianas, os métodos Bayesianos provêm o formalismo para o raciocínio (*reasoning*) sobre convicções ou crenças (parciais) na presença de incertezas. Neste formalismo, parâmetros numéricos são atribuídos às proposições de acordo com o grau de convicção determinado por um corpo de conhecimento. Tais parâmetros podem então ser combinados e manipulados para inferir resultados a partir do modelo, de acordo com as regras da teoria de probabilidade, condicionadas à topologia do grafo que o representa.

A teoria dos grafos, aplicada à representação gráfica dos modelos, fornece duas vantagens significativas: i) uma interface intuitiva que torna possível criar modelos nos quais os conjuntos de variáveis apresentam uma grande interação entre si; (ii) uma forma de estruturar os dados, implícita ao modelo gráfico, que o torna naturalmente eficiente para aplicação em algoritmos computacionais.

Nas aplicações de Redes Bayesianas, e em modelos probabilísticos em geral, pode-se distinguir dois problemas a serem resolvidos. O primeiro corresponde à construção do modelo, que inclui a identificação da topologia do grafo que melhor representa o fenômeno probabilístico que se deseja modelar, bem como a obtenção dos parâmetros que caracterizam as distribuições de probabilidades envolvidas. Concluída a etapa de construção do modelo, o passo seguinte é a utilização do mesmo para inferir sobre os eventos probabilísticos incorporados a ele. O presente trabalho adota como conhecidos a estrutura e os parâmetros do modelo e foca na aplicação do mesmo para inferência de eventos probabilísticos.

O principal objetivo é explorar a analogia entre os algoritmos de inferência em Redes Bayesianas e as técnicas e heurísticas aplicáveis à fatoração simétrica de matrizes esparsas, para desenvolver uma forma computacionalmente mais eficiente para utilizar a rede em exercícios de inferência. Para tanto, nos primeiros capítulos deste trabalho são detalhados três procedimentos para posteriormente relacioná-los entre si, a saber: i) o processo de eliminação de nós em um grafo; ii) o processo de

eliminação de colunas na fatoração de matrizes simétricas (Fatoração de Cholesky); iii) o Algoritmo de Eliminação de Variáveis para inferência em Redes Bayesianas focado em operações algébricas.

Explorando a analogia entre os três processos de eliminação pretende-se implementar um algoritmo de inferência em Redes Bayesianas estruturado em duas fases distintas. Uma primeira fase simbólica e uma segunda fase numérica. A vantagem potencial desta estratégia é o ganho de eficiência ao adotar para cada fase a estrutura de dados mais eficiente e adaptada a cada uma delas. Como ficará claro ao longo da exposição teórica, a análise simbólica permite caracterizar as estruturas de dados que serão manipulados numericamente, identificar a ordem na qual as operações numéricas devem ser realizadas, bem como a possibilidade de explorar a paralelização no processamento numérico, disponível atualmente nos compiladores das principais linguagens de programação e nos processadores.

Concluída a introdução do trabalho, cujo propósito foi, de forma sucinta, apresentar os objetivos do mesmo, contextualizar e demonstrar a aplicação dos modelos gráficos probabilísticos, o Capítulo 2 introduz a teoria de Redes Bayesianas. Para tanto, faz-se necessário abordar os dois arcabouços teóricos intrínsecos às Redes Bayesianas: a teoria dos grafos e teoria probabilística. O capítulo inicia-se com a teoria dos grafos, apresentando formalmente as definições e conceitos como matriz de adjacência, árvores e grafos de eliminação, entre outros, que serão essenciais para o desenvolvimento do trabalho. Em seguida apresentam-se os conceitos e os resultados da teoria de probabilidade e inferência necessários para compreender como os parâmetros dos modelos serão manipulados. Na última seção do capítulo juntam-se os aspectos teóricos das seções anteriores para descrever formalmente as Redes Bayesianas e apresentar o algoritmo de inferência que será utilizado.

O Capítulo 3 aborda a teoria de álgebra linear computacional para descrever o método de fatoração simétrica de matrizes esparsas. Explicita-se a forma como a teoria dos grafos pode ser útil para tornar computacionalmente eficiente este processo. A teoria de grafos incorporada tanto no processo de inferência em Redes Bayesianas quanto na otimização da fatoração de matrizes possibilita a aplicação de técnicas e heurísticas comuns a ambos os processos.

No capítulo 4 os arcabouços teóricos de Redes Bayesianas e Álgebra Linear Computacional são combinados para desenvolver um algoritmo capaz otimizar e paralelizar o processamento computacional do Método de Eliminação de Variáveis para inferência em Redes Bayesianas. Neste capítulo descreve-se o algoritmo proposto e detalha-se a Fase Simbólica e a Fase Numérica com o objetivo de explicitar como as estruturas combinatórias geradas na Fase Simbólica são utilizadas para otimizar o

processamento numérico. Para auxiliar na compreensão integral do procedimento, ao final do capítulo apresenta-se um exemplo numérico completo de inferência no qual detalhe-se e exemplifica-se cada etapa do processo.

No Capítulo 5 descreve-se os principais procedimentos e estruturas de dados utilizados para a implementação computacional de um conjunto de bibliotecas para calcular a inferência em Redes Bayesianas. Este conjunto de bibliotecas explora as analogias entre os processos de fatoração de matrizes esparsas e de eliminação de variáveis para inferência em Redes Bayesianas para desenvolver um algoritmo paralelizável e computacionalmente eficiente de inferência. O capítulo apresenta o pseudocódigo e a representação gráfica dos principais algoritmos implementados.

O Capítulo 6 apresenta os procedimentos e os principais resultados dos testes realizados no conjunto de bibliotecas implementados. Foram feitos testes de duas naturezas: (i) Precisão numérica dos resultados; (ii) Testes comparativos de performance. O primeiro tipo de teste tem o objetivo de verificar a qualidade dos resultados obtidos e para tanto, os mesmos foram comparados com simulações feitas manualmente e com os resultados de aplicações similares disponíveis na comunidade científica. Os testes comparativos de performance foram executados comparando duas aplicações similares que foram desenvolvidas com um conjunto comum de bibliotecas básicas. No entanto, elas se diferenciam pois uma executa o procedimento de eliminação de variáveis explorando as possibilidades de paralelização das operações e a outra o faz de forma sequencial e seriada.

O último capítulo faz uma compilação dos principais resultados e propõe extensões e possibilidades de aperfeiçoamento para a continuidade do projeto que apenas iniciou-se com este trabalho. Ao longo da elaboração deste trabalho e através de sugestões de colegas foram enumeradas possibilidades de melhorias e otimização dos algoritmos, bem como possibilidade de aplicação das técnicas aqui propostas, que por uma questão de tempo e escopo do mesmo não puderam ser incorporadas.

Capítulo 2

Introdução à Teoria de Redes Bayesianas

O objetivo deste capítulo é introduzir os conceitos básicos de Redes Bayesianas que serão essenciais o entendimento do presente trabalho. Sucintamente, uma Rede Bayesiana é a representação gráfica de uma distribuição conjunta de probabilidades. Formalmente a estrutura gráfica de uma Rede Bayesiana é um *grafo*. Por este motivo iniciou-se o capítulo com um resumo da teoria dos grafos para nas seções seguintes, 2.3 e 2.4, apresentar repectivamente, uma definição formal de Redes Bayesianas e uma descrição teórica do procedimento de inferência. Ao final do capítulo detalha-se o algoritmo Eliminação de Variáveis, cujo foco são as operações algébricas e que será fundamental para a técnica computacional que o trabalho propõe.

2.1 Conceitos Básicos de Grafos

Nesta seção serão apresentados os conceitos básicos da teoria de grafos necessários para embasar teoricamente as seções que se seguem. Um tratamento teórico mais abrangente sobre grafos pode ser encontrado em [20], [9].

Um **grafo direcionado** é um par ordenado $G = (V_G, \Gamma_G)$ no qual o primeiro elemento é um conjunto finito de **vértices**, também denominados de **nós**, e o segundo elemento é uma **função de filiação** $\Gamma_G : V_G \to P(V_G)$, onde $P(V_G)$ é conjunto das partes de V_G.

Um grafo direcionado pode ser definido também como um par ordenado $G = (V_G, A_G)$, onde, A_G é um subconjunto do conjunto V_G x V_G de pares ordenados de vértices denominados **arestas**, ou **arcos direcionados**, de G no qual $(i, j) \in A_G \Leftrightarrow j \in \Gamma_G(i)$. Portanto, diz-se que dois vértices i e j são **vértices adjacentes**, ou vizinhos, se o par (i, j) pertence a A_G.

Dado um grafo $G = (V_G, \Gamma_G)$ define-se a **função de paternidade** Γ_G^{-1} como: $i \in \Gamma_G^{-1}(j) \Leftrightarrow j \in \Gamma_G(i)$.

Uma forma alternativa de definir um grafo direcionado, e que será particularmente útil para o presente trabalho, é pelo par ordenado $G = (V_G, B_G)$ onde B_G é a **matriz de adjacência** definida a partir de uma matriz booleana tal que $B_i^j = 1 \Leftrightarrow j \in \Gamma_G(i)$. Serão utilizadas as notações: B_i^j e $B(i, j)$ para indicar é o elemento da *i-ésima* linha e *j-ésima* coluna da matriz B.

Adotar-se-á V_G como um conjunto indexado, $V_G = v_1, v_2, ..., v_n$, ou então, V_G como sendo o próprio conjunto com os n primeiros inteiros positivos $N = 1, 2, ..., n$. Para não sobrecarregar a notação quando não houver ambigüidade (V_G, Γ_G) será indicado simplesmente por (V, Γ).

Graficamente é comum representar um grafo como um conjunto de pontos, os vértices, e um conjunto de setas, os arcos direcionados, que vão de cada vértice $v \in V$ para os elementos de $\Gamma(v)$.

Conforme será possível verificar no Exemplo 2.1.1 abaixo, considerando a representação gráfica usualmente utilizada pode-se visualizar as seguintes definições da teoria dos grafos:

- Um **loop** é uma aresta que parte e chega no mesmo vértice.
- Duas arestas, (i, j) e (k, l), são **contíguas** se a primeira chega no vértice de que a segunda parte, isto é, se $j = k$.
- Um **passeio** é uma seqüência não vazia e finita de arestas contíguas.
- Um **circuito** é um passeio que inicia-se e termina no mesmo nó, ou seja, o primeiro vértice da primeira aresta será o segundo vértice da última aresta do passeio.
- Uma **trilha** é um passeio no qual não se repete nenhuma aresta.
- Um **caminho** é uma trilha na qual não há circuitos.
- Um **ciclo** é uma trilha na qual o único circuito é toda a trilha.

Note que um passeio $C = ((w_1, w_2), (w_2, w_3), ..., (w_{k-1}, w_k))$ está igualmente determinado pela seqüência de vértices $C = (w_1, w_2, w_3, ..., w_{k-1}, w_k)$.

Exemplo 2.1.1

Considere o grafo abaixo:

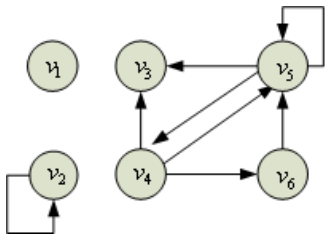


Figura 2.1: Grafo $G = (V_G, \Gamma_G)$

O conjunto de vértices do grafo da figura $V_G = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ e função de filiação: $\Gamma(v_1) = \emptyset$, $\Gamma(v_2) = \{v_2\}$, $\Gamma(v_3) = \emptyset$, $\Gamma(v_4) = \{v_3, v_5, v_6\}$, $\Gamma(v_5) = \{v_3, v_5, v_4\}$, $\Gamma(v_6) = \{v_5\}$. O conjunto de arestas $A_G = \{(v_2, v_2), (v_4, v_3), (v_4, v_5), (v_4, v_6), (v_5, v_3), (v_5, v_4), (v_5, v_5), (v_6, v_5)\}$.

A matriz de adjacência que representa o grafo:

$$\mathbf{B_G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- loop: (v_2, v_2)
- passeio: $((v_4, v_5), (v_5, v_5)(v_5, v_5)(v_5, v_4)(v_4, v_5))$
- circuito: $(v_5, v_5, v_4, v_6, v_5, v_5)$
- trilha: $(v_5, v_4, v_6, v_5, v_5, v_3)$
- caminho: (v_6, v_5, v_4, v_3)
- ciclo: (v_5, v_4, v_6, v_5)

Define-se **grau** de um nó $v \in V_G$ de grafo $G = (V_G, A_G)$ como o número de arcos de A_G contendo v, ou seja, o **grau** de um nó corresponde ao número de arcos que entram ou saem do nó v. Por exemplo, na Figura 2.1 o grau do nó v_5 é 5.

Em um grafo direcionado $G = (V_G, \Gamma_G)$ o nó i pertence ao conjunto dos **pais** do nó j, $i \in pa(j)$, ou j pertence ao conjunto dos **filhos** do nó i, $j \in ch(i)$, se $j \in \Gamma_G(i)$, ou seja, se existe um arco que sai de i e chega em j. Os filhos de i, os filhos dos filhos de i, e assim por diante, são **descendentes** do nó i. Para representar o conjuntos de descendentes de um nó define-se a **função de descendência** $\bar{\Gamma}_G$. O conjunto de descentes do nó i será portanto $\bar{\Gamma}_G(i)$. Se um nó j é um descendente de i, ou seja $j \in \bar{\Gamma}_G(i)$, então existe um caminho em G que inicia-se em i e termina em j.

O nó j pertence ao conjunto dos nós **esposos** de i, $j \in sp(i)$, se existe ao menos um nó k tal que $k \in \Gamma_G(i)$ e $k \in \Gamma_G(j)$, ou seja, i e j tem um filho em comum.

Um subconjunto de vértices, $S \subset V_G$, é um **separador** entre dois vértices quaisquer $a \in V_G$ e $b \in V_G$ se e somente se a remoção de S deixa os vértices a e b em duas componentes conexas distintas. Decorre portanto que S separa a e b se qualquer caminho de a a b passa por nós em S. Estendendo o conceito de separador para subconjuntos de vértices, dir-se-á que $S \subset V_G$ é um separador entre os subconjuntos de vértices $A \subset V_G$ e $B \subset V_G$ se e somente se, $\forall a \in A$, $\forall b \in B$, S separa a e b.

Um **Grafo Direcionado Acíclico**, ou GDA, é um grafo direcionado que não possui ciclos.

Exemplo 2.1.2

Grafo Direcionado Acíclico

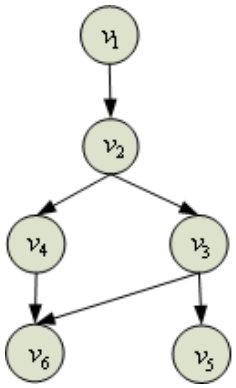


Figura 2.2: Exemplo de Grafo Direcionado Acíclico

Valem as seguintes relações:

- v_2 é filho de v_1.

- v_3 e v_4 são filhos de v_2.
- v_3 é pai de v_5 e v_6.
- v_3 e v_4 são pais de v_6.
- v_3 e v_4 são esposos.
- v_3, v_4, v_5, v_6 são descendentes de v_2.
- v_5 e v_3 não são descendentes de v_4.

§

Uma **árvore** $H = (V_H, \Gamma_H)$ de raiz $v \in V_H$ é um GDA, no qual todos os vértices tem no máximo um pai e apenas a raiz não tem pai. Denominar-se-ão **folhas** aos vértices sem filhos de uma árvore. Um grafo composto por uma conjunto de árvores denomina-se **floresta**.

Um **grafo não direcionado** é um grafo $G = (V_G, \Gamma_G)$ direcionado no qual $j \in \Gamma_G(i) \Leftrightarrow i \in \Gamma_G(j)$, ou seja, os arcos (i, j) e (j, i) pertencem ao grafo. Um grafo não direcionado pode ser representado pelo par ordenado $G = (V_G, E_G)$ onde cada arco não direcionado $\{i, j\} \in E_G$ corresponde ao par de arcos direcionados em sentidos opostos (i, j) e (j, i). Analogamente a um grafo direcionado, um grafo não direcionado pode ser representado por uma matriz booleana B_G, denominada matriz de adjacência, na qual $B_G(i, j) = B_G(j, i) = 1$ se $\{i, j\} \in E_G$, e $B_G(i, j) = B_G(j, i) = 0$ caso contrário.

O **Grafo Moral** (*Moral Graph*) $M(G)$ de um GDA G é um grafo não direcionado com o mesmo conjunto de nós de G, arcos não direcionais unindo os nós que estavam unidos em G e adicionalmente contendo arcos não direcionais unindo os nós i e j se eles forem parentes imediatos em G. O conjunto dos parentes imediatos de um nó i em no grafo G inclui seus pais, $pa(i)$, seus filhos, $ch(i)$, e seus esposos, $sp(i)$. A este conjunto de parentes imediatos denomina-se **Markov Blanket** de i, mb(i). Se $j \in mb(i)$ então j será um nó adjacente a i no Grafo Moral. A Figura 2.3 apresenta o grafo direcionado (a) do exemplo anterior, seu Grafo Moral (b) e o Markov Blanket para o nó v_4, (c). Os arcos não direcionados adicionais incorporados ao Grafo Moral são usualmente denominados de **arcos morais** (*moral links*) pois conectam dois nós com um ou mais filhos em comum.

A ordenação de um conjunto de vértices $V = \{v_1, v_2, ..., v_n\}$ de um grafo pode ser descrita utilizando um vetor de índices q. O vetor de índices $q = [1, 2, ..., N]$ representa a ordem natural dos nós. A partir de uma permutação σ da ordem natural $[1, 2, ..., N]$ pode-se obter uma nova ordenação, a qual será indicada pelo vetor $q = [\sigma(1), \sigma(2), ..., \sigma(N)]$ ou, quando não houver ambiguidade, simplesmente por $q = [(1), (2), ..., (N)]$.

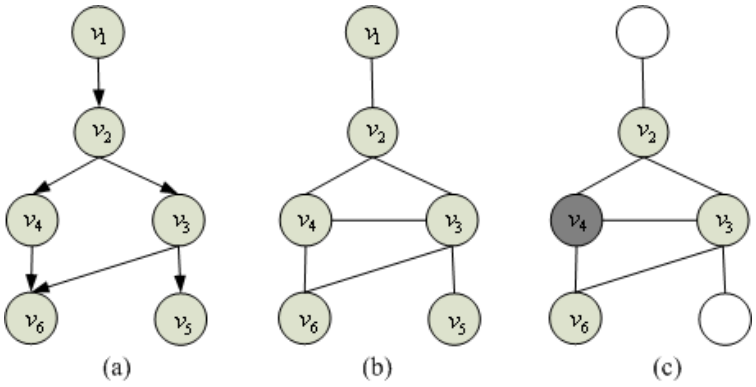


Figura 2.3: Exemplo de GDA e Grafo Moral

Define-se o processo de eliminação em um grafo não direcionado $G = (V, E)$, para uma dada uma ordem, $q = [(1), (2), ..., (N)]$, como uma seqüência de **grafos de eliminação** $G_k = (V_k, E_k)$ na qual, para $k = 1, 2, ..., n$, tem-se:

$V_k = \{V_{q(k)}, V_{q(k+1)}, ..., V_{q(n)}\}$;

$E_1 = E$;

para $k > 1$, $\{i, j\} \in E_k$ ou $\{q(k-1), i\}$,$\{q(k-1), j\} \in E_{k-1}$;

A ordem $q = [(1), (2), ..., (N)]$ será denominada de **ordem de eliminação** na qual $q(k) = i$ significa que o vértice i foi o *k-ésimo* vértice de G a ser eliminado. O **inverso da ordem de eliminação** será definido como, $\bar{q}(i) = k \Leftrightarrow q(k) = i$.

Ao eliminar o vértice $q(k)$, os arcos não direcionados adicionados para unir seus vizinhos são denominados de **arcos de preenchimento**. A inclusão de todos os arcos de preenchimento unindo os vizinhos de $q(k)$ transforma-os em um **clique**.

Uma ordem de eliminação, q, será **perfeita** se e somente se a eliminação de nenhum dos vértices na ordem q acarreta a incorporação ao grafo de um arco de preenchimento.

A partir do processo de eliminação é possível construir o **Grafo Preenchido** (*Filled Graph*) $P = (V, F)$, onde $F = \bigcup_{k=1}^{n} E_k$. Considerando que o grafo originalmente possuía E arcos, os arcos de preenchimento correspondem à diferença $F - E$.

A Figura 2.4 apresenta um grafo original com 6 vértices (a), e uma seqüência de grafos de eliminação (b) a (e). Em cada etapa está em destaque o nó que será eliminado e na etapa seguinte os arcos de preenchimento adicionados. A figura (f) mostra o grafo preenchido, novamente com os arcos adicionados em destaque. Neste exemplo a ordem de eliminação adotada corresponde ao vetor $q = [1, 3, 6, 2, 4, 5]$.

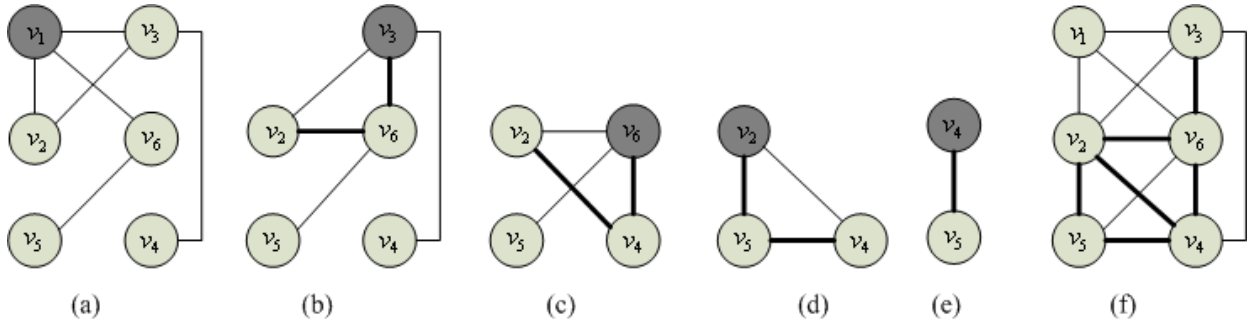


Figura 2.4: Processo de eliminação e grafo preenchido.

O algoritmo de **eliminação simplificada** é uma maneira computacionalmente mais eficiente para se obter o gráfico preenchido. Nesta versão simplificada a cada iteração o grafo de eliminação G_k^* é obtido quando ao eliminar o nó $q(k)$ são preenchidos apenas os arcos incidentes ao seu vizinho **mais** próximo de ser eliminado.

A Figura 2.5 mostra o mesmo grafo com 6 vértices (a), e uma seqüência de grafos de eliminação (b) a (e), gerada com o algoritmo de eliminação simplificada, mantida a ordem $q = [1, 3, 6, 2, 4, 5]$. Em cada etapa estão em destaque: o nó que será eliminado, o nó que será eliminado na etapa seguinte e os arcos de preenchimento adicionados. A figura (f) mostra o grafo preenchido idêntico ao anterior.

2.2 Conceitos de Probabilidade e Potencial

No presente trabalho a teoria clássica do cálculo de probabilidade será utilizada para caracterizar quantitativamente a incerteza. Nesta seção serão apresentados os principais conceitos e resultados desta teoria, os quais serão posteriormente utilizados para trabalhar com Redes Bayesianas.

Considere as seguintes definições:

1. um **espaço amostral** ou **espaço de possibilidades** [23] Ω é o conjunto de todos os resultados

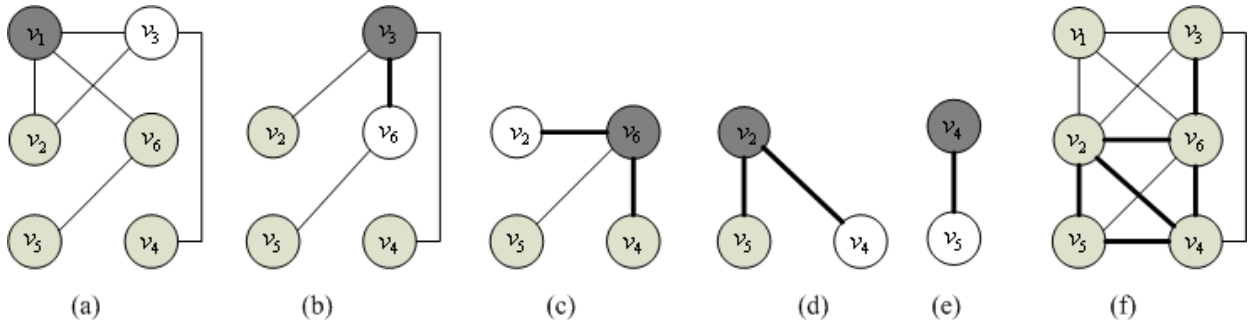


Figura 2.5: Processo de eliminação simplificada

possíveis de um experimento

2. os elementos $\omega \in \Omega$ são os **estados possíveis** ou **eventos elementares**. Os estados são mutuamente exclusivos e exaustivos: a cada experimento o resultado será necessariamente um único estado $\omega \in \Omega$.

3. um **evento aleatório** ou simplesmente um **evento** A é um subconjunto de Ω. Um evento A ocorre quando um estado $\omega \in A$ é observado.

Definido o espaço amostral, o modelo probabilístico de um experimento ou de um fenômeno que incorpore causalidade fica especificado ao se estabelecer uma **probabilidade** $P(\omega)$ para cada $\omega \in \Omega$, de forma que seja possível determinar a probabilidade de qualquer evento $P(A)$ para $A \subset \Omega$. Neste caso, tais probabilidades devem obedecer os seguintes axiomas:

Axioma 1 *Para qualquer evento* A, $P(A) \geq 0$

Axioma 2 *O espaço amostral* Ω *tem probabilidade um:* $P(\Omega) = 1$

Axioma 3 *Dados* $A_1, A_2, ..., A_n$ *disjuntos dois a dois* ($A_i \cap A_j = \emptyset$ *para* $i \neq j$) *então* $P(\cup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i)$

Um conceito fundamental na teoria de probabilidade e que deve estar muito bem compreendido para lidar adequadamente com Redes Bayesianas é o conceito de **probabilidade condicional**: "Se dado um evento B a probabilidade do evento A for igual a π, e indica-se por $P(A|B) = \pi$, se,

conhecido que o evento B ocorreu *e qualquer outro evento é irrelevante para o evento* A, então a probabilidade do evento A ser observado é π.

A regra fundamental do cálculo de probabilidades é

$$P(A|B).P(B) = P(A,B) \tag{2.1}$$

onde $P(A,B)$ é a probabilidade de A e B ocorrerem conjuntamente e será denominada de probabilidade conjunta dos eventos A e B.

De (2.1) segue-se ainda que $P(A|B).P(B) = P(B|A).P(A)$, e desta relação deriva-se a regra conhecida como **Regra de Bayes**:

$$P(B|A) = \frac{P(A|B).P(B)}{P(A)} \tag{2.2}$$

Se o evento B não tiver qualquer efeito sobre a probabilidade do evento A então $P(A|B) = P(A)$, A e B são ditos **eventos independentes** e, neste caso, da regra fundamental (2.1):

$$P(A,B) = P(A).P(B) \tag{2.3}$$

Estabelecido um espaço amostral Ω, uma **Variável Aleatória** $X : \Omega \rightarrow R$ é uma função definida em Ω que assume valores na reta dos número reais. As variáveis aleatórias podem portanto ser utilizadas para definir eventos de interesse em Ω associando-os a números reais. No presente trabalho, quando não causar ambiguidades, o termo *variável* será empregado como sinônimo de *variável aleatória*.

Partindo da hipótese de que as variáveis aleatórias podem assumir um número finito de valores possíveis, então $P(X)$ denota a distribuição de probabilidade da variável X entre os valores possíveis $(x_1, ..., x_n)$.

$$P(X) = (\pi_1, ..., \pi_n); \qquad \pi_i \geq 0; \qquad \sum_{n}^{i=1} \pi_i = 1 \tag{2.4}$$

onde π_i é a probabilidade de X assumir o valor x_i

Na notação adotada $P(X)$ indica a distribuição de probabilidades de X; $P(X = x_i)$, ou simplesmente $P(x_i)$, a probabilidade do evento $\{X = x_i\}$; $P(X|Y)$ a distribuição de probabilidades de X condicionada a valores de Y.

Para ilustrar os conceitos abordados considere duas variáveis X e Y e os seus respectivos conjuntos de possíveis valores $(x_1, ..., x_n)$ e $(y_1, ..., y_m)$. A distribuição conjunta de probabilidade entre as variáveis X e Y, $P(X, Y)$, pode ser representada em uma tabela n x m, na qual cada elemento é a probabilidade conjunta dos eventos $X = x_i$ e $Y = y_j$. A Tabela 2.1 apresenta os valores das probabilidades para os estados possíveis das variáveis X e Y.

	y_1	y_2	y_3	$\sum_Y$
x_1	0.16	0.12	0.12	0.40
x_2	0.24	0.28	0.08	0.60
$\sum_X$	0.40	0.40	0.20	1.00

Tabela 2.1: Exemplo de distribuição conjunta de probabilidade $P(X, Y)$

Assim como a distribuição conjunta, a distribuição de probabilidade condicional $P(X|Y)$ será representada por uma tabela n x m contendo as probabilidades $P(x_i|y_j)$. Seguindo com o exemplo anterior, a Tabela 2.2 apresenta um exemplo numérico de uma distribuição de probabilidade condicional para $n = 2$ e $m = 3$. Observe que a soma dos valores de cada coluna é 1.

	y_1	y_2	y_3
x_1	0.40	0.30	0.60
x_2	0.60	0.70	0.40
$\sum_X$	1.00	1.00	1.00

Tabela 2.2: Exemplo de distribuição de probabilidade condicional $P(X|Y)$

Note a diferença entre a Tabela 2.2 e a Tabela 2.1, enquanto na primeira, a soma de *todas* as probabilidades, ou de todos os valores da tabela, deve ser iqual a 1, na última, a soma de cada coluna deve ser igual a 1.

A partir da tabela de distribuição conjunta $P(X, Y)$ é possível calcular a distribuição $P(X)$, denominada **distribuição marginal** de X. Para compreender como este calculo é feito considere que $X = x_i$, nestas circunstâncias há exatamente m diferentes valores e portanto, m mutuamente exclusivos eventos: $(x_i, y_1), ..., (x_i, y_m)$. Generalizando o Axioma 3 e aplicando-o para variáveis:

$$P(x_i) = \sum_{j=1}^{m} P(x_i, y_j) \tag{2.5}$$

Este cálculo é chamado de **marginalização**. No caso particular do exemplo, dir-se-á que a *variável Y foi marginalizada para fora da distribuição conjunta* $P(X,Y)$, resultando portanto, a distribuição marginal $P(X)$. A notação utilizada para representar este cálculo será:

$$P(X) = \sum_{Y} P(X, Y) \tag{2.6}$$

No exemplo numérico em questão, a marginalização da variável Y para fora de $P(X,Y)$, representada pela Tabela 2.1, resulta em $P(X) = (0.40, 0.60)$.

Determinado o valor de $P(X)$ podemos obter a distribuição $P(Y|X)$ diretamente da Tabela 2.1 de distribuição conjunta, ou a partir da Regra de Bayes (2.2) e da distribuição marginal $P(Y) = (0.40, 0.40, 0.2)$. Os valores da distribuição $P(Y|X)$ estão apresentados Tabela 2.3.

	x_1	x_2
y_1	0.40	0.40
y_2	0.30	0.47
y_3	0.30	0.13
$\sum_Y$	1.00	1.00

Tabela 2.3: Distribuição de probabilidade condicional $P(Y|X)$

Para se manipular tabelas de probabilidade pode-se recorrer ao conceito de **potenciais** e às regras de álgebra para operá-los [12]. Define-se **potencial** ϕ como uma tabela de valores reais para um conjunto finito de variáveis. A este conjunto de variáveis denomina-se **domínio** e será utilizada a notação $dom(\phi)$ para indicar o domínio do potencial ϕ. Para que seja um potencial os valores da tabela não precisam ser medidas de probabilidades. Assim, a tabela de distribuição de probabilidades é um caso particular de potencial.

Os potenciais podem ser *marginalizados* ou *multiplicados* entre si. A multiplicação de potenciais tem as seguintes propriedades:

P. 1 $dom(\phi_1.\phi_2) = dom(\phi_1) \cup dom(\phi_2)$

P. 2 *Propriedade comutativa:* $\phi_1.\phi_2 = \phi_2.\phi_1$

P. 3 *Propriedade associativa:* $(\phi_1.\phi_2).\phi_3 = \phi_1.(\phi_1.\phi_3)$

P. 4 *Existência da Unidade: o número* 1 *é o potencial de um domínio vazio e* $1.\phi = \phi$

Um potencial pode ser marginalizado, e a marginalização de uma variável Y para fora de um potencial será representada por $\sum_Y \phi$. Por sua vez, $\sum_Y \phi$ será o potencial sobre o domínio $dom(\phi)\backslash\{Y\}$, onde $dom(\phi)\backslash\{Y\}$ representa o domínio de ϕ excluída a variável Y.

P. 5 *Propriedade comutativa da marginalização:* $\sum_X \sum_Y \phi = \sum_X \sum_Y \phi$

P. 6 *Potencial unitário: Para potenciais da forma* $P(X|V)$*, onde* V *é um conjunto de variáveis,* $\sum_X P(X|V) = 1$

P. 7 *Propriedade distributiva da marginalização do produto: se* $X \notin dom(\phi_1)$*, então* $\sum_X \phi_1.\phi_2 = \phi_1.\sum_X \phi_2$

Para ilustrar a manipulação algébrica e as propriedades dos potenciais descritas acima, considere os potenciais $\phi_1(A, B)$ e $\phi_2(C, B)$ definidos pelas Tabelas 2.4. Observe que cada umas das variáveis, A, B e C, pode assumir dois estados distintos, respectivamente: (a_1, a_2), (b_1, b_2) e (c_1, c_2). Nas tabelas estão apresentados os valores correspondentes às possíveis combinações de estados das variáveis.

$B\backslash A$	a_1	a_2
b_1	x_1	x_2
b_2	x_3	x_4

$B\backslash C$	c_1	c_2
b_1	y_1	y_2
b_2	y_3	y_4

Tabela 2.4: Potenciais $\phi_1(A, B)$ e $\phi_2(C, B)$

O produto entre os potenciais é obtido multiplicando os elementos das tabelas respeitando os estados das variáveis. A Tabela 2.5 apresenta o resultado do produto $\phi_1(A, B).\phi_2(C, B)$. Os dois números dentro de cada célula da tabela, que representa o produto dos potenciais, correspondem respectivamente aos estados c_1 e c_2.

$B\backslash A$	a_1	a_2
b_1	(x_1y_1, x_1y_2)	(x_2y_1, x_2y_2)
b_2	(x_3y_3, x_3y_4)	(x_4y_3, x_4y_4)

Tabela 2.5: Produto $\phi_1(A, B).\phi_2(C, B)$. Cada célula da tabela contém os estados possíveis de C.

Analisando o resultado do produto dos potenciais e considerando a propriedade comutativa da multiplicação, característica da operação com números reais, pode-se verificar facilmente a propriedade comutativa do produto de potenciais: $\phi_1(A,B).\phi_2(C,B) = \phi_2(C,B).\phi_1(A,B)$

Para marginalizar a variável C do produto de potenciais, ou seja, calcular $\sum_C \phi_1(A,B).\phi_2(C,B)$, conforme visto anteriormente, deve-se somar as componentes de cada célula da Tabela 2.5. O resultado esta apresentado na tabela a seguir.

$B \backslash A$	a_1	a_2
b_1	$x_1y_1 + x_1y_2$	$x_2y_1 + x_2y_2$
b_2	$x_3y_3 + x_3y_4$	$x_4y_3 + x_4y_4$

Tabela 2.6: Marginalização da variável C do produto de potenciais:$\sum_C \phi_1(A,B).\phi_2(C,B)$.

Para verificar a propriedade distributiva dos potenciais, inicialmente pode-se marginalizar a variável C do potencial $\phi_2(C,B)$. Processo representado pela Tabela 2.7.

B	
b_1	$y_1 + y_2$
b_2	$y_3 + y_4$

Tabela 2.7: Marginalização da variável C do potencial $\phi_2(C,B)$: $\sum_C \phi_2(C,B)$.

A Tabela 2.8 é então obtida, através da multiplicação dos potenciais $\phi_1(A,B)$ e $\sum_C \phi_2(C,B)$. Considerando a propriedade distributiva da soma e multiplicação de números reais constata-se que as tabelas 2.6 e 2.8 são equivalentes.

$B \backslash A$	a_1	a_2
b_1	$x_1(y_1 + y_2)$	$x_2(y_1 + y_2)$
b_2	$x_3(y_3 + y_4)$	$x_4(y_3 + y_4)$

Tabela 2.8: Produto dos potenciais $\phi_1(A,B).\sum_C \phi_2(C,B)$.

Como poderá ser constado adiante, o conceito e as regras de manipulação de potenciais serão fundamentais para compreender como tornar computacionalmente mais eficiente os cálculos necessários para inferência em uma Rede Bayesiana.

2.3 Redes Bayesianas

Para apresentar a teoria de Redes Bayesianas, principal objetivo da presente seção, convêm estabelecer algumas premissas e notações que serão adotadas:

- Cada variável tem um conjunto finito e mutuamente excludente de valores ou estados possíveis.
- Um conjunto de variáveis será indicado em negrito, por exemplo, $\mathbf{X} = \{X_1, X_2, ..., X_n\}$.
- A densidade de probabilidade de uma variável X será indica por $P(X)$, a probabilidade de um valor específico $\{X = x\}$ será indicada por $P(X = x)$ ou simplesmente $P(x)$.
- A densidade de probabilidade da variável X condicional a uma variável Y será indicada por $P(X|Y)$.
- Dados dois conjuntos de variáveis $\mathbf{X}$ e $\mathbf{Y}$ a notação $\mathbf{X}\backslash\mathbf{Y}$ indica o conjunto de variáveis que pertencem a $\mathbf{X}$ e não a $\mathbf{Y}$.
- A notação $\sum_{\mathbf{X}} f(\mathbf{X}, \mathbf{Y})$ indica que através do procedimento de marginalização todas as variáveis do conjunto $\mathbf{X}$ foram eliminadas da função $f(\mathbf{X}, \mathbf{Y})$.

2.3.1 Definição de Redes Bayesianas

Uma Rede Bayesiana [15] [14] é uma representação gráfica de uma distribuição conjunta de probabilidade para um conjunto de variáveis $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ [12]. Graficamente, a distribuição conjunta de probabilidade é representada através de uma Rede Bayesiana por um Grafo Direcional Acíclico, GDA, no qual cada vértice i representa uma variável $X_i \in \mathbf{X}$. No GDA que representa a Rede Bayesiana os arcos representam existência direta de dependência entre nós ligados e quantitativamente a importância desta influencia é expressa pela probabilidade condicional. Desta forma, um arco do nó i ao nó j significa que a distribuição de probabilidade da variável X_j esta diretamente condicionada a variável X_i. Em modelos estatísticos específicos o arco pode ser interpretado como uma influência direta ou um efeito causal [16] [14] de X_i sobre X_j.

Em uma Rede Bayesiana, cada variável aleatória X_i está associada a uma probabilidade condicional $P(X_i|pa(X_i))$ [15]. A Figura 2.6 mostra uma Rede Bayesiana simples, as distribuições de probabilidade para cada variável e algumas relações entre as mesmas que derivam da teoria de grafos direcionados apresentadas na seção 2.1.

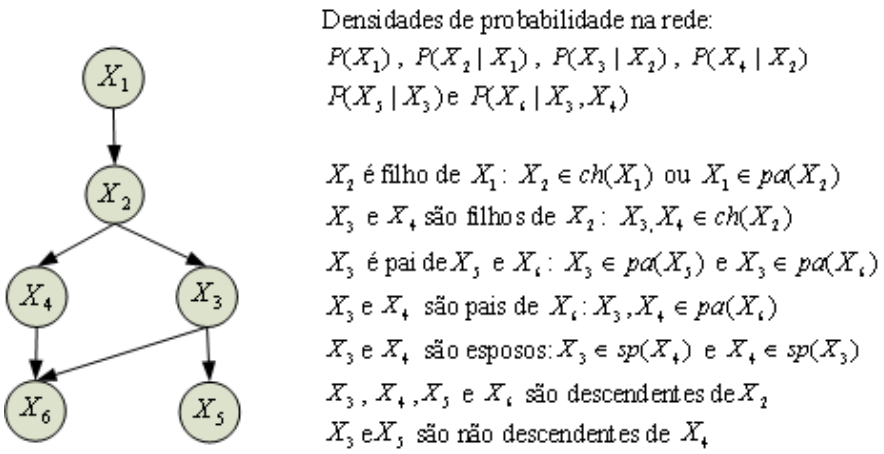


Figura 2.6: Exemplo de uma Rede Bayesiana e algumas das relações

2.3.2 Condição de Markov e o Critério de d-separação

A semântica de uma Rede Bayesiana implica em uma correspondência clara entre a topologia do GDA e as relações de independência expressa pelo mesmo. Esta semântica deriva da **Condição de Markov**: Cada variável é condicionalmente independente de seus não-descendentes não-pais dados seus pais. Desta condição decorre uma única distribuição conjunta de probabilidades [15]:

$$P(\mathbf{X}) = \prod_i p(X_i|pa(X_i)) \tag{2.7}$$

Desta correspondência entre a topologia do GDA e a independência probabilística entre as variáveis decorre o critério de **d-separação** [15]. Para facilitar a compreensão deste critério, convêm ilustrar três situações possíveis que podem estar presentes em uma Rede Bayesiana e que estão representadas na Figura 2.7.

Analisando inicialmente o caso (a) da Figura 2.7, pela Equação 2.7 a distribuição de probabilidade conjunta das variáveis X, Y e Z pode ser escrita como:

$$P(X, Y, Z) = P(X).P(Z|X).P(Y|Z)$$

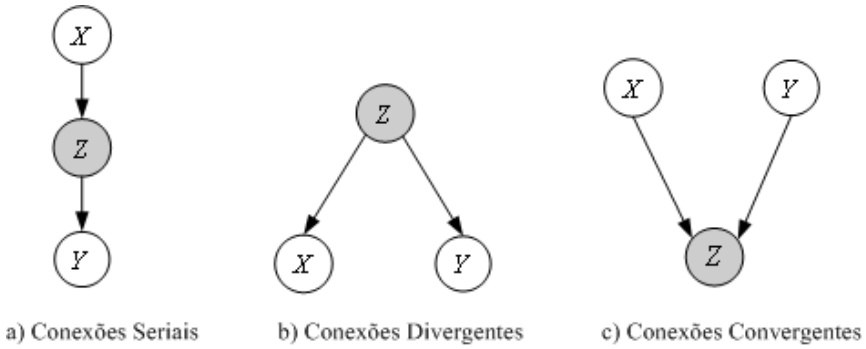


Figura 2.7: Tipos possíveis de conexão em uma Rede Bayesiana

Utilizando a relação anterior e considerando $P(X,Z) = P(X).P(Z|X)$ a expressão para $P(Y|X,Z)$ pode ser escrita como

$$P(Y|X,Z) = \frac{P(X,Y,Z)}{P(X,Z)}$$

$$P(Y|X,Z) = \frac{P(X).P(Z|X).P(Y|Z)}{P(X,Z)}$$

portanto,

$$P(Y|X,Z) = P(Y|Z) \tag{2.8}$$

A partir do resultado representado pela Equação 2.8 conclue-se que $P(Y|X,Z=z) = P(Y|Z=z)$, ou seja, Y é condicionalmente independente de X dado Z e indicaremos esta relação por $Y \perp X|Z$. Analogamente, pode-se demonstrar que no caso (b), conexões Divergentes, vale a mesma relação.

No caso de nós com conexões convergentes, caso (c), a distribuição conjunta é calculada pela expressão

$$P(X,Y,Z) = P(Z|X,Y).P(X).P(Y) \tag{2.9}$$

Portanto, à priori, se o valor da variável Z for desconhecido X e Y serão independentes. Contudo, se o valor de $Z = z$ for observado, então, as combinações de probabilidades de estados de X e Y devem ser tais que a probabilidade condicional $P(Z = z|X,Y)$ seja factível, e portanto, as variáveis

X e Y deixam de ser independentes. O exemplo clássico para compreender este último caso é a rede: (regador → grama molhada ← chuva) que pode ser encontrado em [16], [12].

Conclui-se que nas Conexões Seriais e Divergentes, Figura 2.7 (a) e (b) respectivamente, as variáveis X e Y serão condicionalmente independentes se o valor de Z estiver determinado. Por outro lado, no caso de Conexões Convergentes, Figura 2.7 (c), a determinação do valor de Z torna as variáveis X e Y dependentes entre si.

Formalmente a independência entre variáveis em uma Rede Bayesiana está diretamente ligada ao critério de d-separação. O **critério de d-separação** postula que se **X**, **Y** e **Z** são três conjuntos disjuntos de nós de um Grafo Direcional Acíclico, então, **Z** d-separa **X** de **Y** se ao longo de todo caminho entre **X** e **Y** há um nó W que satisfaz a uma das duas condições seguintes:

i. W tem conexões convergentes e nem W e nenhum de seus descendentes pertencem a **Z**;
ii. W não tem conexões convergentes e W pertence a **Z**.

Uma propriedade fundamental em de Redes Bayesianas é que se duas variáveis, representadas por nós de um GDA, estão graficamente d-separadas então estas variáveis são independentes [15]. Esta propriedade é importante pois permite identificar o conjunto mínimo de variáveis que são estatisticamente relevantes para se inferir sobre uma variável ou um conjunto de variáveis da rede. Esta possibilidade pode reduzir significativamente as dimensões do problema de inferência e portanto a necessidade de processamento numérico.

O conceito de *Markov Blanket* apresentado no contexto da teoria de grafos direcionais, pode ser estendido para variáveis em Rede Bayesiana. O *Markov Blanket* de uma variável X é o conjunto dos parentes imediatos de X, formado pela união de três conjuntos de variáveis: os pais X, pa(X), os filhos diretos de X, ch(X), e os pais dos filhos de X, sp(X). Utilizando o critério de d-separação pode-se demonstrar que o *Markov Blanket* é o conjunto mínimo de variáveis para tornar a variável X independente de todas as demais variáveis da rede [15].

2.3.3 Inferência em Redes Bayesianas

Uma Rede Bayesiana especifica completamente uma única distribuição conjunta de probabilidades envolvendo todas as variáveis que a compõe. Esta distribuição pode ser obtida pela Equação 2.7. Para descrever o procedimento de inferência será utilizado o conceito de potencial, assim, é conveniente reescrever a equação da distribuição conjunta utilizando a notação de potencial, Equação 2.10.

$$P(\mathbf{X}) = \prod_i \phi_i(p(X_i, pa(X_i))) \tag{2.10}$$

Exemplo 2.3.1

Considere o conjunto de variáveis $\mathbf{X} = \{X_1, X_2, X_3, X_4, X_5, X_6\}$ e a Rede Bayesiana da Figura 2.8.

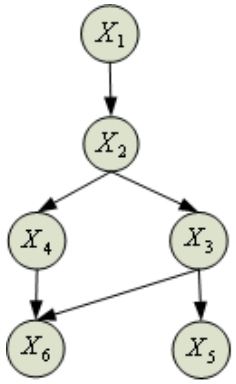


Figura 2.8: Exemplo de uma Rede Bayesiana

As probabilidades e os potenciais, com os respectivos domínios indicados entre parenteses: $P(X_1) = \phi_1(X_1)$, $P(X_2|X_1) = \phi_2(X_1, X_2)$, $P(X_3|X_2) = \phi_3(X_2, X_3)$, $P(X_4|X_2) = \phi_4(X_2, X_4)$, $P(X_5|X_3) = \phi_3(X_3, X_5)$, $P(X_6|X_3, X_4) = \phi_2(X_3, X_4, X_6)$

A distribuição conjunta das variáveis $\mathbf{X}$ será portanto:

$$P(\mathbf{X}) = P(X_1).P(X_2|X_1).P(X_3|X_2).P(X_3|X_2).P(X_4|X_2).P(X_5|X_3).P(X_6|X_3, X_4)$$

ou, utilizando a notação de potencial,

$$P(\mathbf{X}) = \phi_1(X_1).\phi_2(X_2, X_1).\phi_3(X_2, X_3).\phi_4(X_2, X_4).\phi_5(X_3, X_5).\phi_6(X_3, X_4, X_6)$$

§

Conhecida a distribuição conjunta de probabilidade torna-se factível responder a todas as possíveis demandas por inferência sobre uma variável ou um conjunto de variáveis da mesma. Para tanto, o procedimento básico é a marginalização de todas as variáveis da distribuição conjunta, exceto daquelas das quais se deseja obter a distribuição marginal. Tal operação segue os mesmos princípios do procedimento de marginalização descrito e exemplificado na seção 2.2.

Exemplo 2.3.2

Continuando o Exemplo 2.3.1, para calcular o valor de $P(X_4)$ deve-se *eliminar por marginalização* da distribuição conjunta as demais variáveis.

$$P(X_4) = \sum_{X_1,X_2,X_3,X_5,X_6} \phi_1(X_1).\phi_2(X_2,X_1).\phi_3(X_3,X_2).\phi_4(X_2,X_4).\phi_5(X_3,X_5).\phi_6(X_3,X_4,X_6)$$

Para se evitar de calcular $P(\mathbf{X}_4)$ utilizando todos os potenciais simultaneamente, pode-se aplicar a propriedade distributiva dos potenciais e posteriormente executar sucessivas operações de produto e a marginalização envolvendo um número menor de potenciais em cada operação isoladamente.

$$P(X_4) = \sum_{X_1} \phi_1(X_1). \sum_{X_2} \phi_2(X_2).\phi_4(X_2,X_4). \sum_{X_3} \phi_3(X_2,X_3). \sum_{X_5} \phi_5(X_3,X_5). \sum_{X_6} \phi_6(X_3,X_4,X_6)$$

Desta forma, inicialmente calcula-se $\phi_6^{'} = \sum_{X_6} \phi_6(X_3,X_4,X_6)$. Multiplica-se $\phi_6^{'}$ por ϕ_5 para calcular $\phi_5^{'} = \sum_{X_5} \phi_5(X_3,X_5).\phi_6^{'}$. O resultado $\phi_5^{'}$ é então multiplicado por ϕ_3 e assim por diante. Observe que no cálculo de $\phi_5^{'}$ é possível aplicar novamente a propriedade distributiva, ou seja, não é necessário multiplicar $\phi_5(X_3,X_5)$ por $\phi_6^{'}$ antes de marginalizar X_5. Esta independência entre as operações implica que ambas poderiam ser executadas simultaneamente. O conjunto de operações para calcular $P(\mathbf{X}_4)$ está esquematizado na figura Figura 2.9. A forma de representação das operações na figura indica quais operações poderiam ser executadas em paralelo. §

A cada operação de marginalização elimina-se uma variável da distribuição de probabilidades, por este motivo este processo também é chamado de **eliminação de variáveis**. Uma **ordem de eliminação** predefinida especifica a ordem na qual as mesmas devem ser marginalizadas. A complexidade computacional do cálculo da probabilidade marginal é determinada pela escolha da ordem para eliminação das variáveis da distribuição conjunta. A escolha de uma boa ordem de eliminação está diretamente ligada à topologia do grafo direcionado que define a rede.

Para compreender esta relação convêm definir o conceito de **Grafo de Domínio** (*Domain Graph*). Seja $\Phi = \{\phi_1, ..., \phi_n\}$ um conjunto de potenciais com domínios $dom(\phi_i)$ envolvendo um conjunto de variáveis $\mathbf{X} = \{X_1, ..., X_n\}$. O Grafo de Domínio para Φ é um grafo não direcionado no qual cada vértice está associado a uma variável de $\mathbf{X}$ e os arcos unem vértices que estejam associados a variáveis que pertençam a um mesmo domínio $dom(\phi_i)$. A Figura 2.10 apresenta uma Rede Bayesiana e o seu

respectivo Grafo de Domínio.

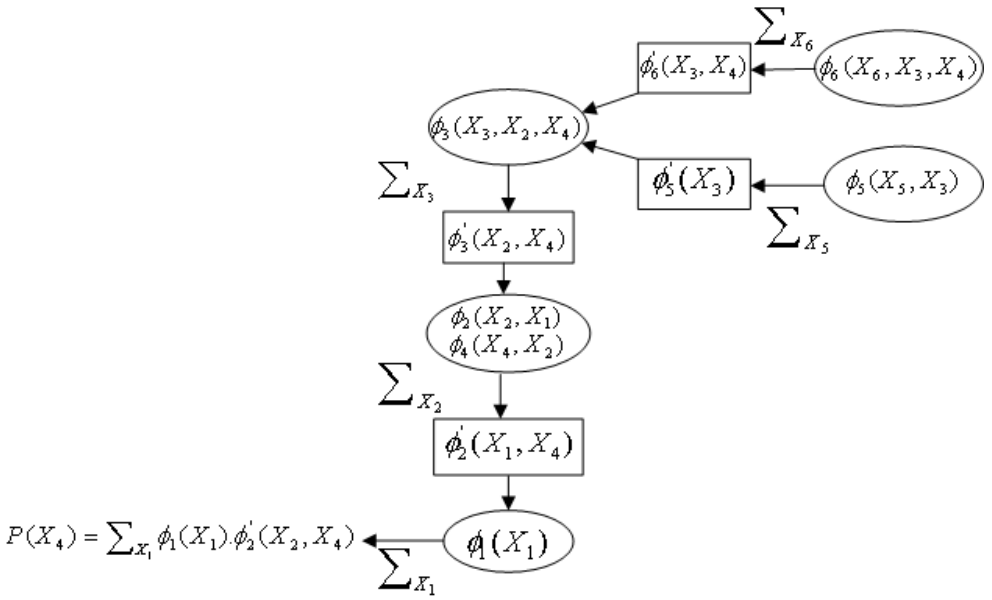


Figura 2.9: Operações para marginalização de X_4

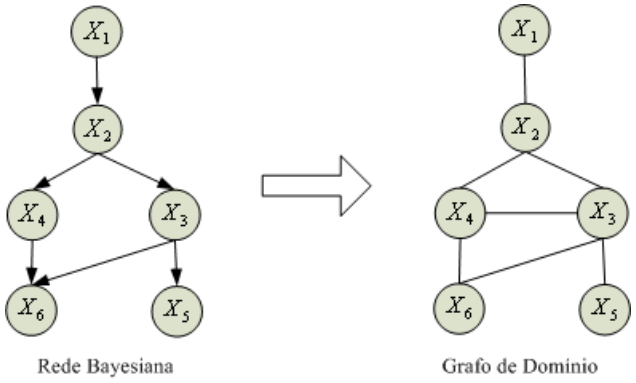


Figura 2.10: Rede Bayesiana e Grafo de Domínio

Por construção o Grafo de Domínio de uma Rede Bayesiana será exatamente o Grafo Moral do grafo direcional acíclico que define a topologia da mesma. Pode-se demonstrar que uma ordem de eliminação das variáveis no processo de marginalização das probabilidades é computacionalmente mais eficiente se a mesma corresponder a uma ordem eficiente de eliminação dos nós do Grafo

Moral [12]. Como foi visto anteriormente, uma ordem de eliminação é eficiente se ao longo da mesma o menor número possível de arcos de preenchimento for introduzidos.

Exemplo 2.3.3

Graficamente a operação de eliminação de variáveis exemplificada no Exemplo 2.3.2 corresponde ao processo de eliminação dos vértices do Grafo Moral. A cada variável que é eliminada por marginalização da distribuição, o respectivo nó é retirado do Grafo Moral. A Figura 2.11 apresenta a sequência de eliminação correspondente ao cálculo da distribuição $P(X_4)$, representada graficamente pela Figura 2.9. Convêm observar a cada etapa a relação entre o grau do vértice e a dimensão do domínio correspondente a variável por ele representada. O grau do vértice será igual ao número de variáveis do domínio a menos da própria variável representada pelo nó e quando for o caso, da variável, ou variáveis, sobre a quais se deseja fazer a inferência. O arco em linha descontínua é um arco de preenchimento incluído ao longo do processo de eliminação dos vértices.

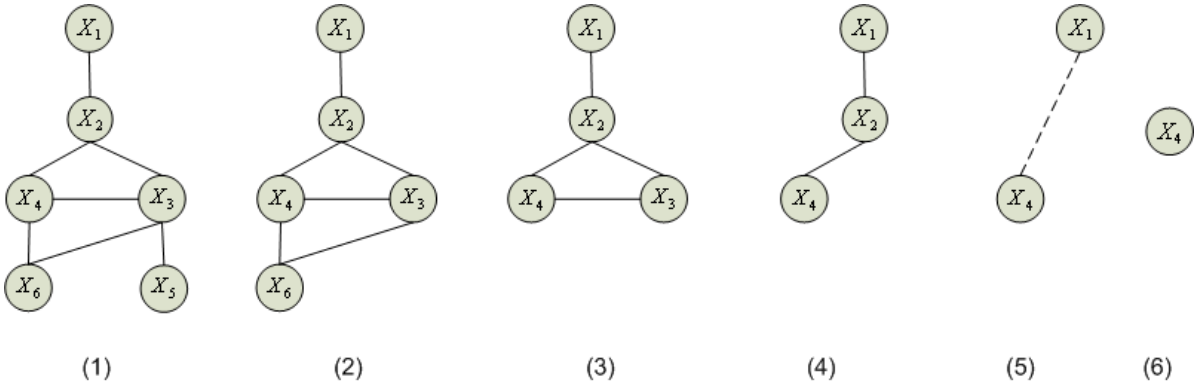


Figura 2.11: Eliminação dos vértices do Grafo Moral.

§

No exemplo anterior, através de operações algébricas com os potenciais obteve-se a distribuição marginal desejada. A sistematização deste procedimento é a essência dos algoritmos para inferência em Redes Bayesianas focado em operações algébricas, baseados nos quais neste trabalho foi implementado um conjunto de bibliotecas computacionais para inferência em Redes Bayesianas.

2.3.4 Evidência

A eficiência no processo de marginalização é importante pois as Redes Bayesianas são particularmente úteis para atualizar ou calcular novas distribuições de probabilidades para uma variável ou

um conjunto de variáveis à medida que novas informações são obtidas. O aumento da quantidade de informação ocorre com a observação dos valores de um subconjunto de variáveis que compõe a rede. Dada uma Rede Bayesiana a **Evidência**, E, corresponde ao conjunto de informações em relação aos estados observados para este subconjunto de variáveis e $\mathbf{X}_E$ indica o conjunto das variáveis das quais os valores ou os estados são conhecidos.

Semanticamente, observar o valor de uma determinada variável equivale a afirmar que o valor observado terá probabilidade 1 e os demais valores serão impossíveis, ou seja, terão probabilidades de ocorrer iguais a zero.

Para incorporar ao modelo variáveis das quais se dispõe de informações adicionais, seja porque o estado da variável foi observado, seja porque a partir da observação da realidade alguns estados que antes eram possíveis tornaram-se inviáveis, utiliza-se um estrutura de dados denominada *finding* [12]. Seja A uma variável com, a priori, n estados possíveis e com distribuição de probabilidade $P(A) = (x_1, ..., x_n)$, onde x_i corresponde à probabilidade de A estar no estado i. Assuma que foi obtida uma informação $\mathbf{e}$ de que A pode estar apenas em 2 estados quaisquer, i e j, dos estados possíveis a priori. Esta afirmação implica que todos os estados de A exceto os estados i e j são impossíveis e pode ser expressa pelo potencial $P(A, \mathbf{e}) = (0, ..., 0, x_i, 0, ..., 0, x_j, 0, ...)$. Observe que $P(A, \mathbf{e})$ foi obtido multiplicando-se $P(A)$ por um vetor de dimensão n no qual a *i-ésima* e a *j-ésima* posições tem valores *1* e todas as demais posições tem valores *0*. A este vetor de *0*s e *1*s denomina-se *finding* e para distingui-lo da própria informação $\mathbf{e}$ utiliza-se a notação $\underline{e}$. Para efeito de cálculo o *finding* pode ser considerado como um potencial com uma única variável no domínio e cujos valores são *0*s e *1*s.

Dada uma evidência E a inferência em uma Rede Bayesiana envolve o cálculo de uma distribuição *marginal a posteriori* para uma variável ou um conjunto de variáveis inquiridas ou questionadas (*query variables*) $\mathbf{X}_q$ [8]. Para simplificar a notação $\mathbf{X}_q$ representará tanto o conjunto de variáveis questionadas quanto o evento no qual as variáveis questionadas assumem um valor específico.

A probabilidade marginal a posteriori de $\mathbf{X}_q$ dada a evidência E será calculada por

$$P(\mathbf{X}_q|E) = \frac{P(\mathbf{X}_q, E)}{P(E)} = \frac{\sum_{\mathbf{X}\setminus\{\mathbf{X}_q,\mathbf{X}_E\}} P(\mathbf{X})}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} P(\mathbf{X})} \tag{2.11}$$

Do ponto de vista computacional, o termo relevante na Equação 2.11 é o numerador, uma vez que, considerando fixa a evidência E, o denominador $P(E)$ da Equação 2.11 é simplesmente uma

constante de normalização e pode ser facilmente obtido uma vez calculado o numerador [5]. Portanto o probabilidade marginal a posteriori será proporcional ao valor do numerador.

$$P(\mathbf{X}_q|E) \propto \sum_{\mathbf{X}\setminus\{\mathbf{X}_q,\mathbf{X}_E\}} P(\mathbf{X}) \tag{2.12}$$

O numerador isoladamente não é uma medida de probabilidade, pois não obedece aos axiomas básicos apresentados no início da Seção 2.2, contudo, para efeito de cálculo o mesmo pode ser tratado como um potencial ϕ cujo domínio será: $dom(\phi) = \mathbf{X}_q \cup \mathbf{X}_E$.

Para calcular $P(\mathbf{X}_q|E)$ usualmente se utiliza apenas um subconjunto de $\mathbf{X}$ composto pelas variáveis estatisticamente relevantes para a consulta. Ao conjunto destas variáveis X_i estatisticamente relevantes, ou seja, cujas distribuições $P(X_i|pa(X_i))$ forem necessárias para obter $P(\mathbf{X}_q|E)$, denomina-se conjunto das **variáveis requisitadas** [19] e será indicado por $\mathbf{X}_R$. Variáveis pertencentes a $\mathbf{X}_q$ necessariamente pertencem a $\mathbf{X}_R$ mas nem todas as variáveis observadas $\mathbf{X}_E$ pertencem a $\mathbf{X}_R$.

A determinação da relevância estatística decorre do critério de d-separação, o qual, conforme descrito na Seção 2.3.2, está baseado na topologia do grafo. Este último fato torna a determinação das relações de independência computacionalmente eficiente por não envolver explicitamente o processamento numérico dos potenciais. Existem algoritmos polinomiais para determinar o conjunto das variáveis requisitadas [19] [6].

Exemplo 2.3.4

No Exemplo 2.3.3 assuma conhecida a evidência $X_5 = x_5$. Esta evidência forma o conjunto de informações $\mathbf{e}$, o qual neste caso é composto apenas pelo respectivo *finding* $\underline{e}_5$. Para efeito de cálculo $\underline{e}_5$ pode ser considerado como um potencial e desta forma ele pode ser incorporado ao cálculo da distribuição marginal.

A distribuição $P(X_4, \mathbf{e})$ seria calculada por

$$P(X_4, \mathbf{e}) = \sum_{X_1,X_2,X_3,X_5,X_6} \phi_1(X_1).\phi_2(X_2, X_1).\phi_3(X_3, X_2).\phi_4(X_2, X_4).\phi_5(X_3, X_5).\phi_6(X_3, X_4, X_6).\underline{e}_5$$

Aplicando a propriedade distributiva dos potenciais

$$P(X_4, \mathbf{e}) = \sum_{X_1} \phi_1(X_1). \sum_{X_2} \phi_2(X_2).\phi_4(X_2, X_4). \sum_{X_3} \phi_3(X_2, X_3). \sum_{X_5} \phi_5(X_3, X_5).\underline{e}_5. \sum_{X_6} \phi_6(X_3, X_4, X_6)$$

Portanto, a probabilidade marginal a posteriori de X_4 dada a evidência será calculada por

$$P(X_4|\mathbf{e}) = \frac{P(X_4, \mathbf{e})}{\sum_{X_4} P(X_4, \mathbf{e})}$$

A Figura 2.12 representa a sequência de operações incorporando a evidência

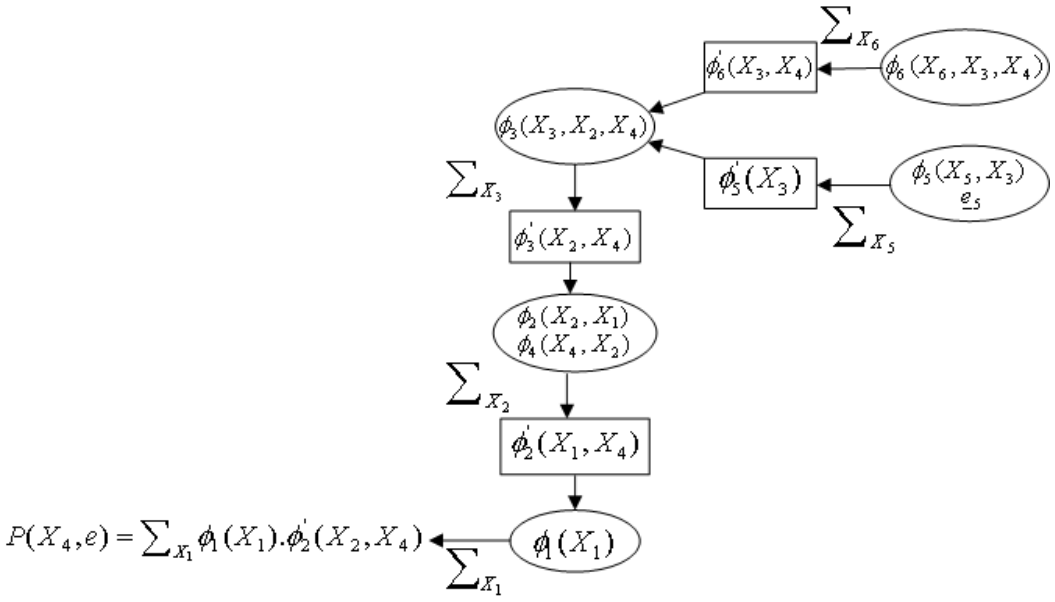


Figura 2.12: Operações para marginalização de X_4 incorporando a evidência observada para X_5.

§

2.3.5 Determinação das Variáveis Requisitadas (*Requisite Variables*)

Para executar de forma eficiente a inferências em uma Rede Bayesiana é fundamental a identificação das informação requisitadas e dos vértices não relevantes para a operação. Para especificar completamente uma Rede Bayesiana é necessário obter para todas as variáveis que a compõe os

possíveis estados e as respectivas distribuições de probabilidades. No entanto, para executar uma particular consulta, ou seja, inferir sobre uma variável ou um conjunto de variáveis, apenas parte da informação codificada na Rede Bayesiana poderá ser suficiente. Em função da importância deste procedimento esta seção apresenta o algoritmo utilizado para identificar as relações de relevância probabilística entre as variáveis.

O fato de uma parte relevante do conteúdo do modelo estar codificada em uma estrutura de rede permite que importantes propriedades do mesmo possam ser identificadas de forma eficiente, pois fazê-lo não envolve o processamento numérico ou mesmo o conhecimento dos estados das variáveis e suas distribuições de probabilidades [15]. No presente trabalho utilizou-se o algoritmo *Bayes-Ball*, descrito em detalhes em [19], para identificar as relações de independência e as informações requisitadas uma vez definidas a consulta que se deseja realizar e a evidência disponível. O Bayes-Ball é um algoritmo simples e eficiente que para Redes Bayesianas é executado em tempo linear em relação ao tamanho do grafo que define a topologia da rede. Este algoritmo é aplicável para modelos que incorporam nós probabilísticos e determinístico, mas devido ao escopo do trabalho ser Redes Bayesianas a descrição que segue aborda apenas nós probabilísticos.

O algoritmo Bayes-Ball está fortemente baseado no conceito de d-separação [15] para determinar a irrelevância probabilística entre conjunto de variáveis, conforme descrito na Seção 2.3.2. Considere $\mathbf{X}$, $\mathbf{Y}$ e $\mathbf{Z}$ três subconjuntos disjuntos de variáveis, representadas em uma Rede Bayesiana pelos nós de um Grafo Direcional Acíclico. Se $\mathbf{Z}$ d-separa $\mathbf{X}$ de $\mathbf{Y}$ então $\mathbf{X}$ é condicionalmente independente de $\mathbf{Y}$ dado $\mathbf{Z}$ e denota-se por $\mathbf{Y} \perp \mathbf{X}|\mathbf{Z}$. Dada a relação de independência, pode-se afirmar que $\mathbf{Y}$ é *probabilisticamente irrelevante* para $\mathbf{X}$ dado $\mathbf{Z}$.

Retomando a notação utilizada na Seção 2.4.1, dada uma Rede Bayesiana composta por um conjunto de variáveis $\mathbf{X} = \{X_1, ..., X_n\}$, observada a Evidência E deseja-se inferir sobre um conjunto de variáveis questionadas $\mathbf{X}_q$. Indicar-se-a por $\mathbf{X}_R$ ao conjunto das variáveis requisitadas X_i cujas distribuições $P(X_i|pa(X_i))$ estão envolvidas na determinação de $P(\mathbf{X}_q|E)$. Variáveis pertencentes a $\mathbf{X}_q$ necessariamente pertencem a $\mathbf{X}_R$ mas nem todas as variáveis observadas pertencem a $\mathbf{X}_R$.

O princípio básico do algoritmo segue uma analogia mecânica do deslocamento de uma bola pelos nós e arcos do grafo: Dois conjuntos de nós $\mathbf{X}$ e $\mathbf{Y}$ estão d-separados, e portanto são condicionalmente independentes, dado um conjunto $\mathbf{Z}$ se e somente se não houver nenhum caminho para que uma bola se desloque de $\mathbf{X}$ até $\mathbf{Y}$ no grafo seguindo regras de movimentação previamente estipuladas que irão condicionar o deslocamentos da bola.

A versão mais simples do algoritmo Bayes-Ball para $\mathbf{X}_q$ dado E consiste em lançar bolas para visitar os nós do grafo da Rede Bayesiana a partir dos nós de $\mathbf{X}_q$. As regras que determinam os movimentos das bolas dependem do tipo de nó e da direção do arco do qual a bola chega para visitar um determinado nó (do pai para o filho ou do filho para o pai). De acordo com estes condicionantes a bola pode:

- *Passar através* do nó;
- *Rebater e voltar*;
- *Ser bloquada* e interromper a movimentação.

A Figura 2.13 resume graficamente as regras de movimentação das bolas lançadas para visitar os nós do grafo. Na figura, os nós observados e portanto pertencentes a E estão representados por círculos preenchidos. Os arcos direcionados do grafo que compõe a Rede Bayesiana estão representados com setas contínuas. As setas tracejadas indicam a direção de deslocamento da bola ao visitar o nó.

Observe que na Figura 2.13:

- Um nó probabilístico não observado permite que a bola passe através de si, casos (a) e (b), e rebate de volta as bolas vindas de nós que sejam seus filhos, caso (b).
- Um nó probabilístico observado rebate de volta as bolas provenientes de seus pais, caso (c), mas bloqueia as bolas vindas de seus filhos, caso (d).

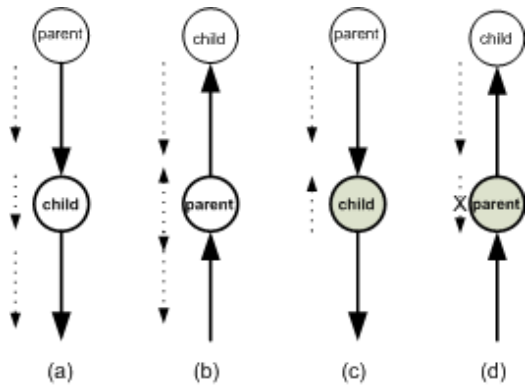


Figura 2.13: Movimentação das bolas no grafo em função do tipo de nó e da direção do movimento.

Partindo-se do conjunto de regras de movimentação descrito acima fez-se a implementação do algoritmo incorporando duas modificações sugeridos em [19] e importantes para garantir a performance

do mesmo. A primeira foi manter uma lista de nós que deveriam ser visitados a partir dos pais e a partir dos filhos. A segunda foi adotar um procedimento de "marcação" dos nós para garantir que um mesmo arco fosse percorrido em uma mesma direção uma única vez. O procedimento de marcar os nós além de tornar o algoritmo mais eficiente garante que não irão ocorrer loops infinitos.

A marcação dos nós segue a seguinte lógica:

- A parte superior do nó deve ser marcada quando o mesmo enviar a bola para seus pais;
- A parte inferior do nó deve ser marcada quando o mesmo enviar a bola para seus filhos;
- Um nó observado deve ser assinalado (*checked*) quando for visitado.

Resumindo e consolidando as regras de movimentação e marcação dos nós:

1. Um nó probabilístico *não* observado:
   (a) Visitado a partir de um de seus filhos: enviar a bola para todos seus pais e marcar a parte superior, enviar a bola para todos seus filhos e marcar a parte inferior.
   (b) Visitado a partir de um de seus pais: enviar a bola para todos seus filhos e marcar a parte inferior.
2. Um nó probabilístico observado:
   (a) Visitado a partir de um de seus filhos: assinalar o nó e interromper a movimentação da bola.
   (b) Visitado a partir de um de seus pais: assinalar o nó, enviar a bola para todos seus pais e marcar a parte superior.

O exemplo que se segue, extraído de [19], ilustra a dinâmica do algoritmo Bayes-Ball. A Figura 2.14 apresenta uma a Rede Bayesiana na qual $\mathbf{X}_q = \{6\}$ e os elementos do conjunto de nós observados $E = \{2, 5\}$ estão indicados por círculos preenchidos.

- Inicialmente o nó 6 é visitado, como se a bola tivesse sido lançada de um nó virtual, filho do nó 6, representado com linha tracejada.
- O nó 6 envia a bola para seus pais e seus filhos e são marcadas tanto a parte superior quanto inferior do nó.
- O nó 5 não envia a bola para lugar algum, por ser um nó observado que recebeu um bola de um filho. O nó 5 é assinalado.

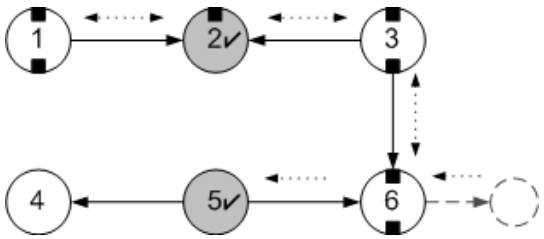


Figura 2.14: Exemplo da dinâmica do Bayes Ball.

- O nó 3 é visitado a partir de seu filho nó 6, e portanto repassa a bola para seus pais e seus filhos e marca-se as partes superior e inferior do nó.
- O nó 6 recebe a bola do nó 3 e a enviaria para seus filhos, mas não o faz pois a parte inferior do nós 6 esta marcada indicando que este procedimento foi executado anteriormente.
- O nó 2 recebe a bola de seu pai, o nó 3, e a repassa para seus pais, sendo portanto assinalado, por ser um nó observado e marcado na parte superior por enviar a bola para seus pais.
- O nó 1 recebe a bola de seu filho, o nó 2, e a envia para seus pais e filhos. Portanto o nó 1 é marcado tanto na parte superior quanto na parte inferior.
- O nó 2 e nó 3 recebem a bola proveniente do nó 1 mas as marcações em ambos os nós indicam que não há envios ou repasses adicionais a serem executados.

Algoritmo Bayes Ball

Considere uma Rede Bayesiana composta por um conjunto de variáveis $\mathbf{X} = \{X_1, ..., X_n\}$, da qual se deseja calcular $P(\mathbf{X}_q|E)$. O algoritmo descrito explora o grafo da Rede Bayesiana e constrói $\mathbf{X}_R$.

1. Inicialize todos os nós do grafo como não visitados e não marcados na parte superior e inferior.

2. Crie um programa de nós a serem visitados, inicializando com cada nó de $\mathbf{X}_q$ como se o mesmo fosse visitado a partir de um nó filho virtual.

3. Enquanto tiver nós para serem visitados:

   3.1. Pegue um nó i programado para ser visitado e retire-o do programa. O nó i pode estar programado para ser visitado por um pai, ou para ser visitado por um filho ou por ambos.

   3.2. Marcar i como visitado.

   3.3. Se $i \notin E$ e a visita é de um filho:

Se a parte superior de i não estiver marcada, marque-a e programe cada um de seus pais para serem visitados;

Se a parte inferior de i não estiver marcada, marque-a e programe cada um de seus filhos para serem visitados.

3.4. Se a visita a i é de um pai:

Se $i \in E$ e a parte superior de i não estiver marcada, marque-a e programe cada um de seus pais para serem visitados;

Se $i \notin E$ e a parte inferior de i não estiver marcada, marque-a e programe cada um de seus filhos para serem visitados.

4. Os nós irrelevantes são os nós **não** marcados na parte inferior.

5. Os nós probabilísticos requisitados são os nós marcados na parte superior

6. Os nós observados requisitados são os nós marcados pertencentes a E marcados como visitados

7. O conjunto $\mathbf{X}_R$ será a união dos conjuntos de nós probabilísticos e observados requisitados.

§

2.4 Algoritmos de Inferência em Redes Bayesianas

Para inferência em Redes Bayesianas o presente trabalho implementou um algoritmo que paraleliza as operações algébricas envolvidas no cálculo da probabilidade marginal a posteriori para um conjunto de variáveis questionadas dado um conjunto de variáveis observadas. O algoritmo adotado, descrito na Seção 5.5, está baseado e segue de maneira muito próxima dois algoritmos essencialmente iguais [5], ambos focados em operações algébricas: o algoritmo de Eliminação de Variáveis [5] e o algoritmo de Eliminação de *Buckets* [8]. Estes esquemas algébricos para inferência em Redes Bayesianas são relativamente simples de serem entendidos e implementados e permitiram a aplicação de técnicas, heurísticas e estruturas combinatórias abstratas características da fatoração de Cholesky em matrizes esparsas para otimizar e paralelizar o processamento numérico.

Algoritmos alternativos para inferência em Redes Bayesianas exploram as propriedades dos grafos que definem a mesmas e na sua maioria lidam com operações que envolvem *junction trees* [11]. Os algoritmos podem ainda ser classificados em duas classes distintas: algoritmos de inferência

exata e algoritmos de inferência aproximada. Estes últimos são aplicados quando, em função das características da rede, o cálculo exato torna-se excessivamente lento.

Dada a relevância para o presente trabalho e para viabilizar o entendimento do algoritmo desenvolvido a seção subsequente descreve o **Algoritmo de Eliminação de Variáveis** para inferência em Redes Bayesianas.

2.4.1 Método de Eliminação de Variáveis

A inferência em uma Rede Bayesiana composta por um conjunto de variáveis $\mathbf{X} = \{X_1, ..., X_n\}$ envolve o cálculo da distribuição marginal a posteriori $P(\mathbf{X}_q|E)$ para um conjunto de variáveis questionadas $\mathbf{X}_q$ dada uma Evidência E, e o respectivo conjunto de variáveis observadas $\mathbf{X}_E$, utilizando um conjunto de variáveis requisitadas $\mathbf{X}_R$.

Para calcular $P(\mathbf{X}_q|E)$ devem ser executadas operações de marginalização e multiplicação com os potenciais de $\mathbf{X}_R$ para se obter a distribuição $P(\mathbf{X}_q, E)$, a qual, uma vez normalizada, será a distribuição procurada. A equação 2.13 indica como calcular a distribuição conjunta $P(\mathbf{X}_q, E)$.

$$P(\mathbf{X}_q, E) = \sum_{\mathbf{X}_R \setminus \{\mathbf{X}_q, \mathbf{X}_E\}} \Big(\prod_{X_i \in \mathbf{X}_R} P(X_i|pa(X_i)) \Big) \tag{2.13}$$

Seja N o número de elementos de $\mathbf{X}_R$ que não foram observadas e não são variáveis questionadas, ou seja, não pertencem a $\mathbf{X}_E$ nem a $\mathbf{X}_q$. Considere que as estas variáveis estão dispostas em uma determinada ordem cujo vetor de índices é dado por $q = [(1), (2), ..., (N)]$. Nestas condições, as equação anterior pode torna-se:

$$P(\mathbf{X}_q, E) = \sum_{\mathbf{X}_{(N)}} ... \sum_{\mathbf{X}_{(1)}} P(X_{(N)}|pa(X_{(N)}))...P(X_{(1)}|pa(X_{(1)})) \tag{2.14}$$

Partindo do princípio de que a normalização será executada apenas no final do processo de eliminação, ao invés de trabalhar com as probabilidades, utilizar-se-á o correspondente conjunto de potenciais de valores reais $\Phi = \{\phi_1, ..., \phi_n\}$ para o conjunto de variáveis $\mathbf{X} = \{X_1, ..., X_n\}$. Utilizando potenciais a Equação 2.14 pode ser reescrita como:

$$\phi(\mathbf{X}_q, E) = \sum_{\mathbf{X}_{(N)}} ... \sum_{\mathbf{X}_{(1)}} \phi_{(N)} ... \phi_{(1)} \tag{2.15}$$

Convêm recordar que dada uma variável aleatória X_i, cujos valores possíveis são $(x_{i,1}, ..., x_{i,k})$, o processo de eliminação da variável do potencial $\phi(X_1, ..., X_i, ..., X_n)$ por marginalização é executar a soma:
$\sum_{X_i} \phi(X_1, ..., X_i, ..., X_n) = \phi(X_1, ..., x_{i,1}, ..., X_n) + \phi(X_1, ..., x_{i,2}, ..., X_n) + ... + \phi(X_1, ..., x_{i,k}, ..., X_n)$.
Além disto, a notação $\sum_{\mathbf{V}}$ onde $\mathbf{V}$ é um conjunto de variáveis, indica a eliminação por marginalização de um determinado potencial de todas as variáveis de $\mathbf{V}$. Devido à propriedade comutativa dos potenciais, a notação não acarreta ambiguidade.

Note que para uma determinada variável X_i:

- A variável X_i aparece uma única vez como variável condicional para cada um de seus filhos, e portanto, está presente nos respectivos potenciais. Por exemplo: X_j é filho de X_i então X_i deve necessariamente aparecer como variável condicionante na densidade $P(X_j|X_i, ...)$ e no potencial $\phi_j(X_j, X_i, ...)$.
- A variável X_i aparece uma única vez como variável não-condicional na densidade $P(X_i|pa(X_i))$, e portanto, faz parte do potencial correspondente, $\phi_i(X_i, pa(X_i))$

Indicar-se-á por $\Phi_{X_{(i)}}$ o conjunto de todos os potenciais com $X_{(i)}$ no seu domínio e por $\prod_k^{j=1} \varphi_j$, ou simplesmente $\prod \varphi_j$, como o produto de k potenciais: $\varphi_1, ..., \varphi_k$. Considerando a propriedade distributiva dos potenciais a operação indicada pela Equação 2.15 pode ser executada de uma forma computacionalmente mais eficiente.

$$\phi(\mathbf{X}_q, E) = \sum_{\mathbf{X}_{(N)}} ... \sum_{\mathbf{X}_{(2)}} \phi_{(N)} ... \phi_{(2)} \cdot \sum_{\mathbf{X}_{(1)}} \prod \Phi_{X_{(1)}}$$

A sistematização do procedimento de eliminação de variáveis por marginalização dos potenciais das variáveis requisitadas não pertencentes a $\mathbf{X}_q$ nem a $\mathbf{X}_E$ define o Algoritmo de Eliminação de Variáveis descrito abaixo.

Algoritmo de Eliminação de Variáveis para Inferência em Redes Bayesianas

Seja $\Phi = \{\phi_1, ..., \phi_n\}$ os potenciais de um conjunto de variáveis $\mathbf{X} = \{X_1, ..., X_n\}$ de uma Rede Bayesiana

1. Gerar uma ordem de eliminação para as N variáveis requisitadas não-observadas e não-questionadas.

2. Para cada variável i de 1 até N executar os seguintes passos:

   (a) Remover de Φ todos os potenciais que tenham $X_{(i)}$ em seu domínio e com eles compor um conjunto indicado por $\Phi_{X_{(i)}}$ que será denominado *bucket* B_i da variável $X_{(i)}$.

   (b) Calcular o *separador* S_i de B_i como o potencial $\phi^{-X_{(i)}} = \sum_{\mathbf{X}_{(i)}} \prod \Phi_{X_{(i)}}$. O produto $\prod \Phi_{X_{(i)}}$ será chamado de *cluster* B_i.

   (c) Adicionar $\phi^{-X_{(i)}}$ em Φ, o resultado será indicado por $\Phi^{-X_{(i)}}$.

3. Colete de $\Phi^{-X_{(N)}}$ todos os potenciais que contenham variáveis pertencentes a $\mathbf{X}_q$. Multiplique-as e normalize o resultado.

§

Definida a ordem de eliminação das variáveis, a cada passo do algoritmo descrito cria-se uma estrutura de dados, o *bucket*, que contêm a variável a ser eliminada e todos os potenciais nos quais ela está presente. A seqüência de operações envolvendo os *buckets* pode ser representada como uma árvore análoga a Figura 2.12 na qual os nós são exatamente os *buckets*. A raiz da árvore, uma vez normalizada, será a distribuição $P(\mathbf{X}_q, E)$. Esta forma de representação ilustra a seqüência que inclui as etapas de processamento do *bucket*, cálculo do separador do *bucket* - representados na figura por retângulos - e envio do separador ao longo da árvore de *buckets*.

Capítulo 3

Esparsidade na Fatoração de Cholesky

3.1 Permutações e Operações Elementares

Conforme mencionado na introdução, o objetivo do presente trabalho é aplicar técnicas de álgebra linear, em particular as utilizadas na Fatoração de Cholesky de matrizes esparsas, para desenvolver um algoritmo computacionalmente eficiente para inferência em Redes Bayesianas. Com este propósito, esta seção apresenta os conceitos e as técnicas de álgebra linear que serão necessárias para atingir o objetivo proposto. Contudo a apresentação do conteúdo se restringe basicamente aos principais resultados sem se ater às demonstrações dos mesmos. O conteúdo aqui apresentado segue de maneira bastante próxima o desenvolvimento teórico de [20] e [21], no qual uma teorização mais formal, informações e exemplos adicionais podem ser encontrados.

Antes de adentrar na teoria propriamente, convêm descrever a notação que será utilizada. Letras maiúsculas: A, B,... serão utilizadas para indicar matrizes; letras minúsculas: a, b,... serão utilizadas para vetores. Tanto em matrizes quanto em vetores, os índices de linha e coluna serão respectivamente subscritos e superescritos à direita. Desta forma: A_i^j será o elemento da *i-ésima* linha e na *j-ésima* coluna; b_i indica o elemento da *i-ésima* linha do vetor coluna; c^j será o elemento na *j-ésima* coluna do vetor linha. Finalmente, índices e sinais à esquerda, subscritos ou superescritos identificam matrizes e vetores distintos, assim: A, ${}^jA$ e ${}_iA$ são matrizes distintas entre si.

A primeira definição necessária é a definição de **matriz de permutação**. A permutação de linhas de uma matriz identidade gera uma **matriz de permutação de linhas** P. O correspondente

vetor de linhas permutadas será:

$$p = (P. \begin{bmatrix} 1 \\ 2 \\ \vdots \\ m \end{bmatrix})'$$

Analogamente, através da permutação de colunas da matriz identidade obtém-se a correspondente **matriz de permutação de colunas** Q. O vetor de permutação de colunas será:

$$q = \begin{bmatrix} 1 & 2 & \dots & n \end{bmatrix}.Q$$

Seja A uma matriz qualquer, realizar uma matriz permutação de linhas para se obter uma matriz permutada $\tilde{A}$ equivale a multiplicá-la à esquerda pela matriz de permutação de linhas P. Ademais, se p é o correspondente vetor de índices de linha permutadas, então

$$\tilde{A}_i^j = (P.A)_i^j = A_{p(i)}^j$$

De modo análogo, realizar permutação de colunas de uma matriz A equivale a multiplicá-la à direita pela matriz de permutação de colunas Q. Ademais, se q é o correspondente vetor de índices de colunas permutadas, então

$$\tilde{A}_i^j = (A.Q)_i^j = A_i^{q(j)}$$

Exemplo 3.1.1

Para tornar clara a notação utilizada, dada a matriz A a matriz de permutação de linhas P e a matriz de permutação de colunas Q, analise as operações de permutação:

$$A = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix}, P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, Q = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$p = (P.\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix})' = \begin{bmatrix} 3 & 1 & 2 \end{bmatrix}, q = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}.Q = \begin{bmatrix} 3 & 1 & 2 \end{bmatrix}$$

$$P.A = \begin{bmatrix} 31 & 32 & 33 \\ 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix}, A.Q = \begin{bmatrix} 13 & 11 & 12 \\ 23 & 21 & 22 \\ 33 & 31 & 32 \end{bmatrix}$$

§

Uma matriz quadrada A será:

- **simétrica** se e somente se $A = A'$, ou seja, a matriz for igual a sua transposta.
- **ortogonal** se e somente se $A^{-1} = A'$, ou seja, a sua inversa for igual a sua transposta.
- **triangular superior** se e somente se todos os elementos abaixo da diagonal principal são nulos, ou seja, para $i > j \Rightarrow A_i^j = 0$
- **triangular inferior** se e somente se todos os elementos acima da diagonal principal são nulos, ou seja, para $i < j \Rightarrow A_i^j = 0$

Uma **permutação simétrica** de uma matriz quadrada qualquer é uma permutação da forma $\tilde{A} = P.A.P'$, onde P é uma matriz de permutação. A matriz resultante de uma permutação simétrica de uma matriz simétrica será necessariamente simétrica [20].

Denominam-se **operações elementares** sobre uma matriz n x m:

1. Multiplicar uma linha por um escalar não nulo;
2. Adicionar, a uma linha da matriz, uma outra linha da matriz;
3. Subtrair de uma linha da matriz, uma outra linha da matriz multiplicada por um escalar não nulo.

A aplicação de uma operação elementar a uma matriz identidade I $n \times n$ produz uma **matriz elementar** E. Realizar um operação elementar sobre o uma matriz A qualquer equivale a multiplicá-la à esquerda pela matriz elementar correspondente.

3.2 Método de Eliminação de Gauss

Dada uma matriz quadrada A, o objetivo do Método de Eliminação de Gauss, também chamado de triangularização, é anular ('eliminar') os elementos A_i^j de forma a torná-la uma matriz triangular. A eliminação dos elementos da matriz é feita por meio de operações elementares e permutações.

Inicialmente, aplicar o método de Gauss em uma matriz A $n \times n$ consiste em um processo iterativo no qual para cada linha $k = 1, 2, ...n$, na *k-ésima* etapa utiliza-se o elemento A_k^k para anular os elementos da *k-ésima* coluna que estão abaixo da diagonal. Ou seja, nesta ordem, utilize o elemento A_1^1 da primeira linha para anular os elementos abaixo da diagonal principal da primeira coluna, em seguida, o elemento A_2^2 da segunda linha para anular os elementos abaixo da diagonal principal da segunda coluna, e assim por diante.

A cada iteração a matriz A se transforma. Assim, a *k-transformada* da matriz A, indicada por ${}^kA$, correspondente a matriz transformada resultante na *k-ésima* etapa do processo de triangularização. Em cada etapa a linha que está sendo multiplicada e subtraída das demais será denominada de **linha pivô**, seu elemento diagonal de **elemento pivô** e os fatores de multiplicação de **multiplicadores**. O Exemplo 3.2.1 demonstra o processo, indicando ao lado de cada linha o fator pelo qual multiplica-se a *k-ésima* linha antes de subtraí-la da mesma.

Exemplo 3.2.1

$$\left[{}^0A\right] = \begin{bmatrix} 2 & 1 & 3 \\ 2 & 3 & 6 \\ 4 & 4 & 6 \end{bmatrix} \begin{matrix} \\ 1 \\ 2 \end{matrix} \rightarrow \left[{}^1A\right] = \begin{bmatrix} 2 & 1 & 3 \\ 0 & 2 & 3 \\ 0 & 2 & 0 \end{bmatrix} \begin{matrix} \\ \\ 1 \end{matrix} \rightarrow \left[{}^2A\right] = \begin{bmatrix} 2 & 1 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & -3 \end{bmatrix}$$

§

Com o propósito de armazenar os multiplicadores para serem utilizados posteriormente, definem-se as matrizes $n \times n$, ${}^1M, {}^2M, ..., {}^{n-1}M = M$, na qual se $j \leq k$ e $i > j$ então ${}^kM_i^j$ é o multiplicador utilizado na *k-ésima* etapa para anular o elemento da *i-ésima* linha e da *j-ésima* coluna; caso contrário ${}^kM_i^j = 0$.

Por construção, na *k-ésima* etapa, os elementos não nulos de ${}^kM$ correspondem a elementos nulos gerados pelo processo de eliminação na ${}^kA$, e vice-versa. Esta característica pode ser explorada para armazenar em uma única matriz $[{}^kA + {}^kM]$ os elementos não nulos restantes na matriz A e os multiplicadores.

Utilizando a notação mais compacta e indicando os multiplicadores em itálico, no Exemplo 3.2.1 para $k = 0, 1, 2$ o método de eliminação de Gauss pode ser representado pelas matrizes:

$$\left[{}^0A +^0 M\right] = \begin{bmatrix} 2 & 1 & 3 \\ 2 & 3 & 6 \\ 4 & 4 & 6 \end{bmatrix} \rightarrow \left[{}^1A +^1 M\right] = \begin{bmatrix} 2 & 1 & 3 \\ \mathit{1} & 2 & 3 \\ \mathit{2} & 2 & 0 \end{bmatrix} \rightarrow \left[{}^2A +^2 M\right] = \begin{bmatrix} 2 & 1 & 3 \\ \mathit{1} & 2 & 3 \\ \mathit{2} & \mathit{1} & -3 \end{bmatrix}$$

3.3 Pivotamento

Ao longo do processo de triangularização, a operação de executar uma permutação de linhas para trocar o elemento pivô a ser utilizado denomina-se de **pivotamento**. Este tipo de operação pode ser útil ou mesmo necessária:

1. O pivotamento é necessário quando surge um elemento nulo na posição do pivô da etapa seguinte no processo de triangularização, isto é, em ${}^{k-1}A$ anula-se o elemento ${}^{k-1}A^k_k$. Se na coluna ${}^{k-1}A^k$ houver ao menos um elemento não nulo em uma linha l, $l > k$, pode-se permutar as linhas k e l e prosseguir com o processo de triangularização.

2. A estratégia de colocar a cada etapa como pivô o elemento de máximo módulo da coluna é denominada de **pivotamento parcial**. Esta estratégia é recomendável para garantir termos $\left|M^j_i\right| \leq 1$ e desta forma garantir a estabilidade numérica do processo e controlar a propagação de erros de arredondamento.

Com o objetivo de preservar o histórico dos pivotamentos realizados ao longo do processo devem ser guardados os vetores de índices de linha permutados da permutação corrente em relação a matriz original, estes serão os vetores ${}^1p, {}^2p, ..., {}^{n-1}p = p$. O Exemplo 3.3.1 ilustra o processo de triangularização com pivotamento.

Exemplo 3.3.1

A coluna de números à direita da matriz identifica o número de cada linha, e portanto, estão inicialmente ordenados na matriz original, ${}^0A$. Para eliminar os elementos abaixo da diagonal principal na primeira coluna faz-se o pivotamento parcial, ou seja, permuta-se a linha 4 e a linha 1 para que o elemento de máximo módulo da coluna seja o elemento pivo. Adotando estratégia análoga

nas etapas subsequentes, observe que ao final do processo a coluna de números à direita da matriz triangularizada preserva o histórico das permutações que foram realizadas.

$$\left[{}^0A +^0 M\right] = \left[\begin{array}{cccc|c} 2 & 1 & 9 & -1 & 1 \\ 1 & 3 & 7 & 7 & 2 \\ 2 & 8 & 4 & 2 & 3 \\ 3 & 9 & 6 & 6 & 4 \end{array}\right] \rightarrow \left[{}^1A +^1 M\right] = \left[\begin{array}{cccc|c} 3 & 9 & 6 & 6 & 4 \\ \textit{1/3} & 0 & 5 & 5 & 2 \\ \textit{2/3} & 2 & 0 & -2 & 3 \\ \textit{2/3} & -5 & 5 & -5 & 1 \end{array}\right] \rightarrow$$

$$\left[{}^2A +^2 M\right] = \left[\begin{array}{cccc|c} 3 & 9 & 6 & 6 & 4 \\ \textit{2/3} & -5 & 5 & -5 & 1 \\ \textit{1/3} & \textit{-2/5} & 2 & -4 & 3 \\ \textit{2/3} & \textit{0} & 5 & 5 & 2 \end{array}\right] \rightarrow \left[{}^3A +^3 M\right] = \left[\begin{array}{cccc|c} 3 & 9 & 6 & 6 & 4 \\ \textit{2/3} & -5 & 5 & -5 & 1 \\ \textit{2/3} & \textit{0} & 5 & 5 & 2 \\ \textit{1/3} & \textit{-2/5} & \textit{2/5} & -6 & 3 \end{array}\right]$$

§

Se na *k-ésima* etapa para $l = k, ...n$, ${}^{k-1}A^k_l = 0$, isto é o pivô e todos os elementos da coluna abaixo dele na coluna ${}^{k-1}A^k$ se anulam, então as linhas ${}^{k-1}A_l$, $l = k, ...n$ são linearmente dependentes e o processo de triangularização deve ser interrompido.

Uma propriedade importante é que se o vetor de permutação, ${}^{n-1}p = p$, for conhecido a priori, ou seja, antes de iniciar o processo de triangularização, pode-se permutar as linhas da matriz original ${}^0A$ como indicado no vetor de permutação, obtendo uma nova matriz $P.{}^0A$, cujas linhas são as mesmas que a matriz original, a menos da ordem em que estão dispostas. A triangularização poderia então ser executada sem a necessidade de pivotamentos intermediários, obtendo ao final uma matriz triangular equivalente.

Para realizar uma operação de pivotamento não é necessário efetivamente permutar linhas da matriz A, basta que na *k-ésima* etapa da triangularização ao invés de utilizar o índice da linha i, utilizar o índice do vetor de permutação ${}^{k-1}p(i)$.

Lema da Fatoração [20]: Dada uma matriz ${}^0A$ inversível e triangularizável pelo método de Gauss sem nenhum pivotamento, e sejam, $A =^0 A,^1 A,^2 A, ...,^{n-1} A$ e ${}^1M,^2 M,^3 M, ...,^{n-1} M = M$, repectivamente, a *k-transformada* da matriz A e a *k-ésima* matriz dos multiplicadores. Fazendo $U =^{n-1} A$ e $L = M + I$, temos que L é uma matriz triangular inferior e U uma matriz triangular superior tais que $A = L.U$.

O Lema anterior, decorre o teorema que garante que sendo A uma matriz inversível $n \times n$, então existe uma matriz de permutação P de modo que $\tilde{A} = P.A$ é triangularizável pelo método de Gauss e pode ser fatorada em $\tilde{A} = L.U$.

Quando a matriz a ser fatorada é uma matriz simétrica S, a partir da fatoração de Gauss calcula-se $S = L.U$. Com os elementos da diagonal da matriz U pode-se construir uma matriz diagonal D de forma que $U = D.L'$. A decomposição pode portanto ser reescrita como $S = L.D.L'$. Se S for positiva definida, a matriz diagonal D também será positiva definida de forma que a mesma pode ser expressa pelo produto: $D = D^{1/2}.D^{1/2'}$, onde $D^{1/2}$ é uma matriz diagonal contendo a raiz quadrada dos elementos em D, logo, $S = L.D^{1/2}.D^{1/2'}.L' = L.D^{1/2}.(L.D^{1/2})'$.

Definindo a matriz triangular C como $C = L.D^{1/2}$, a decomposição da matriz S pode finalmente ser reescrita como $S = C.C'$. A esta fatoração dá-se o nome de **Fatoração de Cholesky** de S.

A **eliminação simétrica** consiste na fatoração de uma matriz simétrica utilizando apenas permutações simétricas com o objetivo de preservar a simetria de uma matriz original dada. Neste caso, se a matriz original for positiva definida tem-se que $\tilde{A} = Q.A.Q' = C.C'$.

3.4 Esparsidade na Fatoração de Cholesky

Uma **matriz esparsa** é uma matriz na qual apenas uma pequena parte dos elementos não são nulos. Ao executar operações com uma matriz esparsa, por exemplo ao fatorá-la, deve-se procurar manter o padrão de esparsidade da mesma ao longo das transformações. Desta forma é possível minimizar o número de operações aritméticas necessárias a serem realizadas e, do ponto de vista computacional, pode-se explorar esta esparsidade para reduzir os recursos computacionais de memória e processamento necessários para armazená-la e operá-la [20]. Se a matriz for esparsa e simétrica, ampliam-se as possibilidades de otimização na manipulação da mesma.

Em função da forma como são construídas, as matrizes de adjacência de grafos não direcionados cujos vértices têm poucas conexões entre si são matrizes esparsas e simétricas. O Exemplo 3.4.1 ilustra este fato.

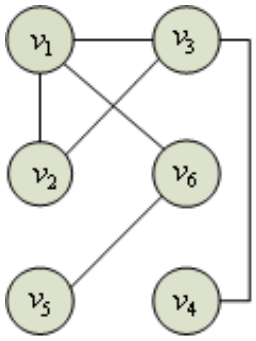
Exemplo 3.4.1

O exemplo apresenta um grafo não direcionado e a respectiva matriz A de adjacência, indicada em (a). Nesta matriz esparsa apenas os elementos indicados com a letra 'x' são diferentes de zero.

Os números na diagonal principal, correspondem ao número da linha (ou coluna) e foram utilizados apenas para facilitar a visualização das operações de permutação apresentadas abaixo.

Conforme descrito anteriormente, os elementos A_j^i e A_i^j serão não nulos se o vértice i estiver conectado ao vértice j por um arco.

As matrizes (b) e (c) são permutações simétricas da matriz A. Note que permutações simétricas mantêm as matrizes de adjacência fiéis ao grafo original.



$$\begin{bmatrix} 1 & x & x & & & x \\ x & 2 & x & & & \\ x & x & 3 & x & & \\ & & x & 4 & & \\ & & & & 5 & x \\ x & & & & x & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & x & x & x & & \\ x & 3 & & x & x & \\ x & & 6 & & & x \\ x & x & & 2 & & \\ & x & & & 4 & \\ & & x & & & 5 \end{bmatrix} \quad \begin{bmatrix} 5 & & & x & & \\ & 4 & & & x & \\ & & 2 & & x & x \\ x & & & 6 & & x \\ & x & x & & 3 & x \\ & & x & x & x & 1 \end{bmatrix}$$

(a) (b) (c)

§

Para fatoração da matriz utilizando o método de Gauss as transformações ${}^kA \rightarrow^{k+1} A$ podem diminuir o número de elementos nulos. Neste caso dir-se-á que houve um **preenchimento** de algumas posições. Seria computacionalmente eficiente e desejável escolher o elemento pivô ${}^kA$ de modo a minimizar este preenchimento.

A teoria de grafos apresentada na Seção 2.1 será útil para estudar o processo de eliminação simétrica visando torná-lo computacionalmente mais eficiente. O primeiro conceito que convêm recapitular é o conceito de **Grafo de Eliminação**.

Inicialmente, a um grafo não direcionado $G = (V, E)$ com n vértices está associada uma matriz de adjacência ${}^0A$ $n \times n$ e, portanto, cada vértice i do grafo está associado a uma coluna em ${}^0A$. Para uma determinada ordem de eliminação $q = [(1), (2), ..., (N)]$, obtém-se uma seqüência de grafos de eliminação $G_k = (V_k, E_k)$ para $k = 1, 2, ..., n$, na qual $q(k) = i$ significa que o vértice i foi o *k-ésimo* vértice de G a ser eliminado. A cada grafo G_k está associada uma matriz de adjacência transformada ${}^kA$ resultado da anulação na matriz ${}^{k-1}A$ dos elementos abaixo da diagonal principal da coluna associada ao vértice i.

Os grafos de eliminação G_k são grafos cujas matrizes de adjacência são as matrizes transformadas

^kA do processo de fatoração. Esta associação implica que ao eliminar a *j-ésima* coluna da matriz na Fatoração de Cholesky serão preenchidas na matriz exatamente as posições correspondentes aos arcos de preenchimento adicionados ao grafo preenchido quando da eliminação do vértice $q(j)$.

Analogamente à eliminação de vértices em um grafo, dir-se-á que a ordem de eliminação, q, é perfeita quando não ocorrer o preenchimento de nenhuma posição ao longo do processo de eliminação das colunas na Fatoração de Cholesky.

Exemplo 3.4.2

A Figura 3.1 ilustra a seqüência de grafos de eliminação para a permutação representada pela matriz (b) do Exemplo 3.4.1.

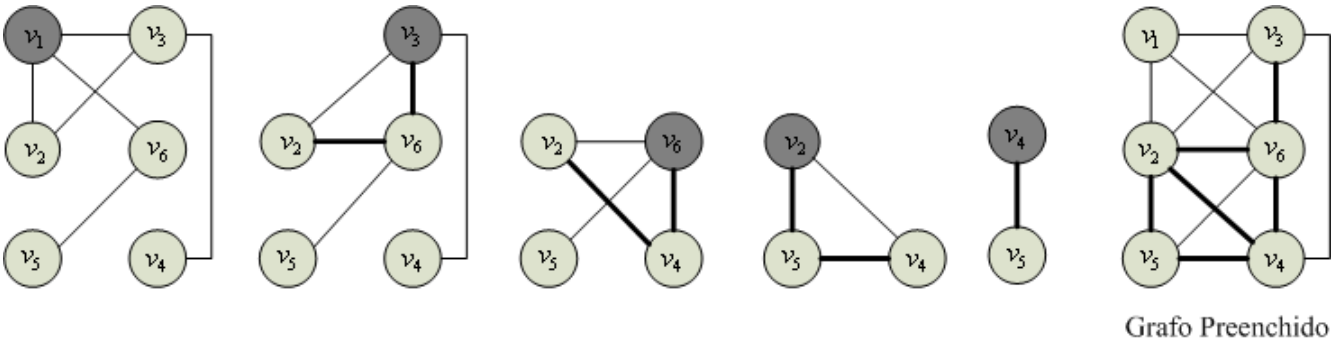


Figura 3.1: Grafos de Eliminação da matriz de permutação (b)

Abaixo estão as mesmas matrizes do exemplo anterior, nas quais estão indicados por '0' os elementos que foram preenchidos durante o processo de fatoração de Cholesky. Observe na matriz (b) a correspondência entre os elementos que foram preenchidos na matriz com os arcos de preenchimento indicados no grafo preenchido da Figura 3.1.

$$\begin{bmatrix} 1 & x & x & & & x \\ x & 2 & x & & & 0 \\ x & x & 3 & x & & 0 \\ & & x & 4 & & 0 \\ & & & & 5 & x \\ x & 0 & 0 & 0 & x & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & x & x & x & & \\ x & 3 & & x & x & \\ x & 0 & 6 & 0 & 0 & x \\ x & x & 0 & 2 & 0 & 0 \\ & x & 0 & 0 & 4 & 0 \\ & & x & 0 & 0 & 5 \end{bmatrix} \quad \begin{bmatrix} 5 & & & x & & \\ & 4 & & & x & \\ & & 2 & & x & x \\ x & & & 6 & & x \\ & x & x & & 3 & x \\ & & x & x & x & 1 \end{bmatrix}$$

(a) (b) (c)

§

Seja A uma matriz de adjacência de um grafo $G = (N, E)$, uma ordem de eliminação q, e o respectivo grafo preenchido F, consideremos o conjunto de índices de linha de Elementos Não Nulos

(ENNs) na coluna j do fator de Cholesky, $L^j|Q.A.Q' = L.L'$:

$$enn(L^j) = \{i|i > j \wedge \{q(i), q(j)\} \in F\} + \{j\}$$

Define-se a **Árvore de Eliminação**, H, como

$$\Gamma_H^{-1}(j) = \begin{cases} j \text{ , se } enn(L^j) = \{j\} \\ min\left\{i > j | i \in enn(L^j)\right\} \text{ , caso contrário} \end{cases}$$

A expressão acima indica que o pai do nó j em H, é simplesmente o primeiro elemento não nulo, e não diagonal na *j-ésima* coluna do fator de Cholesky L.

Exemplo 3.4.3

A Figura 3.2 mostra as três Árvores de Eliminação para fatoração de Cholesky respectivamente para as matrizes (a), (b) e (c) do Exemplo 3.4.1.

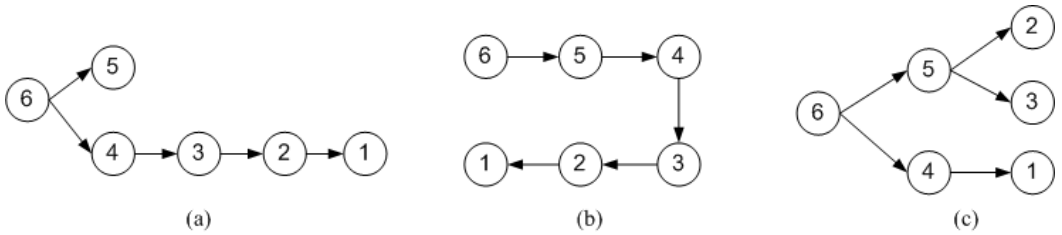


Figura 3.2: Árvores de Eliminação para as matrizes do Exemplo 3.4.1

§

Pode-se demonstrar [20] que: Dado $i > j$,

$$i \in enn(L^j) \Rightarrow j \in \bar{\Gamma}(i)$$

onde, $\bar{\Gamma}(i)$ é a uma função de descendência, ou seja, representa todos os descendentes do nó i, conforme definido na seção 2.1. A expressão acima indica portanto que qualquer índice de linha abaixo da diagonal principal na *j-ésima* coluna de L é um ascendente de j na árvore de eliminação.

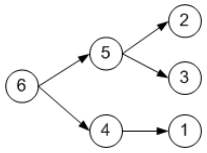
A árvore de eliminação retrata portanto as dependências entre as colunas para o processo de fatoração da matriz: a *j-ésima* coluna da matriz poderá ser eliminada se e somente se todos os descendentes de j na árvore de eliminação já tiverem sido eliminados.

Tomando como base a topologia da Árvore de Eliminação e considerando a possibilidade de executar o processamento em paralelo, pode-se eliminar simultaneamente todas as colunas em um mesmo nível da árvore, começando pelas folhas e terminando na raiz.

Exemplo 3.4.4

Considere a matriz abaixo que apresenta o mesmo padrão de esparsidade da matriz (c) do exemplo 3.4.1, obtida pela permutação simétrica da matriz de adjacência A. A árvore de eliminação correspondente está representada pelo grafo (c) na Figura 3.2.

No exemplo estão representados em negrito, os elementos da diagonal correspondentes às colunas que serão eliminadas e os multiplicadores estão indicados em itálico.



$$\begin{bmatrix} \mathbf{1} & & & 7 & & \\ & \mathbf{2} & & & 8 & \\ & & \mathbf{3} & & 6 & 9 \\ 7 & & & 53 & & 2 \\ & 8 & 6 & & 49 & 23 \\ & & 9 & 2 & 23 & 39 \end{bmatrix} \quad \begin{bmatrix} 1 & & & 7 & & \\ & 2 & & & 8 & \\ & & 3 & & 6 & 9 \\ \mathit{7} & & & \mathbf{4} & & 2 \\ & \mathit{4} & \mathit{2} & & \mathbf{5} & 5 \\ & & \mathit{3} & 2 & 5 & 12 \end{bmatrix} \quad \begin{bmatrix} 1 & & & 7 & & \\ & 2 & & & 8 & \\ & & 3 & & 6 & 9 \\ \mathit{7} & & & 4 & & 2 \\ & \mathit{4} & \mathit{2} & & 5 & 5 \\ & & \mathit{3} & \mathit{0.5} & \mathit{1} & \mathbf{6} \end{bmatrix}$$

Passo 1 Passo 2 Passo 3

A árvore tem 3 níveis cujos conjuntos de nós das folhas para a raiz são respectivamente: $\{1,2,3\}$, $\{4,5\}$ e $\{6\}$. Esta matriz poderá portanto ser fatorada em três passos. §

O **Teorema da Fatoração Simbólica** [20] apresenta uma seqüência de passos que torna computacionalmente mais eficiente a obtenção do grafo preenchido, $P = (V, F)$, e da árvore de eliminação, H, dado o padrão de esparsidade da matriz original, $G = (V, E)$, e a ordem de eliminação, q. Assim, como descrito no final da seção 2.1, nesta versão simplificada a eliminação do vértice $q(j)$ gera o grafo de eliminação G_j^* com o preenchimento apenas dos lados incidentes ao seu vizinho mais próximo de ser eliminado.

A versão simplificada do processo de eliminação pode portanto ser descrita pelos seguintes passos:

$G_j^* = (V_j, E_j^*)$,

$E_1^* = E$,

$\Gamma_{H^*}^{-1}(j) = min\left\{i > j | \{q(j), q(i)\} \in E_j^*\right\}$,

$E_{j+1}^* = \left\{\{a,b\} \in E_j^* | \bar{q}(a), \bar{q}(b) > j\right\} \cup \left\{\{q(h(j)), v\}, v | \bar{q}(v) \geq j \wedge \{q(j), v\} \in E_j^*\right\}$,

$F^* = \cup_1^n E_j^*$

Pode-se demonstrar que nestas condições o grafo preenchido e a árvore de eliminação obtidos no processo simplificado coincidem com a definições dadas anteriormente, ou seja, $F^* = F$ e $H^* = H$.

3.5 Determinação da Ordem de Eliminação

Esta seção tem como objetivo descrever a aplicação das Heurísticas de Busca em Largura e da Heurística de Gibbs para se obter uma boa ordenação para eliminação dos vértices. Conforme procurou-se deixar claro nas seções anteriores, a ordem de eliminação das variáveis é fundamental para garantir que o processo seja computacionalmente eficiente. No entanto, até o presente momento adotou-se a ordem de eliminação dos vértices de um grafo ou, o seu análogo para matrizes, a ordem de eliminação das colunas como conhecida. Preencher esta lacuna é o objetivo desta seção.

Considere um grafo $G = (V, E)$ e um separador S em G que divide $V - S$ em componentes conexas $C_1, C_2, C_3, ..., C_k$. Recursivamente para cada uma das componentes podemos considerar um separador que a divida em novas componentes conexas. Este processo pode ser representado pela **árvore de dissecção**, D, na qual o separador S é a raiz, uma componente separada por S, ou um separador dentro da mesma, é filha de S e as componentes não divididas são as folhas da árvore.

Exemplo 3.5.1

Considere o grafo da Figura 3.3. Utilizando inicialmente o separador $\{4, 5\}$ divide-se o grafo G nas componentes conexas $C_1 = \{1, 3, 8, 9, 10\}$, $C_2 = \{11\}$ e $C_3 = \{2, 6, 7, 12, 13\}$, conforme indicado na Figura 3.4.

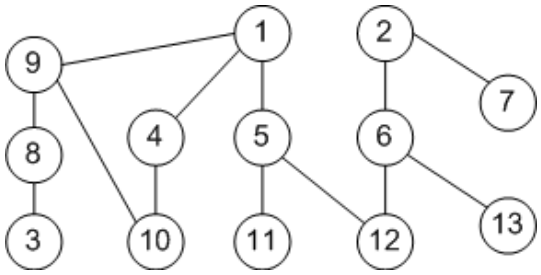


Figura 3.3: Grafo original $G = (V, E)$

Em seguida adotando para a componente C_1 o separador $\{9\}$ e para a componente C_3 o separador $\{6\}$ obtêm-se a árvore de dissecção D representada na Figura 3.5.

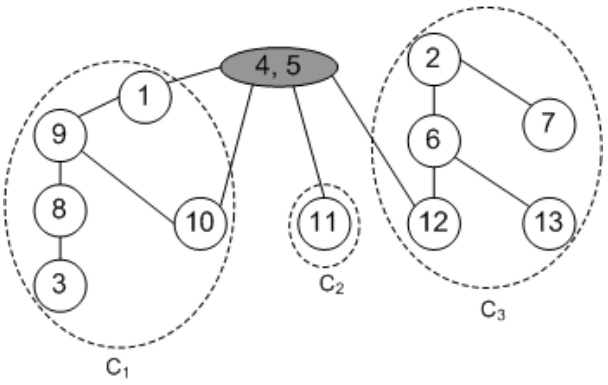


Figura 3.4: Grafo resultante da separação adotando separador $\{4, 5\}$

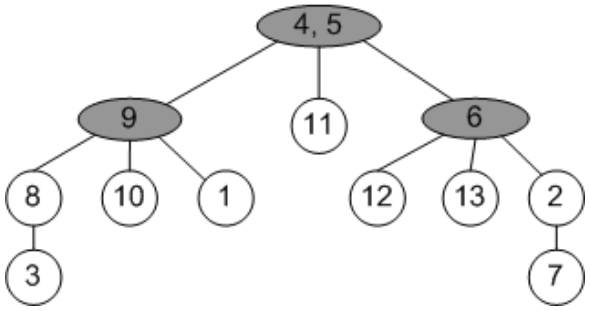


Figura 3.5: Árvore de dissecção resultante da separação adotando os separadores $\{9\}$ e $\{6\}$

§

Dada uma árvore H de raiz r, uma **pós-ordem**, $\tilde{q}$, dos vértices de H é uma ordem que enumera os vértices de cada uma as árvores $H - r$, recursivamente em pós-ordem e ao final a raíz r. Sendo $\tilde{q}$ uma pós-ordem da árvore de dissecção D gerada a partir de um grafo G, a **ordem de dissecção**, q, é obtida substituindo em $\tilde{q}$ cada nó, d, da árvore de disseção pelos vértices correspondentes do grafo original G que compõe d.

Uma possível ordem de dissecção para a árvore de dissecção, D, representada pela Figura 3.5 seria $q = [3, 8, 1, 10, 9, 12, 13, 2, 7, 6, 11, 4, 5]$.

Considere uma árvore de dissecção, D, de nós d, gerada a partir de um grafo $G = (V, E)$ com $v \in V$ e uma na ordem de dissecção q. Pode-se demonstrar [20] que a eliminação de um dado vértice $v \in d$, pode gerar apenas o inclusão de arcos de preenchimento entre os vértices de G pertencentes ao nó d, ou entre vértices de d e vértices em nós ancestrais de d em D ou ainda entre vértices em ascendentes de d.

Retomando a analogia entre eliminação de vértices em um grafo e a eliminação simétrica de colunas na matriz de adjacência que o representa, a eliminação de uma coluna relativa a um determinado vértice irá preencher as posições da matriz correspondentes aos arcos de preenchimento adicionados ao grafo, respeitando as condições descritas no parágrafo anterior.

Exemplo 3.5.2

A eliminação dos vértices do grafo original do exemplo 3.5.1, Figura 3.3, na ordem de dissecção $q = [3, 8, 1, 10, 9, 12, 13, 2, 7, 6, 11, 4, 5]$, obtida a partir da árvore da dissecção representada pela Figura 3.5, gera o grafo preenchido representado pela figura abaixo.

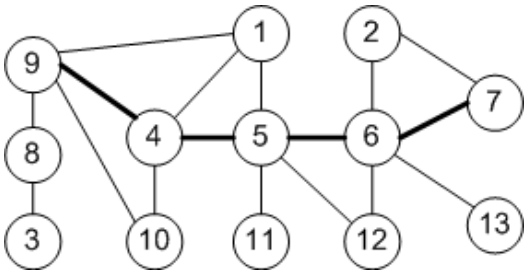


Figura 3.6: Grafo preenchido decorrente da eliminação dos vértices no grafo $G = (V, E)$

A eliminação simétrica correspondente, mantendo a ordem de eliminação, preenche as posições indicadas com **0** na matriz abaixo.

$$
\begin{array}{c}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13
\end{array}
\begin{bmatrix}
3 & x & & & & & & & & & & & \\
x & 8 & & & x & & & & & & & & \\
 & & 1 & & x & & & & & & & x & x \\
 & & & 10 & x & & & & & & & x & \\
 & x & x & x & 9 & & & & & & & \mathbf{0} & \\
 & & & & & 12 & & & & x & & & x \\
 & & & & & & 13 & & & x & & & \\
 & & & & & & & 2 & x & x & & & \\
 & & & & & & & x & 7 & \mathbf{0} & & & \\
 & & & & & x & x & x & \mathbf{0} & 6 & & & \mathbf{0} \\
 & & & & & & & & & & 11 & & x \\
 & & x & x & \mathbf{0} & & & & & & & 4 & \mathbf{0} \\
 & & x & & & x & & & & \mathbf{0} & x & \mathbf{0} & 5
\end{bmatrix}
$$

§

Considerando os possíveis arcos de preenchimento que podem ser adicionados ao grafo G, ou analogamente, as possíveis posições que podem ser preenchidas na matriz, para uma ordem de

eliminação, para minimizar tais possibilidades de preenchimento, é desejável que o separador S_k seja "pequeno e balanceado", i.e. tal que

- O número de vértices de G em S_k seja o menor possível.
- As sub-árvores tenham aproximadamente o mesmo número de vértices de G.

Recursivamente, é desejável que a raiz de cada uma das sub-árvores seja um bom separador.

Para se obter, para um grafo qualquer, bons separadores que posteriormente determinem uma ordem de eliminação que minimize o surgimento de arcos de preenchimento, ou o preenchimento de posições na matriz, bem como gerem Árvores de Eliminação que possibilitem a paralelização das operações, podem ser utilizadas a **Heurística de Busca em Largura** (BEL) e a **Heurística de Gibbs** [21] e [17].

Heurística da Busca em Largura

Dado um grafo $G = (V, E)$, adotando um vértice $v \in V$ como raiz particione os vértices de G em níveis $L_0, L_1, ...L_k$, definidos por:

$$L_0 = \{v\}, L_{i+1} = adj(L_i) - L_{i-1}$$

§

A **profundidade** do nível L_i é i, e a **largura** do nível L_i, definida como o número de elementos do nível L_i, é $\#L_i$. A máxima profundidade e a máxima largura da BEL são respectivamente a sua profundidade e a sua largura.

O nível L_i separa em G os vértices em níveis mais profundos dos vértices em níveis menos profundos profundos que i [20].

A heurística de BEL procura um separador balanceado $S \subseteq L_i$ tomando $i \approx k/2$, ou então tomando:

$$i | \sum_{j=0}^{i-1} \#L_j < n/2 \wedge \sum_{j=i+1}^{k} \#L_j < n/2$$

Para reduzir a largura da BEL, com o objetivo de se obter separadores pequenos, a heurística procura para raiz um vértice $v \in V$ que gere uma BEL de máxima profundidade.

No grafo G a **distância** de um vértice v a um vértice w, representa-se por $dist(v, w)$, é número de arcos do caminho mais curto entre ambos os vértices. A **excentricidade** de um vértice v é $exc(v) = max_{w \in V} dist(v, w)$. Um vértice de máxima excentricidade se diz **periférico**, e sua excentricidade é o **diâmetro** de G.

Uma BEL cuja raiz seja v terá a profundidade igual a excentricidade de v. Desta forma, com o intuito de gerar a BEL de máxima profundidade é desejável que a raiz da BEL seja um vértice periférico. No entanto, encontrar um vértice periférico é um problema computacionalmente difícil.

Para contornar o problema de selecionar um vértice periférico adotou-se a Heurística de **Gibbs** para encontrar um vértice **quase-periférico**, conforme proposto por [20] e [17].

Heurística de Gibbs

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

1. Escolher como raiz um vértice de grau mínimo [1]
2. Forme os níveis da BEL com raiz v, $L_0, L_1, ...L_k$, particione o nível mais profundo em suas l componentes conexas, $l_k = \cup_{j=1}^{l}$, e tome um vértice de grau mínimo, v_j, em cada componente.
3. Para $j = 1 : l$

   Tome v_j como nova raiz e encontre os níveis de BEL $L_0, L_1, ...L_{k'}$

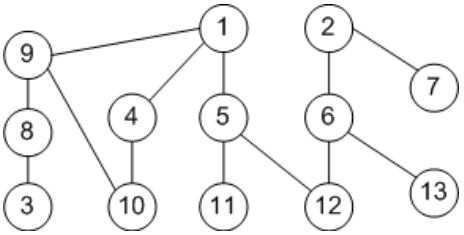
   Até que $k' > k$ ou $j = l$
4. Se o passo 3 terminou com $k' > k$, volte ao passo 2. Caso contrário a atual raiz é um vértice quase-periférico.

§

[1] Grau de um vértice é igual ao número de arcos que entram ou saem do mesmo

Exemplo 3.5.3

Retomando o grafo e a árvore de dissecção estudados nos Exemplos 3.5.1 e 3.5.2.



A heurística de Gibbs encontra 3 como vértice quase-periférico. Tomando 3 como raiz gera-se a árvore H por BEL:

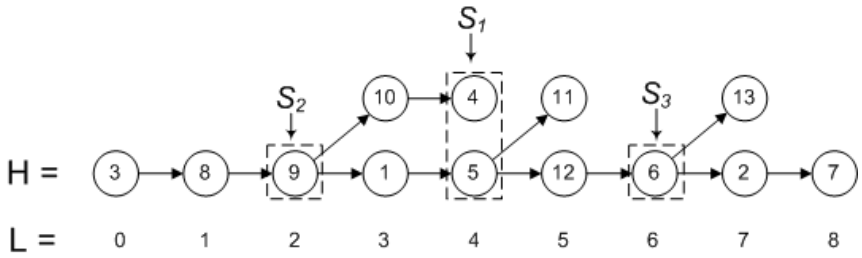
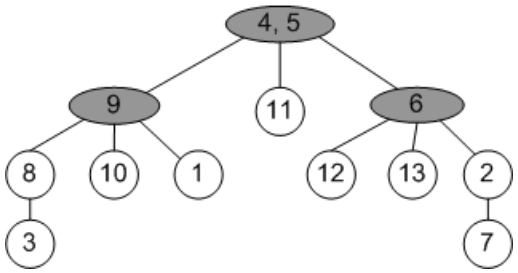


Figura 3.7: Exemplo árvore gerada utilizando as Heurísticas de Gibbs e BEL

Utilizando inicialmente o separador $S_1 = \{4, 5\}$ obtêm-se as componentes conexas $C_1 = \{1, 3, 8, 9, 10\}$, $C_2 = \{11\}$ e $C_3 = \{2, 6, 7, 12, 13\}$. Em seguida, para a componente C_1 escolhe-se o separador $S_2 = \{9\}$ e para a componente C_3 o separador $S_3 = \{6\}$. Obtêm-se portanto a árvore de dissecção da Figura 3.5.



Listando-se os nós da árvore de disseção assim obtida em pós-ordem obtêm-se a ordem de dissecção $q = [3, 8, 1, 10, 9, 12, 13, 2, 7, 6, 11, 4, 5]$. §

Capítulo 4

Algoritmo Paralelo para Inferência em Redes Bayesianas

O objetivo deste capítulo é combinar os arcabouços teóricos de Redes Bayesianas e Álgebra Linear Computacional para desenvolver um algoritmo capaz otimizar e paralelizar o processamento computacional do Método de Eliminação de Variáveis para inferência em Redes Bayesianas. No Capitulo 2 foram introduzidos os conceitos básicos de Redes Bayesianas e dos procedimentos para inferência nesta classe de modelos probabilísticos, bem como os pontos teóricos correlatos das teorias de probabilidade e de grafos. No Capítulo 3 apresentou-se métodos característicos de Álgebra Linear Computacional utilizados para Fatoração de Cholesky de matrizes esparsas. No presente capítulo os arcabouços teóricos apresentados nos capítulos anteriores serão combinados para desenvolver um algoritmo computacional que utiliza técnicas de Fatoração de Cholesky para matrizes esparsas para otimizar e paralelizar o processamento numérico necessário para inferência em Redes Bayesianas.

Para o desenvolvimento de algoritmo paralelo para inferência em redes Bayesianas o processo de inferência foi separado em duas fases, a primeira Fase Simbólica e uma segunda Fase Numérica. As estruturas combinatórias geradas na Fase Simbólica, e comum aos métodos de inferência em Redes Bayesianas e de fatoração de matrizes esparsas, são a chave para a implementação computacionalmente eficiente de um algoritmo capaz de lidar com grandes modelos. Estas estruturas possibilitam a otimização do processamento numérico na inferência de Redes Bayesianas. Tais estruturas geram informações que permitem que os recursos computacionais necessários para o processamento sejam

alocados estaticamente antes do início do processamento. Além disso, e talvez mais importante, uma das estruturas geradas na Fase Simbólica, a Árvore de Eliminação, possibilita o controle e a paralelização das operações de eliminação de variáveis à medida que codifica as relações de dependência entre as mesmas.

Com o objetivo de ilustrar a algoritmo proposto, ao final da seção foi inserido um exemplo completo que detalha e exemplificada cada um dos procedimentos descritos nesta e nas seções anteriores.

4.1 Paralelização do Método de Eliminação de Variáveis

O algoritmo de paralelização do Método de Eliminação de Variáveis, conforme representado graficamente na Figura 4.1, tem duas fases distintas:

1. **Fase Simbólica**: Fase na qual são definidos o conjunto das variáveis requisitadas, a ordem de eliminação e a Árvore de Eliminação;
2. **Fase Numérica**: Fase na qual ocorre o processamento numérico que inclui basicamente as operações de produto, marginalização de variáveis e normalização envolvendo os potenciais da Rede Bayesiana.

4.1.1 Fase Simbólica

A Fase Simbólica do algoritmo paralelo para inferência em uma Rede Bayesiana é assim denominada pois baseia-se unicamente na topologia do grafo da rede. A Fase Simbólica não envolve a manipulação de valores numéricos que definem a rede e portanto independe das distribuições de probabilidades das variáveis e do estados das variáveis observadas. Esta característica é importante pois potencialmente exige um menor processamento computacional e possibilita que, uma vez que tenha sido executada e mantida constante a topologia da rede, as estruturas resultantes possam ser aproveitadas em sucessivos e distintos processamentos numéricos de inferência mesmo que os valores das distribuições de probabilidades e/ou os estados observados das variáveis sejam alterados.

Por uma questão de coerência, optou-se por incluir na Fase Simbólica o procedimento para a determinação do conjunto de variáveis requisitadas, para o qual foi utilizado o algoritmo Bayes-Ball descrito na Seção 2.3.5. No entanto, este é um procedimento padrão, usualmente aplicado a modelos

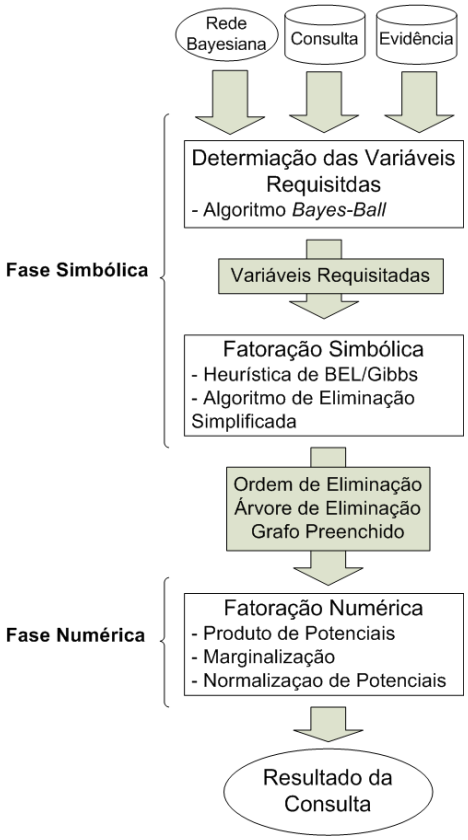


Figura 4.1: Representação gráfica do algoritmo paralelo para inferência em Redes Bayesianas.

gráficos análogos a Redes Bayesianas e característico dos algoritmos para inferência nos mesmos. Por estes motivos o mesmo não será detalhado nesta seção.

O principal procedimento executado na Fase Simbólica será denominado genericamente de *Fatoração Simbólica*. A utilização deste procedimento constitui a contribuição mais importante do presente trabalho pois o mesmo resulta da aplicação para inferência em Redes Bayesianas de técnicas de Álgebra Linear Computacional para fatoração de matrizes esparsa. A Fatoração Simbólica consiste na determinação da ordem de eliminação e na construção da Árvore de Eliminação dos vértices do grafo da Rede Bayesiana. O objetivo de gerar estas duas estruturas de dados é possibilitar a paralelização e a otimização do processamento numérico necessário para inferência.

Convêm ressaltar que os algoritmos utilizados na Fatoração Simbólica tanto para a determinação

da ordem de eliminação como para a construção da Árvore de Eliminação utilizam basicamente o Grafo Moral associado a Rede Bayesiana. Ou seja, os mesmos assumem como dado de entrada um grafo não direcional.

A denominação Fatoração Simbólica foi adotada em analogia ao processo de Fatoração de Cholesky de matrizes esparsas. No caso da Fatoração de Cholesky o procedimento para a determinação da Árvore de Eliminação, uma vez definido o grafo não direcional que representa a matriz, ocorre sem a necessidade de utilizar os valores numéricos da matriz. Analogamente, a Fatoração Simbólica de uma Rede Bayesiana, que utiliza para construção da Árvore de Eliminação exatamente o mesmo algoritmo que a Fatoração de Cholesky em questão, tem como base apenas a topologia do Grafo Moral associado a mesma.

Para a determinação da ordem de eliminação adotou-se a **Heurística de Busca em Largura** (BEL) e a **Heurística de Gibbs** ambas detalhadas na Seção 3.5. Cabe mencionar que existem formas alternativas para se determinar a ordem de eliminação, no entanto, as heurísticas adotadas são particularmente interessantes pois estão estreitamente relacionadas com a etapa seguinte da Fatoração Simbólica na qual será gerada a Árvore de Eliminação. Conforme mencionado anteriormente, o que torna as heurísticas adotadas interessantes para o algoritmo que será proposto é o fato as mesmas buscarem simultaneamente dois objetivos principais:

1. Gerar uma ordem eficiente de eliminação, o que basicamente significa minimizar a inserção de arcos de preenchimento à medida que os vértices do grafo são eliminados;

2. Gerar uma ordem que maximize as possibilidades de paralelização do processamento numérico.

Os passos básicos das heurísticas foram detalhados ao descrevê-las na Seção 3.5. O algoritmo descrito abaixo é apenas uma sistematização teórica das mesmas aplicada ao Grafo Moral de uma Rede Bayesiana. Detalhes adicionais para implementação computacional do algoritmo para se obter a ordem de eliminação estão presentes na Seção 5.4.

Algoritmo para Determinação da Ordem de Eliminação

Dado um Grafo Moral $G = (V, E)$ associado a uma Rede Bayesiana:

1. Inicialize todos os vértices do grafo como não testados.

2. Escolha um vértice de grau mínimo v do grafo e marque-o como testado.

3. Construa a BEL adotando como raiz o vértice v e a defina como a $BEL_{corrente}$.

4. Construa uma lista de nós a serem testados com os nós do nível mais profundo da $BEL_{corrente}$.

5. Enquanto a lista de nós a serem testados não for vazia:

   5.1. Para cada nó v da lista de nós a serem testados faça:

   Marque o nó v como testado.

   Construa a BEL adotando como raiz o vértice v e a defina como BEL_{nova}.

   Se, segundo os critérios de comparação, a BEL_{nova} for superior a $BEL_{corrente}$ então adote a BEL_{nova} como a nova $BEL_{corrente}$.

   5.2. Se houve a substituição da $BEL_{corrente}$ construa uma nova lista de nós a serem testados com os nós **não marcados como testados** do nível mais profundo da $BEL_{corrente}$.

6. Para a $BEL_{corrente}$ construa a Árvore de Dissecção.

7. A partir da raiz percorra a Árvore de Dissecção recursivamente montando a ordenação do vértices em pós-ordem.

8. Monte a ordem de eliminação percorrendo a lista dos vértices da Árvore de Dissecção listados em pós-ordem substituindo cada vértice da Árvore de Dissecção pelos vértices correspondentes do grafo original.

§

Definida a ordem de eliminação, a etapa seguinte da Fatoração Simbólica da qual resultam a Árvore de Eliminação e o Grafo Preenchido foi implementada utilizando o algoritmo de eliminação simplificada. Este algoritmo de eliminação dos vértices de uma grafo foi introduzido na Seção 2.1 e formalmente detalhado na Seção 3.4 com Teorema da Fatoração Simbólica. O algoritmo de eliminação simplificada é computacionalmente mais eficiente pois ao longo do processo de eliminação dos vértices adiciona-se ao grafo apenas os arcos de preenchimento que incidem sobre o vértice vizinho *mais* próximo de ser eliminado.

Os arcos existentes unindo os vértices do grafo inicial, bem como os arcos de preenchimento inseridos ao longo da seqüência de grafos de eliminação associada ao processo de eliminação dos vértices, expressam a relação de dependências entre as operações de marginalização de variáveis na Rede Bayesiana. Da mesma forma que na Fatoração de Cholesky os arcos mencionados expressam a relação de dependência na eliminação das colunas da matriz. A identificação destas relações de

dependência é a base para a construção da Árvore de Eliminação e para a paralelização do processamento numérico. A partir das relações identificadas é possível estabelecer quais operações são independentes entre si e portanto poderão ser executadas paralelamente.

Algoritmo de Fatoração Simbólica para Construção da Árvore de Eliminação

Dado um Grafo Moral $G = (V, E)$ associado a uma Rede Bayesiana e uma ordem de eliminação q:

1. Inicializar com os nós de G a estrutura para armazenar a Árvore de Eliminação H, mas sem estabelecer ainda os arcos de H.

2. Inicializar com os nós e os arcos de G a estrutura para armazenar o Grafo Preenchido F, ligando em F apenas os nós que estejam ligados em G.

3. $j = 0$;

4. Enquanto $j <$ (Número de Vértices de G - 1) faça

   4.1. Definir v_j como o *j-ésimo* nó de q;

   4.2. $i = j + 1$;

   4.3. Enquanto $i <$ (Número de Vértices G) faça

      4.3.1. Definir v_i como o *i-ésimo* nó de q;

      4.3.2. Se o existe em F o arco $\{v_j, v_i\}$ então

         Acrescentar em H um arco direcional $\{v_i, v_j\}$;

         Definir v_i como o vizinho de v_j mais próximo de ser eliminado;

         Interromper o *loop* e as iterações de i;

      4.3.3. Incrementar $i = i + 1$;

   4.4. $k = i + 1$;

   4.5. Enquanto $k <$ (Número de Vértices G) faça

      4.5.1. Definir v_k como o *k-ésimo* nó de q;

      4.5.2. Se existir em F um arco $\{v_j, v_k\}$ e não existir em F o arco $\{v_i, v_k\}$ então

         Acrescentar em F o arco $\{v_i, v_k\}$.

      4.5.3. Incrementar $k = k + 1$;

   4.6. Incrementar $j = j + 1$;

§

Descrito o processo de Fatoração Simbólica observe que realmente não foi utilizado em nenhum momento nenhuma informação numérica relativa às distribuições de probabilidade ou estados das variáveis da Rede Bayesiana. Isto significa que, mantidos constantes os elementos dos conjuntos $\mathbf{X}_q$, E e estrutura da rede, é possível fazer inúmeros exercício de inferência alterando os estados de E ou mesmo as distribuições de probabilidades das variáveis sem que para isto seja necessário refazer a Fatoração Simbólica, ou seja, não é necessário refazer o processo para obter a ordem de eliminação e a Árvore de Eliminação.

4.1.2 Fase Numérica

Na fase numérica do algoritmo paralelo para inferência em uma Rede Bayesiana são feitas as operações de produto entre potenciais, marginalização das variáveis e normalização dos mesmos, ou seja, operações que envolvem os números que definem as distribuições de probabilidades da Rede Bayesiana. Por este motivo, este conjunto de operações será denominado: *Fatoração Numérica*. Estas operações são as mesmas realizadas por outros métodos de inferência em modelos gráficos probabilísticos, em particular no Método de Eliminação de Variáveis descrito na Seção 2.4.

O aspecto que distingue o método proposto no presente trabalho dos demais métodos existentes para o processamento numérico é a utilização das estruturas de dados obtidas na etapa de Fatoração Simbólica para otimizar e paralelizar as operações. Após a execução da fase simbólica as informações que podem ser extraídas a partir da ordem de eliminação e da Árvore de Eliminação permitem que, antes que se inicie o processamento numérico, se conheça quais as operações de produto, de marginalização e de normalização envolvendo os potenciais serão realizadas; as dimensões dos potenciais envolvidos; e principalmente quais operações poderão ser executadas em paralelo. Estas informações podem ser utilizadas para alocar de forma estática os recursos computacionais que serão necessários antes de se iniciar o processamento, o que do ponto de vista computacional pode representar um ganho significativo de performance.

Um aspecto fundamental do método proposto é a possibilidade de paralelizar as operações de eliminação das variáveis com base na topologia da Árvore de Eliminação. Caso se disponha de recursos computacionais para a execução em paralelo das operações, a partir das relações de dependência entre as mesmas, codificadas na Árvore de Eliminação, é computacionalmente possível construir estruturas de dados especiais para controlar a execução dos processos que irão realizá-las e desta forma executar em paralelo operações que sejam independentes entre si.

Para compreender como as operações poderão ser paralelizadas, considere um exemplo hipotético representado pela Figura 4.2. O grafo mais a esquerda corresponde a uma Árvore de Eliminação obtida para uma Rede Bayesiana sobre a qual se deseja fazer um exercício de inferência. Cada vértice da Árvore de Eliminação pode ser interpretado como uma operação de eliminação de uma variável. Os arcos direcionais da árvore expressam uma relação de dependência entre as operações. Assim, a operação que um determinado vértice representa poderá iniciar sua execução desde que as execuções de todos seus descendentes tenham sido concluídas. A seqüência de processamento está indicada na figura da esquerda para a direita. Em cada um dos passos, os vértices que teoricamente poderiam ser executados em paralelo estão assinalados. Observe que o processamento inicia-se pelas *folhas da árvore*, ou seja, pelos vértices da extremidade da árvore, os quais não possuem descendentes. O processamento avança com a execução dos vértices cujos filhos tenham sido executados até que a *raiz da árvore* seja executada, concluindo o processamento.

Observe ainda na Figura 4.2 que, se o processamento fosse feito sequencialmente, seriam teoricamente necessários 11 ciclos de processamento, ou seja, um ciclo para cada operação representada por um vértice da árvore. Por outro lado, com base na topologia da Árvore de Eliminação e dispondo de processamento paralelo, o número de ciclos de processamento que seriam necessário se reduz para 4.

A estrutura de dados utilizada com o objetivo de possibilitar a execução em paralelo das operações está descrita na Seção 5.2.2. A descrição detalhada de como foi feita a implementação computacional da Fatoração Numérica está presente na Seção 5.5.

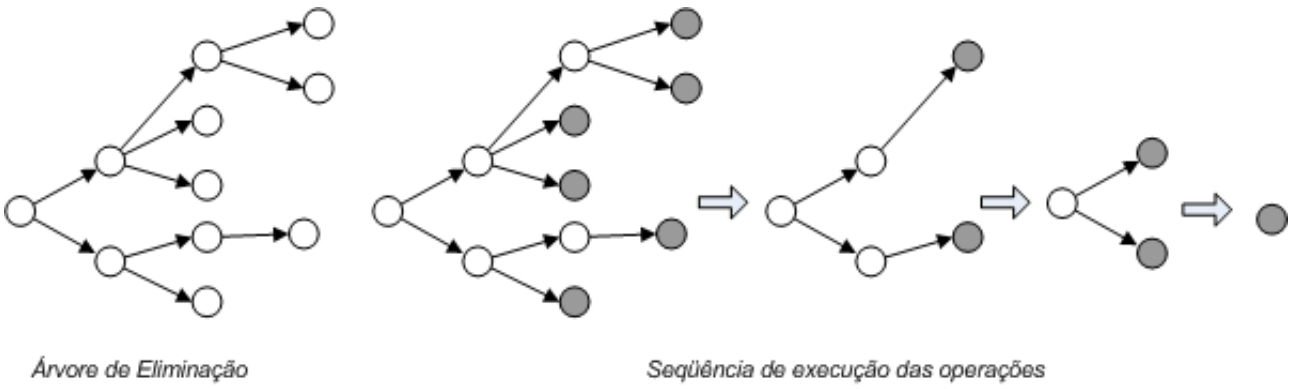


Figura 4.2: Ilustração da seqüência de execução de uma árvore de threads.

4.2 Exemplo de Inferência em uma Rede Bayesiana

Esta seção apresenta um exemplo passo-a-passo de inferência em uma Rede Bayesiana executado utilizando o algoritmo para a paralelização do procedimento de eliminação de variáveis proposto na seção anterior. Conforme descrito no algoritmo, inicialmente executa-se a Fatoração Simbólica para a determinação da ordem de eliminação das variáveis e a construção da Árvore de Eliminação. Estas duas estruturas de dados são posteriormente utilizadas para otimizar o Fatoração Numérica.

Procurou-se explicitar de forma didática cada uma das etapas envolvidas no processo. Esta mesma sequência de passos será executado pela aplicação computacional cuja implementação está descrita no Capítulo 5.

Considere a Rede Bayesiana representada na Figura 4.3, da qual as distribuições de probabilidades estão especificadas pelo conjunto de tabelas da Figura 4.4. Considere ainda que o objetivo seja calcular a distribuição marginal a posteriori da variável I observados os estados das variáveis $A = A1$, $H = H2$ e $J = J3$, ou seja, deseja-se obter $P(I|A = A1, H = H2, J = J3)$.

Para possibilitar o processamento computacional, as informações necessárias foram codificadas em arquivos texto no formato XML. Os Apêndices A.1, A.2 e A.3 apresentam respectivamente a listagem dos arquivos com a definição da Rede Bayesiana no formato XMLBIF, a definição da Consulta e da Evidência nos formatos definidos na Seção 5.3.

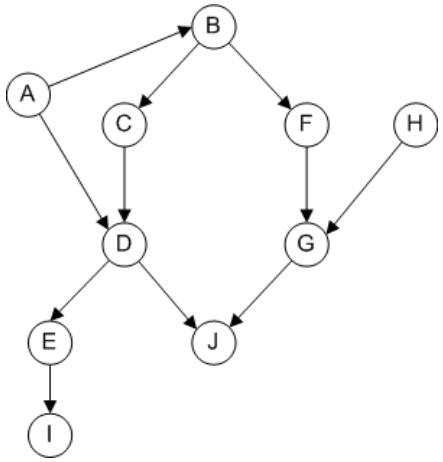


Figura 4.3: Exemplo de Rede Bayesiana.

P(A)	
A1	0,60
A2	0,09
A3	0,31

P(B|A)	A1	A2	A3
B1	0,35	0,04	0,48
B2	0,50	0,40	0,48
B3	0,15	0,56	0,04

P(C|B)	B1	B2	B3
C1	0,08	0,51	0,86
C2	0,92	0,49	0,14

P(D|A,C)	A1		A2		A3	
	C1	C2	C1	C2	C1	C2
D1	0,01	0,23	0,51	0,26	0,40	0,27
D2	0,47	0,05	0,23	0,32	0,46	0,38
D3	0,52	0,72	0,26	0,42	0,14	0,35

P(E|D)	D1	D2	D3
E1	0,38	0,92	0,12
E2	0,62	0,08	0,88

P(F|B)	B1	B2	B3
F1	0,79	0,43	0,45
F2	0,21	0,57	0,55

P(G|F,H)	F1			F2		
	H1	H2	H3	H1	H2	H3
G1	0,03	0,39	0,48	0,96	0,51	0,60
G2	0,97	0,61	0,52	0,04	0,49	0,40

P(H)	
H1	0,50
H2	0,03
H3	0,47

P(I|E)	E1	E2
I1	0,42	0,79
I2	0,58	0,21

P(J|D,G)	D1		D2		D3	
	G1	G2	G1	G2	G1	G2
J1	0,43	0,21	0,07	0,14	0,33	0,35
J2	0,32	0,53	0,39	0,68	0,40	0,56
J3	0,25	0,26	0,54	0,18	0,27	0,09

Figura 4.4: Distribuições de probabilidade que definem a Rede Bayesiana.

A Rede Bayesiana codifica uma única distribuição conjunta de probabilidades para o conjunto de variáveis aleatórias $\mathbf{X} = \{A, B, C, D, E, F, G, H, I, J\}$. A topologia do grafo da Rede Bayesiana define as relações de dependência entre as variáveis, permite especificar as distribuições de probabilidades envolvidas e, por conseguinte, os potenciais envolvidos: $P(A) = \phi_A(A)$, $P(B|A) = \phi_B(B, A)$, $P(C|B) = \phi_C(C, B)$, $P(D|A, C) = \phi_D(D, A, C)$, $P(E|D) = \phi_E(E, D)$, $P(F|B) = \phi_F(F, B)$, $P(G|F, H) = \phi_G(G, F, H)$, $P(H) = \phi_H(H)$, $P(I|E) = \phi_I(I, E)$, $P(J|D, G) = \phi_J(J, D, G)$.

Com base na Equação 2.7 distribuição codificada pelo rede deste exemplo pode ser calculada por:

$$P(\mathbf{X}) = P(A).P(B|A).P(C|B).P(D|A,C).P(E|D).P(F|B).P(G|F,H).P(H).P(I|E).P(J|D,G)$$

Utilizando a notação de potencial para mesma distribuição:

$$P(\mathbf{X}) = \phi_A.\phi_B.\phi_C.\phi_D.\phi_E.\phi_F.\phi_G.\phi_H.\phi_I.\phi_J$$

O conjunto das variáveis questionadas que define a Consulta será $\mathbf{X}_q = \{I\}$, e o conjunto das variáveis observadas que compõe a Evidência será $\mathbf{X}_E = \{A, H, J\}$. A partir da análise da Evidência montam-se os *findings* correspondentes às variáveis observadas, no caso do exemplo têm-se: $\underline{e}_A = \{1, 0, 0\}$, $\underline{e}_H = \{0, 1, 0\}$ e $\underline{e}_J = \{0, 0, 1\}$. O conjunto de informações que compõe a Evidência será indicado por $\mathbf{e}$.

Considerando agora a Evidência, a distribuição conjunta de probabilidade será dada por:

$$P(\mathbf{X}, \mathbf{e}) = \phi_A.\underline{e}_A.\phi_B.\phi_C.\phi_D.\phi_E.\phi_F.\phi_G.\phi_H.\underline{e}_H.\phi_I.\phi_J.\underline{e}_J \tag{4.1}$$

Para se calcular a distribuição marginal a posteriori da variável I é necessário marginalizar todas as variáveis de $P(\mathbf{X}, \mathbf{e})$, exceto I.

$$P(I, \mathbf{e}) = \sum_{A,B,C,D,E,F,G,H,J} \phi_A.\underline{e}_A.\phi_B.\phi_C.\phi_D.\phi_E.\phi_F.\phi_G.\phi_H.\underline{e}_H.\phi_I.\phi_J.\underline{e}_J \tag{4.2}$$

A distribuição desejada poderá então ser obtida normalizando o resultado da marginalização:

$$P(I|\mathbf{e}) = \frac{P(I, \mathbf{e})}{\sum_I P(I, \mathbf{e})} \tag{4.3}$$

O passo seguinte é a definição das variáveis estatisticamente relevantes para o problema, o que potencialmente poderia reduzir as dimensões do problema. Para tanto utiliza-se o algoritmo Bayes-Ball descrito na Seção 2.3.5. A Figura 4.5 apresenta o resultado da aplicação do algoritmo para o problema em questão, a partir da qual conclui-se que o conjunto das variáveis requisitadas será $\mathbf{X}_R = \{A, B, C, D, E, F, G, H, I, J\}$. Neste caso em particular, em função da Consulta que se deseja executar e da Evidência disponível, todas as variáveis serão requisitadas para a inferência.

A etapa seguinte inicia-se com a construção do Grafo Moral para a Rede Bayesiana utilizando apenas as variáveis presentes em $\mathbf{X}_R$. A Figura 4.6 apresenta o Grafo Moral com os *Moral links*, acrescentados para unir os vértices com filhos em comum, indicados em destaque.

Para a determinação da ordem de eliminação utiliza-se a Heurística de Gibbs/BEL. A Árvore BEL apresentada na Figura 4.7(a) foi obtida a partir do Grafo Moral e corresponde ao resultado final da heurística, ou seja, pela Heurística de Gibbs H é um vértice quase-periférico. Esta é, provavelmente,

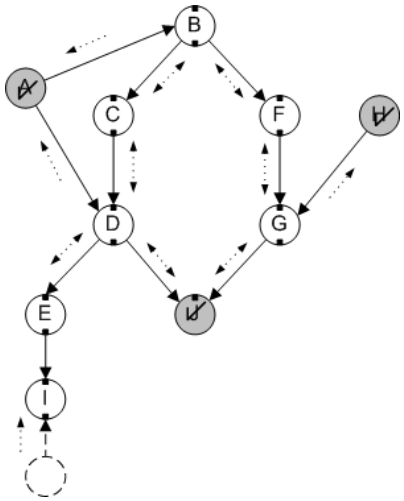


Figura 4.5: Bayes-Ball: Definição de $\mathbf{X}_R$

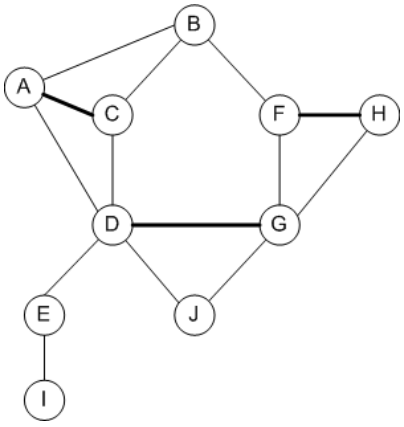


Figura 4.6: Grafo Moral

uma das Árvores BEL possíveis de serem geradas que apresentam profundidade 5. Muitas vezes apenas a profundidade da Árvore BEL não é suficiente para definir um nó quase-periférico pois nós distintos podem gerar Árvores BEL com mesma profundidade. Portanto para escolher uma dentre as árvores de mesma profundidade são necessários critérios adicionais. Tais critérios, detalhados na Seção 5.4, consideram outras características além da profundidade para escolher a árvore capaz de gerar uma boa ordem de eliminação, que minimize o processamento numérico e possibilite a maior paralelização possível do mesmo.

A mesma figura indica os separadores S_1, S_2 e S_3 escolhidos como os vértices da Árvore de Dissecção, esta última representada na Figura 4.7(b).

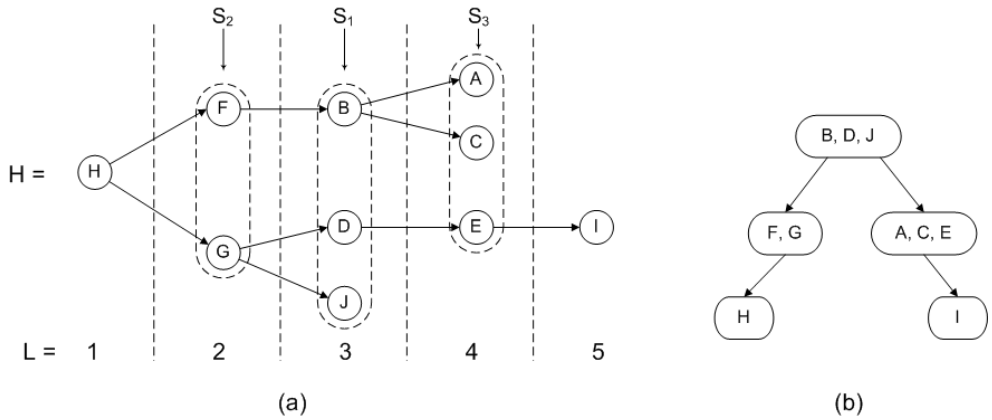


Figura 4.7: (a)Árvore BEL e (b) Árvore de Dissecção.

A enumeração dos vértices da Árvore de Disseção em pós-ordem gera a ordem $\tilde{q} = [\{H\}, \{F, G\}, \{A, E, C\}, \{B, J, D\}]$. A ordem de dissecção, q, que será a ordem de eliminação, é obtida substituindo em $\tilde{q}$ cada vértice da Árvore de Disseção pelos vértices correspondentes do Grafo Moral que o compõe. Tem-se portanto que uma possível ordem de eliminação obtida desta forma será: $q = [H, F, G, I, A, C, J, B, D]$.

Tomando como base a ordem de eliminação aplica-se à Equação 4.1 a propriedade distributiva dos potenciais, descrita no Seção 2.2 e obtêm-se a Equação 4.4. Procedendo desta forma evita-se trabalhar com todos os potenciais simultaneamente, o que poderia tornar o processamento numérico complexo e ineficiente.

$$P(I, \mathbf{e}) = \sum_D \sum_E \phi_I(I, E).\phi_E(E, D) \sum_B \sum_C \phi_C(C, B) \sum_A \phi_A(A).\underline{e}_A.\phi_B(B, A).$$
$$.\phi_D(D, A, C) \sum_J \sum_G \phi_J(J, D, G).\underline{e}_J \sum_F \phi_F(F, B) \sum_H \phi_G(G, F, H).\phi_H(H).\underline{e}_H \qquad (4.4)$$

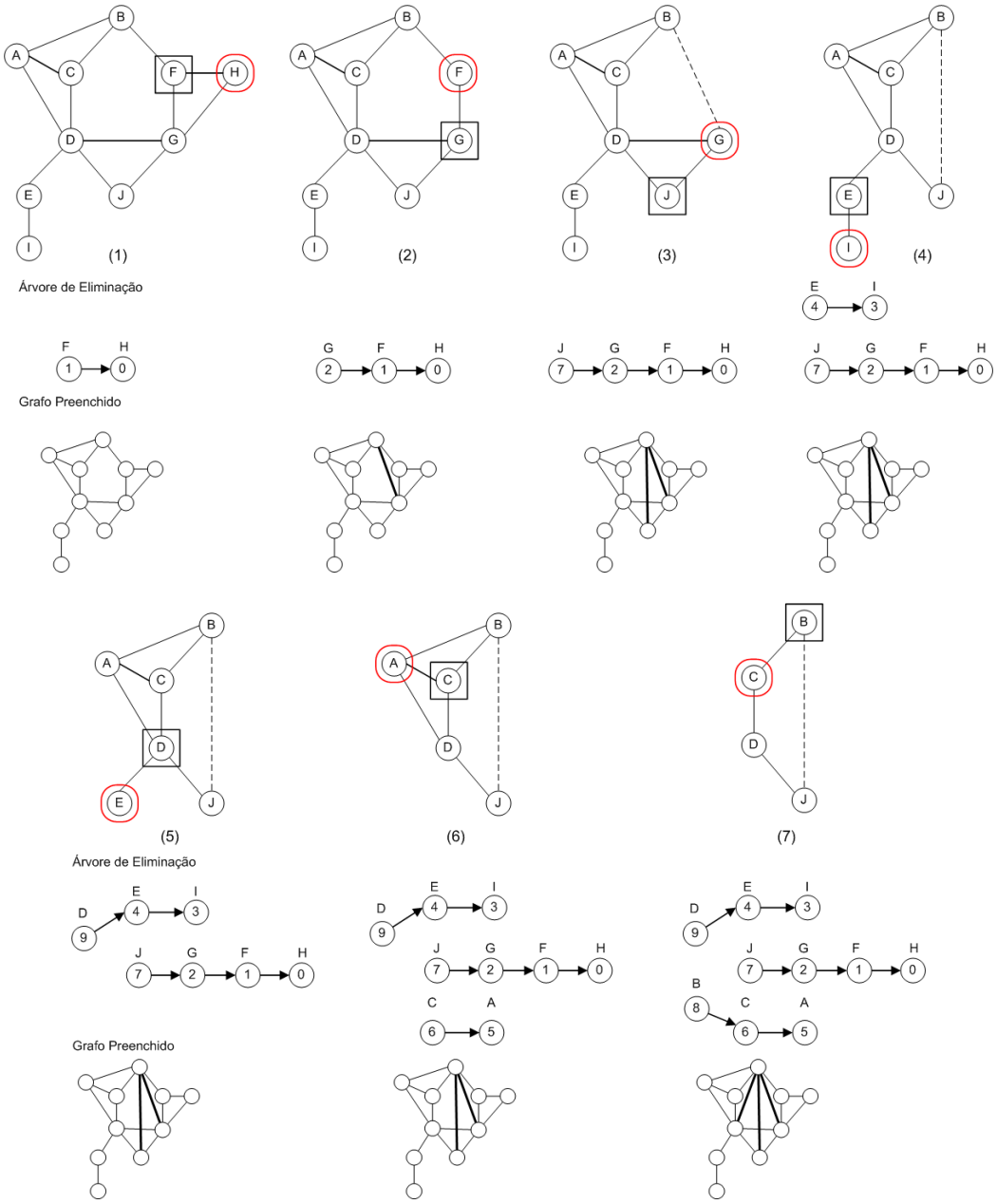
O cálculo de $P(I, \mathbf{e})$ poderia portanto ser feito sequencialmente iniciando-se com o cálculo de $\phi'_H = \sum_H \phi_G(G, F, H).\phi_H(H).\underline{e}_H$. Em seguida multiplica-se $\phi'_H(G, F)$ por $\phi_F(F, B)$ para cacular

$\phi'_F = \sum_F \phi_F(F,B).\phi'_H(G,F)$. O resultado ϕ'_F seria então multiplicado por $(\phi_J.\underline{e}_J)$ e assim por diante. No entanto, devido a propriedade distributiva dos potenciais algumas das operações de eliminação poderiam ser feitas simultaneamente, pois seriam independentes entre si. Como por exemplo a marginalização da variável A poderia ser feita paralelamente a marginalização da variável H uma vez que reciprocamente uma variável não pertence ao domínio do potencial da outra. Uma forma computacionalmente eficiente de se identificar tais relações de dependência entre as operações é construir a Árvore de Eliminação a partir da eliminação dos vértice do Grafo Moral.

A construção da Árvore de Eliminação realizada a partir da eliminação dos vértices do Grafo Moral é o principal processo da etapa de Fatoração Simbólica. Para tanto, seguindo a ordem de eliminação faz-se a eliminação dos vértices do Grafo Moral adotando-se, por uma questão de eficiência computacional, o algoritmo de eliminação simplificada descrito na Seção 2.1. A Figura 4.8 a presenta o processo de eliminação dos vértices. O vértice circunscrito no círculo é o vértice a ser eliminado em cada etapa e o vértice inscrito no quadrado é vizinho mais próximo a ser eliminado, sobre o qual, de acordo com o algoritmo da eliminação simplificada, quando necessário incidirão os arcos de preenchimento.

À medida que os vértices são eliminados definem-se as relações de dependência entre cada uma das operações de eliminação com base nas quais constrói-se a Árvore de Eliminação. Vértices cujas eliminações são independentes, e que portanto poderão ser eliminados em operações simultâneas ou paralelamente, são vértices entre os quais não existe um arco no Grafo Moral e ao longo de todo o processo de eliminação não foram conectados por um arco de preenchimento. A Figura 4.9 apresenta a Árvore de Eliminação. No interior de cada vértice da árvore está a ordem na qual o mesmo será eliminado e sobre o vértice está indicada a variável ou o vértice correspondente no Grafo Moral.

A construção da Árvore de Eliminação sistematiza a identificação da relação entre as operações algébrica envolvidas no processo de eliminação, ou marginalização das variáveis. Outra forma de se identificar tais relações de dependência seria analisando os domínios dos potenciais envolvidos nas operações da Equação 4.4.


(1)
(2)
(3)
(4)
Árvore de Eliminação
Grafo Preenchido
(5)
(6)
(7)
Árvore de Eliminação
Grafo Preenchido

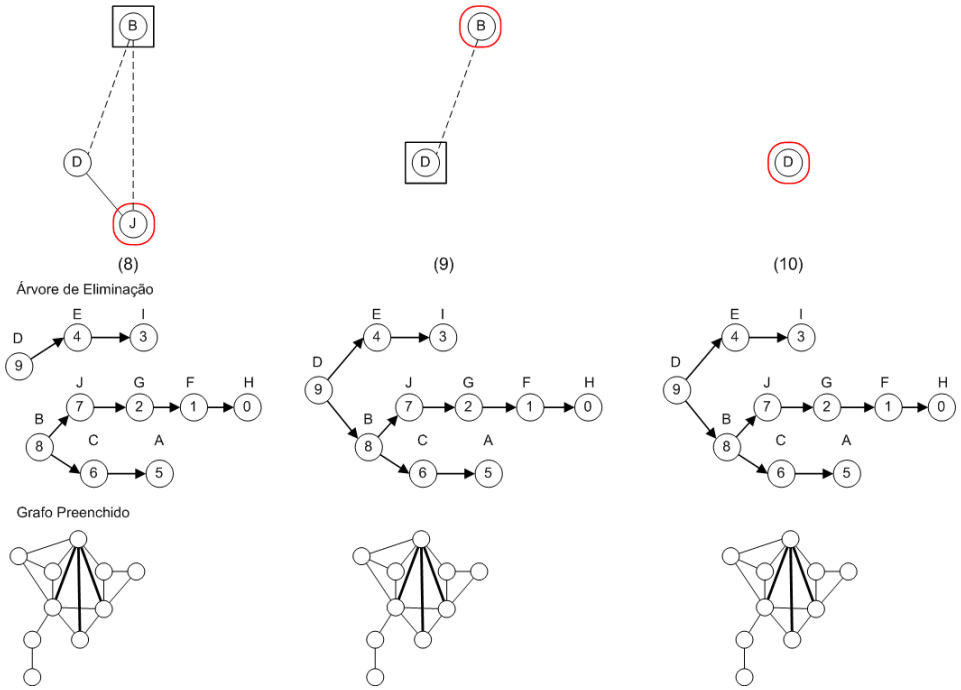


Figura 4.8: Eliminação dos vértices do Grafo Moral.

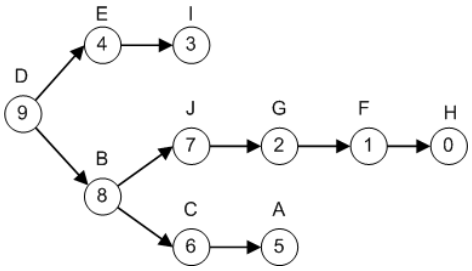


Figura 4.9: Árvore de Eliminação.

Definida a Árvore de Eliminação fica simples identificar quais operações indicadas na Equação 4.4 podem ser feitas em paralelo. Conforme feito anteriormente, este conjunto de operações está representado na Figura 4.10 por uma árvore a qual, não por acaso, apresenta a mesma estrutura da Árvore de Eliminação.

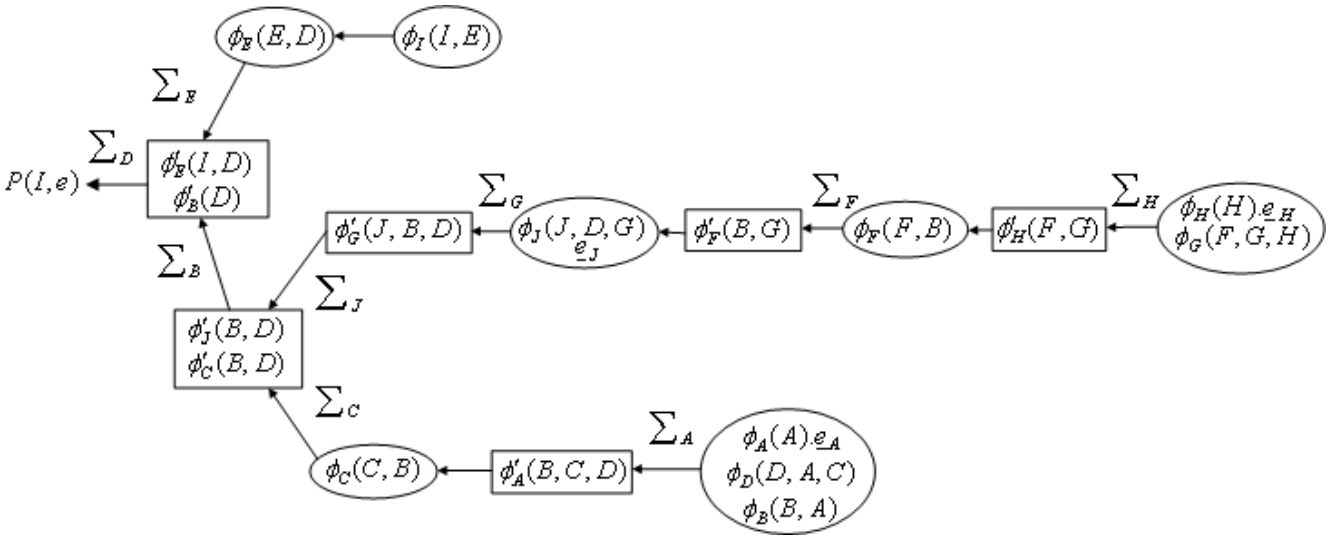


Figura 4.10: Processo de marginalização da variável I.

Como subproduto da etapa de Fatoração Simbólica obtêm-se o grafo preenchido, representado na Figura 4.11. Os arcos representados em linhas descontínuas são arcos de preenchimento adicionados ao longo do processo de eliminação dos vértices do Grafo Moral.

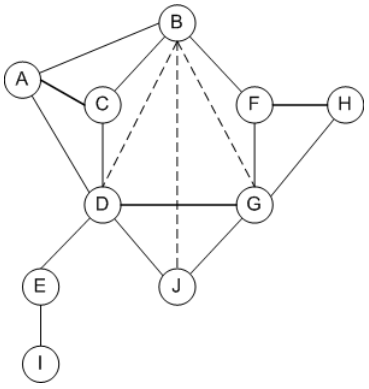


Figura 4.11: Grafo Preenchido com os arcos de preenchimento.

Na etapa de Fatoração Simbólica determinou-se a ordem de eliminação e a Árvore de Eliminação. As informações contidas nestas duas estruturas de dados permitirão otimizar e paralelizar as operações numéricas na etapa de Fatoração Numérica. Do ponto de vista computacional, conforme

descrito em detalhes na Seção 4.1.2, com as informações obtidas na etapa simbólica é possível determinar quais operações poderão ser executadas em paralelo e alocar de forma estática os recursos computacionais que serão necessários para realizá-las.

Considerando que cada vértice da Árvore de Eliminação pode ser interpretado como uma operação de eliminação de uma variável, a Figura 4.12 mostra a seqüência de execução dos vértices da Árvore de Eliminação. O processamento inicia-se nos vértices da extremidade que não possuem descendentes. À medida que o processamento avança, os demais vértices aguardam até que seu último filho tenha sido executado para então iniciar o próprio processamento.

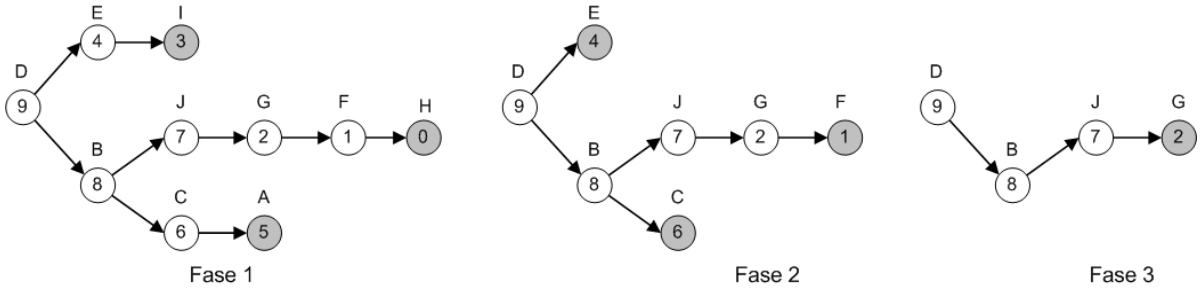


Figura 4.12: Execução da Árvore de Threads.

Na Figura 4.12, na seqüência de grafos da esquerda para a direita, estão assinalados os vértices que *podem* ser executados em paralelo. Em função da diferença de complexidade das operações, ramos independentes poderão ser executados em diferentes velocidades. O caminho crítico, que determinará o tempo necessário para o processo como um todo, será o ramo mais lento.

As operações numéricas com os potenciais serão apresentadas a seguir de acordo com a representação esquemática da Figura 4.10, simulando a ordem de processamento definida pela Árvore de Eliminação, ou seja, as operações em um mesmo ciclo de processamento estariam sendo executadas paralelamente.

Ciclo 1: Variáveis H, A e I.

- Cálculo de $\phi'_H(F,G) = \sum_H \phi_H(H).\underline{e}_H.\phi_G(F,G,H)$

$\phi_H(H).\underline{e}_H$

$H1$	-
$H2$	0.030
$H3$	-

$\phi_H(H).\underline{e}_H.\phi_G(F,G,H)$

	$F1$			$F2$		
	$H1$	$H2$	$H3$	$H1$	$H2$	$H3$
$G1$	-	0.01170	-	-	0.01530	-
$G2$	-	0.01830	-	-	0.01470	-

$\sum_H \phi_H(H).\underline{e}_H.\phi_G(F,G,H)$

	$F1$	$F2$
$G1$	0,01170	0,01530
$G2$	0,01830	0,01470

- Cálculo de $\phi'_A(B,C,D) = \sum_A \phi_A(A).\underline{e}_A.\phi_D(D,A,C).\phi_B(B,A)$

$\phi_A(A).\underline{e}_A$

$A1$	0.600
$A2$	-
$A3$	-

$\phi_B(B,A).\phi_A(A).\underline{e}_A$

	$A1$	$A2$	$A3$
$B1$	0.21000	-	-
$B2$	0.30000	-	-
$B3$	0.09000	-	-

$\sum_A \phi_A(A).\underline{e}_A.\phi_D(D,A,C).\phi_B(B,A)$

	$B1$			$B1$			$B3$		
	$D1$	$D2$	$D3$	$D1$	$D2$	$D3$	$D1$	$D2$	$D3$
$C1$	0.00210	0.09870	0.10920	0.00300	0.14100	0.15600	0.00090	0.04230	0.04680
$C2$	0.04830	0.01050	0.15120	0.06900	0.01500	0.21600	0.02070	0.00450	0.06480

Como o objetivo é calcular $P(I,(e))$ a operação de marginalização da variável I para fora o potencial $\phi_I(I,E)$ não deve ser executada.

Ciclo 2: Variáveis F, C, E.

- Cálculo de $\phi'_F(B,G) = \sum_F \phi_F(F,B).\phi_H(F,G)$

$\sum_F \phi_F(F,B).\phi_H(F,G)$

	$B1$	$B2$	$B3$
$G1$	0.01246	0.01375	0.01368
$G2$	0.01754	0.01625	0.01632

- Cálculo de $\phi'_C(B,D) = \sum_C \phi_C(C,B).\phi'_A(B,C,D)$

$\sum_F \phi_F(F,B).\phi_H(F,G)$

	$B1$	$B2$	$B3$
$D1$	0.04460	0.03534	0.00367
$D2$	0.01756	0.07926	0.03701
$D3$	0.14784	0.18540	0.04932

- Cálculo de $\phi'_E(D,I) = \sum_E \phi_E(E,D).\phi_I(I,E)$

$\sum_E \phi_E(E,D).\phi_I(I,E)$

	$D1$	$D2$	$D3$
$I1$	0.64940	0.44960	0.74560
$I2$	0.35060	0.55040	0.25440

Concluído o Ciclo 2, os demais deverão necessariamente ser executadas sequencialmente pois,

como é possível observar analisando a Árvore de Eliminação, existe uma relação de dependência entre as operações restantes e portanto não podem ser executadas em paralelo.

Ciclo 3: Variável G.

- Cálculo de $\phi'_G(B,G) = \sum_G \phi_J(J,D,G).\underline{e}_J.\phi'_F(B,G)$

$\phi_J(J,D,G).\underline{e}_J$

	$D1$		$D2$		$D3$	
	$G1$	$G2$	$G1$	$G2$	$G1$	$G2$
$J1$	-	-	-	-	-	-
$J2$	-	-	-	-	-	-
$J3$	0.250	0.260	0.540	0.180	0.270	0.090

$\sum_G \phi_J(J,D,G).\underline{e}_J.\phi'_F(B,G)$

	$D1$			$D1$			$D3$		
	$B1$	$B2$	$B3$	$B1$	$B2$	$B3$	$B1$	$B2$	$B3$
$J1$	-	-	-	-	-	-	-	-	-
$J2$	-	-	-	-	-	-	-	-	-
$J3$	0.00768	0.00766	0.00766	0.00988	0.01035	0.01032	0.00494	0.00518	0.00516

Ciclo 4: Variável J.

- Cálculo de $\phi'_J(B,D) = \sum_J \phi'_G(B,D,J)$

$\sum_J \phi'_G(B,D,J)$

	$B1$	$B2$	$B3$
$D1$	0.00768	0.00766	0.00766
$D2$	0.00988	0.01035	0.01032
$D3$	0.00494	0.00518	0.00516

Ciclo 5: Variável B.

- Cálculo de $\phi'_B(D) = \sum_B \phi'_C(B,D).\phi'_J(B,D)$

$\sum_B \phi'_C(B,D).\phi'_J(B,D)$

$D1$	0.00064
$D2$	0.00138
$D3$	0.00194

Ciclo 6: Variável D.

- Cálculo de $\phi'_D(I) = \sum_D \phi'_B(D).\phi'_E(D,I)$

	$\sum_D \phi'_B(D).\phi'_E(D,I)$
$I1$	0.00249
$I2$	0.00148

Normalizando o resultado, obtêm-se a distribuição $P(I|\mathbf{e})$ desejada

	$P(I\|\mathbf{e})$
$I1$	0.62723
$I2$	0.37277

Com este exemplo procurou-se mostrar passo-a-passo a seqüência de execução das etapas de uma aplicação típica de inferência utilizando o algoritmo de paralelização proposto. Convêm reiterar que o conteúdo numérico e os componentes potenciais envolvidos foram utilizados apenas na última etapa, Fase Numérica. Este fato é relevante pois o presente trabalho procurou exatamente demonstrar que a análise da estrutura da rede permite em um etapa anterior ao processamento numérico, aqui denominada de Fatoração Simbólica, determinar quais operações serão executadas, em que ordem e quais poderão ser feitas paralelamente. Estas informações permitiram otimizar o processamento numérico.

Capítulo 5

Implementação Computacional

5.1 Introdução

Com o intuito de verificar a viabilidade técnica da proposta de aplicação de métodos de fatoração de matrizes esparsas para inferência em Redes Bayesianas criou-se em linguagem C um conjunto de bibliotecas que implementam os métodos e os algoritmos descritos nas seções anteriores. O conjunto de bibliotecas foi desenvolvido para possibilitar a criação de aplicações eficientes através da separação completa do procedimento de inferência em duas principais etapas: uma etapa simbólica (*Fatoração Simbólica*) e uma etapa numérica (*Fatoração Numérica).* A Fatoração Simbólica gera estruturas de dados que permitem a otimização e a paralelização do processamento numérico executado na Fatoração Numérica.

Seguem abaixo as principais premissas que nortearam o projeto e o desenvolvimento das bibliotecas:

- As bibliotecas devem ser, na medida do possível, genéricas, flexíveis e com uma arquitetura que permita o desenvolvimento de aplicações com diferentes propósitos e/ou otimizadas para uma determinada etapa do processo de inferência.
- Deve haver uma separação clara e bem definida dos diferentes procedimentos e algoritmos, de forma que as bibliotecas possam ser utilizadas independentemente, ou seja, a utilização de uma biblioteca não pode estar vinculada a outra biblioteca do conjunto. Por este motivo procurou-se

isolar os procedimentos para que possam ser desenvolvidas ou utilizadas formas alternativas de realizar uma determinada tarefa, inclusive utilizando funções desenvolvidas por terceiros, sem inviabilizar adoção das demais rotinas que integram o conjunto de bibliotecas.

- As bibliotecas devem encapsular a complexidade dos procedimentos por meio de interfaces que tornem mais fácil e intuitivo o desenvolvimento das aplicações e possibilitem que no futuro as bibliotecas de operações mais elementares, de mais baixo nível, possam ser reescritas ou substituídas mantendo a compatibilidade com as aplicações desenvolvidas até então.

- Para facilitar a integração com aplicações e bibliotecas de terceiros, bem como para utilizar bases de testes e repositórios de dados de domínio público, sempre que possível, devem ser utilizados padrões abertos e reconhecidamente adotados pela comunidade científica.

Em linhas gerais, e por motivos didáticos, pode-se considerar que uma aplicação típica que utilize as funcionalidades disponíveis nas bibliotecas implementadas pode ser dividida em 3 etapas. Nas subseções que se seguem as etapas serão detalhadas uma a uma, mas sucintamente podemos descrever as mesmas por seus principais processos como:

Etapa 1 Entrada de dados:

1.1 Leitura dos arquivo que define a Rede Bayesiana

1.2 Leitura dos arquivo que define a Consulta (*Query*);

1.3 Leitura dos arquivo que define as variáveis observadas (*Evidence*);

1.4 Determinação as Variáveis Requisitadas (*Requisite Variables*) para processar a inferência.

Etapa 2 Fatoração Simbólica:

2.1 Construção do Grafo Moral;

2.2 Determinação da ordem de eliminação;

2.3 Fatoração Simbólica: Construção da Árvore de Eliminação e do Grafo Preenchido

Etapa 3 Processamento Numérico:

3.1 Preparação e inicialização das estruturas para executar o processamento numérico com os dados dos potenciais.;

3.2 Construção e (Re)Inicialização da Árvore de Threads para executar o processamento numérico;

3.3 Execução da árvore de threads para o cálculo da inferência utilizando o Método de Eliminação de Variáveis.

A descrição sequencial das etapas representa uma aplicação típica de inferência, na qual as etapas seriam executadas seqüencialmente. No entanto dependendo do propósito da aplicação podem ser adotadas configurações alternativas para os processos. A título de exemplo, considere uma aplicação cujo objetivo seja executar diversas inferências para uma mesma Rede Bayesiana e uma mesma Consulta, apenas variando a Evidência. Após executar as Etapas 1 e 2 seria possível executar sucessivas vezes a Etapa 3 apenas atualizando a Evidência, ou seja, poderiam ser utilizadas as estruturas obtidas nas etapas anteriores sem a necessidade de executá-las novamente. Este procedimento pode ser aplicado para simulações.

5.2 Bibliotecas e Estruturas de Dados

5.2.1 Organização das Bibliotecas

Partindo-se das premissas mencionadas na introdução do capítulo, as bibliotecas ou pacotes (*packages*), foram organizadas hierarquicamente conforme o diagrama indicado na Figura 5.1. As mesmas foram desenvolvidas seguindo uma linha lógica de isolar a implementação de métodos e algoritmos aplicáveis em álgebra linear computacional dos procedimentos utilizados para inferência em Redes Bayesianas. Neste sentido, na organização dos arquivos procurou-se delimitar claramente os métodos que executam operações elementares, como a manipulação de grafos, e os métodos específicos para de Redes Bayesianas. Os métodos aplicáveis a álgebra linear computacional foram isolados em um conjunto de arquivos genéricos que poderiam ser utilizados tanto para implementar uma aplicação específica de álgebra linear computacional, como por exemplo, a fatoração de Cholesky de matrizes esparsas, quanto para compor um conjunto de operações elementares para inferência em Redes Bayesianas.

Nível 1 Aplicação ou Programa Principal - A aplicação não é propriamente um arquivo da biblioteca, ela foi indicada no diagrama para facilitar o entendimento. A aplicação terá que ser desenvolvida pelo programador que fará uso do conjunto de bibliotecas.

Nível 2 Bibliotecas e Interfaces específicas para Redes Bayesianas - Neste nível estão os pacotes ou bibliotecas que foram efetivamente utilizados e incluídos nos códigos das aplicações que serão desenvolvidas. As principais funções deste conjunto de arquivos são:

i. Definir a estrutura de dados que irá representar internamente uma Rede Bayesiana;
ii. Funcionar como uma interface, dentro do possível, imutável para abstrair e isolar das aplicações as bibliotecas e funcionalidades de baixo nível, ou seja, mais elementares.

Desta forma, no futuro, caso seja feita a otimização ou a alteração do método utilizado para implementar um operação mais elementar, se a interface permanecer inalterada as aplicações, que idealmente devem invocar apenas os métodos destas interfaces, não precisaram ser reescritas.

BNMain: Biblioteca que define a estrutura de dados que representa uma Rede Bayesiana e implementa funções para contruí-la, destruí-la e obter informações da mesma.
BNUtils: Interface/Biblioteca que implementa funções genéricas e úteis para o processamento de uma Rede Bayesiana, como por exemplo, a construção do Grafo Moral, a fatoração simbólica ou a interface para o algoritmo de determinação das variáveis requisitadas.
BNVarElimination: Biblioteca que implementa o algoritmo de inferência utilizando o método de eliminação de variáveis.
BNLoader: Biblioteca com as funções para carregar os arquivos de definição da Rede Bayesiana, da Consulta e da Evidência.

Nível 3 Bibliotecas específicas para Redes Bayesianas - Neste nível estão colocados os arquivos que implementam funcionalidades específicas para Redes Bayesianas, mas que por serem operações intermediárias e utilizarem algoritmos específicos com funções bem definidas, poderão no futuro ser substituídas ou otimizadas.

bayesball: Implementação o algoritmo Bayes-Ball, descrito na Seção 2.3.5, para a determinação das variáveis requisitadas para executar um procedimento de inferência dado um conjunto de variáveis observadas.

Nível 4 Biblioteca para manipulação de potenciais.

potential: Define as estruturas que representam internamente uma variável e um potencial. Implementa um conjunto de operações básicas envolvendo as duas estruturas como multiplicação e marginalização de uma variável de um potencial.

Nível 5 Bibliotecas para operações com grafos - Implementam operações e heurísticas com grafos utilizadas em álgebra linear computacional.

graphtools: Implementação da função de construção do Grafo Moral e da Fatoração Simbólica de grafos genéricos.
gibbs: Executa a Heurística de BEL/Gibbs para a determinação da ordem de eliminação dos nós de um grafo.

Nível 6 Bibliotecas e estruturas de dados genéricas - Definições e operações de estruturas genéricas de dados utilizadas pelos níveis superiores.

graph: Define a estrutura e implementa operações com grafos.
threadtree: Define a estrutura e implementa as funções para utilização das árvores de threads para o processamento paralelo.
singlelinkedlist: Define a estrutura e implementa operações com lista ligadas simples.
hash: Implementa operações com *hash*.

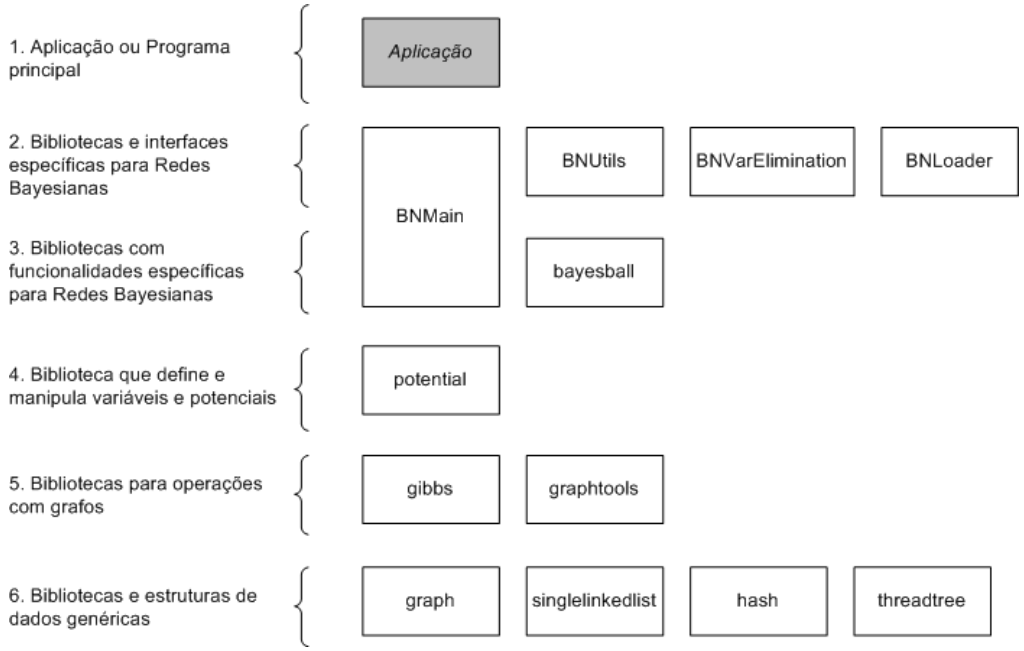


Figura 5.1: Organização dos arquivos e bibliotecas.

Da maneira como o conjunto de bibliotecas foi implementado a estrutura de dados que representa uma Rede Bayesiana está definida no arquivo *BNMain*. Esta estrutura é conhecida e utilizada apenas

pelas bibliotecas e interfaces no Níveis 1, 2 e 3, nas quais estão definidas estruturas e implementadas as funções específicas para trabalhar com Redes Bayesianas. Para garantir o grau de abstração desejado, as bibliotecas dos níveis mais baixos não conhecem a estrutura de dados que representa a Rede Bayesiana, mas apenas seus componentes, ou seja, o grafo e os conjuntos de variáveis e potenciais.

5.2.2 Principais Estruturas de Dados

As principais estruturas de dados, bem como seus componentes e a forma como se relacionam entre si, estão representadas nas Figuras 5.2 a 5.7 e descritas brevemente nesta seção. A descrição aqui apresentada limita-se a necessária para entender a maneira como foram implementadas as bibliotecas. Uma descrição mais detalhada está disponível na documentação de desenvolvimento.

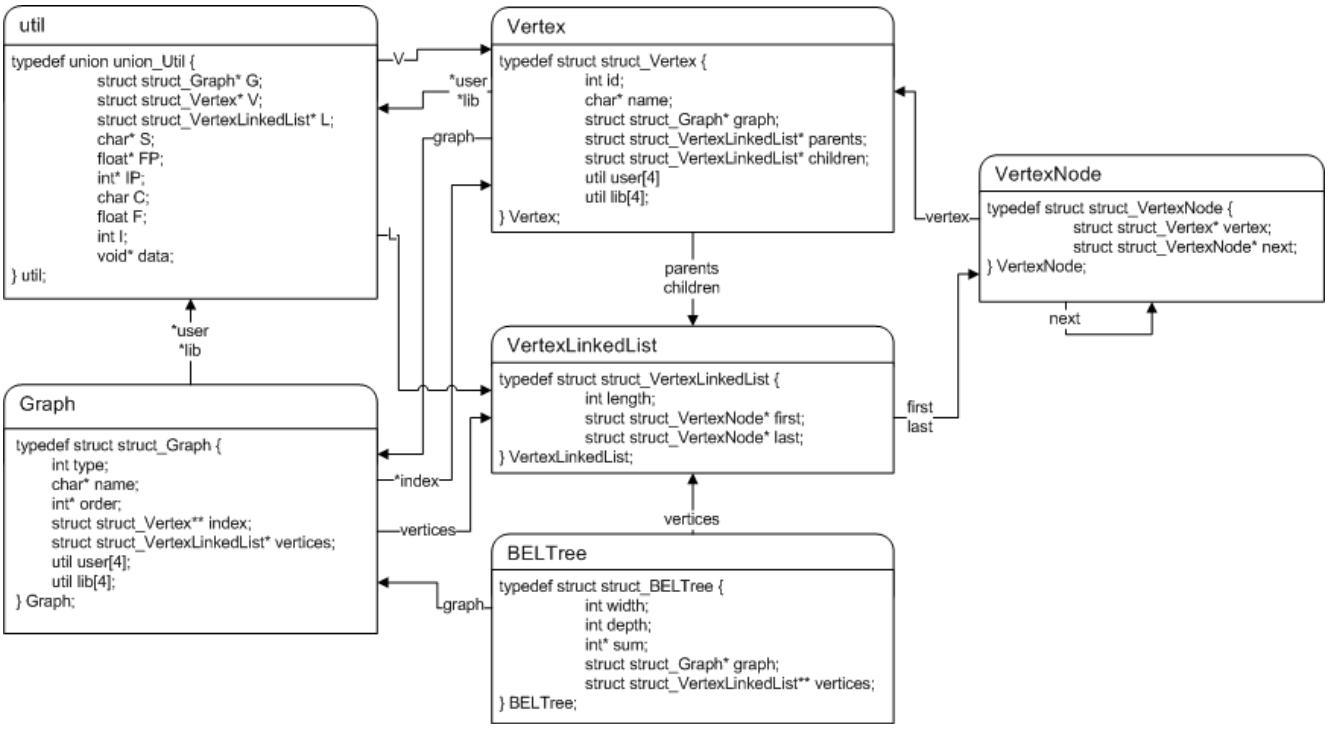


Figura 5.2: Principais estruturas de dados utilizadas para definição e manipulação de grafos.

A estrutura *Graph*, Figura 5.2, é estrutura utilizada em todas as bibliotecas que implementam

operações com grafos ou estruturas de que utilizam internamente um grafo, como por exemplo a Rede Bayesiana, descrita a seguir. Da forma como foi implementada, um único tipo de estrutura foi utilizado para representar grafos direcionados, como uma Rede Bayesiana, ou não direcionados, como um *Moral Graph*, de acordo com o valor do campo *type*. O campo mais importante da estrutura *Graph* é o campo *vertices*, o qual armazena os vértices do grafo em uma lista ligada simples. Para se criar um vértice utiliza-se a estrutura *Vertex*. Cada vértice possui a lista dos seus 'pais' (*parents*) e de seus 'filhos' (*children*) que definem os arcos do grafo. No caso de grafos não direcionados dois vértices conectados serão pais e filhos reciproca e simultaneamente. Internamente no grafo os vértices estão indexados pelos respectivos *id*s e portanto pode-se acessar qualquer vértice do grafo em tempo constante, apesar deles estarem armazenados em uma lista ligada. Os *id*s dos vértices devem ser únicos mas não necessariamente sequenciais. Por uma questão de performance e otimização do espaço necessário para armazenar e indexar os vértices é desejável que os *id*s sejam sequenciais.

Por uma questão de performance optou-se por utilizar uma estrutura específica para implementar uma lista ligada simples de vértices, a *VertexLinkedList*, com dois ponteiros, um para o primeiro e um para o último elemento da lista. Desta forma a inserção de um novo elemento na lista no início ou no final da lista pode ser feito em tempo constante.

Conforme descrito na Seção 3.5 para determinação da ordem de eliminação foram utilizadas as Heurísticas de Gibbs e da Busca Em Largura (BEL). Para implementá-las criou-se a estrutura *BELTree*, a qual internamente possui um grafo direcionado (*graph*) e um vetor de listas ligadas (*vertices*). O vetor de listas ligadas é utilizado para armazenar a lista de vértices de cada nível da árvore da BEL. Tecnicamente uma estrutura específica para armazenar esta informação é redundante, pois um campo da própria estrutura do vértice poderia armazená-la. No entanto, o vetor auxiliar adotado funciona como um indexador dos níveis da BEL, o que otimiza e simplifica a implementação dos algoritmos que a manipulam.

A estrutura *BELTree* possui ainda um campo *width* para armazenar a sua largura, um campo *depth* para armazenar a sua profundidade e um vetor *sum* cujo conteúdo correspondente a cada nível da árvore é a soma acumulada do número de vértices desde o primeiro nível. Conforme descrito na seção abaixo, estas informações serão utilizadas para escolher uma entre as diversas árvores geradas pela Heurística de Gibbs.

Coerente com a premissa de isolar as funcionalidades para viabilizar otimizações futuras é interessante observar que a *BELTree*, utilizada unicamente na Heurística de Gibbs para determinar a

ordem de eliminação dos vértices do grafo, existe exclusivamente dentro do arquivo *gibbs.h*. Assim, caso se deseje utilizar outro algoritmo para a definição da ordem de eliminação bastaria substituir esta biblioteca, mantendo os demais arquivos inalterados.

Duas importantes estruturas de dados genéricas auxiliares forma criadas. Uma lista ligada simples a *SingleLinkedList*, Figura 5.3, e uma estrutura de *Hash* para armazenar pares *chave-valor* onde *chave* é uma palavra e *valor* é um valor real, Figura 5.4. As implementações seguem o padrão usualmente descrito na teoria de ciência da computação para estes tipos de estrutura e portanto as mesmas não serão detalhadas neste documento. Convêm no entanto, mencionar duas características da biblioteca *SingleLinkedList*:

i. Para garantir que as operações de inserção e remoção de elementos da lista ligada serão seguras e atômicas, ou seja, *thread-safe*, foram criadas funções específicas que utilizam *mutexes* para sincronizar o acesso aos elementos da lista e desta forma evitam a concorrências entre as threads e garantem a singularidade na manipulação dos mesmos para a realização de operações críticas.
ii. Analogamente à lista ligada criada para armazenar vértices, a lista ligada genérica foi implementada com dois ponteiros, um para o primeiro e um para o último elemento da lista para que a inserção de um novo elemento no início ou no final da lista seja feito em tempo constante;

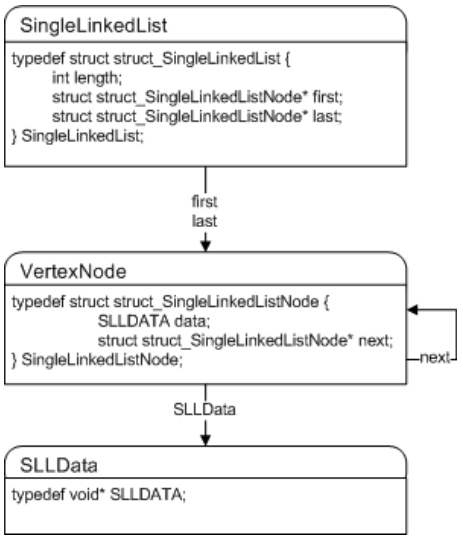


Figura 5.3: Estrutura de dados utilizada na implementação da Lista Ligada Simples.

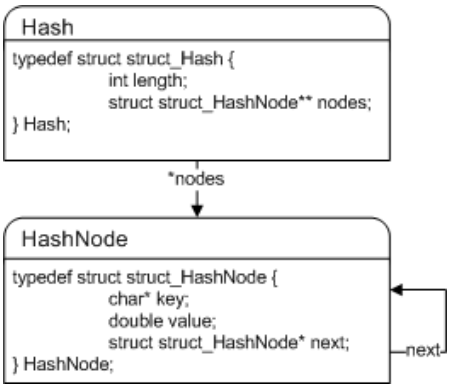


Figura 5.4: Estrutura de dados utilizada na implementação do *Hash*.

Para paralelizar o processamento das operações envolvidas na eliminação das variáveis foi definida uma estrutura que pode ser caracterizada como '*uma árvore de threads*' a qual denominou-se *ThreadTree*, Figura 5.5. Cada vértice da Árvore de Threads, um *ThreadVertex*, é uma thread que pode executar operações predefinidas paralelamente com outros *ThreadVertex*, desde que todos seus respectivos 'filhos' tenham sido executados. A topologia da Árvore de Threads é definida pela topologia da Árvore de Eliminação, esta última instanciada como um grafo direcional.

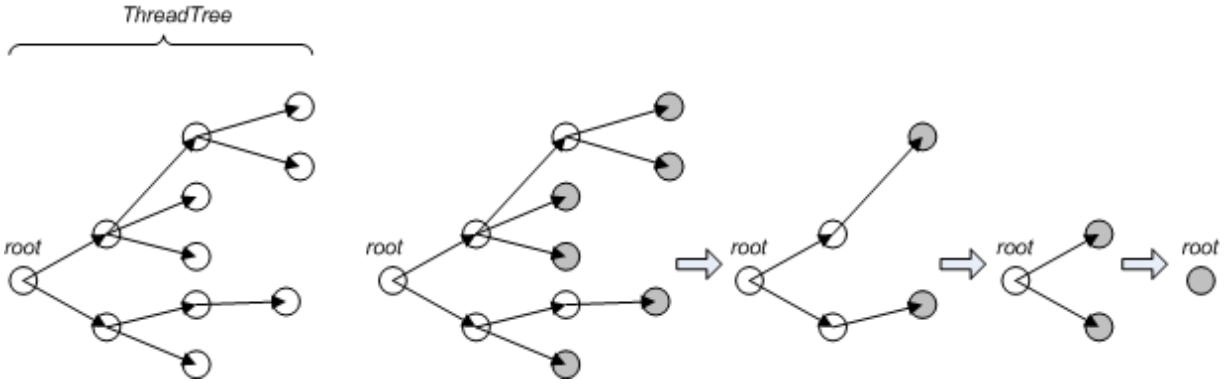


Figura 5.5: Ilustração da seqüência de execução de uma árvore de threads.

A Figura 5.5 exemplifica como ocorre o processamento dos *ThreadVertex* mostrando a seqüência de operações para se executar uma árvore de threads. Ao iniciar o processamento da árvore todas as threads são iniciadas, mas o processamento avança efetivamente apenas nos vértices mais periféricas que não possuem filhos. O processamento prossegue executando as threads das quais todos filhos

tenham sido finalizados e termina após a conclusão da execução da raiz (*root*) da árvore. Na figura, os vértices marcados em cada passo correspondem às threads que estão em execução.

As estruturas de dados utilizadas para implementar a *ThreadTree* estão representadas na Figura 5.6. A implementação da *ThreadTree* é genérica o suficiente para que possa ser utilizada em diversas aplicações nas quais existam relações de dependência entre as etapas necessárias para se resolver um problema.

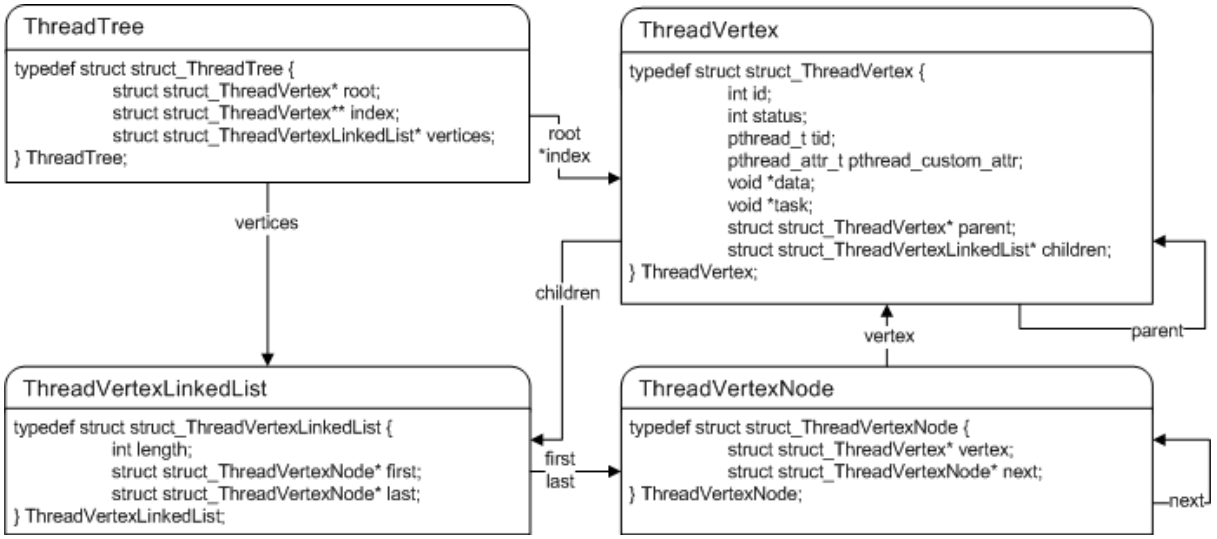


Figura 5.6: Estrutura de dados para controlar a execução das threads.

Uma Rede Bayesiana é instanciada utilizando a estrutura *BayesNet*, Figura 5.7. Os componentes principais desta estrutura são:

graph: grafo direcionado que define a topologia da rede;
variables: vetor de variáveis presentes na rede;
potential: vetor de potenciais que define as distribuições de probabilidade da rede.

Logicamente que o número de variáveis, o número de vértices e de potenciais iniciais da rede são iguais. Em função da forma como foi feita a implementação a *i-ésima* variável corresponde ao *i-ésimo* vértice do grafo e ao *i-ésimo* potencial. Optando por estas associações eliminou-se a necessidade de trabalhar com estruturas mais sofisticadas de indexação, como por exemplo *hashs*, e todas as indexações foram feitas com vetores numéricos, com o objetivo de ganhar performance.

Cada variável possui um número finito (*nstates*) de estados mutuamente exclusivos, e identificados

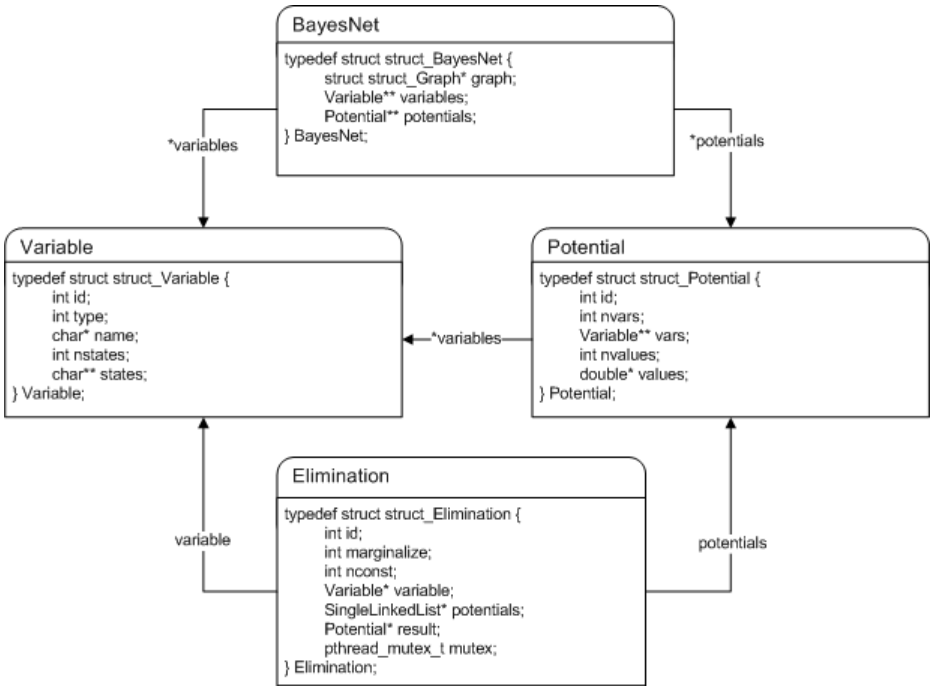


Figura 5.7: Principais estruturas de dados utilizadas para representar uma Rede Bayesiana e a operação de eliminação de uma variável.

por nomes armazenados em um vetor de *strings* (*states*). O potencial possui um vetor de variáveis (*variables*). Os valores relativos à distribuição de probabilidades das possíveis combinações de estados das variáveis de um potencial estão armazenados no vetor *values*, cuja dimensão é igual ao produto do número de estados possíveis de cada variável do potencial.

A operação de eliminação de uma variável da Rede Bayesiana é definida pela estrutura *Elimination*. Cada estrutura de eliminação esta associada a uma única variável (*variable*) e contem uma lista de potenciais que incluem a variável a ser eliminada (*potentials*). Dependendo da operação e da consulta que está sendo executada o resultado da eliminação poderá ou não ser marginalizado. Para indicar quando deve ou não ser marginalizado utiliza-se o campo *marginalize*.

A execução das operações de eliminação podem ocorrer simultaneamente em diferentes threads e os resultados de tais operações devem ser inseridos nas listas de potenciais dos pais das threads que as executaram. Por este motivo, cada estrutura Elimination possui um *mutex* utilizado para evitar problemas de concorrência entre as threads-filhas no momento da inserção dos resultados na

lista de potenciais das respectivas threads-pais. Este mutex é passado para as funções que fazem as operações críticas com as estruturas de dados que compõe a Elimination.

Algumas estruturas secundárias foram implementadas diretamente utilizando as estruturas mais elementares ou recursos disponíveis na própria linguagem. Devido à importância no processo de eliminação convêm mencionar:

Elimination Order: instanciada utilizando diretamente um vetor de números inteiros, no qual na *i-ésima* posição está o *id* da *i-ésima* variável a ser eliminada;

Moral Graph: para representar o Grafo Moral correspondente à Rede Bayesiana adotou-se diretamente a estrutura *Graph* para instanciar um grafo não direcionado;

Elimination Tree: a árvore de eliminação, cuja topologia explicita a relação de dependência entre os vértices do grafo e serve de base para a definição da topologia da árvore de threads, foi implementada utilizando um grafo direcionado.

Finding: para incorporar evidências no processo de inferência, conforme descrito no seção 2.3.4, as informações a priori sobre os estados das variáveis que compõe a Evidência são representadas por vetores de *0*s e *1*s. Seguindo a teoria, a estas estruturas denominou-se *findings*. A implementação foi feita com um *Potential* instanciado com uma única variável do qual a dimensão do vetor de valores é igual ao número de estados da variável. Neste vetor as posições tem valor *1* para o estado observado, ou os estados que passam a ser os únicos possíveis dadas as informações a priori, e todas as demais posições tem valores *0*.

A incorporação da Evidência utilizando *findings* torna a aplicação mais genérica pois possibilita que a mesma indique não apenas que a variável foi observada em um único estado, mas também para uma determinada realidade quais os estados possíveis caso um ou mais estados que antes eram possíveis se tornaram inviáveis. Além disso, a possibilidade de implementar um *finding* como um *Potential* torna a codificação mais simples pois a estrutura e as operações com a mesma já estão implementadas.

As desvantagens de utilizar estruturas como *finding* para representar a Evidência são: (i) a necessidade de se executar um produto de potenciais a mais no momento que estiver sendo feita a eliminação de variáveis; (ii) utilizar espaço adicional desnecessário por subutilizar a estrutura *Potential* para armazenar um potencial de uma única variável.

No entanto, o número de estados de uma variável é relativamente pequeno comparado com o número de estados da rede como um todo e o tamanho do *Potential* para representar o *finding* também será relativamente pequeno. Portanto, as vantagens listadas anteriormente aparentemente superam as desvantagens justificando a solução adotada. Se no futuro melhorias de performance se fizerem necessárias, uma alternativa será aprofundar a comparação entre das vantagens e as desvantagens da mesma.

5.3 Entrada de Dados

O primeiro procedimento para executar o processamento de inferência é obter as informações que definem: (i) a Rede Bayesiana; (ii) a Consulta que se deseja executar; e (iii) a Evidência observada. As aplicações desenvolvidas não incorporam nenhum tipo de iteração com o usuário. Todas as informações devem estar definidas antes de iniciar a aplicação por meio de arquivos textos em formatos predefinidos.

A Figura 5.8 apresenta um diagrama no qual estão representados os principais processos e as respectivas estruturas de dados resultantes nesta etapa.

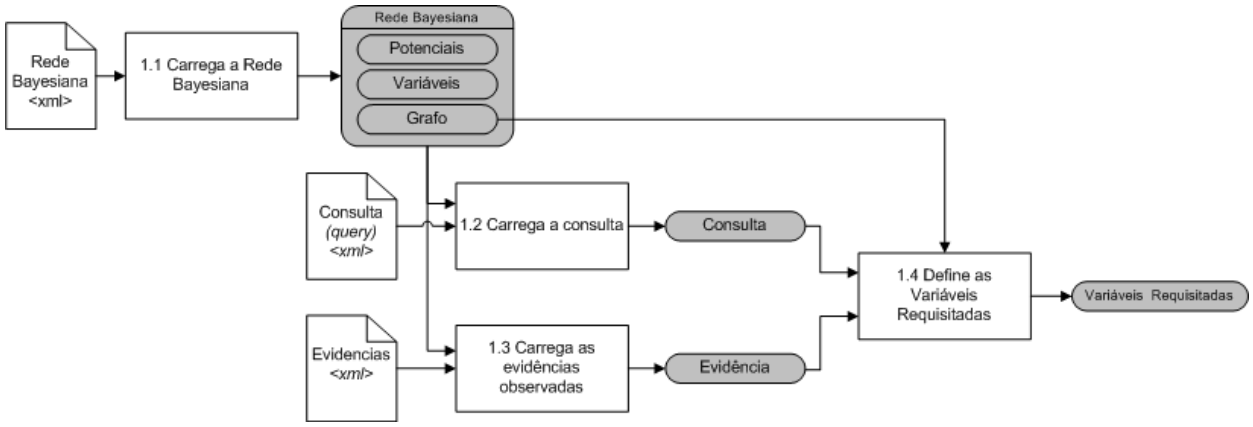


Figura 5.8: Etapa 01 - Entrada de dados.

Para definição da Rede Bayesiana, da Consulta (*Query*) e das variáveis observadas (*Evidence*) optou-se por utilizar arquivos XML (*Extensible Markup Language*). O formato XML está se tornando um padrão de mercado para arquivos de trocas de informação e diversas linguagens imple-

mentam interpretadores de arquivos XML. De forma sucinta, a linguagem XML utiliza um conjunto de palavras (*tags*) ou sentenças especiais, compreendidas entre "<" e ">" para atribuir significado ao conteúdo do documento. Uma especificação completa do formato XML pode ser encontrada no site: *http://www.w3.org/XML/*.

A adoção deste formato, é coerente com a premissa de utilizar padrões abertos e reconhecidamente adotados pela comunidade científica com o objetivo de facilitar a integração com aplicações e bibliotecas de terceiros e utilizar bases de testes e repositórios de dados de domínio público.

Para descrever uma Rede Bayesiana adotou-se o padrão XMLBIF (*XML-based BayesNets Interchange Format*). Este formato foi inicialmente desenvolvido por Fábio Cozman e é suportado por alguns dos principais aplicativos para processar Redes Bayesianas. O objetivo do formato XMLBIF é descrever grafos direcionados acíclicos aos quais podem ser associadas medidas de distribuições de probabilidade condicionais para variáveis discretas. Como se trata de um formato padrão o mesmo não será detalhado, neste documento. Um descrição completa do padrão pode ser encontrada em [4].

Para definir a Consulta e a Evidência envolvidas na inferência, como não foram encontrados padrões, optou-se por criar os formatos dos arquivos XML proprietários para descrevê-los. Ambos arquivos são bastante simples.

Para a especificação de uma *Query* são necessárias apenas 2 tags: *query* e *variable*. O elemento mais externo é a tag *query*. Dentro da tag *query* utiliza-se a tag *variable* para especificar as variáveis a serem consultadas. O nome da variável no interior da tag *variable* deve ser escrito de forma idêntica ao nome utilizado definição da Rede Bayesiana. A listagem abaixo representa uma *Query* da qual o resultado será uma distribuição conjunta das variáveis D e J.

```
<query>
    <variable>D</variable>
    <variable>J</variable>
</query>
```

O padrão definido para a especificação da *Evidence* utiliza 4 tags: *evidences*, *variable*, *name*, *value*. O elemento mais externo é a tag *evidences*. Dentro desta tag para cada variável que se deseja especificar um ou mais estados observados, ou possíveis, deve-se utilizar a tag *variable*. Para cada variável, é necessário indicar o nome da mesma utilizando a tag *name*, e um ou mais estados, utilizando a tag *value*. Tanto o nome da variável quanto a identificação do estado devem estar escritos exatamente como estão escritos no arquivo no qual a Rede Bayesiana foi definido. A listagem abaixo reproduz um arquivo no qual compõe a Evidência a variável A observada no estado *A1*, e a variável

B da qual se têm a informação de que os únicos estados possíveis são os estados *B2* ou *B3*.

```
<evidences>
    <variable>
        <name>A</name>
        <value>A1</value>
    </variable>
    <variable>
        <name>B</name>
        <value>B2</value>
        <value>B3</value>
    </variable>
</evidences>
```

O último procedimento a ser realizado na etapa de entrada de dados é a determinação das variáveis requisitadas, ou seja, as variáveis probabilisticamente relevantes para executar um procedimento de inferência dado um conjunto de variáveis observadas. A determinação das variáveis requisitadas pode reduzir de maneira significativa as dimensões do problema a ser tratado e, portanto, reduzir o tempo necessário para processá-lo. Para determinar o conjunto de variáveis requisitadas utilizou-se o algoritmo *Bayes-Ball*, descrito em detalhes na Seção 2.3.5. O Bayes-Ball é um algoritmo simples e eficiente que identifica as relações de independência e define o conjunto de variáveis requisitadas baseado apenas na topologia do grafo da Rede Bayesiana, ou seja, sem envolver processamento processamento numérico ou o conhecimento dos estados das variáveis e suas respectivas distribuições de probabilidades.

A implementação do algoritmo Bayes-Ball está no arquivo *bayesball.c* e a funcionalidade de selecionar as variáveis requisitadas está encapsulada pela interface *BNUtils.c*.

5.4 Fatoração Simbólica

O objetivo da etapa aqui denominda de *Fatoração Simbólica* é a determinação da Ordem de Eliminação e na construção da Árvore de Eliminação dos vértices do Grafo Moral associado a uma Rede Bayesiana. Conforme mencionada reiteradas vezes nas seções anteriores, tais estruturas de dados permitiram paralelizar e otimizar o processamento numérico da etapa subsequente.

De forma gráfica, os principais processos envolvidos na Fatoração Simbólida e as respectivas

estruturas de dados resultantes estão representados na Figura 5.9. Observe no diagrama que, das estruturas internas que compõe a representação computacional da Rede Bayesiana, será utilizada nesta etapa apenas o grafo que define a topologia da rede. Desta forma procurou-se explicitar que a Fatoração Simbólica de uma a Rede Bayesiana não envolve valores numéricos que a definem, ou seja, este procedimento independe das distribuições de probabilidades das variáveis e dos os estados das variáveis observadas.

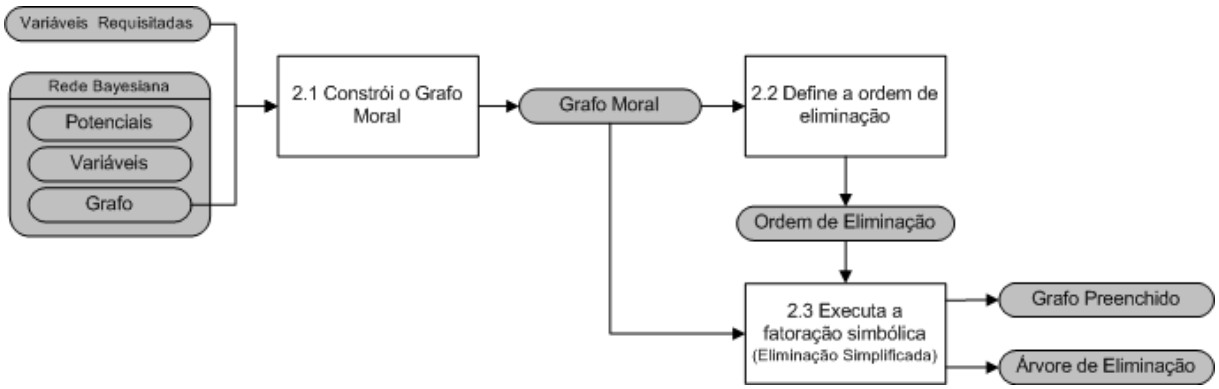


Figura 5.9: Etapa 02 - Fatoração Simbólica.

Conforme indicado na Figura 5.9 as principais estruturas resultantes desta etapa são: O Grafo Moral, a Ordem de Eliminação e a Árvore de Eliminação. Adicionalmente obtêm-se como um subproduto o Grafo Preenchido com os arcos de preenchimento gerados durante o processo de eliminação.

A construção do Grafo Moral segue um algoritmo bastante simples de percorrer o grafo, unindo com um arco os pais que tenham um filho em comum. Convêm ressaltar que o Grafo Moral é construído apenas com o subconjunto dos nós requeridos obtidos na primeira etapa.

A determinação da ordem de eliminação segue o arcabouço teórico proposto na na Seção 3.5 e detalhado na Seção 4.1.1. Sucintamente, utilizou-se a Heurística de Gibbs para a determinação de um nó quase-periférico. Como resultado da Heurística da Gibbs obtem-se para o nó quase-periférico a Árvore da Busca Em Largura (BEL), com base na qual constrói-se a Árvore de Dissecção. A partir da enumeração dos vértices da Árvore de Dissecção em pós-ordem gera-se a ordem de dissecção da qual deriva a ordem de eliminação das variáveis.

No processo de procura do vértice quase-periférico a Heurística de Gibbs constrói sucessivas Árvores BEL das quais uma deve ser selecionada. O critério básico proposto pela teoria define que

seria desejável que a BEL obtida tivesse a máxima profundidade possível. Observou-se que para vários vértices a BEL gerada apresentava o mesma profundidade, de forma que seria conveniente estabelecer critérios adicionais para se comparar as BELs obtidas com o objetivo selecionar a BEL que geraria a melhor Árvore de Dissecção e posteriormente uma ordem de eliminação melhor.

Considerando as características desejáveis para uma Árvore de Dissecção capaz de gerar boas ordens de eliminação (separadores pequenos e simetria) estipulou-se critérios adicionais para comparar as BELs geradas na Heurística de Gibbs. Tais critérios adicionais foram adotados "ad hoc" na tentativa de otimizar a de implementação computacional, com base em um conjunto restrito de testes empíricos. A validação destes carecem de uma justificativa teórica formal e/ou de testes empíricos exaustivos, não executados no presente trabalho.

Para selecionar BEL, e portanto o nó quase-periférico, adotou-se os seguintes critérios de comparação, aplicados na ordem em que estão listados:

1. *Maior Profundidade*: Conforme proposto pela teoria, com objetivo de reduzir a largura da BEL e obter separadores pequenos deve-se escolher a BEL de máxima profundidade.

2. *Menor Largura*: BELs com menor largura são desejáveis pois potencialmente gerariam separadores menores.

3. *Balanceamento da BEL*: BELs melhor balanceadas, ou seja, cujo nível central divida a BEL em dois conjuntos de nós de tamanhos próximos seriam preferíveis por potencialmente gerarem Árvores de Dissecção mais simétricas.

4. *Grau da Raiz*: BELs cujas raízes sejam nós com menor grau, ou seja, ligado a menos nós, são preferíveis pois evitaria a concentração nós na extremidade da BEL.

5. *Variância do números de nós*: BELs mais uniformes, ou seja, que apresentem a menor variância do número de nós nos níveis são mais desejáveis pois potencialmente gerariam Árvores de Dissecção melhor distribuídas.

A Figura 5.10 apresenta graficamente os principais processos e as estruturas de dados da implementação computacional do algoritmo descrito na Seção 4.1.1 e que sistematiza a Heurística de BEL/Gibbs para determinação da ordem de eliminação.

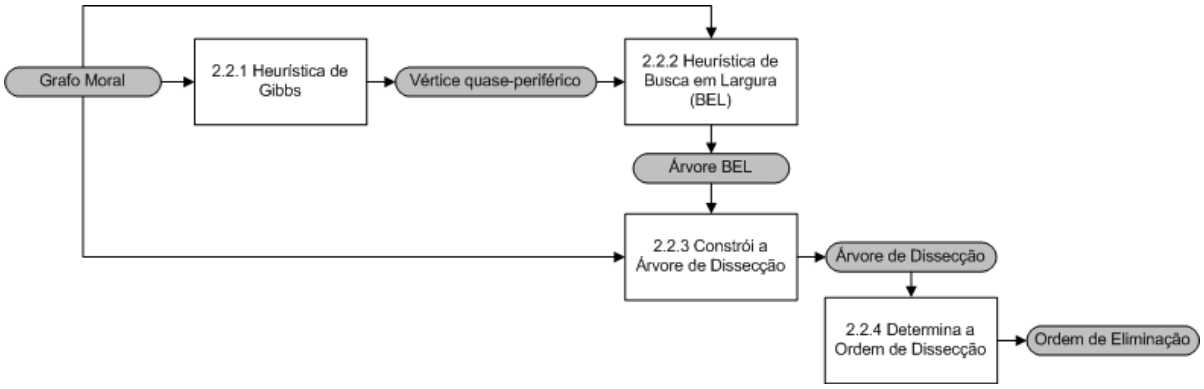


Figura 5.10: Etapa 02 - Determinação da Ordem de Eliminação.

Para a construção da Árvore de Eliminação utilizou-se o algoritmo de Fatoração Simbólica detalhado e exemplificado na Seção 4.1.1, o qual, por uma questão de eficiência computacional, utiliza método de eliminação simplificada dos vértices de uma grafo descrito na Seção 2.1. Cada vértice da Árvore de Eliminação corresponde a operação de eliminação de uma variável. Os arcos que unem os vértices do Grafo Moral inicial e os arcos de preenchimento inseridos nos grafos de eliminação ao longo do processo de eliminação dos vértices, permitem identificar a relação de dependência entre as operações para uma dada ordem de eliminação das variáveis. Com base nestas relações monta-se a Árvore de Eliminação.

A construção da Árvore de Eliminação conclui a Fase Simbólica, da qual como subproduto do processo de construção da Árvore de Eliminação obtêm-se o Grafo Preenchido. Do ponto de vista computacional o resultado desta etapa são ponteiros para duas estruturas uma que instancia a ordem de eliminação e outra a Árvore de Eliminação. Estas estruturas serão posteriormente fornecidas para as funções que irão executar o processamento numérico.

5.5 Fatoração Numérica

A terceira etapa, denominada de *Fatoração Numérica*, é provavelmente a etapa na qual ficará mais evidente contribuição do presente trabalho. Ela explicita a importância do procedimento simbólico, derivado da Álgebra Linear computacional, para o processamento numérico das operações envolvendo as distribuições de probabilidade da Rede Bayesiana. A denominação desta etapa, na qual serão

realizadas as operações de produto, de marginalização e de normalização envolvendo os potenciais, é novamente uma analogia ao procedimento de fatoração de matrizes. A Figura 5.11 apresenta um diagrama no qual estão representados os principais processos e as respectivas estruturas envolvidos na Fatoração Numérica.

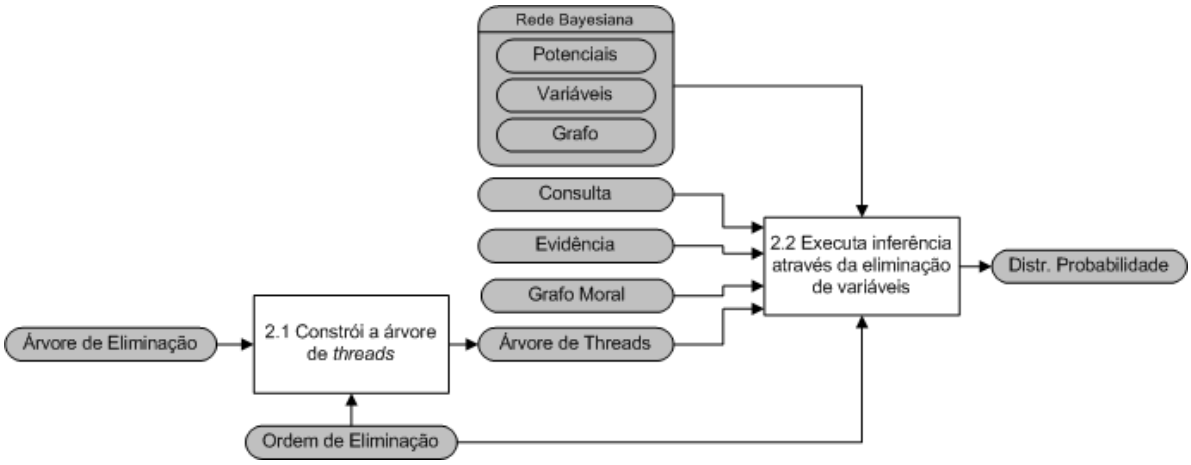


Figura 5.11: Etapa 03 - Fatoração Numérica.

A primeira vantagem da solução adotada é que o número de operações de eliminações, a ordem em que serão realizadas e quais poderão ser feitas em paralelo, são informações disponíveis antes de se iniciar o processamento numérico. Tais informações foram obtidas diretamente das características próprias do problema ou a partir das estruturas geradas pela Fatoração Simbólica. Estas informações permitem que todas as estruturas e todos os recursos computacionais sejam alocados a priori antes de se iniciar propriamente o processamento, o que torna o código computacionalmente mais eficiente.

A segunda vantagem, e talvez a mais relevante, é que as estruturas resultantes da Fatoração Simbólica, e em especial a Árvore de Eliminação, são fundamentais para permitir a otimização e a paralelização do processamento numérico. A topologia da Árvore de Eliminação define claramente a relação de dependência entre as operações de eliminação de variáveis da Rede Bayesianas, ou seja, permite identificar a priori quais operações devem estar concluídas para que uma determinada operação possa ser então executada. Desta forma, com base na Árvore de Eliminação, é possível definir quais operações podem ser executadas em paralelo o que torna possível implementar uma aplicação que explore as possibilidades de processamento paralelo disponíveis nos recursos computacionais atuais.

Na etapa de processamento numérico duas estruturas desempenham papéis especialmente im-

portantes, ambas descritas na Seção 5.2.2: (i) *ThreadTree* Árvore de Threads para paralelizar o processamento das eliminações; (ii) *Elimination* estrutura de dados utilizada para representar uma operação de eliminação. A Árvore de Threads é construída replicando a topologia e as relações de dependência da Árvore de Eliminação. O número de operações de eliminações é igual ao número de variáveis requisitadas, portanto, com base nesta informação aloca-se previamente um vetor de eliminações. Esta coleção de eliminações será armazenado em uma área genérica de dados da Árvore de Threads na qual os nós da árvore, ou seja, as threads, irão buscar as informações no momento que iniciarem a execução.

Convêm ressaltar uma vez mais que tanto a Árvore de Threads quanto a coleção de operações de eliminações foram criadas ou alocadas sem envolver informações sobre os valores numéricos das distribuições de probabilidades das variáveis ou os valores do estados das variáveis que formam a Evidência. Por este motivo, uma vez alocadas estas estruturas, se forem mantidos constantes os elementos dos conjuntos $\mathbf{X}_q$ e E, as mesmas podem ser reutilizadas repetidas vezes para inferência alterando os estados de E ou mesmo as distribuições de probabilidades das variáveis sem que para isto seja necessário recriar ou realocar a Árvore de Threads ou o vetor para armazenar a coleção de operações de eliminação.

Outro cuidado tomado na implementação foi manter a definição da Rede Bayesiana inalterada ao longo do processamento. Como os potenciais e a estrutura da rede permanecem intactos é possível executar repetidos exercícios de inferência, alterando-se a evidência E, sem que para isto seja necessário recarregar ou reiniciar a rede.

Seqüência de passos para o processamento numérico

Concluídas a criação da Árvore de Thread e a alocação do vetor para armazenar as operações de eliminação o processamento numérico segue a seqüência de passos descrita abaixo:

1. Inicializar cada operação de eliminação do vetor de operações associando à *i-ésima* operação a *i-ésima* variável da ordem de eliminação q e alocando uma lista vazia para armazenar os potenciais que serão utilizados na operação.

2. Para cada variável pertencente a Evidência E marginalizar dos potenciais associados a elas as variáveis dos mesmos que não pertençam a $\mathbf{X}_R$. Em seguida, executar o produto do potencial de cada variável pelo respectivo *finding*. O novo potencial gerado após estes dois procedimentos deverá ser associado à variável e será utilizado nas operações de eliminação.

3. Percorrer $\mathbf{X}_R$ acrescentando os respectivos potenciais às operações de eliminação de acordo com a ordem de eliminação. Considere o potencial ϕ_j associado a $X_j \in \mathbf{X}_R$ com domínio $dom(\phi_j)$, seja ainda $X_k^i \in dom(\phi_j)$ a *i-ésima* variável em q e a primeira do $dom(\phi_j)$ a ser eliminada. O potencial ϕ_j deve ser acrescentado à lista de potenciais da *i-ésima* operação de eliminação.

4. Disparar a execução da Árvore de Threads. Os primeiros vértices a serem executados são os vértices que não possuem descendentes, ou seja, estão na extremidade da árvore. Os demais seguem a seguinte sequência de operações:

   5.1. Cada vértice da Árvore de Threads aguarda o momento em que as execuções de todos seus descendentes tenham sido concluídas para executar a respectiva operação de eliminação.

   5.2. Cada vértice da Árvore de Threads ao executar a respectiva operação de eliminação deverá:

      5.2.1. Calcular o produto dos potenciais da sua lista de potenciais.

      5.2.2. Se a variável associada a operação de eliminação não pertencer a $\mathbf{X}_q$ então marginalizar o resultado do produto dos potenciais.

      5.2.3. Se o vértice executado não for a raiz da Árvore de Threads

      Atribuir o resultado do seu processamento à lista de potenciais de seu pai.

      Senão

      Concluir a execução da Árvore de Threads.

5. Aguardar a conclusão da execução da Árvore de Threads.

6. Normalizar o resultado o última operação de eliminação, executada pela raiz da Árvore de Threads. O potencial assim obtido é a distribuição de probabilidades resultante da inferência.

§

No que se refere a implementação computacional desta fase, tomou-se especial cuidado para minimizar os procedimentos que pudessem causar problemas de concorrência entre threads. Desta forma procurou-se maximizar a possibilidade de execução de processos paralelos com o mínimo custo de gerenciamento de concorrência e trocas de contexto de execução. No entanto, ao longo da execução da Árvore de Threads o procedimento de inserção do potencial resultante de uma operação de eliminação na lista ligada de potenciais do pai pode claramente gerar uma situação de concorrência entre threads. Conforme descrito anteriormente, para garantir que as operações com os elementos da lista ligada serão seguras e atômicas foram desenvolvidas funções específicas que sincronizam o acesso

aos elementos da lista e garantem a singularidade na manipulação dos mesmos durante operações críticas.

Capítulo 6

Testes e Comparação de Performance

Para testar o conjunto de bibliotecas implementados foram feitos testes de duas naturezas:

1. Precisão numérica dos resultados: Testes com o objetivo de verificar se os resultados obtidos estavam corretos.
2. Testes comparativos de performance: Testes para verificar se houve ganho com as otimizações propostas a partir da utilização das estruturas geradas na etapa de Fatoração Simbólica.

6.1 Testes de Precisão dos Resultados

Os testes iniciais para avaliar a precisão dos resultados foram realizados com 4 redes, 3 das quais disponíveis para testes na internet e uma criada especificamente para este projeto. O diagrama das redes estão indicados no Apêndice B.

1. *Exemplo01*: Rede com 10 variáveis cada qual com 2 ou 3 estados possíveis. Esta é a rede utilizada no exemplo apresentado na seção anterior e foi desenvolvida com propósitos didáticos de simular diversas situações que devem ser consideradas quando se trabalha com inferência em Redes Bayesianas.
2. *DogProblem*: Rede simples, com 5 variáveis binárias, normalmente utilizada em cursos introdutórios neste tipo de modelo.

3. *Asia*: Rede com 8 variáveis binárias a cerca do diagnóstico de Tuberculose e Cancer de Pulmão.

4. *Hailfinder25*: Rede com 55 variáveis, cada uma com múltiplos estados e relacionada a previsões meteorológicas. Das redes utilizadas esta é talvez a que mais se aproxima de um modelo real pela complexidade e número de nós.

O objetivo de se utilizar redes pequenas (com poucos nós) foi possibilitar a simulação manual da solução dos problemas de inferências, viabilizando a verificação dos resultados a cada passo do processamento. A adoção deste expediente foi essencial para auxiliar a elaboração dos algoritmos e a implementação computacional dos mesmos. No entanto, para obter ganhos de performance com a paralelização das operações foi necessário adotar uma rede maior, pois em uma rede com poucos nós o número de operações e a possibilidade de executá-las em paralelo são reduzidas. Por este motivo incorporou-se ao conjunto de testes a rede Hailfinder25 com 55 variáveis.

Além dos testes executados durante o desenvolvimento, foi estabelecido um conjunto básico de 36 testes envolvendo as redes acima descritas, nos quais procurou-se simular todas as situações e configurações possíveis de topologia de rede, de evidência e de consulta. Foram feitos vários testes em diversos exercícios de inferência para cada uma das redes mencionadas acima, incluindo testes com e sem evidência e consultas com uma ou mais variáveis. Os resultados, quando possível, foram confrontados com os resultados obtidos manualmente. No caso de modelos maiores, inviáveis de serem resolvidos manualmente, a verificação dos resultados foi feita confrontando os mesmos com os resultados gerados pelo software JavaBayes (*http://www.pmr.poli.usp.br/ltd/Software/javabayes/*). O JavaBayes foi escolhido por ser aplicação de referência entre as aplicações para inferência em Redes Bayesianas.

Para todos os testes realizados os resultados foram os resultados esperados, considerando tanto os resultados intermediários e finais para os problemas simulados manualmente, quanto os valores obtidos com auxílio computacional utilizando o JavaBayes. Tal constatação sugere que a implementação computacional está muito provavelmente correta e portanto que os resultados obtidos são confiáveis.

Por ser a primeira versão da biblioteca, testes exaustivos devem ainda ser feitos para atestar a robustez dos resultados e a estabilidade numérica das operações executadas, principalmente para redes mais complexas que incluam centenas ou milhares de variáveis. Tais procedimentos são parte do projeto de continuidade deste trabalho.

6.2 Testes Comparativos de Performance

6.2.1 Implementação Linear e Sequencial

Para possibilitar a comparação entre a performance obtida pelo método de paralelização das operações de eliminação, proposto pelo presente trabalho, e o método tradicional de execução linear e sequencial das mesmas, implementou-se uma versão alternativa da aplicação. Esta versão alternativa utiliza exatamente os mesmos procedimentos desde a entrada de dados até a determinação da ordem de eliminação, mas não executa a Fatoração Simbólica, nem tão pouco cria a Árvore de Threads. A versão alternatica utiliza exatamente a mesma função para executar as operações de eliminação que a versão paralela (*multithread*), mas as executa de forma serial, eliminando uma variável por vez, seguindo a ordem de eliminação previamente determinada.

A adoção desta estratégia isolou de forma inequívoca o efeito da paralelização do processamento uma vez que as demais funcionalidades são comuns a ambas as implementações. Optou-se pela mesma uma vez que o foco da presente trabalho foi testar a viabilidade e a eficácia da aplicação de métodos de fatoração de matrizes esparsa para a inferência em Redes Bayesianas. Utilizou-se algoritmos e heurísticas característicos da álgebra linear computacional, campo do conhecimento no qual a fatoração de matrizes esparsas está inserida, sem a preocupação, no primeiro momento, de buscar os algoritmos mais otimizados para cada etapa do processo.

A comparação da performance da aplicação implementada com aplicações semelhantes escritas em outras linguagens e com formas alternativas para entrada de dados ou determinação da ordem de eliminação poderiam ter resultados distorcidos dada a dificuldade ou mesmo a inviabilidade de obter exatamente o tempo alocado para o processamento das operações de eliminação das variáveis. Por não ser o escopo do trabalho, não houve neste primeiro momento a preocupação em otimizar ou buscar os algoritmos mais eficientes para executar operações secundárias. Como por exemplo não foram testadas formas alternativas para se obter a ordem de eliminação, assim como não foram pesquisados os algoritmos mais eficientes para executar operações de produto ou marginalização de potenciais. Dada a importância de tais operações para o processo, as mesmas deverão ser alvos de pesquisas e aperfeiçoamentos futuros.

6.2.2 Plataforma e Descrição dos Testes

Para comparar a performance da versão serial e da versão paralela do algoritmo de eliminação de variáveis foram feitos diversos testes utilizando os modelos listados na seção anterior. Os principais resultados estão apresentados nas Tabelas 6.1 - 6.4.

Os aplicativos e as bibliotecas foram compilados e testados em um sistema operacional Linux rodando em uma computador com plataforma Intel com as seguintes configuração:

Processadores:	2x Intel(R) Xeon(TM) CPU 3.60GHz
Memória:	RAM 2.0GB
Sistema Operacional:	Linux Debian 3.4.3-13sarge1
Versão:	Linux version 2.6.8-12-em64t-p4-smp
Compilador:	gcc (GCC) 3.3.5 (Debian 1:3.3.5-13)

A versão paralela do algoritmo de eliminação de variáveis antes de iniciar o processamento numérico executa a Fatoração Simbólica para construir a Árvore de Eliminação e posteriormente monta a Árvore de Threads. Estes procedimentos consomem tempo de processamento e não são executados na versão serial da aplicação. Mas por outro lado, uma vez executada a Fatoração Simbólica e construída a Árvore de Threads, as estruturas geradas podem ser reaproveitadas para executar inferências quantas vezes forem necessárias. Este fato para aplicações específicas poderia implicar em ganhos significativos de performance.

Em função das diferenças entre as versões paralela e serial das implementações foram definidos dois objetivos:

1. Estimar o processamento adicional necessário para executar a Fatoração Simbólica e construção da Árvore de Threads e o impacto deste processamento no procedimento de inferência como um todo;

2. Isolar tanto na versão serial quanto na paralela especificamente o tempo e o processamento alocados para a Fatoração Numérica, para que seja possível avaliar o impacto da paralelização.

Para atingir os objetivos propostos acima dois procedimentos de testes distintos foram executados:

1. *Procedimento Completo:* Para a versão paralela executa-se a Fatoração Simbólica, a construção do Árvore de Threads e a Fatoração Numérica. No caso da versão serial determina-se a ordem

de eliminação e executa-se diretamente a Fatoração Numérica.

2. *Procedimento de Inferência:* Para a versão paralela executa-se uma única vez a Fatoração Simbólica e a construção do Árvore de Threads e diversas vezes a Fatoração Numérica utilizando as estruturas obtidas na fase simbólica. No caso da versão serial determina-se a ordem de eliminação e executa-se sucessivas vezes a Fatoração Numérica.

6.2.3 Resultados e Análises

Os principais resultados dos testes estão resumidos nas Tabelas 6.1 - 6.4. As tabelas apresentam as seguintes colunas:

Versão: Versão da aplicação utilizada: (i) *paralela* é a versão que executa a Fatoração Simbólica e com base na Árvore de Eliminação paraleliza as operações;(ii) *serial* é a verão da aplicação que executa as operações do processamento numérico sequencialmente.

Modelo: Identificação da rede utilizada para testes.

Teste: Identificação do teste realizado. Para cada rede os testes variaram quanto a evidência e a consulta.

 # Var.: Número de variáveis requeridas, ou seja, a dimensão de X_R.

N: Número de procedimentos completos realizados, conforme descrito anteriormente.

NTtotal: Tempo total de processamento em segundos alocados das CPUs para realizar os N procedimentos completos.

NTmédio: Tempo médio de processamento em segundos alocados das CPUs para realizar cada procedimento completo, calculado como o Ntotal/N.

L: Número total de procedimentos de inferência realizados: L = N x (Inferências por procedimento).

LTtotal: Tempo total de processamento em segundos alocados das CPUs para realizar os L procedimentos de inferência.

LTmédio: Tempo médio de processamento em segundos alocados das CPUs para realizar cada procedimento de inferência, calculado como o Ltotal/L.

Tempo: Intervalo real de tempo em segundos necessários para executar um determinado teste.

***Context Switch*:** Número total de mudanças de contexto voluntárias e involuntárias.

***CPU Usage*:** Porcentagem de utilização da CPU. Observar que a máquina tem 2 processadores.

Convêm ainda mencionar que: (i) Para cada um dos testes foram realizadas 5 amostras e que os valores zeros na tabela correspondem a valores inferiores a milésimos de segundo; (ii) O tempo de processamento alocado das CPUs é a soma dos tempos em que o processo foi executado em *user mode* e *kernel mode*; (iii) Considerando que para a execução dos testes utilizou-se um hardware multiprocessando, a soma da alocação de tempo de processamento para cada uma das CPUs pode superar o intervalo de tempo real necessário para executar o aplicativo.

Versão	Modelo	Teste	# Var.	N	NTtotal (s)	NTmédio (s)	L	LTtotal (s)	LTmédio (s)	Tempo (s)	Context Switched	CPU Usage
paralela	Exemplo01	1	4	100	1.306	0.0131	10000	1.284	0.000	0.790	22804.60	166.0%
serial	Exemplo01	1	4	100	0.110	0.0011	10000	0.106	0.000	0.114	2.20	98.4%
paralela	Exemplo01	2	8	100	4.070	0.0407	10000	4.050	0.000	1.894	101441.60	214.4%
serial	Exemplo01	2	8	100	0.368	0.0037	10000	0.360	0.000	0.370	2.00	99.0%
paralela	Exemplo01	3	8	100	4.108	0.0411	10000	4.084	0.000	1.904	101568.20	215.8%
serial	Exemplo01	3	8	100	0.398	0.0040	10000	0.388	0.000	0.400	1.60	99.0%
paralela	Exemplo01	4	2	100	0.480	0.0048	10000	0.456	0.000	0.370	10243.20	134.0%
serial	Exemplo01	4	2	100	0.040	0.0004	10000	0.038	0.000	0.040	2.20	95.4%
paralela	Exemplo01	5	10	100	4.842	0.0484	10000	4.818	0.001	2.180	108856.40	222.0%
serial	Exemplo01	5	10	100	0.444	0.0044	10000	0.430	0.000	0.450	2.00	99.0%
paralela	Exemplo01	6	10	100	4.680	0.0468	10000	4.640	0.000	2.160	107621.20	216.8%
serial	Exemplo01	6	10	100	0.472	0.0047	10000	0.462	0.000	0.474	8.00	99.2%
paralela	Exemplo01	7	6	100	2.136	0.0214	10000	2.114	0.000	1.040	41997.60	205.4%
serial	Exemplo01	7	6	100	0.226	0.0023	10000	0.218	0.000	0.226	3.60	98.8%
paralela	Exemplo01	8	10	100	4.804	0.0480	10000	4.788	0.001	2.148	108391.80	223.2%
serial	Exemplo01	8	10	100	0.454	0.0045	10000	0.448	0.000	0.460	5.60	99.0%
paralela	Exemplo01	9	10	100	4.814	0.0481	10000	4.772	0.001	2.144	107497.40	224.2%
serial	Exemplo01	9	10	100	0.456	0.0046	10000	0.448	0.000	0.460	1.80	99.0%

Tabela 6.1: Modelo pequeno e elevado número de procedimentos completos e de inferência

Observando a Tabela 6.1, que utilizou um modelo pequeno com poucas variáveis e 10000 procedimentos de inferência, observa-se que apesar dos dois aplicativos serem muito eficientes, cada procedimento é executado em uma fração de segundo, na operação de inferência a aplicação serial leva vantagem sobre a aplicação paralela. A vantagem é ainda discretamente maior quando

Versão	Modelo	Teste	# Var.	N	NTtotal (s)	NTmédio (s)	L	LTtotal (s)	LTmédio (s)	Tempo (s)	Context Switched	CPU Usage
paralela	hailfinder25	1	13	10	0.078	0.0078	100	0.076	0.001	0.050	1299.60	197.2%
serial	hailfinder25	1	13	10	0.030	0.0030	100	0.028	0.000	0.030	2.00	95.4%
paralela	hailfinder25	2	6	10	0.014	0.0014	100	0.014	0.000	0.020	748.80	162.0%
serial	hailfinder25	2	6	10	0.008	0.0008	100	0.008	0.000	0.010	2.00	88.2%
paralela	hailfinder25	3	30	10	76.724	7.6724	100	76.710	0.767	76.580	4253.60	100.0%
serial	hailfinder25	3	30	10	45.408	4.5408	100	45.404	0.454	45.414	40.40	99.0%
paralela	hailfinder25	4	28	10	6.736	0.6736	100	6.728	0.067	6.598	4004.40	102.0%
serial	hailfinder25	4	28	10	6.656	0.6656	100	6.650	0.067	6.660	16.80	99.0%
paralela	hailfinder25	5	40	10	122.774	12.2774	100	122.740	1.227	79.028	7806.20	155.0%
serial	hailfinder25	5	40	10	176.922	17.6922	100	176.914	1.769	176.928	184.40	99.0%
paralela	hailfinder25	6	3	10	0.000	0.0000	100	0.000	0.000	0.010	205.40	119.6%
serial	hailfinder25	6	3	10	0.000	0.0000	100	0.000	0.000	0.000	1.80	88.0%
paralela	hailfinder25	7	29	10	6.740	0.6740	100	6.732	0.067	6.604	4155.80	101.8%
serial	hailfinder25	7	29	10	6.664	0.6664	100	6.662	0.067	6.664	16.40	99.2%
paralela	hailfinder25	8	22	10	97.960	9.7960	100	97.942	0.979	97.882	2984.80	100.0%
serial	hailfinder25	8	22	10	66.540	6.6540	100	66.536	0.665	66.548	79.80	99.0%
paralela	hailfinder25	9	44	10	174.700	17.4700	100	174.662	1.747	172.936	6537.20	100.6%
serial	hailfinder25	9	44	10	812.346	81.2346	100	812.346	8.123	812.332	858.00	99.0%
paralela	hailfinder25	10	44	10	97.510	9.7510	100	97.478	0.975	95.766	6495.80	101.0%
serial	hailfinder25	10	44	10	125.434	12.5434	100	125.424	1.254	125.436	162.80	99.0%
paralela	hailfinder25	11	45	10	72.986	7.2986	100	72.964	0.730	71.236	6732.80	102.0%
serial	hailfinder25	11	45	10	154.862	15.4862	100	154.856	1.549	154.866	183.40	99.0%
paralela	hailfinder25	12	46	10	76.670	7.6670	100	76.650	0.767	74.904	6799.00	102.0%
serial	hailfinder25	12	46	10	154.742	15.4742	100	154.732	1.547	154.750	158.40	99.0%
paralela	hailfinder25	13	46	10	105.516	10.5516	100	105.470	1.055	103.772	6927.00	101.0%
serial	hailfinder25	13	46	10	125.582	12.5582	100	125.578	1.256	125.584	154.40	99.0%
paralela	hailfinder25	14	48	10	107.960	10.7960	100	107.926	1.079	106.210	7193.20	101.0%
serial	hailfinder25	14	48	10	125.564	12.5564	100	125.554	1.256	125.560	133.80	99.0%

Tabela 6.2: Modelo maior com repetições de procedimento completo e inferência

Versão	Modelo	Teste	# Var.	N	NTtotal (s)	NTmédio (s)	L	LTtotal (s)	LTmédio (s)	Tempo (s)	Context Switched	CPU Usage
paralela	hailfinder25	1	13	1	0.070	0.0700	100	0.070	0.001	0.040	1247.80	186.6%
serial	hailfinder25	1	13	1	0.030	0.0300	100	0.030	0.000	0.030	2.00	95.0%
paralela	hailfinder25	2	6	1	0.014	0.0140	100	0.014	0.000	0.020	722.60	157.8%
serial	hailfinder25	2	6	1	0.010	0.0100	100	0.010	0.000	0.010	2.00	88.2%
paralela	hailfinder25	3	30	1	76.684	76.6840	100	76.684	0.767	76.548	4208.20	100.0%
serial	hailfinder25	3	30	1	45.302	45.3020	100	45.302	0.453	45.300	46.20	99.0%
paralela	hailfinder25	4	28	1	6.726	6.7260	100	6.726	0.067	6.590	3907.40	101.8%
serial	hailfinder25	4	28	1	6.662	6.6620	100	6.662	0.067	6.672	15.20	99.4%
paralela	hailfinder25	5	40	1	123.006	123.0060	100	123.006	1.230	79.002	7769.60	155.0%
serial	hailfinder25	5	40	1	176.622	176.6220	100	176.622	1.766	176.626	201.80	99.0%
paralela	hailfinder25	6	3	1	0.000	0.0000	100	0.000	0.000	0.010	206.40	111.0%
serial	hailfinder25	6	3	1	0.000	0.0000	100	0.000	0.000	0.000	1.40	79.4%
paralela	hailfinder25	7	29	1	6.742	6.7420	100	6.742	0.067	6.592	4114.00	102.0%
serial	hailfinder25	7	29	1	6.622	6.6220	100	6.622	0.066	6.624	2.40	99.0%
paralela	hailfinder25	8	22	1	97.798	97.7980	100	97.798	0.978	97.726	2948.40	100.0%
serial	hailfinder25	8	22	1	66.430	66.4300	100	66.430	0.664	66.434	78.00	99.0%
paralela	hailfinder25	9	44	1	175.938	175.9380	100	175.932	1.759	174.154	6498.40	100.6%
serial	hailfinder25	9	44	1	812.184	812.1840	100	812.184	8.122	812.158	901.40	99.0%
paralela	hailfinder25	10	44	1	97.056	97.0560	100	97.052	0.971	95.304	6513.80	101.0%
serial	hailfinder25	10	44	1	125.310	125.3100	100	125.310	1.253	125.316	142.20	99.0%
paralela	hailfinder25	11	45	1	72.962	72.9620	100	72.958	0.730	71.210	6553.00	102.0%
serial	hailfinder25	11	45	1	155.170	155.1700	100	155.170	1.552	155.176	183.40	99.0%
paralela	hailfinder25	12	46	1	75.770	75.7700	100	75.770	0.758	74.014	6680.60	102.0%
serial	hailfinder25	12	46	1	154.780	154.7800	100	154.780	1.548	154.780	164.40	99.0%
paralela	hailfinder25	13	46	1	105.870	105.8700	100	105.870	1.059	104.136	6816.60	101.0%
serial	hailfinder25	13	46	1	125.508	125.5080	100	125.508	1.255	125.512	137.60	99.0%
paralela	hailfinder25	14	48	1	108.012	108.0120	100	108.008	1.080	106.250	7067.40	101.0%
serial	hailfinder25	14	48	1	125.360	125.3600	100	125.358	1.254	125.362	152.20	99.0%

Tabela 6.3: Modelo maior com repetições da operação de inferência

Versão	Modelo	Teste	# Var.	N	NTtotal (s)	NTmédio (s)	L	LTtotal (s)	LTmédio (s)	Tempo (s)	Context Switched	CPU Usage
paralela	hailfinder25	1	13	100	0.104	0.0010	100	0.084	0.001	0.070	1764.80	166.2%
serial	hailfinder25	1	13	100	0.032	0.0003	100	0.032	0.000	0.036	2.00	96.2%
paralela	hailfinder25	2	6	100	0.032	0.0003	100	0.018	0.000	0.030	763.80	145.8%
serial	hailfinder25	2	6	100	0.010	0.0001	100	0.004	0.000	0.010	1.80	90.8%
paralela	hailfinder25	3	30	100	76.974	0.7697	100	76.860	0.769	76.838	4376.80	100.0%
serial	hailfinder25	3	30	100	45.550	0.4555	100	45.516	0.455	45.554	60.00	99.0%
paralela	hailfinder25	4	28	100	6.866	0.0687	100	6.766	0.068	6.722	4275.60	102.0%
serial	hailfinder25	4	28	100	6.670	0.0667	100	6.650	0.067	6.676	2.60	99.2%
paralela	hailfinder25	5	40	100	123.338	1.2334	100	123.088	1.231	79.348	8040.60	155.0%
serial	hailfinder25	5	40	100	176.994	1.7699	100	176.904	1.769	176.994	208.20	99.0%
paralela	hailfinder25	6	3	100	0.012	0.0001	100	0.004	0.000	0.020	207.20	119.6%
serial	hailfinder25	6	3	100	0.006	0.0001	100	0.004	0.000	0.010	2.20	86.6%
paralela	hailfinder25	7	29	100	6.868	0.0687	100	6.770	0.068	6.716	4276.40	102.0%
serial	hailfinder25	7	29	100	6.720	0.0672	100	6.692	0.067	6.726	19.80	99.2%
paralela	hailfinder25	8	22	100	98.290	0.9829	100	98.168	0.982	98.218	2984.20	100.0%
serial	hailfinder25	8	22	100	66.558	0.6656	100	66.530	0.665	66.562	74.00	99.2%
paralela	hailfinder25	9	44	100	177.958	1.7796	100	177.550	1.776	176.202	6692.00	100.6%
serial	hailfinder25	9	44	100	813.374	8.1337	100	813.272	8.133	813.346	878.00	99.0%
paralela	hailfinder25	10	44	100	98.390	0.9839	100	97.980	0.980	96.640	6840.80	101.0%
serial	hailfinder25	10	44	100	125.452	1.2545	100	125.374	1.254	125.454	160.80	99.0%
paralela	hailfinder25	11	45	100	73.806	0.7381	100	73.576	0.736	72.064	6796.20	102.0%
serial	hailfinder25	11	45	100	154.170	1.5417	100	154.092	1.541	154.164	156.20	99.0%
paralela	hailfinder25	12	46	100	83.688	0.8369	100	83.446	0.834	81.922	7005.20	102.0%
serial	hailfinder25	12	46	100	154.366	1.5437	100	154.278	1.543	154.368	175.60	99.0%
paralela	hailfinder25	13	46	100	107.978	1.0798	100	107.562	1.076	106.210	7113.20	101.0%
serial	hailfinder25	13	46	100	125.362	1.2536	100	125.286	1.253	125.366	136.80	99.0%
paralela	hailfinder25	14	48	100	110.334	1.1033	100	109.910	1.099	108.588	7548.60	101.0%
serial	hailfinder25	14	48	100	125.246	1.2525	100	125.154	1.252	125.244	143.80	99.0%

Tabela 6.4: Modelo maior com repetições do procedimento completo

comparamos os procedimentos completos devido provavelmente ao processamento adicional para a Fatoração Simbólica. No entanto, quando analisamos o tempo realmente necessário para executar a aplicação (coluna Tempo), em função da máquina possuir dois processadores, a diferença de tempo real entre as duas aplicações se reduz significativamente, se comparada com a diferença no tempo de processamento de CPU.

Quando o tamanho do modelo aumenta, ou seja, aumentam o número de variáveis e as possibilidades de executar operações em paralelo, a aplicação paralela supera a aplicação serial. Analise por exemplo os Testes 9, 10, 11 e 12 nas Tabelas 6.2 - 6.4. Em particular, a partir da análise da Árvore de Eliminação do Teste 9 do modelo Hailfinder25 - Figura 6.1 - é possível verificar que a mesma possibilita uma considerável paralelização das operações numéricas, o que gerou um ganho de performance significativo. Na figura, os números dentro dos círculos, que representam os vértices da Árvore de Eliminação, correspondem à ordem de eliminação e o número sobre o círculo é o ciclo de processamento no qual o mesmo poderia ser eliminado.

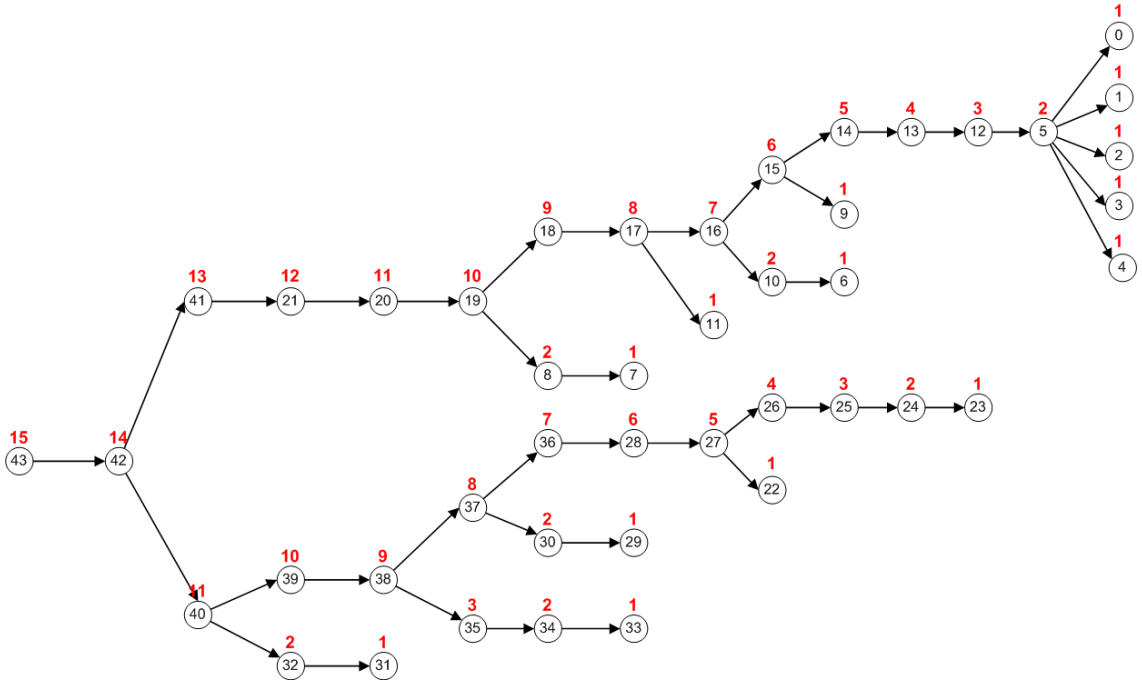


Figura 6.1: Hailfinder25 - Árvore de eliminação do Teste 09.

No entanto, o modelo ter grande número de variáveis não garante que a versão paralela terá desempenho superior. Para modelos com maior número de variáveis mas cuja topologia da Árvore de Eliminação seja mais linear, as relações de dependência fazem com que as mesmas sejam pouco paralelizáveis. Neste caso, a versão serial poderá ter um desempenho igual ou superior, como sugerem o Teste 3 - Figura 6.3 - e o Teste 8 - Figura 6.2 do modelo Hailfinder25. A comparação da relação entre o número de nós e a profundidade das Árvores de Eliminação dos Testes 3 e 8 com a do Teste 9 sugere que a Árvore de Eliminação desta última é mais paralelizável pois a topologia é menos linear, uma vez que para um número maior de nós a Árvore de Eliminação do Teste 9 possui aproximadamente mesma profundidade que as duas outras.

Além da possibilidade ou não de paralelizar as operações, é interessante notar que outro redutor do potencial ganho com a paralelização das operações numéricas é o tamanho dos potenciais e das operações com os mesmos, presentes no caminho crítico das série de operações (seqüência de nós que determina o tempo de processamento).

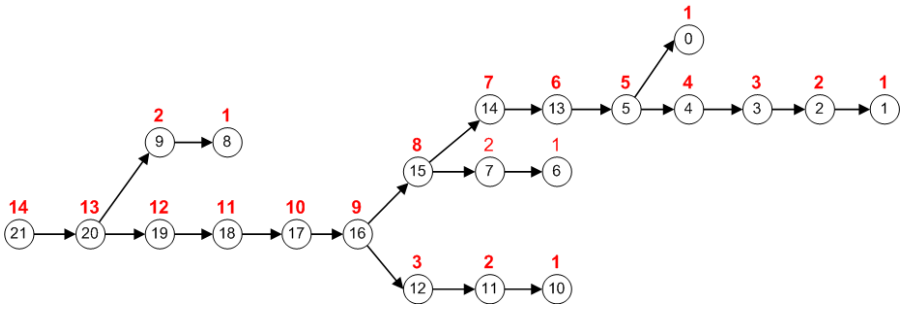


Figura 6.2: Hailfinder25 - Árvore de eliminação do Teste 08.

Considerando os testes realizados com a versão paralela da aplicação que envolveram um maior número de variáveis, comparando os tempos totais para os procedimentos completos e o tempos totais para os procedimentos de inferência pode-se inferir que o processamento adicional para a Fatoração Simbólica na aplicação paralela deixa de ser significativo com o aumento do número de variáveis, e para modelos maiores e paralelizáveis o ganho de performance com a paralelização supera significativamente o tempo de processamento adicional necessário. Observe que é relativamente pequena a diferença entre os valores do tempo total para realizar 1 procedimento completo e 100 inferências, Tabela 6.3, e do tempo total para realizar 100 procedimentos completos com 1 inferência em cada procedimento (totalizando L=100 inferências), Tabela 6.4.

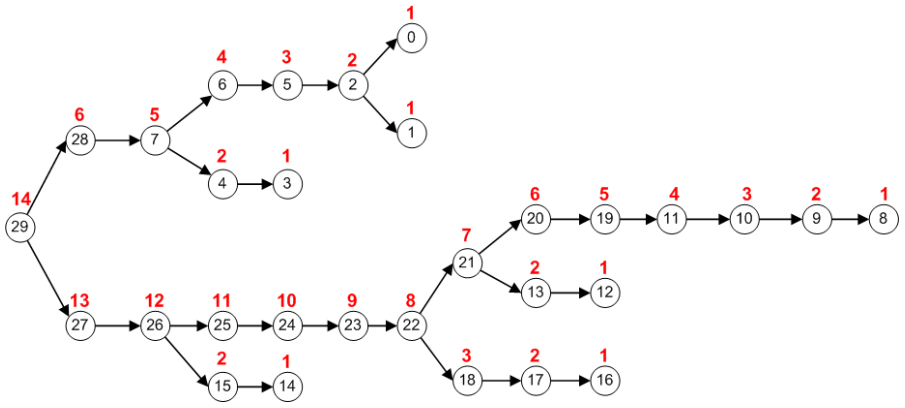


Figura 6.3: Hailfinder25 - Árvore de eliminação do Teste 03.

Mesmo analisando isoladamente as Tabelas 6.2 - 6.4, a diferença entre o tempo total necessário para realizar os procedimentos completos e o tempo total para realizar os procedimentos de inferência é possível concluir que à medida que o número de nós aumenta e se obtém maiores ganhos com a paralelização (para os modelos testados aproximadamente acima de 20 variáveis), o tempo alocado para a Fatoração Simbólica, torna-se quase desprezível, sendo para os testes realizados de ordem de grandeza inferior a 0,4% do tempo total para maioria dos casos. Considerando-se ainda que uma vez feita a Fatoração Simbólica é possível reaproveitar as estruturas de dados geradas para realizar diversos procedimentos de inferência, concluí-se que a utilização do algoritmo proposto para inferência em Redes Bayesianas pode otimizar significativamente os procedimentos de inferência.

A partir dos resultados dos experimentos para os modelos com maior número de variáveis requisitadas concluí-se que a versão paralela apresentou na maioria dos casos, ganhos significativos de performance em relação à versão serial. De forma consolidada, a análise integral do conjunto de resultados suscita as seguintes conclusões:

- Para problemas pequenos - quantidade pequena de variáveis e com pouca possibilidade de paralelização das operações - a versão linear mostrou-se mais rápida e eficiente tanto nos teste de Procedimento Completo quanto nos testes de Procedimento de Inferência. Esta diferença de performance deve-se ao fato da versão linear não consumir o processamento adicional necessário para a Fatoração Numérica e para a construção e inicialização da Árvore de Threads.

- Especificamente com relação aos testes com os Procedimentos de Inferência, sobre os quais

teoricamente não têm efeito o processamento adicional da fase simbólica, a versão serial para problemas pequenos mostrou-se ainda sim superior. Esta melhor performance decorre do fato do sistema operacional consumir tempo de processamento para gerenciar os processos e para troca de contexto de execução entre os mesmos ao executar um aplicativo *multithread*. Observe a diferenças relativa ao número de trocada de contexto entre os experimentos realizados utilizando a aplicação paralela e compare os valores com os obtidos com a versão serial.

- Para problemas maiores - quantidade grande de variáveis e/ou com grande possibilidade de paralelização das operações - a versão paralela mostrou-se visivelmente superior. Ocorreram ganhos de performance significativos nos testes de Procedimento de Inferência o que repercutiu em ganhos de performance também no Procedimento Completo. Ou seja, à medida que a complexidade do problema aumentou o impacto do processamento adicional necessário para executar a versão paralela diminui e é superado pelos ganhos de performance decorrentes paralelização das operações na fase numérica. Porém, cabe notar que mesmo para modelos grandes no quais as possibilidades de paralelizar as operações as operações são pequenas, em função da topologia e da relação de dependência entre as variáveis, o modelo serial pode levar vantagem.

- Da mesma forma, observando os resultados para o testes com os Procedimento de Inferência pode-se inferir que o processamento adicional necessário para o sistema operacional executar um aplicativo multihread deixa de ser significativo à medida que o problema aumenta. Assim os ganhos com a paralelização das operações superam o processamento adicional para gerenciar processos paralelos.

As vantagens obtidas com a paralelização das operações poderiam ser maximizadas, e suas desvantagens minimizadas se o aplicativo for executado em uma plataforma com mais processadores, com um sistema operacional otimizado para trabalhar com aplicativos multithread e a aplicação fosse compilada com parâmetros específicos para otimizar o suporte a procedimentos multithread. Além disto, uma análise mais aprofundada nas bibliotecas implementadas indicaram que existem ainda otimizações possíveis de serem introduzidas para reduzir a quantidade de instruções necessárias para a Fatoração Numérica na versão paralela. Tais alterações serão feitas com a continuidade deste projeto.

Os resultados dos testes demonstram que a Fatoração Simbólica pode acarretar ganhos importantes de performance à medida que as dimensões e a complexidade do problema aumenta, ou quando, o que justificaria esforços futuros para aperfeiçoamento dos algoritmos e do conjunto de bibliotecas

implementados.

Capítulo 7

Conclusão

O objetivo deste trabalho foi desenvolver uma aplicação computacional para demonstrar que técnicas de fatoração de matrizes esparsas características de álgebra linear computacional podiam ser utilizadas para construir um algoritmo eficiente e paralelizável para inferência em Redes Bayesianas. Implementou-se uma aplicação computacional que separa o processo de inferência em duas fases, a primeira fase simbólica e uma segunda fase numérica. Utilizando as estruturas geradas na primeira fase, em especial a Árvore de Eliminação e a definição da ordem de eliminação, foi possível otimizar e paralelizar os procedimentos numéricos da segunda fase.

Esta separação foi possível através da análise de algoritmos de fatoração de matrizes esparsas e do algoritmo de eliminação de variáveis para inferência em Redes Bayesianas a partir de um arcabouço combinatório unificado descritos nos capítulos iniciais do trabalho. As estruturas combinatórias geradas na fase simbólica e comum aos dois processos são a chave para a implementação computacionalmente eficiente de um algoritmo capaz de lidar com grandes modelos.

Conforme foi possível constatar pelos resultados dos testes comparativos de performance, a técnica de paralelização da eliminação das variáveis pode gerar ganhos significativos de performance em modelos nos quais o número de variáveis é grande e os mesmos permitem paralelizar uma quantidade relevante das operações. No entanto existem exceções, em particular para modelos pequenos - quantidade pequena de variáveis e com pouca possibilidade de paralelização das operações - realizar as operações de forma serial pode ser mais eficiente pois não consume o processamento adicional necessário para a Fatoração Simbólica e poupa o sistema operacional de gerenciar processos

simultâneos.

A atual disponibilidade de recursos computacionais capazes de fazer processamento paralelo de forma eficiente, bem como os ganhos potenciais de performance com a paralelização das operações sugerem que a metodologia aqui descrita e implementada pode viabilizar ganhos significativos de performance em problemas de grande escala. Por outro lado, os resultados e análise apresentadas na Seção 6.2 colocam como desafio o desenvolvimento de uma algoritmo (ou uma heurística) capaz de, em tempo de execução, identificar quando as operações devem ser paralelizadas e quando seria mais eficiente executá-las em série. Outra possível aplicação da técnica discutida neste trabalho é desenvolver um *compilador* de Redes Bayesianas que pré-processem a rede com o objetivo de otimizar os procedimento numéricos com a mesma.

A primeira versão do conjunto de bibliotecas desenvolvidos, à princípio com base nos testes realizados, implementou satisfatoriamente o algoritmo proposto e mostrou-se capaz de executar operações de inferência e gerar resultados confiáveis. No entanto, não há dúvidas de que as funções dos programas que compõe este conjunto de bibliotecas podem ser otimizadas, seja pela pesquisa de algoritmos mais eficientes para etapas dos processo, como por exemplo para determinação a ordem de eliminação ou para calcular o produto de potenciais, seja por aperfeiçoamento do código da aplicação em si.

Este trabalho foi o resultado de um esforço para compreender a inferência em modelos de Redes Bayesianas com o objetivo de aplicá-los a decisões reais feitas sob incerteza.

Apêndice A

Listagens Exemplo01

A.1 Definição da Rede Bayesiana

```
<?xml version="1.0" encoding="US-ASCII"?>


<!--
    Bayesian network in XMLBIF v0.3 (BayesNet Interchange Format)
    Produced by JavaBayes (http://www.cs.cmu.edu/~javabayes/
    Output created Sun Jul 29 20:18:49 BRT 2007
-->



<!-- DTD for the XMLBIF 0.3 format -->
<!DOCTYPE BIF [
    <!ELEMENT BIF ( NETWORK )*>
          <!ATTLIST BIF VERSION CDATA #REQUIRED>
    <!ELEMENT NETWORK ( NAME, ( PROPERTY | VARIABLE | DEFINITION )* )>
    <!ELEMENT NAME (#PCDATA)>
    <!ELEMENT VARIABLE ( NAME, ( OUTCOME |  PROPERTY )* ) >
```

```
          <!ATTLIST VARIABLE TYPE (nature|decision|utility) "nature">
    <!ELEMENT OUTCOME (#PCDATA)>
    <!ELEMENT DEFINITION ( FOR | GIVEN | TABLE | PROPERTY )* >
    <!ELEMENT FOR (#PCDATA)>
    <!ELEMENT GIVEN (#PCDATA)>
    <!ELEMENT TABLE (#PCDATA)>
    <!ELEMENT PROPERTY (#PCDATA)>
]>


<BIF VERSION="0.3">
<NETWORK>
<NAME>InternalNetwork</NAME>

<!-- Variables -->
<VARIABLE TYPE="nature">
    <NAME>A</NAME>
    <OUTCOME>A1</OUTCOME>
    <OUTCOME>A2</OUTCOME>
    <OUTCOME>A3</OUTCOME>
    <PROPERTY>position = (828, 193)</PROPERTY>
</VARIABLE>

<VARIABLE TYPE="nature">
    <NAME>B</NAME>
    <OUTCOME>B1</OUTCOME>
    <OUTCOME>B2</OUTCOME>
    <OUTCOME>B3</OUTCOME>
    <PROPERTY>position = (957, 142)</PROPERTY>
</VARIABLE>

<VARIABLE TYPE="nature">
    <NAME>C</NAME>
```

```
    <OUTCOME>C1</OUTCOME>
    <OUTCOME>C2</OUTCOME>
    <PROPERTY>position = (899, 219)</PROPERTY>
</VARIABLE>

<VARIABLE TYPE="nature">
    <NAME>D</NAME>
    <OUTCOME>D1</OUTCOME>
    <OUTCOME>D2</OUTCOME>
    <OUTCOME>D3</OUTCOME>
    <PROPERTY>position = (899, 327)</PROPERTY>
</VARIABLE>

<VARIABLE TYPE="nature">
    <NAME>E</NAME>
    <OUTCOME>E1</OUTCOME>
    <OUTCOME>E2</OUTCOME>
    <PROPERTY>position = (826, 376)</PROPERTY>
</VARIABLE>

<VARIABLE TYPE="nature">
    <NAME>F</NAME>
    <OUTCOME>F1</OUTCOME>
    <OUTCOME>F2</OUTCOME>
    <PROPERTY>position = (1021, 231)</PROPERTY>
</VARIABLE>

<VARIABLE TYPE="nature">
    <NAME>G</NAME>
    <OUTCOME>G1</OUTCOME>
    <OUTCOME>G2</OUTCOME>
    <PROPERTY>position = (1021, 326)</PROPERTY>
</VARIABLE>
```

```
<VARIABLE TYPE="nature">
    <NAME>H</NAME>
    <OUTCOME>H1</OUTCOME>
    <OUTCOME>H2</OUTCOME>
    <OUTCOME>H3</OUTCOME>
    <PROPERTY>position = (1089, 235)</PROPERTY>
</VARIABLE>

<VARIABLE TYPE="nature">
    <NAME>I</NAME>
    <OUTCOME>I1</OUTCOME>
    <OUTCOME>I2</OUTCOME>
    <PROPERTY>position = (826, 454)</PROPERTY>
</VARIABLE>

<VARIABLE TYPE="nature">
    <NAME>J</NAME>
    <OUTCOME>J1</OUTCOME>
    <OUTCOME>J2</OUTCOME>
    <OUTCOME>J3</OUTCOME>
    <PROPERTY>position = (956, 377)</PROPERTY>
</VARIABLE>


<!-- Probability distributions -->
<DEFINITION>
    <FOR>A</FOR>
    <TABLE>0.6 0.09 0.31 </TABLE>
</DEFINITION>

<DEFINITION>
    <FOR>B</FOR>
```

```
    <GIVEN>A</GIVEN>
    <TABLE>0.35 0.5 0.15 0.04 0.4 0.56 0.48 0.48 0.04 </TABLE>
</DEFINITION>

<DEFINITION>
    <FOR>C</FOR>
    <GIVEN>B</GIVEN>
    <TABLE>0.08 0.92 0.51 0.49 0.86 0.14 </TABLE>
</DEFINITION>

<DEFINITION>
    <FOR>D</FOR>
    <GIVEN>A</GIVEN>
    <GIVEN>C</GIVEN>
    <TABLE>
    0.01 0.47 0.52 0.23 0.05 0.72 0.51 0.23 0.26 0.26...
0.32 0.42 0.4 0.46 0.14 0.27 0.38 0.35
    </TABLE>
</DEFINITION>

<DEFINITION>
    <FOR>E</FOR>
    <GIVEN>D</GIVEN>
    <TABLE>0.38 0.62 0.92 0.08 0.12 0.88 </TABLE>
</DEFINITION>

<DEFINITION>
    <FOR>F</FOR>
    <GIVEN>B</GIVEN>
    <TABLE>0.79 0.21 0.43 0.57 0.45 0.55 </TABLE>
</DEFINITION>

<DEFINITION>
```

```
    <FOR>G</FOR>
    <GIVEN>F</GIVEN>
    <GIVEN>H</GIVEN>
    <TABLE>
0.03 0.97 0.39 0.61 0.48 0.52 0.96 0.04 0.51 0.49 0.6 0.4
</TABLE>
</DEFINITION>

<DEFINITION>
    <FOR>H</FOR>
    <TABLE>0.5 0.03 0.47 </TABLE>
</DEFINITION>

<DEFINITION>
    <FOR>I</FOR>
    <GIVEN>E</GIVEN>
    <TABLE>0.42 0.58 0.79 0.21 </TABLE>
</DEFINITION>

<DEFINITION>
    <FOR>J</FOR>
    <GIVEN>D</GIVEN>
    <GIVEN>G</GIVEN>
    <TABLE>
    0.43 0.32 0.25 0.21 0.53 0.26 0.07 0.39 0.54 0.14 0.68...
0.18 0.33 0.4 0.27 0.35 0.56 0.09
    </TABLE>
</DEFINITION>


</NETWORK>
</BIF>
```

A.2 Definição da Consulta

```
<query>
    <variable>I</variable>
</query>
```

A.3 Definição da Evidência

```
<evidences>
    <variable>
        <name>A</name>
        <value>A1</value>
    </variable>

    <variable>
        <name>H</name>
        <value>H2</value>
    </variable>

    <variable>
        <name>J</name>
        <value>J3</value>
    </variable>
</evidences>
```

Apêndice B

Redes utilizadas para testes

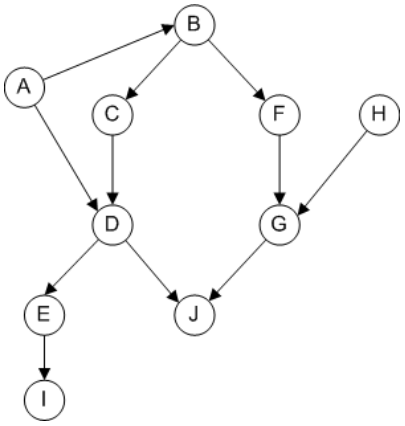


Figura B.1: Rede Bayesiana: Exemplo 01

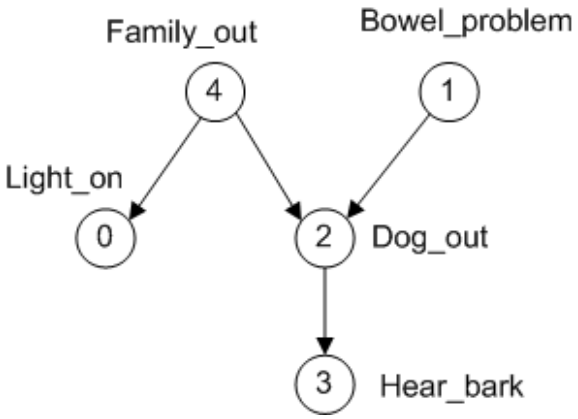


Figura B.2: Rede Bayesiana: DogProblem

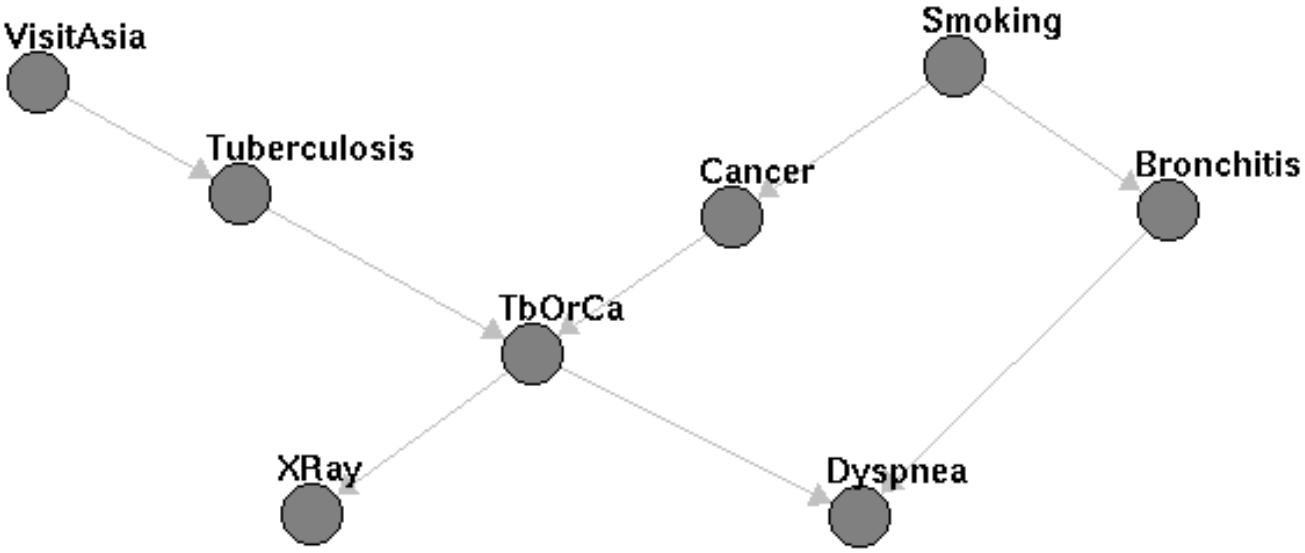


Figura B.3: Rede Bayesiana: Asia

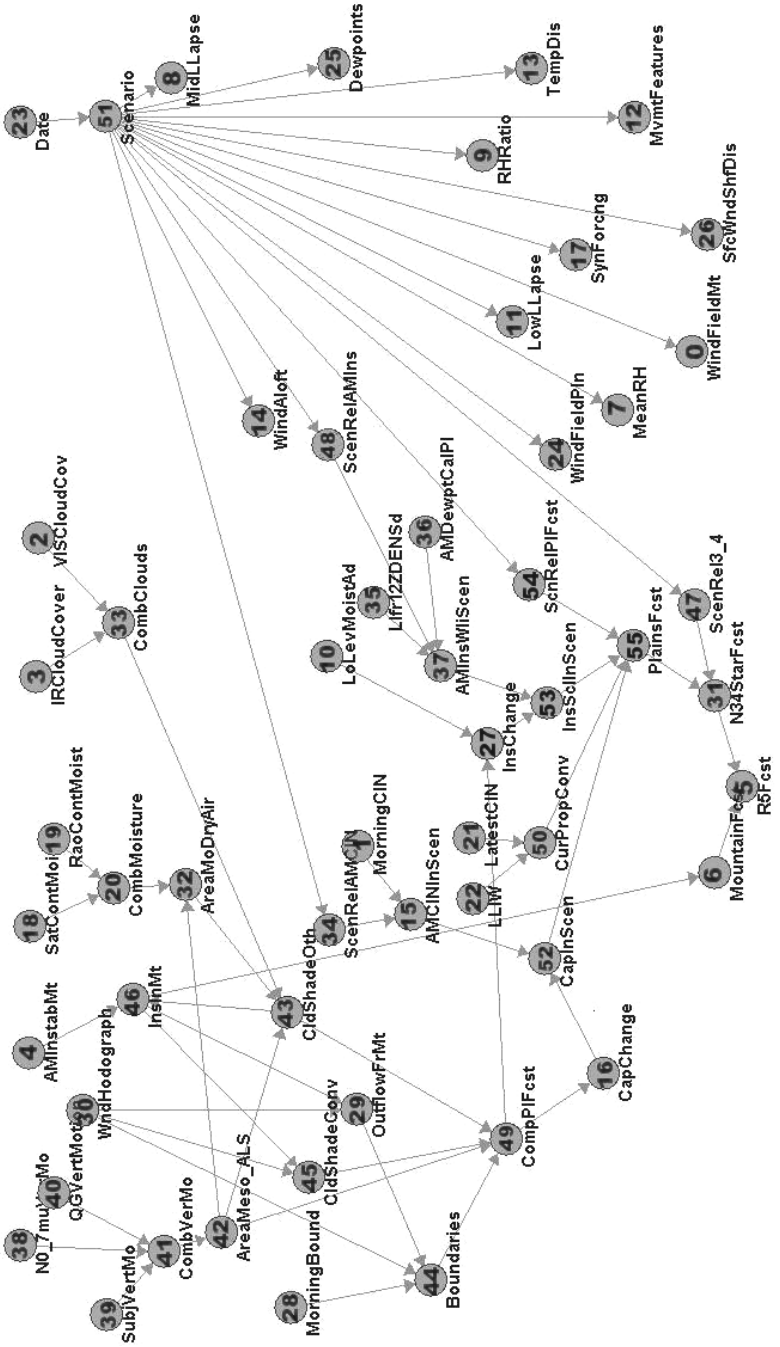


Figura B.4: Rede Bayesiana: Hailfinder25

Referências Bibliográficas

[1] S. Andreassen, *A model-based approach to insulin adjustment*, Proceedings of the Third Conference on Artificial Intelligence in Medicine, Lecture Notes in Medical Informatics, 1991, pp. 239–248.

[2] D. van den Poel M. Egmont-Petersen P. van Kenhove J. Vanthienen B. Baesens, G. Verstraeten, *Bayesian network classifiers for identifying the slope of the customers lifecycle of long-life customers*, European Journal of Operational Research, vol. 156, -, -, 2004, pp. 508–523.

[3] P.P. Shenoy C. Shenoy, *Bayesian network models of portfolio risk and return*, Computational Finance (A W. Lo Y. S. Abu-Mostafa, B. LeBaron and A. S. Weigand, eds.), vol. 156, T2he MIT Press, Cambridge, MA, 1999, pp. 87–106.

[4] F. G. Cozman, *Xml-based bayesnets interchange format*, URL: http://www.cs.cmu.edu/afs/cs/user/fgcozman/www/Research/InterchangeFormat/.

[5] ———, *Generalizing variable elimination in bayesian networks*, Proceedings of the IBERAMIA/SBIA 2000 Workshops (São Paulo), Tec Art Editora, 2000, pp. 27–32.

[6] J. Pearl D. Geiger, T. Verma, *Identifying independence in bayesian networks*, 1990.

[7] A. Mandani D. Heckerman and M. P. Wellman, *Real-world applications of Bayesian networks*, Communications of the ACM **38** (1995), no. 3, 24–26.

[8] Rina Dechter, *Bucket elimination: An unifying framework for probabilistic inference*, Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence UAI-96 (San Francisco) (Eric Horvitz and Finn Jensen, eds.), Morgan Kaufmann Publishers, aug 1–4 1996, pp. 211–219.

[9] O. R. Oellermann G. Chartrand, *Applied and algorithmic graph theory*, McGraw-Hill, New York, USA, 1993.

[10] E. Horvitz, *The lumiere project: Bayesian user modeling for inferring the goals and needs of software users*, Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, 1998, pp. 256–265.

[11] F. V. Jensen, *An introduction to bayesian networks*, Springer Verlag, New York, 1996.

[12] _____, *Bayesian networks and decision graphs*, Statistics for Engineering and Information Science, Springer, New York, 2001.

[13] A. Lucas R. Oliver N. Shikaloff K. C. Chang, R. Fung, *Bayesian network applied to credit scoring*, IMA Journal of Mathematics Applied in Business and Industry **11** (2000), no. 6, 1–18.

[14] R. E. Neapolitan, *Learning bayesian networks*, Prentice Hall, Englewood Cliffs, 1990.

[15] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausive inference*, Morgan Kaufmann, San Francisco, 1988.

[16] _____, *Causality: Models, reasoning and inference*, Cambridge University Press, Cambridge, 2000.

[17] S. Pissanetzky, *Sparse matrix technology*, Academic Press, New York, USA, 2005.

[18] A. H. Saheki, *Construção de uma rede bayesiana aplicada ao diagnóstico de doenças cardíacas*, Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Mestre em Engenharia (2005).

[19] R. Shachter, *Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams)*, Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference (San Francisco, CA.) (S. Moral (eds.) G. F. Cooper, ed.), Morgan Kaufmann Publishers, 1998, pp. 480–487.

[20] J. M. Stern, *Esparsidade, estrutura, estabilidade e escalonamento em Álgebra linear computacional*, IX Escola de Computação (Recife) (Silvio Lemos Meira, ed.), UFPE, 1994, pp. 23–35.

[21] _____, *Decoupling, sparsity, randomization and objective inference*, Relatório Técnico MAC-2006-07 (2006).

[22] J. Sztykiel W. Abramowicz, M. Nowak, *Bayesian networks as a decision support tool in credit scoring domain*, Idea Group Publishing, PA, USA, 2003.

[23] P. Walley, *Statistical reasoning with imprecise probabilities*, Chapman and Hall, London, 1991.