JAIME SHINSUKE IDE

GERAÇÃO DE REDES BAYESIANAS UNIFORMEMENTE DISTRIBUÍDAS

Dissertação Apresentada à Escola Politécnica da Universidade de São Paulo para Obtenção do Título de Mestre em Engenharia.

SÃO PAULO

2002

JAIME SHINSUKE IDE

GERAÇÃO DE REDES BAYESIANAS UNIFORMEMENTE DISTRIBUÍDAS

Dissertação Apresentada à Escola Politécnica da Universidade de São Paulo para Obtenção do Título de Mestre em Engenharia

Área de Concentração: Ciência da Computação

Orientador:
Prof. Dr.
Fabio G. Cozman

São Paulo

2002

Aos meus pais, Mikio e Hoshiko, que tanto se dedicaram à minha formação, a quem devo todas as minhas conquistas.

Agradecimentos

Agradeço ao meu orientador, Prof. Dr. Fabio Gagliardi Cozman, pelas importantes orientações recebidas durante a condução deste trabalho, por ter me orientado pelos caminhos da pesquisa, e por inspirar grande confiança em seus alunos.

Agradeço à Prof. Dra. Márcia D'Elia Branco pelas orientações recebidas na condução da parte estatística do trabalho.

Agradeço ao Prof. Dr. Gilberto Francisco Martha de Souza e ao Prof. Dr. Newton Maruyama pelas sugestões recebidas no Exame de Qualificação.

Agradeço à FAPESP pelo apoio financeiro prestado através da Bolsa de Mestrado (Processo n^o 00/11067-9).

Agradeço ao Dr. Nir Friedman pela sua valiosa sugestão na geração de distribuições condicionais, e ao Dr. Roberto Castelo por ter indicado o trabalho de Melançon.

Agradeço ao Dr. Guy Melançon por confirmar a idéia proposta neste trabalho, e por ter colocado à disposição seu programa DagAlea.

Agradeço ao Dr. Jaap Suermondt e à Dra. Alessandra Potrich por trazerem idéias importantes, e ao Dr. Tomas Kocka, ao Dr. Denver Dash, ao Dr. Eugene Santos e ao Dr. Bruce D'Ambrosio por sugerirem procedimentos valiosos.

Agradeço ao Dr. Haipeng Guo pelas sugestões recebidas na implementação do programa BNGenerator.

Agradeço aos companheiros do Laboratório de Telecomando e Tomada de Decisão do PMR-EPUSP: José Carlos, Marko, Eric, André, Marcelo e Fábio; por criarem um ambiente produtivo de trabalho, e pela amizade demonstrada nos momentos mais difíceis.

Finalmente, agradeço à minha família e a todos os amigos que indiretamente me ajudaram na conclusão deste trabalho de mestrado.

Resumo

Redes Bayesianas são empregadas em Inteligência Artificial para representar incerteza. Não existe, na literatura atual, algoritmo que dê garantias sobre a distribuição de redes Bayesianas geradas aleatoriamente. Este trabalho apresenta novos métodos para geração aleatória de redes Bayesianas. Tais métodos podem ser empregados para se testar algoritmos de inferência e de aprendizado em redes Bayesianas, e para se obter informações sobre propriedades médias de redes Bayesianas. Este trabalho propõe novos algoritmos para geração uniforme de grafos (isto é, todo grafo tem a mesma probabilidade de ser gerado) multi-conectados e polytrees, para um número especificado de nós e de arcos. Após geração uniforme do grafo, distribuições condicionais são construídas, amostrando-se a distribuição Dirichlet. O resultado final do trabalho foi a confecção de um programa livremente distribuído para geração aleatória de redes Bayesianas, BNGenerator. A aplicação de redes Bayesianas geradas aleatoriamente para análise de métodos quasi-Monte Carlo é apresentada.

Abstract

Bayesian networks are employed in Artificial Intelligence to represent uncertainty. No algorithm in the literature currently offers guarantees concerning the distribution of generated Bayesian networks. This work presents new methods for random generation of Bayesian networks. Such methods can be used to test inference and learning algorithms for Bayesian networks, and to obtain insights on average properties of such networks. This work proposes new algorithms that can generate uniformly distributed samples of directed acyclic graphs, like multi-conected networks and polytrees, for a given number of nodes and arcs. After a directed acyclic graph is uniformly generated, the conditional distributions are produced by sampling Dirichlet distributions. The main result of this work is the development of a freely distributed random Bayesian network generator, BNGenerator. An application of random generated Bayesian networks in the analysis of quasi-Monte Carlo methods is presented.

ERRATA - Dissertação de Mestrado, Escola Politécnica (EPUSP)
Jaime Shinsuke Ide

- Pág. 10, linha 5: "valor n".

- Pág. 12, último parágrafo (até o final da Seção 3.1), leia-se:

"Para gerar uma amostra **z** da distribuição de Dirichlet, é necessário amostrar distribuições Gama.(nota de rodapé 1) A amostra **z** é obtida normalizando-se um vetor **y** = $[y_1,..., y_k]$, onde cada y_j é gerado a partir de uma distribuição Gama(nota de rodapé 2) com parâmetro de escala unitário e parâmetro de forma α_j. Suponha então que um grafo direcionado acíclico tenha sido gerado, e uma variável X_i da rede Bayesiana associada a este grafo tenha k valores. A distribuição $p(X_i|\text{pa}(X_i))$ será gerada da seguinte maneira. Uma amostra **z** = $[z_1,...,z_k]$ da distribuição de Dirichlet será gerada para cada valor fixo de pa(X_i), e o valor $p(X_i=j|\text{pa}(X_i))$ será igual a z_i.

Nesse trabalho estamos interessados em um caso particular da distribuição de Dirichlet, o caso em que **z** é gerado de uniformemente no espaço de vetores não-negativos e normalizados. Nesse caso, a geração das amostras **z** é bastante simples. Basta repetir k vezes o seguinte procedimento. Obtenha uma amostra u_j de $U_j \sim U(0,1)$, onde $U(a,b)$ indica a distribuição uniforme entre a e b. Calcule $w_j = \log(1/u_j)$. Normalize o vetor $[w_1,...,w_k]$. O resultado é um vetor não-negativo normalizado e distribuído uniformemente.

Um método alternativo para geração de distribuições uniformes é proposto por Caprile [31].

A maior dificuldade é construir aleatoriamente os DAGs (Etapa 1). Uma discussão mais detalhada é apresentada na próxima seção."

- Pág. 15, na linha 8, leia-se:

" Um estado da cadeia de Markov é *recorrente positivo*, se existe uma probabilidade não nula de se estar retornando ao mesmo estado em um número finito de passos. Caso contrário, o estado é *recorrente nulo*. Uma cadeia de Markov é *recorrente positiva*, se todos os estados forem recorrente positivos. Propriedade que é garantida, se $p_{ii}>0$, para todo i pertencente a **S**."

- Pág. 15, linha 29, leia-se:

"Uma matriz de transição *não negativa*, ou seja, com todos os elementos não-negativos, é denominada [...]"

- Pág. 26, Algoritmo 2:

06. Encontrar o predecessor de j, nó k, no caminho entre i e j.
07. Remover o arco entre k e j e adicionar o arco (i,j) ou (j,i), com probabilidade ½.

- Pág. 29, linha 31, leia-se:

"Como observado na Tabela 3.1, os valores χ^2 diminuem, conforme se aumenta *nTransitions*; e a superioridade do Algoritmo 1 sobre o Algoritmo 2 fica clara, pois o primeiro algoritmo atinge valores aceitáveis de χ^2 muito antes que o segundo algoritmo."

- Na Pág. 30, Tabela 3.1, linha A_5, coluna **Restrições**: onde se lê "(nenhuma)", leia-se "polytree".

- Na Pág. 30, linha 10, leia-se:

"A análise feita a seguir segue o padrão adotado por Spiegel [41]. Comparando-se os valores entre a Tabela 3.1 e 3.2, para a Amostra A_2, como 436,7 < 495,2, a hipótese de uniformidade não é rejeitada, pois o valor χ^2 da amostra está abaixo dos níveis críticos. Para a Amostra A_4, como 331,8 < 355,2, a hipótese de uniformidade também não é rejeitada, pois o valor χ^2 da amostra está abaixo dos níveis críticos. Para a Amostra A_5, como 121,2 < 154,3, a hipótese de uniformidade também não é rejeitada, pois o valor χ^2 da amostra está abaixo dos níveis críticos."

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

INTRODUÇÃO

O ano de 1956 é tomado como sendo o ano de nascimento da Inteligência Artificial e desde então muitos avanços vêm sendo realizados na área [1]. Técnicas de Inteligência Artificial são empregadas tanto em problemas mais simples, como em complexas tentativas de simular a inteligência humana [2]. Além de ser uma atividade de Ciência da Computação, a Inteligência Artificial constitui uma área da Engenharia, que está preocupada como suas técnicas serão implementadas e empregadas na prática. Uma das atividades mais complexas em Inteligência Artificial é o tratamento de incertezas. A Teoria de Probabilidades vem sendo empregada com sucesso no tratamento destas incertezas, particularmente em situações em que medidas de sensores são imprecisas e informações, vagas e incompletas [3].

Modelos probabilísticos baseados em grafos são empregados em problemas mais complexos. Um destes modelos, constituídos por grafos direcionados, são as redes Bayesianas (ou redes de crença, ou redes probabilísticas), que se originaram simultaneamente em Estatística e em Inteligência Artificial [4]. Redes Bayesianas encontram aplicações em diversas áreas, tais como em Medicina (diagnóstico médico), em Administração (sistema de auxílio à tomada de decisão), em Economia (previsão do cenário econômico) e, principalmente, na Engenharia (auto-deteção de falhas, auto-análise de falhas e auto-diagnóstico de máquinas) [5].

Neste trabalho, um novo método para geração uniforme de redes Bayesianas (isto é, cada rede construída possuindo a mesma probabilidade de ser gerada que uma outra qualquer) , é apresentado, através do enfoque em cadeias de Markov. O método desenvolvido apresenta uma solução para um problema que é fácil enunciar,

mas bastante difícil de solucionar. A discussão sobre tal complexidade é apresentado no Capítulo 3.

Por que este tipo de geração é útil? Dois aspectos são suficientes para se mostrar a necessidade de redes Bayesianas aleatórias com distribuição uniforme:

1. Muitos algoritmos de inferência e de aprendizado utilizados em redes Bayesianas devem ser testados, e redes Bayesianas geradas uniformemente oferecem um caminho natural para se produzir experimentos "não viciados", isto é, que resumem o comportamento de todos os tipos possíveis de redes Bayesianas.

2. Propriedades de redes Bayesianas (tais como o número médio de componentes conectados, número médio de variáveis independentes, etc.) são difíceis de serem calculados analiticamente, e redes Bayesianas geradas uniformemente podem ser utilizadas para se estimar tais propriedades de forma empírica.

O fato de redes Bayesianas ocuparem uma posição promissora para modelamento de incerteza em Inteligência Artificial [4] poderia levar a concluir que algoritmos para geração uniforme de redes Bayesianas são amplamente disponíveis. Entretanto, este não é o caso. Não existe, até o momento, geradores que dêem garantia sobre a distribuição das redes geradas.

Para se investigar propriedades relevantes em problemas reais, deve-se gerar redes Bayesianas constituídas por grafos sujeitas à restrições (número de nós, número de arcos, grau dos nós, número de pais para cada nó, etc). A geração de grafos com tais propriedades, garantindo-se a distribuição uniforme, é uma questão desafiadora. Uma das características importantes do método apresentado é o fato de ser flexível o suficiente para possibilitar a adição destas restrições. Esta característica torna o método uma ferramenta efetivamente útil e foi uma das principais contribuições deste trabalho.

No Capítulo 2, apresenta-se um breve resumo sobre redes Bayesianas. A leitura deste capítulo pode ser dispensada aos que já estão bastante familializados com o assunto. No Capítulo 3, apresenta-se a discussão sobre geração aleatória e uniforme de redes Bayesianas, assim como a solução proposta, que é a principal contribuição deste trabalho. No Capítulo 4, apresenta-se um programa, BNGenerator, que implementa os algoritmos desenvolvidos. No Capítulo 5, apresenta-se uma aplicação

de redes Bayesianas geradas aleatoriamente na análise de algoritmos de inferência. No Capítulo 6, apresentam-se conclusões gerais deste trabalho.

Capítulo 2

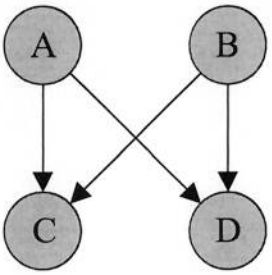
REDES BAYESIANAS

O objetivo deste capítulo é fixar a terminologia e indicar as referências importantes. Nas Seções 2.1 e 2.2, apresentam-se um breve resumo sobre as redes Bayesianas e a terminologia empregada. Na Seção 2.3, diversos métodos de inferência em redes Bayesianas são discutidos. Dentre os diferentes métodos de inferência, métodos MCMC (*Monte Carlo Markov Chain*) são descritos com mais detalhe na Seção 2.4, pois tais métodos são analisados no Capítulo 5.

2.1 Redes Bayesianas

Redes Bayesianas são grafos direcionados acíclicos (abreviados por DAG - *Directed Acyclic Graph*) associados a um conjunto de distribuições de probabilidade [6]. Em redes Bayesianas, "nó" e "variável" são termos análogos. A diferença é que "nó" é empregado quando se fala em grafos (ressaltando o aspecto gráfico), e "variável" , em distribuições condicionais (ressaltando o aspecto paramétrico). Uma rede Bayesiana é constituída por uma parte *qualitativa* (relações de dependência entre variáveis), que é a estrutura do grafo (conjunto de nós interligados por arcos direcionados), e uma parte *quantitativa*, que são as distribuições de probabilidades associadas a cada uma das variáveis (nós) do grafo. As distribuições de probabilidade quantificam a relação de dependência entre as variáveis.

Considere um DAG, onde cada nó é associado a uma variável X_i. Denote por $\text{pa}(X_i)$ o conjunto de variáveis que são pais diretos da variável X_i na rede. Cada variável X_i é associada a uma distribuição $p(X_i|\text{pa}(X_i))$ (uma variável sem pais é



	A=0	A=1
P(A)	x_1	y_1

	B=0	B=1
P(B)	x_2	y_2

P(C/A,B)	C=0	C=1
A=0, B=0	x_3	y_3
A=0, B=1	x_4	y_4
A=1, B=0	x_5	y_5
A=1, B=1	x_6	y_6

P(D/A,B)	D=0	D=1
A=0, B=0	x_7	y_7
A=0, B=1	x_8	y_8
A=1, B=0	x_9	y_9
A=1, B=1	x_{10}	y_{10}

Figura 2.1. Rede Bayesiana com 4 nós e suas distribuições de probabilidade.

associada à distribuição marginal $p(X_i)$).

A propriedade básica em redes Bayesianas é que toda variável X_i é independente de todos os antecessores de X_i, condicional nos pais de X_i. Isto garante que a distribuição conjunta seja definida por [6]:

$$p(\mathbf{X}) = \prod_i p(X_i|\text{pa}(X_i))\,, \tag{2.1}$$

onde $\mathbf{X}$ é o conjunto de todas as variáveis. Essa expressão é importante por garantir que uma distribuição conjunta, potencialmente complexa, possa ser representada através do produto de blocos menores.

Por exemplo, considere uma rede Bayesiana com 4 nós, com variáveis assumindo dois valores: 0 ou 1, onde cada variável está associada a uma distribuição de probabilidade (Figura 2.1). Note que $x_i + y_i = 1$ $(1 \leq i \leq 10)$. A distribuição conjunta é expressa por:

$$p(A, B, C, D) = p(A).p(B).p(C|A, B).p(D|A, B). \tag{2.2}$$

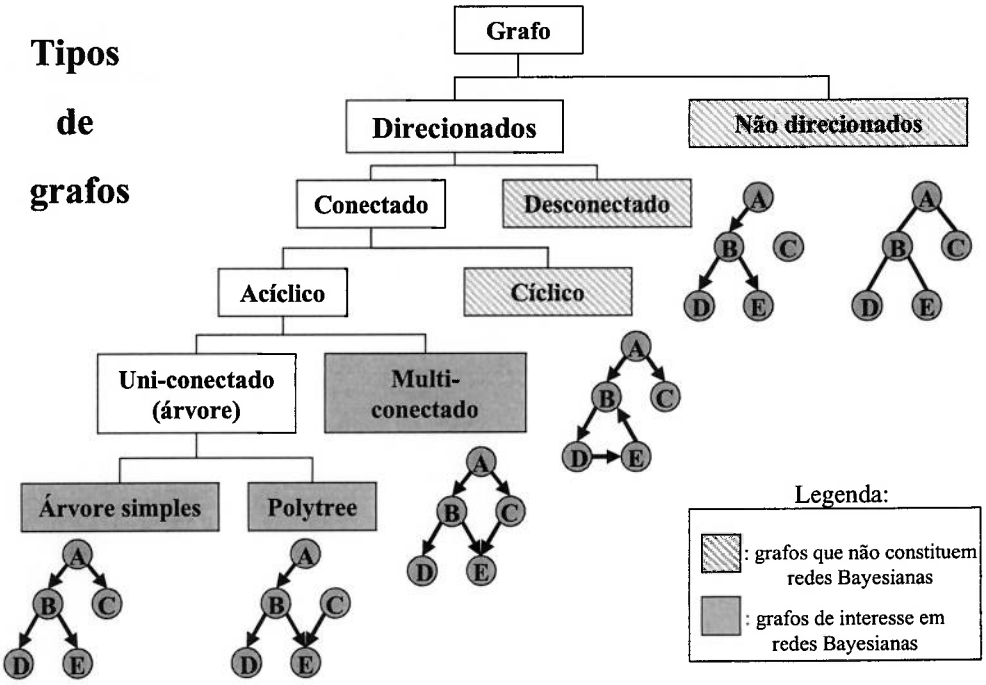


Figura 2.2. Tipos de estruturas de grafos empregados em redes Bayesianas.

2.2 Terminologia básica de grafos

Nesta seção é estabelecida uma terminologia de grafos que será empregada no restante do trabalho. Maiores detalhes sobre teoria de grafos são encontrados em [7]; [8].

Um *grafo direcionado* é denotado por $G = (\mathbf{V}, \mathbf{E})$, onde $\mathbf{V}$ é um conjunto de nós e $\mathbf{E}$ é um conjunto de arcos. Todo arco (u, v) contido em $\mathbf{E}$ é *direcionado*, ou seja, parte de um nó u (pai) e chega a outro nó v (filho). Dois nós u e v são *adjacentes* se o arco (u, v) pertence ao conjunto $\mathbf{E}$. Um *caminho* (*path*) é uma seqüência de nós de modo que cada par de nós consecutivos são adjacentes. Um caminho é um *ciclo* se contém mais de dois nós, e o primeiro e o último nó são iguais. Um ciclo é *direcionado* se é possível alcançar um mesmo nó seguindo arcos que estão na mesma direção. Um grafo é *conectado* se existe caminho entre quaisquer dois nós pertencentes ao conjunto de vértices $\mathbf{V}$. Um *grafo direcionado acíclico* (*Directed Acyclic Graph*-DAG) é um grafo que não contém nenhum ciclo direcionado. Para todo nó v contido em $\mathbf{V}$, o *grau* (*degree*) é o número de arcos contendo v. Em outras palavras, *grau* é o número total de arcos entrando ou saindo do nó v.

Um grafo é uma *árvore* (*tree*), se existe exatamente um caminho entre quaisquer dois nós; caso contrário, o grafo será *multi-conectado*. Se todo nó da árvore possuir apenas um pai, ela será *árvore simples*; senão, ela será *polytree*.

Na Figura 2.2, apresenta-se uma classificação dos tipos de grafos, segundo características previamente descritas. Os DAGs, tais como árvores simples, polytrees e grafos multi-conectados, são grafos utilizados em redes Bayesianas. Os grafos não direcionados, desconectados e cíclicos não constituem redes Bayesianas.

2.3 Algoritmos para inferência em redes Bayesianas

Normalmente, as redes Bayesianas são utilizadas para realizar cálculos de probabilidade condicional em um conjunto de variáveis observadas, o que se denomina por *inferência*. Por exemplo, a evidência $\mathbf{E} = \{X_4 = 0, X_6 = 1\}$ indica que as variáveis X_4 e X_6 estão observadas e portanto fixas. Para calcular a distribuição condicional de uma variável qualquer X_Q, dadas as observações em $\mathbf{E}$, é preciso computar:

$$p(X_Q|\mathbf{E}) = \frac{p(X_Q, X_4 = 0, X_6 = 1)}{p(X_4 = 0, X_6 = 1)}. \tag{2.3}$$

A expressão geral para inferências em redes Bayesianas é:

$$p(X_Q|\mathbf{E}) = \frac{\sum_{\mathbf{X}\setminus\{X_Q,\mathbf{E}\}} p(\mathbf{X})}{\sum_{\mathbf{X}\setminus\mathbf{E}} p(\mathbf{X})}, \tag{2.4}$$

onde X_Q é a variável requerida (ou um conjunto de variáveis) e $\mathbf{E}$ é um conjunto de variáveis observadas. $\mathbf{X}\setminus\{X_Q, \mathbf{E}\}$ representa o conjunto de todas as variáveis contidas em $\mathbf{X}$, excluindo X_Q e $\mathbf{E}$.

Existem algoritmos eficientes para realizar estas operações em redes Bayesianas genéricas. Dois tipos de algoritmos são normalmente utilizados: algoritmos baseados em *junction trees* [4] e algoritmos baseados em eliminação de variáveis [9]; [10]; [11]. Uma apresentação compacta de eliminação de variáveis está em [12]. Todos esses algoritmos são exatos e podem ser exponenciais no tamanho do maior clique existente no grafo da rede triangulado [13]. Isto faz com que algoritmos exatos sejam potencialmente intratáveis.

O caminho para realizar inferências em Redes Bayesianas complexas são os algoritmos aproximados. As aproximações que ocupam menos memória são baseadas

em amostragem, popularmente conhecidos por métodos de *Monte Carlo.*[1]

A idéia fundamental do método de Monte Carlo é obter estimativas a partir de amostras[2] da população geradas por simulação. Por se tratar de uma idéia geral, métodos Monte Carlo são aplicados em diversas áreas. Uma explicação simples e intuitiva do método é encontrada em [16].

Em redes Bayesianas, métodos de Monte Carlo são aplicados no cálculo aproximado da Equação (2.4). Considere $f(X_Q) = p(X_Q|\mathbf{E})$; a partir de n amostras, obtém-se a estimativa:

$$E[f(X_Q)] \approx \widehat{f(X_Q)} = \frac{1}{n}\sum_{s=1}^{n} f(X_Q^{(s)}), \tag{2.5}$$

onde $X_Q^{(s)}$ são amostras de $p(X_Q|\mathbf{E})$. A questão do método de Monte Carlo resume-se em como gerar estas amostras. Um dos métodos pioneiros para geração destas amostras em redes Bayesianas é o *logic sampling* [17].

O problema da aplicação pura e simples do método de Monte Carlo é que, quanto maior for o número de evidências, mais amostras devem ser geradas para que uma amostra seja aceita. Se uma das evidências tem pouca probabilidade para ocorrer, muitas amostras terão que ser geradas até que se consiga uma que seja consistente. Uma maneira de superar este inconveniente é o método conhecido por *amostragem por importância* (*importance sampling*), empregado largamente em econometria na década de 80 e trazido para a área Bayesiana por [18] e [19]. Sua idéia fundamental em redes Bayesianas é concentrar o número de amostras na região de interesse, isto é,

[1] Aplicações de métodos Monte Carlo para Raciocínio Probabilístico em sistemas de processamento embarcado são descritas por Ide e Cozman [14]; Ramos, Cozman e Ide [15].

[2] Uma observação deve ser feita com relação ao emprego do termo "amostra" ao longo do texto. A palavra "amostra" é empregada em dois sentidos. O primeiro siginificado é *amostra* como sendo uma única rede Bayesiana. Este significado é utilizado durante a simulação MCMC (seções 2.4 e 5.2) , onde várias redes (amostras) são selecionadas para realização da estimativa de Monte Carlo. Quando se diz: "tamanho da amostra" , o termo é utilizado com o segundo significado: *amostra* como sendo um conjunto de redes Bayesianas. Diz-se que, ao se gerar um conjunto de redes aleatoriamente dentro de todas as combinações possíveis, obtém-se ao final um conjunto (amostra) de redes Bayesianas. Este significado é empregado principalmente nas Seções 3.7 e 5.3. Na Seção 5.4, os dois significados do termo "amostra" são empregados, sendo possível distinguí-los, através do contexto em que se encontram. Optou-se por utilizar os dois sentidos para um mesmo termo, para não se prejudicar a exatidão do que se quer significar. Este "conflito" entre significados decorre da junção que é feita neste trabalho de duas áreas distintas de pesquisa.

na região em que as evidências ocorrem, de modo a não ter que desperdiçar tantas amostras. A questão difícil é encontrar a função de importância que "detecta" esta região de interesse (conforme as amostras são geradas) sem que a variância da estimativa seja prejudicada [20].

2.4 Métodos MCMC de inferência

Uma importante classe de métodos aproximados baseados em amostragem são os métodos MCMC [21]; [22]; [23]. Esses métodos se baseiam na aplicação de técnicas de Monte Carlo em uma cadeia de Markov especialmente construída para simular o modelo de interesse. A idéia fundamental é gerar amostras dependentes, em que a amostra posterior depende apenas da anterior. As amostras dependentes são geradas segundo uma regra que garante a convergência da cadeia de Markov para uma distribuição limite, tal que a distribuição limite é a distribuição da probabilidade condicional de interesse.

Vários métodos utilizam essa abordagem; entre esses métodos, *Gibbs sampling* é um método de implementação relativamente simples em redes Bayesianas. Gibbs sampling é um caso específico do algoritmo *Metropolis-Hastings* [24], e foi proposto inicialmente por Geman e Geman [25].

A idéia básica do Gibbs sampling é utilizar a probabilidade condicional total (*pct*) de cada variável para gerar as amostras. A *pct* representa a distribuição de probabilidade condicional de uma variável da rede, dado o estado de todas as demais variáveis. Por exemplo, para a variável A da rede Bayesiana da Figura 2.1, a *pct* é computado por:

$$pct(A) = \frac{p(A,B,C,D)}{p(B,C,D)} \propto p(A).p(C|A,B).p(D|A,B), \tag{2.6}$$

onde as variáveis B, C e D são observadas. Note que *pct* é uma distribuição, e portanto, obtém-se dois valores numéricos $pct(A=0)$ e $pct(A=1)$, que ao final, são normalizados.

O emprego da *pct* é mostrado na Figura 2.3, que implementa Gibbs sampling da maneira mais simples possível. X_Q é a variável de interesse, $\mathbf{E}$ é o conjunto de variáveis observadas e n é o número total de amostras a serem geradas. Ao final,

Algoritmo *Gibbs sampler*

Entrada: variáveis solicitadas (X_Q), variáveis observadas ($\mathbf{E}$), número de amostras geradas (n)

Saída: probabilidade condicional, $p(X_Q|\mathbf{E})$

01. Inicializar valores de todas as variáveis não observadas;
02. Repetir o seguinte ciclo n-vezes;
03. Percorrer todas as variáveis e para cada uma das variáveis não observadas;
04. Calcular a probabilidade condicional total (*pct*) e normalizar ao final;
05. Gerar uma variável aleatória $u \sim U(0,1)$;
06. Gerar o próximo valor da variável atualmente percorrida em função da *pct* e da variável aleatória u;
07. Incrementar a posição do valor assumido pela variável X_Q no vetor n_Q;
08. Computar a estimativa da probabilidade condicional $p(X_Q|\mathbf{E})$ dado por $\frac{n_Q}{n}$;
09. Retornar a estimativa $p(X_Q|\mathbf{E})$.

Figura 2.3. Implementação simples do Gibbs sampling com variáveis discretas.

o algoritmo retorna uma estimativa para a expressão $p(X_Q|\mathbf{E})$. Este é o algoritmo que será utilizado nos testes do Capítulo 5.

Sabe-se que quanto maior o número de amostras, melhor será a estimativa fornecida. E uma questão difícil de ser avaliada é determinar o número necessário de amostras, para obter a precisão requerida. Habitualmente, assume-se um valor$n$ alto suficiente para superar a fase de *burn-in* (fase de convergência para a distribuição limite) [26]. Estas e outras questões são discutidas por Casella e George [27] que explicam de uma maneira clara o funcionamento do algoritmo de Gibbs sampling, ilustrando-o para exemplos clássicos da literatura de Estatística Bayesiana .

Capítulo 3

DESENVOLVIMENTO DE UM GERADOR ALEATÓRIO E UNIFORME DE REDES BAYESIANAS

Este capítulo descreve a principal contribuição deste trabalho: um novo método para geração aleatória e uniforme de redes Bayesianas segundo um enfoque em cadeias de Markov.

Inicialmente, na Seção 3.1, apresentam-se as etapas envolvidas na geração aleatória de redes Bayesianas. Depois, na Seção 3.2, discute-se porque a geração aleatória de redes Bayesianas é difícil. Na Seção 3.3, apresenta-se um resumo sobre a teoria de cadeias de Markov, e como ela é empregada para geração de grafos. Na Seção 3.4, apresenta-se como matrizes de transição simétricas são obtidas. A *simetria* é uma propriedade importante para o funcionamento do método proposto. Nas Seções 3.5 e 3.6, apresentam-se algoritmos que implementam o método proposto. E na Seção 3.7, apresentam-se resultados experimentais obtidos a partir dos algoritmos implementados e também uma análise de desempenho.

3.1 Geração aleatória de redes Bayesianas

A geração aleatória de redes Bayesianas se dá em duas etapas:

1. Gerar DAGs (informação qualitativa);

2. Gerar distribuições de probabilidade condicional associadas a cada nó deste grafo (informação quantitativa).

A execução da Etapa 2 é mais simples. Dado um DAG (Etapa 1), é relativamente fácil gerar as distribuições condicionais associadas. Considere a estrutura da rede Bayesiana apresentada no Capítulo 2 (Figura 2.1), um grafo com 4 nós. Para cada nó deste grafo é necessário gerar suas respectivas distribuições de probabilidade. Para tal, geram-se valores de x_i e y_i, tal que $x_i+y_i = 1$ $(1 \leq i \leq 10)$. Gerar distribuições de probabilidade de forma uniforme significa neste caso gerar pontos (x_i, y_i) distribuídos uniformemente ao longo da reta definida pela equação $x_i + y_i = 1$. Se as variáveis assumissem três valores (x_i, y_i, z_i), as distribuições de probabilidade definiriam um plano, e a uniformidade resultaria em pontos uniformemente distribuídos ao longo do plano $x_i + y_i + z_i = 1$.

Um esquema genérico para geração destas distribuições é produzir amostras da distribuição de *Dirichlet*. A distribuição de *Dirichlet* é definida a seguir [28]. O vetor de variáveis aleatórias $\mathbf{X} = [X_1, \ldots, X_k]$ possui distribuição de *Dirichlet* com vetor de parâmetros $\alpha = [\alpha_1, \ldots, \alpha_k]$ $(\alpha_i > 0;\ i = 1, \ldots, k)$, se a distribuição densidade probabilidade de $\mathbf{X}$ satisfaz a seguinte propriedade:

Seja $\mathbf{x} = [x_1, \ldots, x_k]$ um ponto qualquer de R^k, tal que $x_i > 0$ para $i = 1, \ldots, k$ e $\sum_{i=1}^{k} x_i = 1$, então:

$$f(\mathbf{x}|\alpha) = \frac{\Gamma(\alpha_1 + \ldots + \alpha_k)}{\Gamma(\alpha_1) \ldots \Gamma(\alpha_k)} x_1^{\alpha_1 - 1} \ldots x_k^{\alpha_k - 1}, \tag{3.1}$$

e $f(\mathbf{x}|\alpha) = 0$ para qualquer outro ponto $\mathbf{x} \in R^k$.

Para gerar uma amostra $\mathbf{x}$ da distribuição de *Dirichlet*, amostras de distribuições *Gama* são necessárias [1]. A amostra $\mathbf{x}$ é obtida, normalizando-se o vetor $\mathbf{y} = [y_1, \ldots, y_k]$, onde y_i é uma amostra da distribuição *Gama*[2] com parâmetro de escala unitário e parâmetro $\alpha_i \in \alpha$. A amostra gerada $\mathbf{x} = [x_1, \ldots, x_k]$ da distribuição de *Dirichlet* será a distribuição condicional $p(X_i|\text{pa}(X_i))$ para um valor fixo de $\text{pa}(X_i)$, onde X_i assume k valores. Para cada valor fixo de $\text{pa}(X_i)$, será necessário gerar uma amostra da distribuição de *Dirichlet*.

[1] Esquema para geração de amostras da distribuição de *Dirichlet* sugerido por Nir Friedman [29].

[2] Um esquema para geração de amostras da distribuição *Gama* é apresentado por Ripley [30].

Para gerar distribuições condicionais uniformes, toma-se todos os valores de α assumindo 1. Neste caso, a implementação é mais simples. Basta gerar k valores de $\log(1/u)$, onde u é uma amostra de $U \sim (0, 1)$, e normalizá-los ao final. Um método alternativo para geração de distribuições uniformes é proposto por Caprile [31].

A maior dificuldade é construir aleatoriamente os DAGs (Etapa 1). Uma discussão mais detalhada é apresentada na próxima seção.

3.2 Discussão sobre geração aleatória de DAGs

Após pesquisas [3], constatou-se que, não existe atualmente nenhum tipo de trabalho que trata da geração uniforme de redes Bayesianas. Não é fácil gerar grafos de interesse em redes Bayesianas, garantindo-se a uniformidade da amostra. Uma possível solução seria gerar todas as combinações dos grafos, para depois realizar um sorteio uniforme. Mas, esta solução é inviável uma vez que o número de combinações possíveis de grafos tem um crescimento *super-exponencial* [32]. Por exemplo, com 8 nós, existem aproximadamente $7,8\text{x}10^{11}$ combinações de DAGs possíveis [33]. Uma coletânea de métodos de construção de grafos aleatórios em geral é apresentada por Tinhofer [34].

Em redes Bayesianas, é interessante que se consiga:

Controle sobre a distribuição dos grafos gerados . Na maioria dos casos, os grafos são gerados heuristicamente, o que não garante nenhum controle sobre a geração. Por exemplo, pode-se partir de um grafo vazio e ir adicionando arcos de maneira aleatória até atingir um número total desejado de arcos ou de nós. Um exemplo típico deste tipo de construção pode ser visto no trabalho de Xiang e Miller [35]. Uma interessante solução é apresentada por Melançon [36], e sua idéia é aperfeiçoada neste trabalho, fazendo-se as adaptações necessárias para redes Bayesianas.

Controle sobre características do grafo gerado . Gerar grafos distribuídos uniformemente no domínio de todos os grafos possíveis com $nNodes$ nós não é, a princípio, algo tão interessante em redes Bayesianas; pois, em aplicações

[3] A discussão encontrada nesta seção é fruto de sugestões e indicações recebidas na lista de Discussão Eletrônica da UAI [29].

reais, as redes Bayesianas possuem baixa *densidade* de arcos, ou seja, pequeno número de arcos saindo ou chegando em um nó, e em contrapartida, grafos gerados uniformemente com um dado número de nós possuem alta densidade, devido ao alto número de combinações possíveis para um maior número de arcos. Ou seja, gerando-se uma amostra aleatória e uniforme de grafos com $nNodes$ nós, obtém-se um maior número de grafos com maior densidade, pois estão em maior número. Assim, não basta apenas gerar uniformemente grafos com um dado número de nós para se ter uma amostra representativa de redes Bayesianas, mas é necessário gerar DAGs com baixa densidade. Assumindo o número total de arcos ($maxArcs$) como um bom indicador do quão denso é um grafo, pode-se dizer que o problema se resume em gerar uniformemente DAGs com $nNodes$ nós e $maxArcs$ número total de arcos.

3.3 Geração de DAG conectado segundo uma cadeia de Markov

O método proposto neste trabalho é gerar grafos segundo um enfoque de *cadeias de Markov*. Este enfoque é flexível o suficiente para se estar adaptando a diversos tipos de estruturas de interesse, com suas respectivas características desejadas. Significa que o método apresentado é genérico o suficiente para não só gerar grafos multi-conectados e polytrees, mas também para outros tipos de estruturas.

A metodologia adotada foi diretamente inspirada nos trabalhos de Sinclair [37] e de Melançon e Bousque [36]. Sinclair apresenta um estudo mais detalhado sobre a convergência de cadeias de Markov. A principal diferença deste trabalho com o de Melançon e Bousque é que os seus grafos gerados podem ser desconectados, o que acarreta uma considerável diferença nas provas de *irredutibilidade* da cadeia de Markov gerada, condição necessária para convergência da cadeia.

Alguns conceitos fazem-se necessários para a compreensão da metodologia proposta. Considere uma cadeia de Markov, $\{X_n, n \geq 0\}$, com espaço finito de estados $\mathbf{S}$ [38]; [39] com uma matriz de transição NxN, $P = (p_{ij})_{ij=1}^{N}$, onde $p_{ij} = Pr(X_{t+1} = j|X_t = i)$, para todo t; e com distribuição inicial representado pelo vetor $\pi^{(0)}$. A *probabilidade de transição* no passo n é dada por $P^n = p_{ij}^{(n)} = Pr(X_{t+n} = j|X_t = i)$,

independentemente de t. Uma cadeia de Markov é *irredutível* se $p_{ij}^{(n)} > 0$, para todo i e $j \in \mathbf{S}$ e algum $n \geq 0$. Uma cadeia de Markov é irredutível se, somente se, todos os pares de estados se intercomunicarem. Uma cadeia de Markov é *aperiódica* se todos os estados forem aperiódicos. Um estado é aperiódico se o *máximo divisor comum* (mdc) do conjunto de todos os n passos, em que $p_{ii}^{(n)} > 0$, for 1 (isto é, $mdc = G.C.D.\{n|p_{ii}^{(n)} > 0\} = 1$), ou seja, se o período for 1. A aperiodicidade é assegurada se $p_{ii} > 0$, para todo $i \in \mathbf{S}$, ou seja, todas as probabilidades da diagonal positivas. Uma cadeia de Markov é dita *recorrente positiva*, se para todo estado existe uma possibilidade de se estar retornando ao mesmo estado em um número finito de passos, característica que é garantida se $p_{ii} > 0$, para todo $i \in \mathbf{S}$. Note que a condição $p_{ii} > 0$ garante tanto a aperiodicidade como a recorrência positiva da cadeia. Um estado da cadeia de Markov é chamado *ergódico* se ele for recorrente positivo e aperiódico.

Seja $\pi = \{\pi_j, j \in \mathbf{S}\}$ uma distribuição de probabilidade. Esta distribuição é chamada *distribuição estacionária* com matriz de transição P, se $\pi = \pi P$ (isto é, $\pi_j = \sum_{i \in S} \pi_i p_{ij}, j \in \mathbf{S}$). Assim, uma vez que o estado com distribuição estacionária é atingido, os estados subseqüentes terão a mesma distribuição π. Demonstra-se que uma cadeia de Markov irredutível e aperiódica pertence a dois tipos de classes: todos os estados recorrentes nulos ou todos os estados recorrentes positivos, ou seja, $\lim_{n \longrightarrow \infty} p_{ij}^{(n)} = \pi_j$, para todo estado i e $j \in \mathbf{S}$ [38]. Neste último caso, em que todos os estados são recorrentes positivos, o conjunto $\{\pi_j, \forall j \in \mathbf{S}\}$ é uma distribuição estacionária. Deste modo, a recorrência positiva assegura a existência de uma distribuição estacionária, que por sua vez é também limite. A aperiodicidade da cadeia garante que a distribuição estacionária seja única [23]. Deste modo, conclui-se que toda cadeia de Markov irredutível, recorrente positiva e aperiódica possui uma distribuição estacionária única, e esta cadeia de Markov é chamada *ergódica*.

Uma cadeia de Markov é dita *homogênea*, se ela possuir uma matriz de transição estacionária, ou seja, independente do passo n em que se encontra. Uma matriz de transição *não negativa*, ou seja todas as probabilidades não negativas, é denominada *duplamente estocástica* se todas as somas das linhas e colunas assumem 1 (isto é, se $\sum_{j=1}^{s} p_{ij} = 1$, para todo i e $\sum_{i=1}^{s} p_{ij} = 1$, para todo j, onde s é o número total de estados de $\mathbf{S}$). Toda matriz com esta característica possui uma distribuição

estacionária uniforme.

É possível gerar grafos aleatórios simulando uma cadeia de Markov da seguinte maneira. Cada grafo possível é um estado da cadeia. Para se ter uma cadeia de Markov, basta que seja possível "mover" de um grafo para outro de maneira probabilística tal que o próximo grafo gerado dependa somente do grafo atual. Esta cadeia de Markov será irredutível se for possível alcançar qualquer grafo a partir de um outro qualquer, obedecendo uma certa regra de transição. Além disso, a cadeia será aperiódica sempre que existir uma possibilidade de que o próximo grafo gerado ser o mesmo do atual. Se a matriz de transição for duplamente estocástica, a distribuição estacionária será única e uniforme.

3.4 Construção de matrizes de transição simétricas

Uma cadeia de Markov é definida pela sua matriz de transição, composta por probabilidades de transição entre um estado e outro. Uma característica importante desta matriz, para o funcionamento do método proposto, é a sua *simetria*. Pois, desta simetria decorre outras propriedades importantes, como visto nas seções seguintes. Nesta seção, apresenta-se como matrizes de transições simétricas são obtidas.

Sinclair [37] e Melançon [36] propõe que as transições da cadeia de Markov sejam efetuadas conforme dois tipos de operações: adição e remoção de arcos. Este esquema de transição é apresentado no seguinte exemplo de cadeia de Markov.

Considere um grafo com 4 nós (Figura 3.1), sendo ele o estado inicial S_1 da cadeia de Markov; e duas operações possíveis: adição e remoção de arcos. O objetivo é simular DAGs conectados. Para que uma transição entre um estado e outro ocorra, suponha que um arco (i, j) seja sorteado com probabilidade $p(arco(i, j)) = \frac{1}{12}$ (pois existem 12 combinações de se sortear 2 números, no total de 4, sem reposição). Se o arco existir no grafo, ele será adicionado, desde que não torne o grafo acíclico. Senão, o arco será removido, desde que isso não torne o grafo desconectado. Não satisfazendo as condições "acíclico" e "conectado" [4] , não há transição (o estado seguinte e o atual são iguais). O esquema de transição é ilustrado na Figura 3.2.

De S_1 existem 3 possíveis estados para transição: S_2, S_3 e S_4. Para que a

[4]Melançon [36] não leva em consideração a condição de grafo "conectado" em seu método.

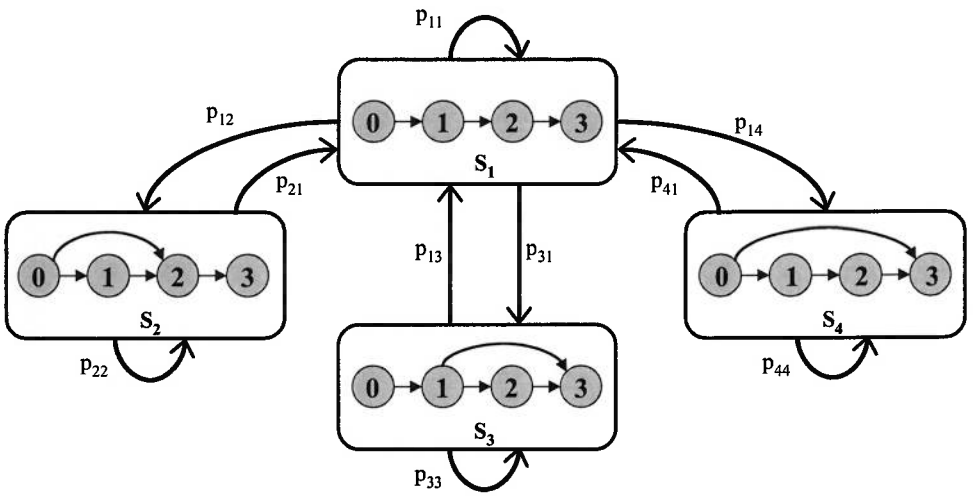


Figura 3.1. Estados de uma cadeia Markov representando grafos com 4 nós, conforme possíveis operações de adição e remoção de arcos.

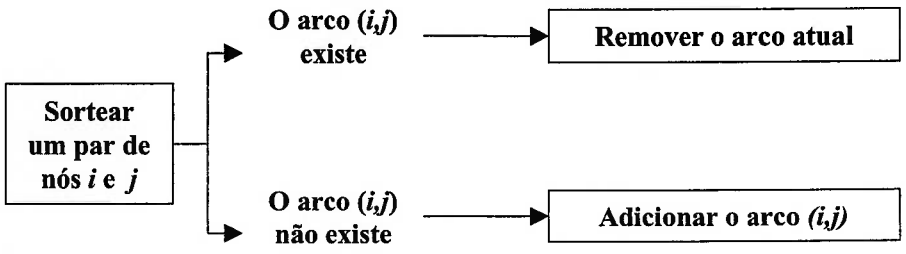


Figura 3.2. Esquematização da transição com 2 operações: adição e remoção.

transição do estado S_1 para S_2 seja efetuada, o arco $(0,2)$ deve ser sorteado. Como o arco $(0,2)$ não existe no estado S_1 e o grafo resultante permanece acíclico, efetua-se a operação de adição. A probabilidade de transição de S_1 para S_2 é $p_{12} = \frac{1}{12}$. Para passar do estado S_2 para S_1, deve-se sortear novamente o arco $(0,2)$ e efetuar a operação de remoção. A probabilidade de transição de S_2 para S_1 é novamente $p_{21} = \frac{1}{12}$. Note que todas as probabilidades de transição entre dois estados vizinhos será $p_{ij} = \frac{1}{12}$, onde i e $j \in \mathbf{S}$. A probabilidade de se permanecer no mesmo estado S_1 será $p_{11} = 1 - \frac{1}{4} = \frac{3}{4}$, pois existem 3 estados possíveis a partir deste com probabilidade de transição valendo $\frac{1}{12}$. Deste modo obtém-se uma matriz de transição de estados da forma (sendo que existem 446 combinações diferentes de DAGs conectados com 4 nós)[5] :

	S_1	S_2	S_3	S_4	$\cdots$	S_{446}
S_1	$\frac{3}{4}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\cdots$	
S_2	$\frac{1}{12}$	p_{22}	0	0	$\cdots$	
S_3	$\frac{1}{12}$	0	p_{33}	0	$\cdots$	
S_4	$\frac{1}{12}$	0	0	p_{44}	$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
S_{446}						$p_{446,446}$

Generalizando o exemplo, para se obter a simetria, a idéia é garantir que a matriz de transição (para geração de grafos segundo uma cadeia de Markov) seja da seguinte forma:

$$\begin{bmatrix} \left(1 - \sum_{i=2}^{s} p_{1i}\right) & p_{12} & p_{13} & \cdots & p_{1s} \\ p_{21} & \left(1 - \sum_{i=1, X \neq 2}^{s} p_{2i}\right) & p_{23} & \cdots & p_{2s} \\ p_{31} & p_{32} & \left(1 - \sum_{i=1, i \neq 3}^{s} p_{3i}\right) & \cdots & p_{3s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{s1} & p_{s2} & p_{s3} & \cdots & \left(1 - \sum_{i=1}^{s-1} p_{si}\right) \end{bmatrix},$$

onde s é o número total de estados de $\mathbf{S}$ (ou seja, o número total de DAGs conectados possíveis com um determinado número de nós) e $p_{ij} = p_{ji}$. Denomina-se que i e j são estados *reversíveis*. Como explicado nas seções que se seguem (prova do Teorema 1),

[5] O número 446 de combinações pode ser calculado teoricamente pela expressão apresentada por Cowell [32], ou ser obtido empiricamente, simulando uma cadeia de Markov.

a reversibilidade é uma propriedade importante para se obter matrizes duplamente estocásticas.

3.5 Método para geração de grafos multi-conectados

Dentre os grafos de interesse em redes Bayesianas, o grafo multi-conectado é o tipo de estrutura de grafo mais genérico. Nesta seção, apresenta-se um método para geração aleatória e uniforme de grafos multi-conectados.[6]

Para geração de tais grafos, segundo o enfoque apresentado na Seção 3.3, as seguintes exigências devem ser satisfeitas:

1. A matriz de transição da cadeia de Markov gerada deve ser simétrica.
2. A cadeia de Markov gerada deve ser ergódica para que a simulação convirja para uma distribuição estacionária.

Além disso, uma característica desejada é a maior velocidade de convergência da cadeia de Markov. Maior convergência é obtida construindo-se matrizes de transição menos *esparsas* (maior número de estados vizinhos, ou seja, menor número de probabilidades de transição nulas). A proposta do método apresentado nesta seção é que matrizes menos esparsas sejam obtidas, aumentando-se o número de operações possíveis durante a transição entre estados da cadeia. O que não é tão simples de ser obtida, uma vez que as exigências de simetria e de ergodicidade devem ser satisfeitas. Detalhes de como novas operações foram adicionadas, garantindo-se as exigências, são apresentados a seguir.

Nos algoritmos de Melançon [36] e Ide e Cozman [40][7], dois tipos de operações são possíveis: adição e remoção de arcos, determinado pela condição de existência, ou não, do arco. Uma vez sorteado o arco, existe uma única possibilidade de transição

[6] O método proposto possui a limitação de não gerar exclusivamente grafos multi-conectados. Ele pode acabar gerando árvores, mas a possibilidade é bastante pequena.

[7] No início deste trabalho, desenvolveu-se um método para construção de grafos multi-conectados para redes Bayesianas, apresentado no artigo. Este método é diferente do apresentado por Melançon, pois Melançon não se preocupa por manter o grafo gerado conectado, o que é necessário para redes Bayesianas e acarreta diferenças substanciais na prova de irredutibilidade (detalhes no artigo).

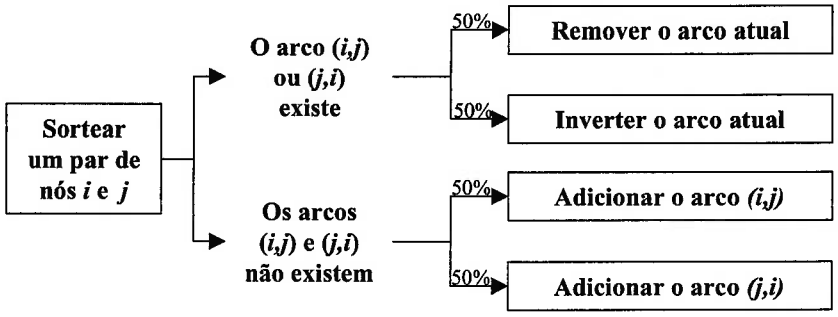


Figura 3.3. Esquematização das 4 operações possíveis do algoritmo.

para um grafo distinto. A idéia desta versão é adicionar um grau de liberdade a mais para cada uma das possibilidades (o par de nós sorteados, i e j já existem, ou não) e obter 4 operações possíveis, como esquematizado na Figura 3.3.

Um aspecto importante a ser ressaltado é que, com o acréscimo da operação de *inversão*, os arcos (i, j) e (j, i) precisam ser tratados de forma indistinta para que a *simetria* da matriz de transição seja satisfeita.

Na Figura 3.4, procura-se ilustrar como a simetria da matriz de transição é obtida. Seja o estado atual S_1. O par de nós 0 e 2 é sorteado com probabilidade igual a $\frac{1}{3}$ (a direção do arco não importa). A probabilidade de transição do estado S_1 para S_3 é $p_{13} = \frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$. A transição inversa, S_3 para S_1, tem probabilidade $p_{31} = \frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$. Como $p_{13} = p_{31}$, S_1 e S_3 são estados *reversíveis*. De modo análogo, quaisquer dois estados são reversíveis, $p_{ij} = p_{ji}$, o que torna a matriz de transição resultante simétrica.

Considere um conjunto de $nNodes$ nós, e uma cadeia de Markov construída segundo o Algoritmo 1 (Figura 3.6), no domínio finito de todos os grafos possíveis no conjunto de $nNodes$ nós, $\mathbf{S}$. Na linha 1, o grafo é inicializado como uma árvore única ordenada[8]. Nas linhas 3 a 7, efetuam-se as transições através das 4 operações (Figura 3.3), desde que o grafo resultante permaneça conectado, acíclico e satisfaça

[8]Nas explicações que se seguem, definem-se 3 tipos de grafos básicos (ilustrados na Figura 3.5). Denomina-se *árvore única* uma árvore simples cujos nós possuem um único pai, exceto o primeiro que não possui pai. Uma *polytree simples* é uma polytree cujo nós possuem no máximo grau igual a 2. E uma árvore *única ordenada* é uma árvore única cujos nós estão ordenados numericamente.

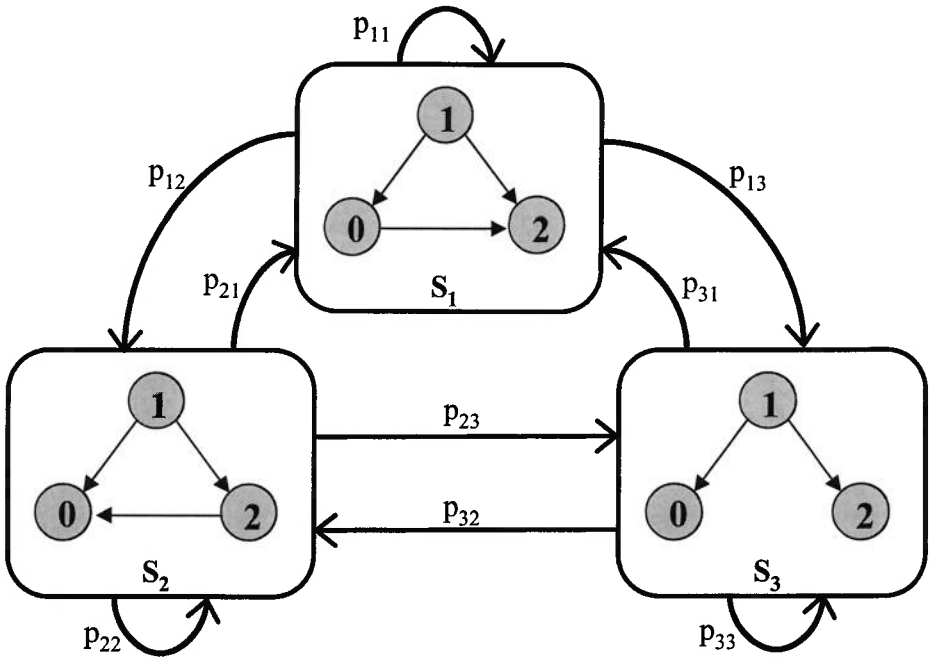


Figura 3.4. Estados de uma cadeia de Markov representando grafos com 3 nós, conforme as 4 operações possíveis da Figura 3.3.

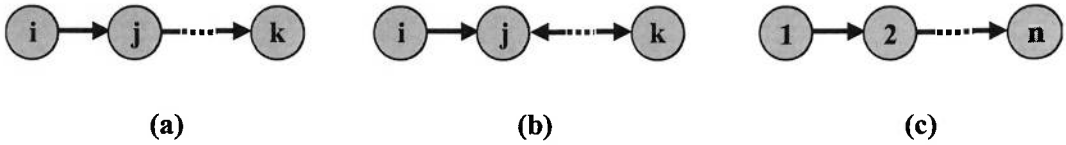


Figura 3.5. Grafos básicos: (a) árvore única, (b) polytree simples, (c) árvore única ordenada.

Algoritmo 1: Gerando DAGs multi-conectados

Entrada: número de nós (*nNodes*), número de iterações até a convergência (*nTransitions*), número total de arcos (*maxArcs*).

Saída: DAG multi-conectado com *nNodes* nós.

01. Inicializar grafo como árvore única ordenada;
02. Repetir o seguinte ciclo *nTransitions* vezes:
03. Gerar um par distinto de nós, i e j, com distribuição uniforme;
04. Se o arco (i, j) ou (j, i) existir no grafo atual, remover o arco existente (com probabilidade $\frac{1}{2}$), ou inverter o arco existente (com probabilidade $\frac{1}{2}$), desde que o grafo resultante permaneça conectado e acíclico;
05. Senão;
06. Adicionar o arco (i, j) ou (i, j) (probabilidade de $\frac{1}{2}$ para cada uma das possibilidades), desde que o grafo resultante permaneça acíclico e que satisfaça a condição *maxArcs*;
07. Senão permanecer no mesmo grafo;
08. Retornar o grafo atual após *nTransitions* iterações.

Figura 3.6. Algoritmo otimizado para geração uniforme de DAGs multi-conectados.

o número total de arcos permitido. Esta última condição (*maxArcs*) é adicionada com o objetivo de se obter grafos menos densos, característicos em redes Bayesianas (ver Seção 3.2).

Falta saber se a cadeia de Markov gerada pelo Algoritmo 1 converge de fato para uma distribuição estacionária única e uniforme. Para tanto, apresentam-se os seguintes teoremas:

Teorema 1 *A matriz de transição definida pelo Algoritmo 1 é duplamente estocástica.*

Prova.

Note que a matriz de transição é construída (linhas 3 a 7) de maneira a obter uma matriz simétrica, ou seja, a probabilidade de transição entre dois estados adjacentes quaisquer é igual em ambas as direções, $p_{ij} = p_{ji}$. A probabilidade de se permanecer no mesmo estado (linha 7) (que nunca é nula, pois sempre existe a possibilidade de

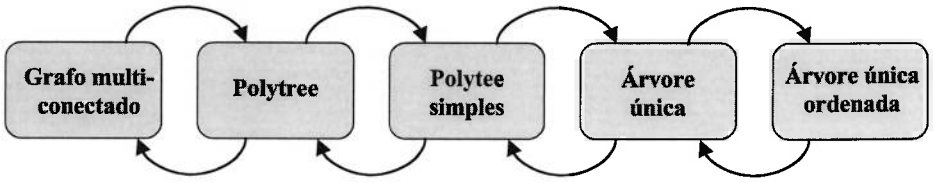


Figura 3.7. Esquematização da prova de irredutibilidade.

o grafo se tornar cíclico) é complementar à soma das probabilidades de transição de todos os estados possíveis adjacentes. Deste modo, a soma de todas as colunas e linhas assumem 1, e portanto, a matriz de transição é duplamente estocástica.

Teorema 2 *A cadeia de Markov gerada pelo Algoritmo 1 é irredutível.*

Prova.

Uma cadeia de Markov é irredutível se quaisquer dois estados desta cadeia se intercomunicam, isto é, existe uma probabilidade não nula de se atingir qualquer estado a partir de um outro qualquer. Suponha um grafo multi-conectado com $nNodes$ nós. Provando-se que deste grafo é possível atingir uma árvore única ordenada (Figura 3.5), o oposto também será verdadeiro devido à simetria da matriz de transição. Portanto, será possível atingir qualquer estado a partir de um outro qualquer. Na Figura 3.7, apresentam-se os estados percorridos para se provar a irredutibilidade da cadeia.

A metodologia adotada para provar a intercomunicação entre estados é a seguinte. Estabelece-se uma "rotina"que dita quais os passos que devem ser seguidos para que de um estado qualquer se chegue a um estado de interesse. Ou seja, desenvolve-se uma rotina para efetuar transições desejadas , para assim provar a possibilidade de comunicação entre dois estados. Para tanto, dispõe-se de 3 operações: adição, remoção e inversão; e deve-se manter as características de grafo acíclico e conectado.

De um grafo multi-conectado é possível obter uma polytree, encontrando-se um *loop cutset* e removendo os arcos necessários [6]. A partir de uma polytree é possível obter uma polytree simples (Figura 3.5) de forma recursiva, descrita a seguir.

Seja uma polytree genérica, note que todas as extremidades dela podem ser enquadradas em uma das três possíveis configurações descritas na Figura 3.8. Em todos os três casos, é possível adicionar um arco entre o último nó de um extremo

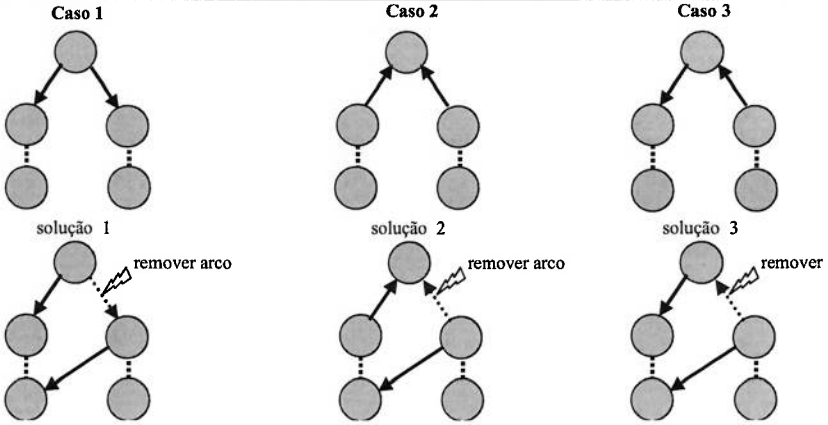


Figura 3.8. Três casos possíveis e suas respectivas soluções para se obter polytree simples a partir de uma polytree qualquer.

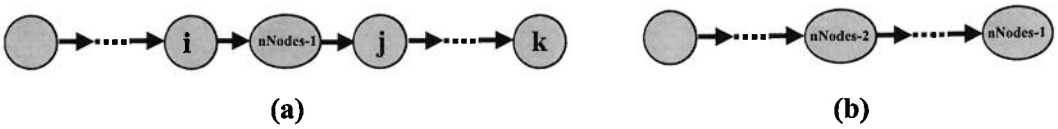


Figura 3.9. Configuração do grafo inicial (a) e final (b) de um procedimento, durante o processo de ordenação.

e o primeiro nó deste sub-grafo, e remover um arco, de modo a obter uma polytree simples, garantindo sempre que o grafo permaneça acíclico. Ou seja, todo par de extremidades pode ser convertido numa única extremidade. No caso de se ter um sub-grafo com mais de 3 extremidades, este processo é repetido em pares até se obter uma única extremidade. Realizando este processo recursivamente para todas as extremidades, obtém-se ao final uma única polytree simples, a partir de uma polytree qualquer.

A seguinte etapa é simples, obter uma árvore única (todos os arcos direcionados numa mesma direção) a partir da polytree simples. Basta inverter todos os arcos para uma mesma direção.

A etapa final consiste em ordenar a árvore única. A idéia da ordenação é a seguinte. Parte-se do último nó da árvore única k (Figura 3.9). Adiciona-se o arco ($nNodes$-1,k). A seguir, adiciona-se o arco (i, j) entre o nó anterior e posterior ao ($nNodes$-1). Ao final, removem-se os arcos $(i, nNodes - 1)$ e $(nNodes - 1, j)$,

e inverte-se o arco $(nNodes - 1, k)$ (configurações inicial e final do procedimento são ilustradas na Figura 3.9). O passo seguinte seria adicionar o arco $(nNodes - 2, nNodes-1)$, pois ($nNodes$-1) passa a ser o último nó. Repete-se este procedimento em todos os nós até se obter uma árvore única ordenada. Dessa forma, prova-se que de qualquer grafo multi-conectado é possível obter uma árvore única ordenada. A prova oposta é análoga, e assim, de qualquer grafo multi-conectado é possível atingir um outro qualquer, isto é, a cadeia de Markov gerada é irredutível.

Teorema 3 *A cadeia de Markov gerada pelo Algoritmo 1 é aperiódica.*

Prova.

Note-se que em qualquer estado da cadeia, sempre existe um arco que torna o grafo acíclico, e portanto, sempre existe uma possibilidade de manter o estado atual (linha 7). Portanto, sendo $p_{ii} > 0$ para todo $i \in \mathbf{S}$, a cadeia de Markov é aperiódica.

Teorema 4 *A cadeia de Markov gerada pelo Algoritmo 1 é ergódica, e possui uma distribuição estacionária única e uniforme, isto é* $p_{ii}^{(n)} = \frac{1}{s}$ *, para todo* $i \in \mathbf{S}$*, onde* s *é o número total de estados possíveis.*

Prova.

A cadeia de Markov gerada pelo Algoritmo 1 é homogênea, pois possui uma matriz de transição constante. Sendo a cadeia recorrente positiva (característica garantida pela diagonal não nula da matriz de transição), homogênea e irredutível, a cadeia possuirá uma distribuição estacionária, que será única por causa da aperiodicidade. Como a matriz de transição é duplamente estocástica, a cadeia de Markov gerada é ergódica, com distribuição estacionária uniforme.

3.6 Método para geração de polytrees

Um tipo de rede Bayesiana muito popular e de grande interesse prático é representado por estruturas conhecidas como polytree [6]. Até o presente momento, não existe na literatura nenhum tipo de gerador uniforme de polytrees. Nesta seção, apresenta-se um método para geração uniforme de polytrees.[9] O método garante um grande controle sobre a geração.

[9] O método proposto possui a limitação de não gerar exclusivamente polytrees. Ele pode acabar gerando árvores simples, mas a possibilidade é pequena, conforme se aumenta o número de nós.

Algoritmo 2: Gerando polytrees

Entrada: número de nós ($nNodes$), número de iterações até a convergência ($nTransitions$).

Saída: polytree com $nNodes$ nós.

01. Inicializar grafo como árvore única ordenada;
02. Repetir o seguinte ciclo $nTransitions$ vezes:
03. Gerar um par distinto de nós, i e j, com distribuição uniforme;
04. Se o arco(i, j) existir no grafo atual, manter o mesmo estado,
05. Senão;
06. Inverter o arco(i, j) com probabilidade de $\frac{1}{2}$, e então;
07. Encontrar o predecessor, nó k, no caminho entre os nós i e j, remover o arco entre k e i, e adicionar o arco (i, j) ou (j, i), dependendo do sorteio na linha 6;
08. Retornar o grafo atual após $nTransitions$ iterações.

Figura 3.10. Algoritmo para geração uniforme de polytrees.

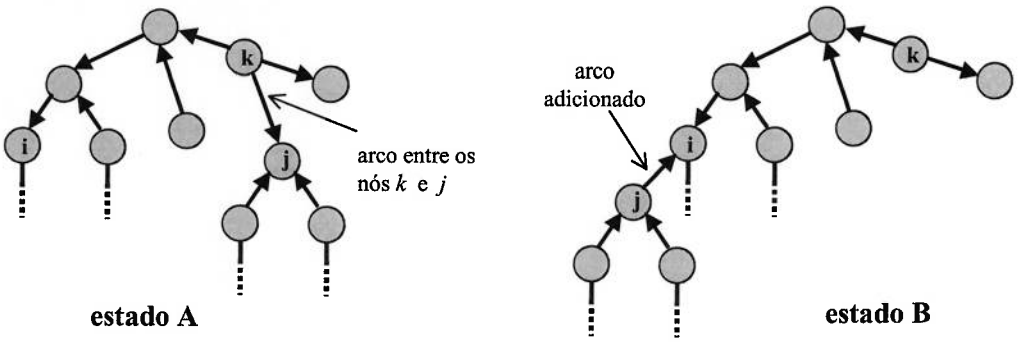


Figura 3.11. Processo de transição entre dois polytrees vizinhos.

O processo para se gerar polytrees é dado pelo Algoritmo 2 e é semelhante ao do Algoritmo 1; portanto, o enfoque desta seção é ressaltar mais as diferenças existentes entre os dois algoritmos. Descrições detalhadas e provas são dispensadas.

Considere novamente um grafo com $nNodes$ nós, e uma matriz de transição definida pelo Algoritmo 2. Na linha 1, inicia-se o grafo como uma árvore única ordenada, a qual é também um tipo de polytree. Entre as linhas 3 e 7, tem-se a função de transição propriamente dita. Na linha 4, tem-se a condição que possibilita a permanência no mesmo estado , assegurando que a cadeia de Markov gerada seja recorrente positiva e aperiódica. A linha 6 é importante para se obter a simetria da matriz de transição. A Figura 3.11 ilustra o processo de transição entre dois estados vizinhos. Suponha que no estado A, sorteia-se o arco (i, j), com probabilidade $p = \frac{1}{nNodes(nNodes-1)}$. Através de uma operação de remoção e adição, descrito na linha 7, obtém-se um dado estado B com probabilidade de transição $p_{AB} = \frac{1}{2nNodes(nNodes-1)}$. Note que a transição oposta do estado B para A também possui a mesma probabilidade de transição $p_{BA} = p_{AB} = \frac{1}{2nNodes(nNodes-1)}$. Isto é possível graças ao 50% de chance de inversão do arco sorteado (linha 6). Portanto, como no Algoritmo 1, o Algoritmo 2 produz uma matriz de transição duplamente estocástica.

No Algoritmo 1, as operações de remoção e adição eram distintas, enquanto que no Algoritmo 2, remoção e adição constituem uma única operação (linha 7), pois não é possível remover arcos de uma polytree e mantê-la ainda conectada, e ainda, não é possível adicionar um arco numa polytree e mantê-la sendo uma polytree. A operação de *inversão* (linha 6) é adicionada para se obter arcos em direções mistas.

A prova de irredutibilidade é similar ao do Algoritmo 1. A operação da linha 7 é simplesmente uma composição de operações (remoção e adição), assim sendo, é fácil notar que qualquer polytree se intercomunica com qualquer outra. Portanto, obtém-se uma cadeia de Markov homogênea, irredutível, recorrente positiva e aperiódica que possui uma única distribuição estacionária uniforme.

3.7 Análise de desempenho do gerador

Nesta seção apresentam-se resultados experimentais obtidos a partir de testes χ^2 *para prova de aderência.* Estes testes foram realizados com o intuito de avaliar o desempenho do gerador de redes Bayesianas desenvolvido.

O teste de χ^2 será empregado para avaliar o quanto as distribuições empíricas (obtidas por dados amostrais) se ajustam às distribuições teóricas (no caso, distribuições uniformes). A este emprego se denomina *prova de aderência.* A estatística χ^2 é definida como sendo a medida de discrepância existente entre as freqüências observadas e esperadas [41]:

$$\chi^2 = \frac{(f_{o_1} - f_{e_1})^2}{f_{e_1}} + \frac{(f_{o_2} - f_{e_2})^2}{f_{e_2}} + \cdots + \frac{(f_{o_k} - f_{e_k})^2}{f_{e_k}} = \sum_{j=1}^{k} \frac{(f_{o_i} - f_{e_i})^2}{f_{e_i}} \tag{3.2}$$

, onde k é o número de *categorias.* Para distribuições uniformes, a freqüência esperada é constante, e portanto, a Equação 3.2 fica:

$$\chi^2 = \sum_{i=1}^{k} \frac{(f_{o_i} - f_{e_i})^2}{f_{e_i}} = \frac{\sum_{i=1}^{k}(f_{o_i} - f_{e_i})^2}{f_e}. \tag{3.3}$$

Na prova de aderência, valores de χ^2 serão calculados e comparados com distribuições χ^2 (cujos valores são normalmente tabelados), através de um teste de hipóteses. Quanto maior for o valor χ^2, maior será a discrepância entre as freqüências observadas e esperadas. Definem-se, normalmente, os valores $\chi^2_{0,95}$ e $\chi^2_{0,99}$ como sendo *valores críticos.* Se o valor χ^2 calculado for maior que os valores críticos, deve-se rejeitar as freqüências observadas. O número de graus de liberdade, v, definido por $v = k-1$ (pois, a freqüência esperada é calculada, e não estimada), será utilizado para escolha da distribuição χ^2.

Amostras de redes Bayesianas com determinadas características (número de nós $nNodes$, número máximo de arcos por nó $maxDegree$ e número máximo total de

arcos $maxArcs$) foram geradas pelo BNGenerator (programa desenvolvido neste trabalho e apresentado no Capítulo 4) para obter amostras de grafos. Cada *amostra* (conjunto de todas as amostras de grafos) gerada possui 100 *mil* grafos, e cada tipo diferente de estrutura de grafo é definido como uma categoria. E para cada categoria, freqüências são computadas na medida em que as estruturas se repetem. Os valores de χ^2 foram calculados para cada tipo de estrutura de grafo como apresentado na Tabela 3.1. A_i representa diferentes amostras, cada uma contendo 100 mil grafos. Algoritmo 0 é o algoritmo implementado para geração de grafos multi-conectados por Ide e Cozman [40]. Algoritmo 1 é a versão otimizada para geração de grafos multi-conectados (Seção 3.5). Algoritmo 2 gera polytrees (Seção 3.6). O parâmetro $nNode$ indica o número de nós. Para número de nós maiores do que 4, o teste de χ^2 passa a ser inviável, pois o número de diferentes estruturas aumenta consideravelmente. Para $nNodes = 5$, sabe-se experimentalmente que existem 7540 diferentes tipos de grafos multi-conectados e 2000 tipos diferentes de polytree. Para ambas amostras, não se possui distribuições de χ^2 de tamanha escala. As *restrições* $maxDegree$ e $maxArcs$ são indicadores da densidade dos grafos gerados. O parâmetro v é o grau de liberdade da amostra, obtido subtraindo-se 1 do número de categorias k. O parâmetro i é o número de transições ($nTransitions$) efetuadas entre um grafo e outro selecionado, durante a geração, segundo uma cadeia de Markov.

Determinar o número suficiente de transições $nTransitions$ é uma questão ainda em aberto. Melançon [36] chega a conclusões experimentais de que um número quadrático de transições é o suficiente, e adota em seu programa *DagAlea* o seguinte valor de transições: $nTransitions = 4 \times nNodes^2$. Neste caso, para $nNodes = 4$, $nTransitions = 64$ e este valor é tomado como valor inicial. Variou-se $nTransitions$ para 100 e para 1000. Teoricamente, quanto maior o valor de $nTransitions$, melhor deve ser a distribuição das amostras geradas. Por exemplo, numa amostra de grafos, com $nNodes = 4$ e $maxDegree = 3$, existem 446 categorias, ou seja, 446 tipos diferentes de estruturas de grafos com a mesma característica. A freqüência esperada para cada categoria é $f_e = \frac{100000}{446} = 224$. Como $k = 446$, escolhe-se a distribuição χ^2 com grau de liberdade $v = 445$ para a prova de aderência.

Como observado na Tabela 3.1, os valores de χ^2 diminuem, conforme se aumenta $nTransitions$; e a superioridade de desempenho do Algoritmo 1 sobre o Algoritmo

Tabela 3.1. Tabela contendo resultados de χ^2 calculados, para cada amostra A_i de grafo gerado. Onde i é o número de transições $nTransitions$.

	Algoritmo	**nNodes**	**Restrição**	υ	χ^2		
					$i = 64$	$i = 100$	$i = 1000$
A_1	0	4	$maxDegree = 3$	445	12 mil	573,2	396,8
A_2	1	4	$maxDegree = 3$	445	509	436,7	418,3
A_3	0	4	$maxArcs = 4$	313	24 mil	4468,5	314,6
A_4	1	4	$maxArcs = 4$	313	367,5	331,8	313,5
A_5	2	4	(nenhuma)	127	131,7	121,2	127,0

Tabela 3.2. Tabela contendo valores aproximados da distribuição de χ^2.

υ	$\chi^2_{0,99}$	$\chi^2_{0,95}$	$\chi^2_{0,05}$	$\chi^2_{0,01}$
127	167,0	154,3	102,0	92,8
313	374,0	355,2	273,0	257,7
445	517,5	495,2	397,2	378,5

0 fica nítida. Nas amostras A_1 e A_3, para $nTransitions = 64$, os valores χ^2 ficam muito acima do aceitável.

Uma vez obtido os valores de χ^2 das amostras geradas, falta compará-los com os valores teóricos da distribuição χ^2. Habitualmente, estes valores são apresentados somente até o grau de liberdade 100 [41]. Sendo assim, valores de χ^2 foram obtidos por programas que fornecem valores aproximados, disponíveis na internet[10], e apresentados na Tabela 3.2. Sabe-se que para valores de υ maiores que 30, a distribuição χ^2 é aproximada por distribuições Gaussianas, fazendo as devidas transformações Z necessárias.

A análise feita a seguir segue o padrão adotado por Spiegel [41]. Comparando-se os valores entre a Tabela 3.1 e 3.2, para a amostra A_2, como $436,7 < 495,2$, o resultado é positivo, pois o valor de χ^2 da amostra observada está acima dos valores críticos ($\chi^2 = 436,7$ equivale a 61,1% de significância). Para a amostra A_4, como

[10] *http : //fonsg3.let.uva.nl/Service/Statistics/ChiSquare − distribution.html*

$331,8 < 355,2$, o resultado é positivo ($\chi^2 = 331,8$ equivale a 23,2% de significância). Para a amostra A_5, como $121,2 < 154,3$, o resultado é positivo ($\chi^2 = 121,2$ equivale a 63,3% de significância).

Dos testes χ^2 para prova de aderência realizados, conclui-se que o valor $nTransitions = 64$, obtido por $nTransitions = 4 \times nNodes^2$, é inadequado, pelo menos para o algoritmo de geração de grafos multi-conectados. Assim, propõe-se uma nova expressão para o cálculo do número de transições necessárias para a convergência, $nTransitions = 6 \times nNodes^2$.

Para finalizar a análise, pode-se concluir o seguinte. O algoritmo desenvolvido por Melançon [36], no seu programa *DagAlea*, para geração de grafos, utiliza a relação $nTransitions = 4 \times nNodes^2$ para o cálculo do número de transições necessárias entre uma amostra de rede e outra; e, pelas análises realizadas nesta seção, para geração de grafos de interesse em redes Bayesianas (multi-conectados e polytrees), deve-se realizar pelo menos $nTransitions = 6 \times nNodes^2$ transições. Estes resultados são bastante razoáveis. No algoritmo de Melançon [36], o fato de não precisar manter o grafo conectado, elimina uma restrição (responsável por diminuir a velocidade de convergência), enquanto que no algoritmo proposto, restrições adicionais naturalmente retardam a convergência, sendo necessário um maior número de transições.

Capítulo 4

SOFTWARE DESENVOLVIDO: BNGenerator

O programa desenvolvido BNGenerator implementa os algoritmos desenvolvidos ao longo deste trabalho (Capítulo 3, e está disponível para a comunidade científica na Internet em: <http://www.pmr.poli.usp.br/ltd/Software/BNGenerator/>. Na Figura 4.1, vizualiza-se a página da internet, de onde é possível adquirir o programa.

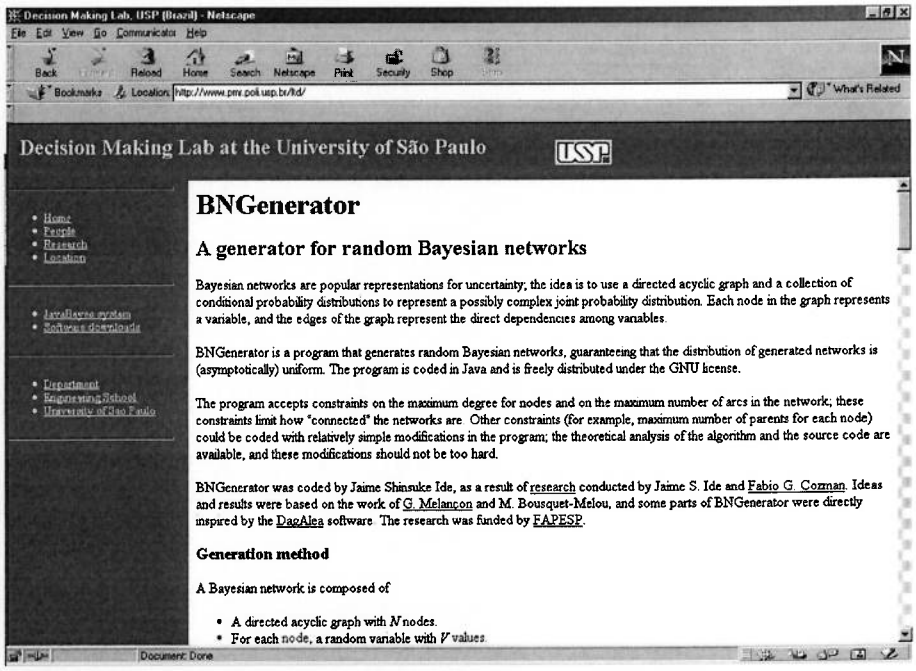


Figura 4.1. Visualização do site de divulgação do BNGenerator.

4.1 Um gerador de redes Bayesianas aleatórias

BNGenerator é um programa que gera redes Bayesianas aleatoriamente, garantindo que a distribuição das estruturas da rede geradas seja uniforme. O programa foi codificado em *Java* e é livremente distribuído pelos termos de licença da GNU.

O programa aceita especificações tais como o número máximo de arcos para cada nó e o número máximo de arcos total da rede; estas especificações limitam o quão "conectadas" são as redes.

Para gerar DAGs conectados, o programa BNGenerator baseia-se na simulação de cadeias de Markov, conforme descrito no Capítulo 3. Uma vez obtido o DAG, o programa gera aleatoriamente o número de valores v assumidos por cada variável da rede. Esse número é sorteado a partir de uma distribuição uniforme $U \sim (2, v_{max})$, onde v_{max} é o número máximo permitido de valores (especificado pelo usuário). Uma vez estabelecido o número de valores para cada variável, o programa gera distribuições de probabilidade, a partir da distribuição de *Dirichlet*, como indicado na Seção 3.1.

BNGenerator inicia a cadeia de Markov por um grafo bem simples (um grafo onde os nós possuem no máximo um pai e um filho) e realiza transições a partir do grafo inicial. Após um número de transições especificado pelo usuário (caso contrário, adota um valor *default*), o grafo é selecionado, os números de valores assumidos pelas variáveis são sorteados, as distribuições de probabilidade são geradas e por fim, a rede Bayesiana é salva em um arquivo. Após uma amostra ser salva, o programa pode continuar executando as transições e gerar mais redes.

O programa permite gerar redes Bayesianas com estruturas de grafos multi-conectadas e polytrees. Os grafos conhecidos por *polytree* possuem características de grande interesse em estudos de rede Bayesiana, e até o presente momento, parece não existir nenhum outro gerador aleatório de polytrees que dê garantias sobre a distribuição das amostras geradas.

4.2 Adquirindo o BNGenerator

O programa é constituído por dois arquivos de código fonte: *BNGenerator.java* e *DFGenerator.java*. Os arquivos compilados correspondentes são: *BNGenerator.class*

e *DFGenerator.class*. BNGenerator ainda necessita de *Matrix.class* (gerado automaticamente ao se compilar *BNGenerator.java*). Além destes arquivos que constituem a geração da rede Bayesiana em si, são necessários: a biblioteca *colt.jar* da CERN [42] para geração de distribuições Gama e o gerador de números pseudo-aleatórios *MersenneTwister.class* [43].

4.3 Rodando o BNGenerator

Como o BNGenerator é codificado em Java, para executá-lo, é necessário ter instalado uma versão de *Java Virtual Machine*[1]. Assumindo -se que o arquivo *colt.jar* e demais arquivos *.class* estejam no mesmo diretório, para rodar o BNGenerator deve-se entrar com a linha de comando:

```
java -classpath .;colt.jar BNGenerator [-opção valor]
```

Onde o par [-opção valor] indica uma seqüência de opções de especificação, que podem ser:

-nNodes : número de nós das redes geradas. O valor *default* é 4.

-maxDegree : *grau* máximo permitido para quaisquer nós das redes. O valor *default* é $(nNodes - 1)$. Limite inferior: $maxDegree$ deve ser maior do que 2.

-maxArcs : número total máximo de arcos que as redes podem possuir. O valor *default* é $\frac{nNodes*maxDegree}{2}$ (número total máximo de arcos possíveis). $maxArcs$ deve ser maior que $(nNodes - 1)$.

-nBNs : número de redes Bayesianas geradas. Valor *dafault* é 1.

-nTransitions : número de iterações (transições efetuadas na cadeia de Markov gerada pelo BNGnerator) entre amostras geradas e salvas. Note que, para cada iteração, uma nova rede é gerada, mas somente após $nTransitions$ iterações, a rede é salva em arquivo (este procedimento repete-se por $nBNs$ vezes). O valor *default* é $(6 \times nNodes^2)$.

[1]Disponível em: <http://java.sun.com>

-format : formato em que as redes são salvas. São disponíveis dois tipos formato: "xml" (caso em as redes são salvas em formato XMLBIF, compatível com o programa *JavaBayes* [2]); e "java"(caso em que as redes são salvas em Java, podendo ser utilizadas para inserir dados na biblioteca *EBayes*[3]). A opção "xmljava"permite salvar em ambos os formatos. O valor *default* é xml.

-nval : máximo número de valores assumidos pelas variáveis . Toda variável assume um número de valores entre 2 e *nval*. O valor *default* é 2.

-fName : nome dos arquivos gerados. Uma extensão será adicionada ao $fName$, dependendo de $format$. Se $nBNs$ é maior do que um, uma seqüência de arquivos é gerada. Assim, os arquivos serão da seguinte forma $fNameX.format$, onde X é um número crescente. O nome *default* é *Graph*.

-structure : Tipo de estrutura das redes geradas. Pode ser "multi"(para se gerar grafos multi-conectados), ou "poly"(para se gerar polytrees). Note que o tipo "multi" é uma classe que abrange as polytrees e as árvores simples. Entretanto, a probabilidade de se gerar polytrees empregando a opção "multi" é muito pequena. "multi" é o tipo de estrutura *default*.

Um comando adicional "-testUnif" está disponível para se computar as freqüências das estruturas que se repetem. Ao final da geração, salva as freqüências computadas num arquivo com o seguinte formato $UniformityTestResults - [caractersticas].doc$. Estas informações servem tanto para realização de testes χ^2 para prova de aderência (ver Seção 3.7), como para *contagem* do número de diferentes combinações possíveis.

Por exemplo, a linha de comando:

```
C:\> java -classpath .;colt.jar BNGenerator -nNodes 20 -maxDegree 4 -nBNs 2 -fName
     exemplo
```

Gera duas redes Bayesianas com 20 nós (variáveis binárias, pois ao não se especificar *nval*, foi adotado o valor *default*), no formato *xml* (valor *default*), nos arquivos

[2]Sistema para manuseio de redes Bayesianas, livremente distribuído em http://www-2.cs.cmu.edu/ javabayes/.

[3]Biblioteca para realização de inferências em redes Bayesianas, disponível em http://www.pmr.poli.usp.br/ltd/Software/EBayes/index.html.

exemplo1.xml e *exemplo2.xml*. Cada nó possui um grau máximo de 4 arcos. Antes que as redes sejam salvas, 2400 transições (usa o valor *default* que é 6*20*20) são efetuadas. Na Figura 4.2, visualiza-se a rede Bayesiana gerada pelo comando anterior.

Para a linha de comando:

C:\> java -classpath .;colt.jar BNGenerator -nNodes 20 -maxDegree 4 - maxArcs 25 -nBNs 2 -fName exemplo

Obtém-se resultados iguais aos da linha de comando anterior, com a única diferença que agora, as redes geradas possuem no máximo 25 arcos. Note que para $nNodes = 20$ e $maxDegree = 4$ poder-se-ia ter até no máximo $\frac{20*4}{2} = 40$ arcos, mas o número total de arcos fica limitado a 25, com a adicional restrição. Na Figura 4.3, visualiza-se a rede Bayesiana gerada pelo comando acima .

Para a linha de comando:

C:\> java -classpath .;colt.jar BNGenerator -nNodes 20 -nBNs 1 -fName exemplo -structure poly

Obtém-se uma polytree com 20 nós e variáveis assumindo dois valores. Na Figura 4.4, visualiza-se a rede Bayesiana gerada pelo comando acima .

Para a linha de comando:

C:\> java -classpath .;colt.jar BNGenerator -nNodes 50 - maxArcs 60 -nBNs 1 -fName exemplo

Obtém-se uma rede Bayesiana com 50 nós e $maxArcs = 60$. Na Figura 4.5, visualiza-se a rede Bayesiana gerada pelo comando acima .

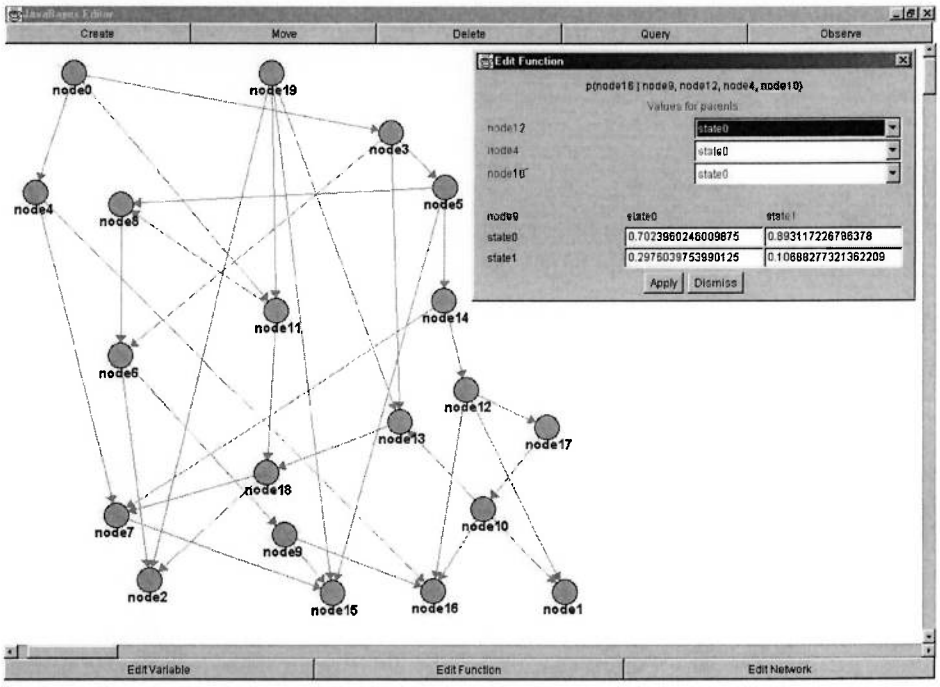


Figura 4.2. Visualização no JavaBayes da rede Bayesiana gerada com $nNodes = 20$ e $maxDegree = 4$ pelo BNGenerator.

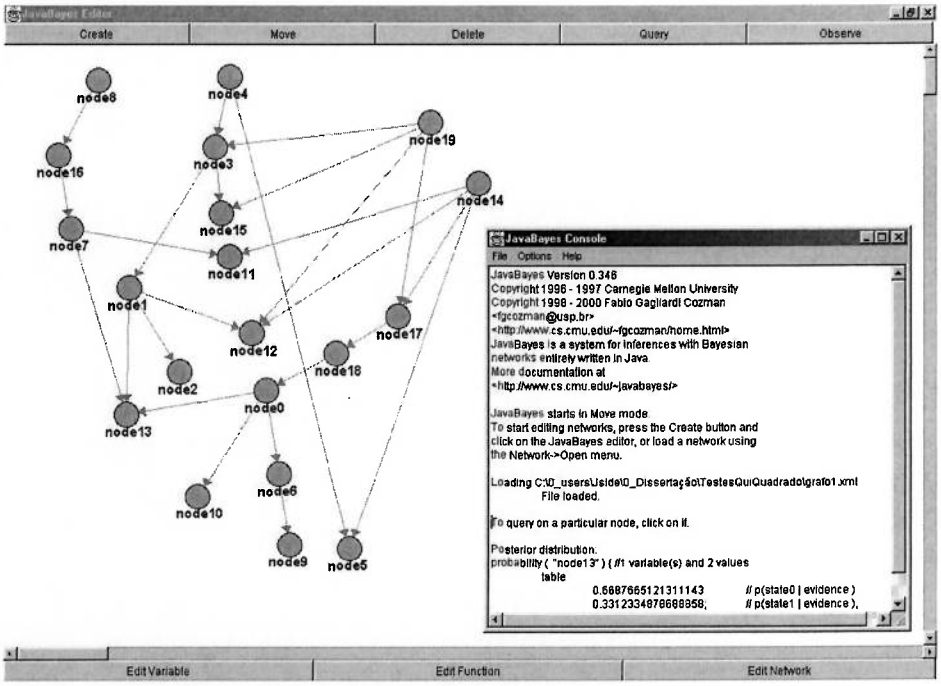


Figura 4.3. Visualização no JavaBayes da rede Bayesiana gerada com $nNodes = 20$, $maxDegree = 4$ e $maxArcs = 25$ pelo BNGenerator.

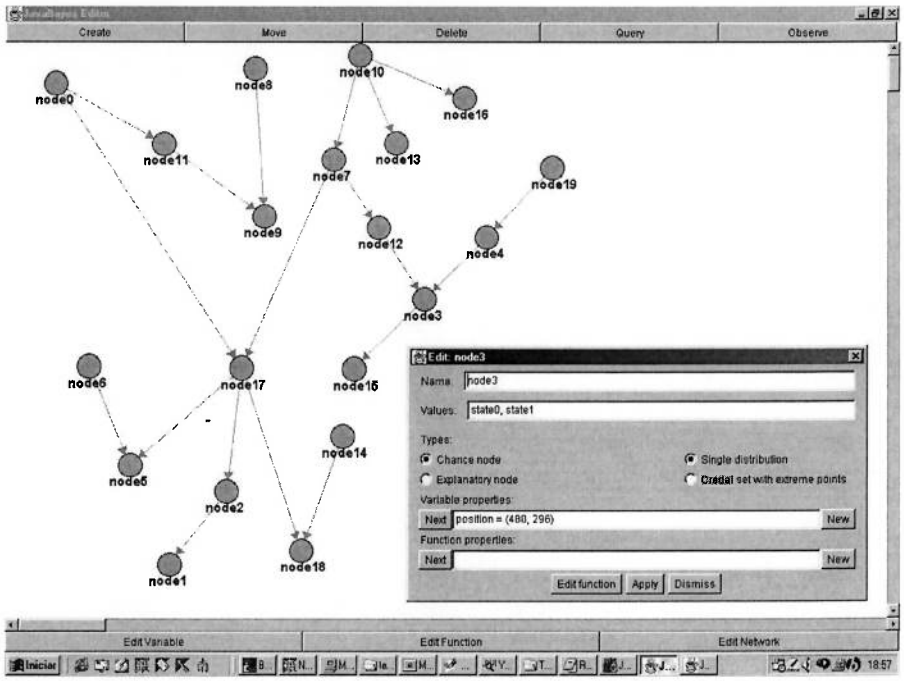


Figura 4.4. Visualização no JavaBayes de uma polytree gerada com $nNodes = 20$ pelo BNGenerator.

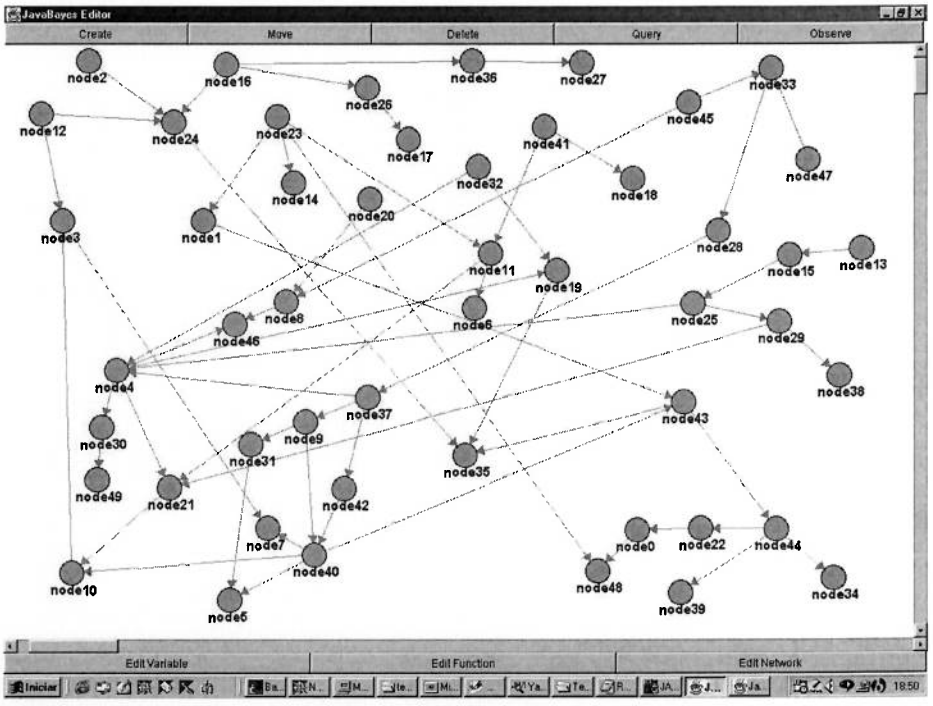


Figura 4.5. Visualização no JavaBayes da rede Bayesiana gerada com $nNodes = 50$ e $maxArcs = 60$ pelo BNGenerator.

4.4 Escolha do número de transições $nTransitions$

Teoricamente, é necessário se esperar por um número infinito de transições para que a cadeia de Markov convirja para uma distribuição estacionária (Seção 3.3), mas na prática, adota-se um número de transições razoavelmente grande. Alguns estudos empíricos [44] apontam para um valor quadrático de transições, em função do número de nós $nNodes$. Melançon [36] adota em seu programa *DagAlea* o valor $nTransitions = 4 \times nNodes^2$ de transições. Mas, análises realizadas neste trabalho (Seção 3.7) apontam para uma número de pelo menos $6 \times nNodes^2$ transições, para grafos de interesse em redes Bayesianas. Este é o número de transições que o BNGenerator adota por *default*, mas o usuário pode especificar um número diferente.

Recomenda-se que, quanto mais restrições ($maxDegree$ e $maxArcs$) se colocam, maior deve ser o parâmetro $nTransitions$, para garantir que a geração segundo cadeia de Markov convirja. Naturalmente, quanto menor forem os valores de $maxDegree$ e $maxArcs$, mais "restritivas" são as condições a serem satisfeitas, e mais transições devem ser efetuadas entre uma amostra e outra salva.

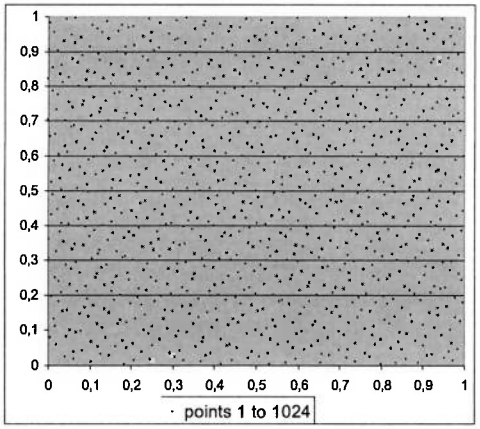
Capítulo 5

APLICAÇÃO PARA ANÁLISE DE MÉTODOS QUASI-MONTE CARLO

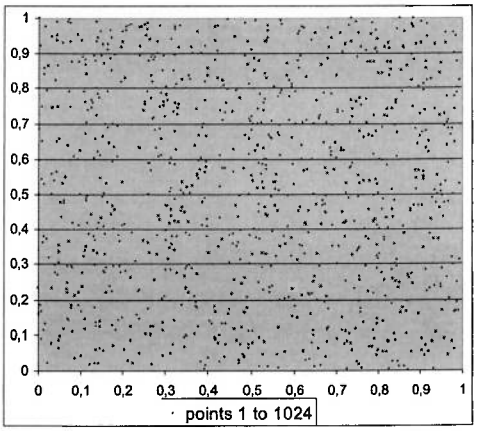
Neste capítulo, apresenta-se uma aplicação para redes Bayesianas aleatórias geradas de forma uniforme. O objetivo é análisar métodos quasi-Monte Carlo. A geração de redes Bayesianas é interessante para se obter uma amostra grande de redes e com ela testar diferentes algoritmos de inferência, obtendo-se resultados comparativos mais genéricos e significativos. Na maioria dos trabalhos que comparam algoritmos de inferência, os resultados baseiam-se em testes realizados em redes Bayesianas conhecidas, encontradas na literatura da UAI [1]. Entretanto, uma maneira mais completa e significativa seria testar os algoritmos de inferência, não apenas para redes conhecidas, mas para um grande número de amostras.

Na Seção 5.1, resume-se o que são as seqüências quasi-aleatórias e qual é o interesse em aplicá-las em algoritmos de inferência. Na Seção 5.2, apresenta-se uma explicação de como será analisado o desempenho do algoritmo proposto. Na Seção 5.3, descreve-se como amostras de redes Bayesianas são geradas para realização dos testes. Na Seção 5.4, apresentam-se os resultados experimentais obtidos. E na Seção 5.5, apresenta-se uma discussão sobre os resultados obtidos.

[1] Repositório de redes Bayesianas está disponível em: <http://www.cs.huji.ac.il/labs/ compbio/Repository/>.



Seqüência de baixa-discrepância (Sobol)



Seqüência pseudo-aleatória

Figura 5.1. Projeções $2D$ de dois tipos de seqüências que mostram fundamentais diferenças. A baixa *discrepância* consiste na maior proximidade média entre os pontos.

5.1 Métodos quasi-Monte Carlo

O termo *quasi-Monte Carlo* nada mais é do que a denominação para os métodos Monte Carlo que empregam seqüências *quasi-aleatórias* ao invés de *pseudo-aleatórias*. Neste trabalho, utiliza-se o termo *pseudo-Monte Carlo* para se referir à métodos Monte Carlo que empregam as tradicionais seqüências aleatórias (pseudo-aleatórias). A aplicação de métodos quasi-Monte Carlo em algoritmos que realizam *Gibbs sampling* (ver Seção 2.4) é um problema ainda em aberto [45].

Métodos MCMC empregam habitualmente seqüências de números aleatórios para alimentar a simulação (linha 5, Algoritmo *Gibbs sampler*, Seção 2.4). Estas seqüências são denominadas de *pseudo-aleatórias*, para diferenciar com as seqüências *quasi-aleatórias*. A diferença entre estes dois tipos de seqüência é a seguinte: na pseudo-aleatória, os números são gerados de modo a "simular amostras aleatórias" a partir de uma distribuição uniforme $U \sim (0, 1)$; enquanto que na quasi-aleatória, os números são gerados de modo a preencher o *hipercubo unitário* $[0, 1]^d$ da maneira mais uniforme possível [20]. Na Figura 5.1, encontra-se uma projeção bidimensional de ambos os tipos de seqüências para ilustrar fundamentais diferenças.

5.1.1 Seqüências quasi-aleatórias

Seqüências quasi-aleatórias são denominadas de seqüências de *baixa discrepância* (*low-discrepancy*). A *discrepância* é uma medida de não uniformidade de uma seqüência de pontos espalhados num hipercubo unitário $[0,1]^d$. A medida mais popular de discrepância é a *star-discrepancy* D_N^* [46]. Uma seqüência $(x_1, x_2, \cdots, x_N)$ de pontos espalhados em $[0,1]^d$ é de baixa discrepância se para qualquer $N > 1$:

$$D_N^*(x_1, \ldots, x_N) < c(d).\frac{(logN)^d}{N}, \tag{5.1}$$

onde, a constante $c(d)$ depende da dimensão d do problema e N é o número de seqüências geradas.

Por causa da característica de baixa discrepância, métodos quasi-Monte Carlo possuem um erro determinístico de $O((logN)^d/N)$, ao invés do erro probabilístico $O(N^{-1/2})$ dos métodos Monte Carlo. Teoricamente, isso significa que métodos quasi-aleatórios possuem melhor desempenho de convergência para baixas dimensões e pior desempenho para altas dimensões.

5.1.2 Construção de seqüências quasi-aleatórias clássicas

As seqüências quasi-aleatórias clássicas propostas na literatura são as de Halton [47], Sobol [48] e Faure [49]. Niederreiter [46] propôs um princípio geral de construção das chamadas *(t,d)-sequences*. As seqüências de Halton, Sobol e Faure podem ser consideradas casos especiais das *(t,d)-sequences*.

Seqüências de Faure e Halton foram implementadas segundo Fox [50]. Uma eficiente variante da seqüência de Sobol baseada em *Gray code*, proposto por Antonov e Saleev [51], foi implementado segundo Bratley e Fox [52].

5.1.3 Discussão sobre o emprego de métodos quasi-Monte Carlo

Métodos quasi-Monte Carlo têm sido aplicados com sucesso para problemas de baixa dimensão em diversas áreas, entre elas na Computação Gráfica, Física Computacional e Engenharia Financeira [53]; [54]; [55]; [56]. Em problemas de alta dimensão, não existe um consenso entre os pesquisadores. Enquanto alguns pesquisadores [57];

[58] acreditam que métodos quasi-Monte Carlo não funcionam bem para problemas de alta dimensão, outros pesquisadores como Paskov e Traub [55], Paskov [59], Cheng e Druzdzel [60] obtiveram ótimos resultados em Aplicações Financeiras, Física Computacional e Computação Bayesiana de alta dimensão, respectivamente.

Cheng e Druzdzel [60] aplicaram métodos quasi-Monte Carlo em algoritmos de amostragem por importância para realização de inferências em redes Bayesianas, obtendo resultados positivos. Shaw [61] faz uma análise do emprego de seqüências quasi-aleatória para integração numérica em Estatística Bayesiana.

Segundo Liao [62], o emprego direto de métodos quasi-Monte Carlo em métodos MCMC não funciona bem, devido à uma dependência entre a seqüência quasi-aleatória gerada e a cadeia de Markov construída; mas, através de uma permutação das seqüências, Liao obtém bons resultados aplicando Gibbs sampling em problemas estatísticos clássicos (de baixa dimensão).

O objetivo da análise que se segue é justamente fornecer subsídios para avaliação de seqüências quasi-aleatórias em métodos MCMC.

5.2 Análise de desempenho de métodos quasi-Monte Carlo

A análise do desempenho de métodos quasi-Monte Carlo é feita da seguinte forma. Métodos quasi-Monte Carlo e pseudo-Monte Carlo são empregados para realizar inferências. A diferença entre os dois métodos é que a primeira utiliza seqüências quasi-aleatórias, e a segunda, pseudo-aleatórias. Os dois tipos de seqüências são empregados para se "alimentar"o Gibbs sampler (Algoritmo *Gibbs sampler*, Seção 2.4). O desempenho destes métodos é quantificado, calculando-se o erro de seus resultados.

A dificuldade em se medir o erro destes métodos é a seguinte. Como o Gibbs sampler é um método MCMC (*Monte Carlo Markov Chain*), o erro resultante das estimativas de Monte Carlo estão atreladas à dependências criadas pela cadeia de Markov, e portanto o estimador de erro deve ser escolhido com bastante cuidado. Literatura sobre diagnóstico de convergência pode ser encontrado em [63] e [64]. O *erro quadrático médio* (MSE) é empregado aqui para se medir o erro proveniente dos

métodos aproximados, como proposto por Cheng e Druzdzel [60]. Deve-se ressaltar que, para se utilizar o MSE, é necessário que cálculos exatos de inferência sejam realizados. O MSE é computado da seguinte forma:

$$\sqrt{\frac{1}{\sum_{x_i \in \{X \setminus E\}} n_i} \sum_{x_i \in \{X \setminus E\}} \sum_{j=1}^{n_i} (p'(x_{ij}) - p(x_{ij}))^2}, \tag{5.2}$$

onde, X representa o conjunto de todas as variáveis, E o conjunto de todas as variáveis observadas e n_i o número de valores assumidos pela variável i, representada por x_i. Note-que x_{ij} é o valor numérico assumido por x_i para cada valor com índice j. O conjunto $\{X \setminus E\}$ representa o conjunto de todas as variáveis não observadas.

Recapitulando (Seção 2.4), métodos Monte Carlo utilizam amostras de redes Bayesianas para obter estimativas das probabilidades condicionais (inferência). Quanto maior o tamanho da amostra, melhores são as estimativas, ou seja, menores são os erros(MSE). Teoricamente, quando o tamanho da amostra tende ao infinito, o MSE tende a zero. Mas na prática, geram-se amostras de redes até se obter níveis de erros aceitáveis. A análise propriamente dita do algoritmo aproximado de inferência é realizada sobre a relação entre *tamanho da amostra* (número de redes geradas para realização das estimativas) e MSE (exatidão da aproximação atingida pela simulação).

Testes com métodos quasi-Monte Carlo e pseudo-Monte Carlo foram realizados em algumas redes clássicas (sem variáveis observadas), e seus resultados são apresentados nas Figuras 5.2, 5.3 e 5.4. Variou-se o tamanho da amostra (*nSimulation*) entre 250 e 250000. Dentre os três tipos de seqüências quasi-aleatórias: Faure, Halton e Sobol, os resultados obtidos por seqüências de Sobol foram superiores. Portanto, foi o tipo de seqüência escolhida para os testes. Seqüências pseudo-aleatórias foram obtidas pelo eficiente gerador *MersenneTwister* [43].

5.3 Empregando redes Bayesianas aleatórias

Como exposto inicialmente no Capítulo 1, o propósito de gerar inúmeras redes Bayesianas aleatoriamente é de obter propriedades médias em um conjunto de redes. Desde modo, resultados do desempenho de um método de inferência não ficam limitados a apenas alguns tipos de redes Bayesianas, com características específicas.

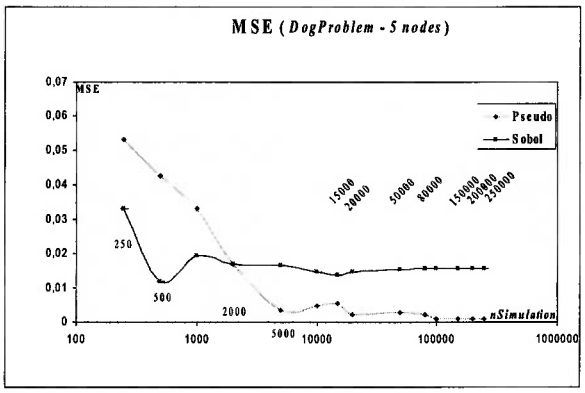


Figura 5.2. Erro quadrático médio (MSE) em função do número de amostras para a rede *DogProblem*. Esta rede é constituída por 5 nós, e é uma polytree.

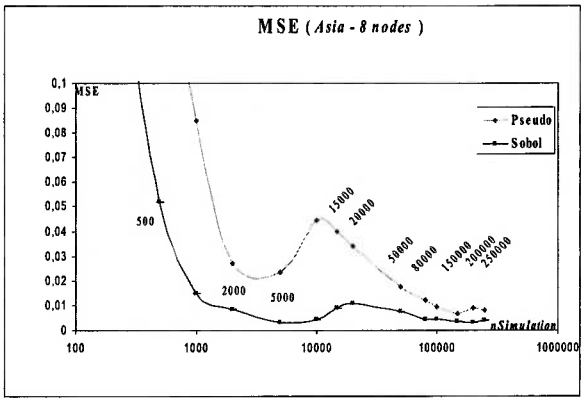


Figura 5.3. Erro quadrático médio (MSE) em função do número de amostras para a rede *Asia*. Esta rede possui 8 nós e 8 arcos.

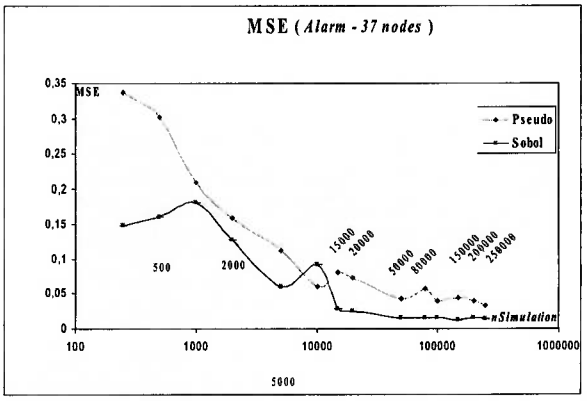


Figura 5.4. Erro quadrático médio (MSE) em função do número de amostras para a rede *Alarm*. Esta rede possui 37 nós e 44 arcos.

Tabela 5.1. Tabela contendo parâmetros de entrada do BNGenerator para geração de amostras de redes Bayesianas.

Amostra	*nNodes*	*maxArcs*	*nBNs*	*nval*	*structure*
A_1	5	-	200	2	poly
A_2	8	8	300	2	multi
A_3	8	10	300	2	multi
A_4	8	12	300	2	multi
A_5	16	16	600	3	multi
A_6	16	20	600	3	multi
A_7	16	24	600	3	multi
A_8	24	24	900	3	multi
A_9	24	30	900	3	multi
A_{10}	24	36	900	3	multi
A_{11}	37	37	2000	4	multi
A_{12}	37	44	2000	4	multi

Obtém-se propriedades médias e gerais de desempenho do método de inferência em análise.

Inúmeras amostras de redes Bayesianas foram geradas pelo BNGenerator, com diferentes características (Tabela 5.1), para serem utilizadas na análise de desempenho de métodos quasi-Monte Carlo e pseudo-Monte Carlo.

A amostra A_1 é constituída por polytrees com o mesmo número de nós da rede *DogProblem*. Note que, em polytrees, não há sentido falar de número máximo de arcos, uma vez que o número de arcos é sempre o mesmo, $(nNodes - 1)$. A amostra A_2 possui redes com características semelhantes à rede *Asia* (8 nós e 8 arcos). A amostra A_{12} possui redes com características da rede *Alarm* (37 nós e 44 arcos). As demais amostras são geradas com a finalidade de se avaliar a influência do aumento do números de arcos, fixo o número de nós.

Uma vez que as amostras de redes Bayesianas são geradas, cada elemento é compilado para o formato $*.class$ compatível com *EBayes*. O algoritmo *Gibbs sampler*, que emprega métodos quasi-Monte Carlo e pseudo-Monte Carlo, foi implementado e

adicionado à biblioteca *EBayes*. E cada rede Bayesiana é processada por *EBayes* e o valor do MSE é calculado e computado.

Para que fosse possível calcular tantos valores do MSE (note que ao total são 9600 redes!), criou-se uma rotina de *teste*, isto é, um programa que processa várias redes Bayesianas e ao final grava em um arquivo texto os os valores do MSE para cada uma das redes.

5.4 Resultados experimentais

Realizaram-se inferências em amostras de redes Bayesianas geradas aleatoriamente, no total de 9600 redes; e para cada uma das redes das amostras, computaram-se os valores do MSE (erro quadrático médio), tanto para os resultados do Gibbs sampling fornecidos pelo emprego de seqüências quasi-aleatórias (método quasi-Monte Carlo), como para seqüências pseudo-aleatórias. O tamanho da amostra, durante a simulação do Gibbs sampling, variou entre 250 e 250000. Observe que, para cada tamanho da amostra, calcula-se o MSE. Em cada conjunto (amostra) de redes, calculou-se a média dos MSE ($\overline{MSE}$) e os respectivos desvios padrões (s) para o mesmo tamanho da amostra (*nSimulation*). Resultados são visualizados nas Figuras 5.5 a 5.16.

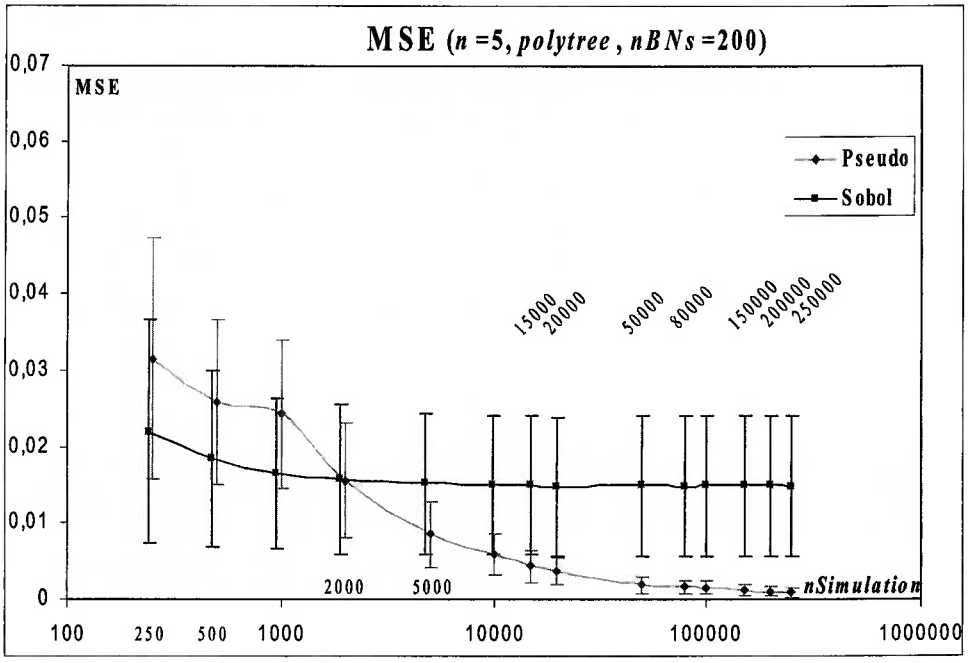


Figura 5.5. Média dos MSE em função de *nSimulation* para redes da amostra A_1. Redes desta amostra são polytrees com 5 nós.

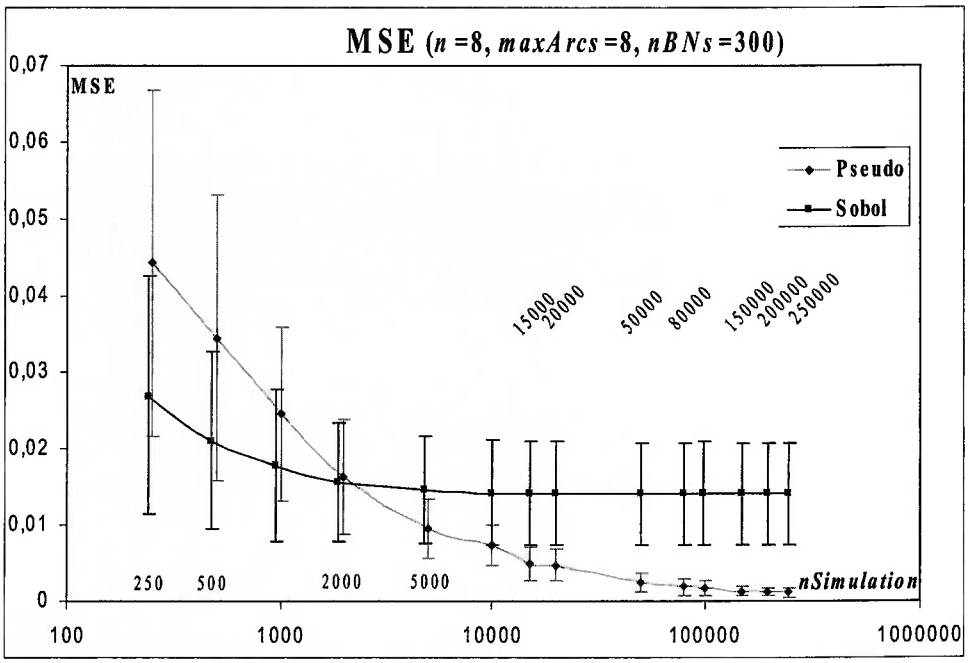


Figura 5.6. Média dos MSE em função de *nSimulation* para redes da amostra A_2. Redes desta amostra possuem 8 nós e 8 arcos.

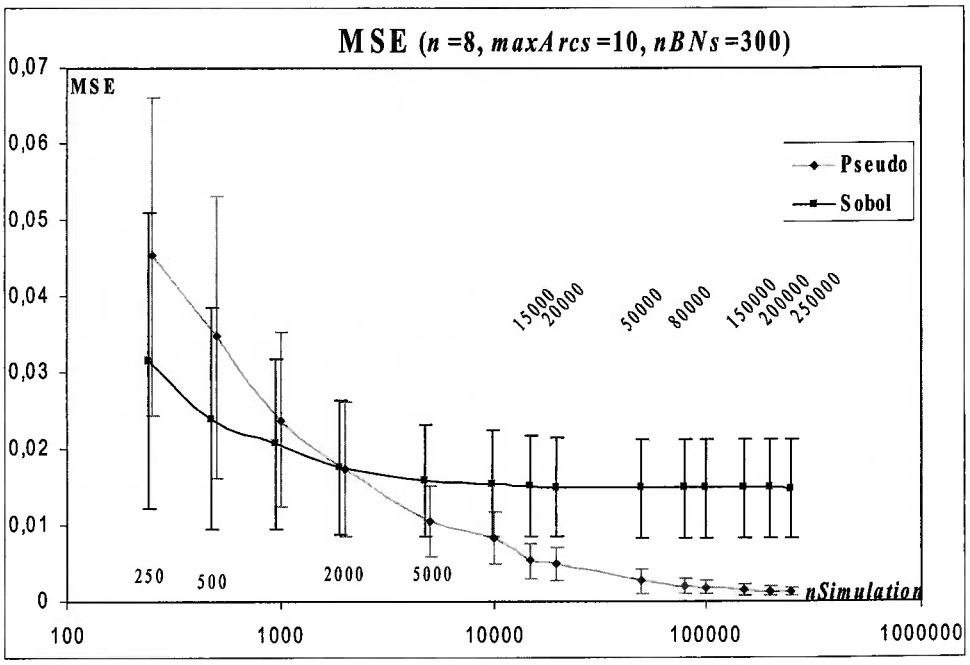


Figura 5.7. Média dos MSE em função de *nSimulation* para redes da amostra A_3. Redes desta amostra possuem 8 nós e 10 arcos.

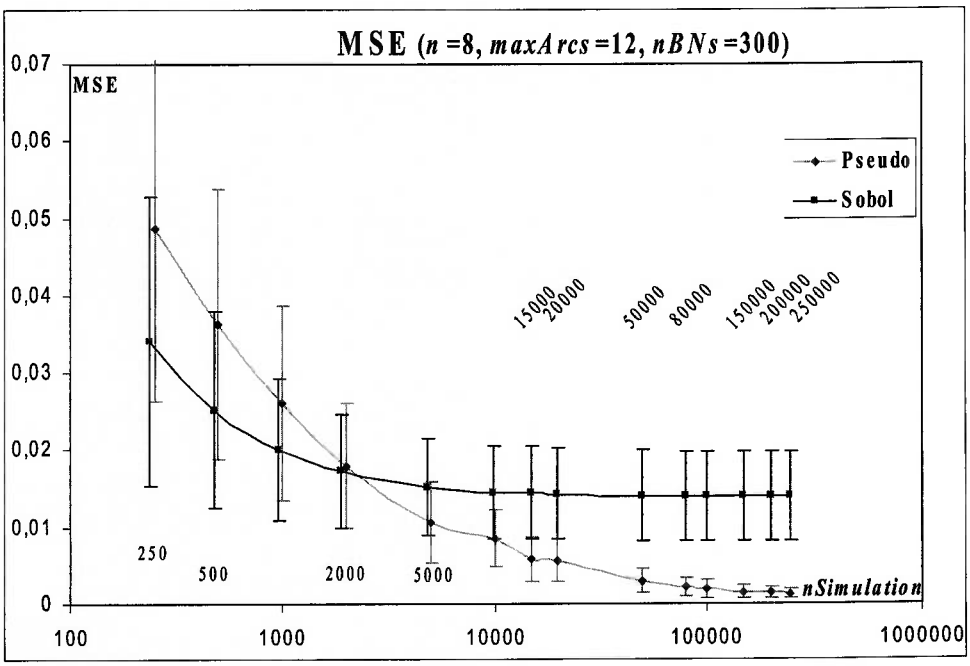


Figura 5.8. Média dos MSE em função de *nSimulation* para redes da amostra A_4. Redes desta amostra possuem 8 nós e 12 arcos.

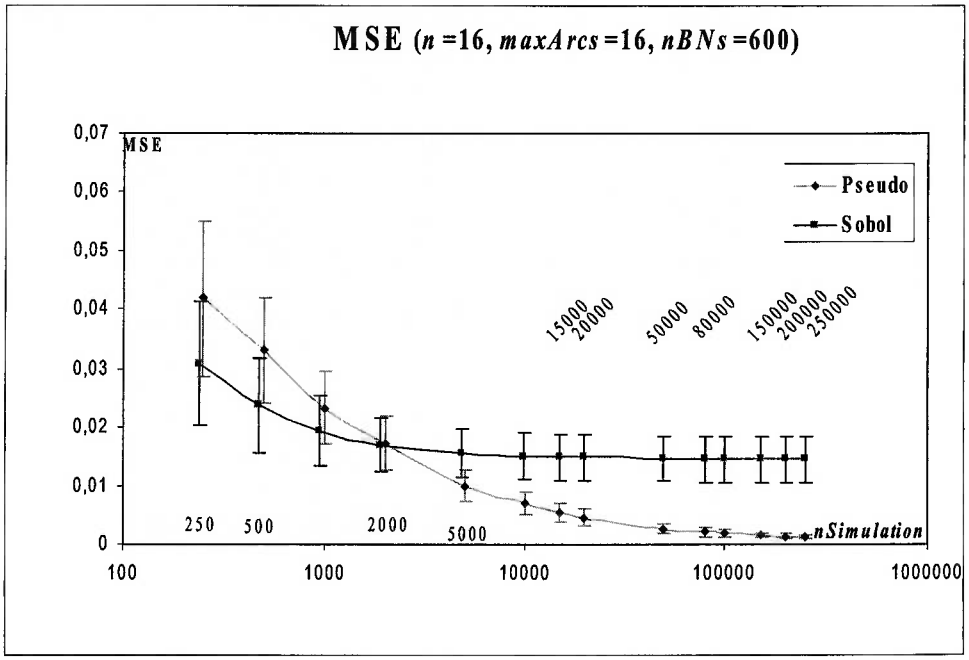


Figura 5.9. Média dos MSE em função de $nSimulation$ para redes da amostra A_5. Redes desta amostra possuem 16 nós e 16 arcos.

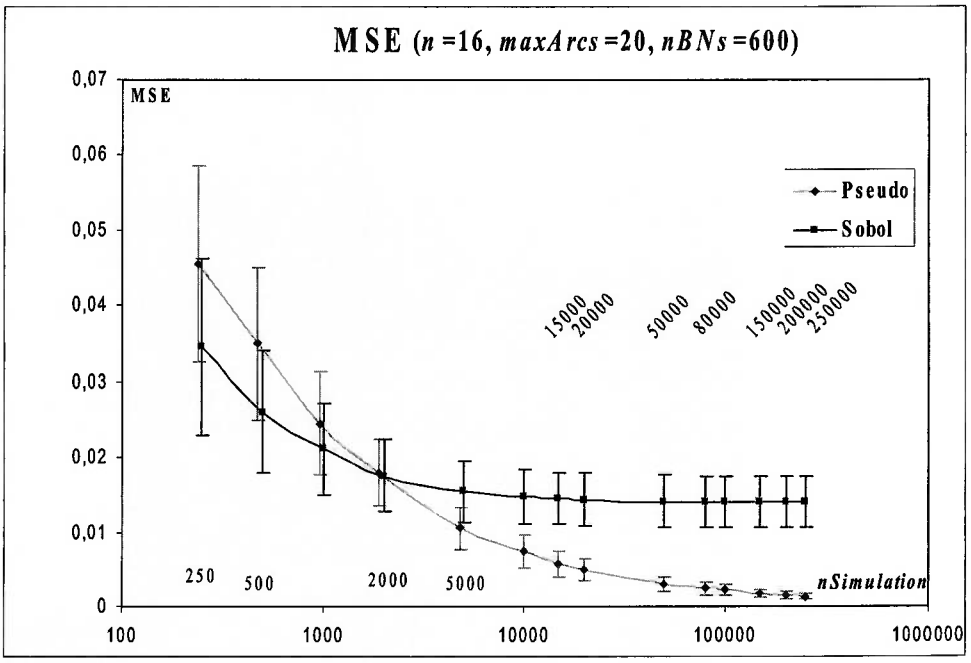


Figura 5.10. Média dos MSE em função de $nSimulation$ para redes da amostra A_6. Redes desta amostra possuem 16 nós e 20 arcos.

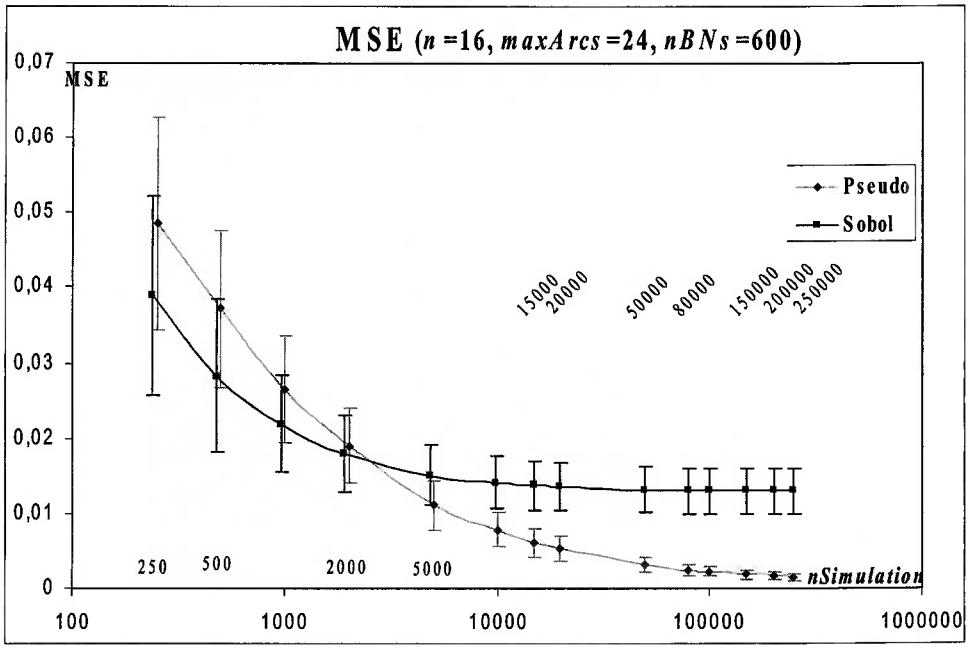


Figura 5.11. Média dos MSE em função de *nSimulation* para redes da amostra A_7. Redes desta amostra possuem 16 nós e 24 arcos.

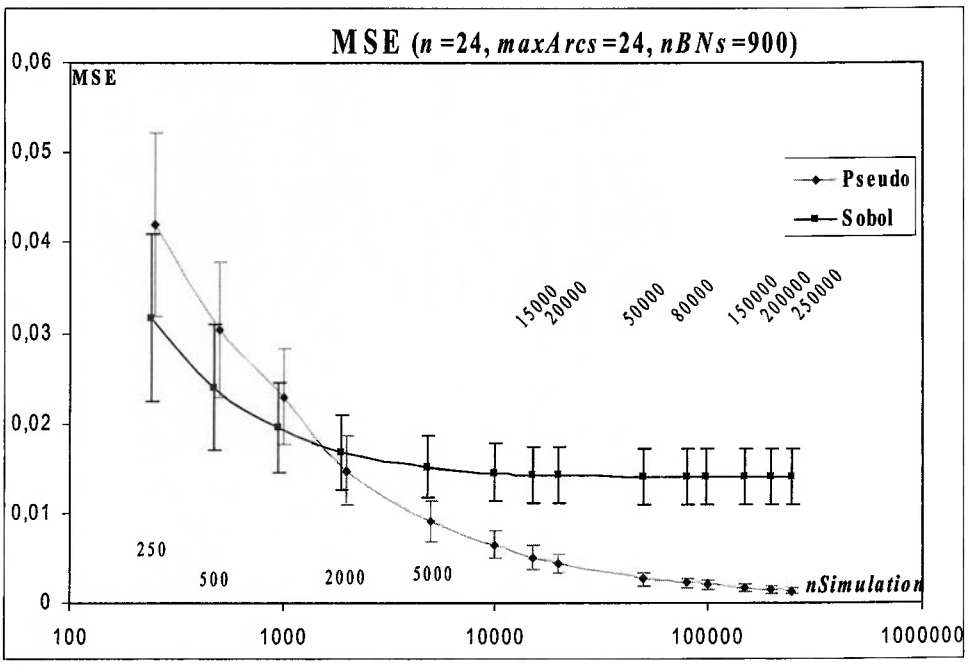


Figura 5.12. Média dos MSE em função de *nSimulation* para redes da amostra A_8. Redes desta amostra possuem 24 nós e 24 arcos.

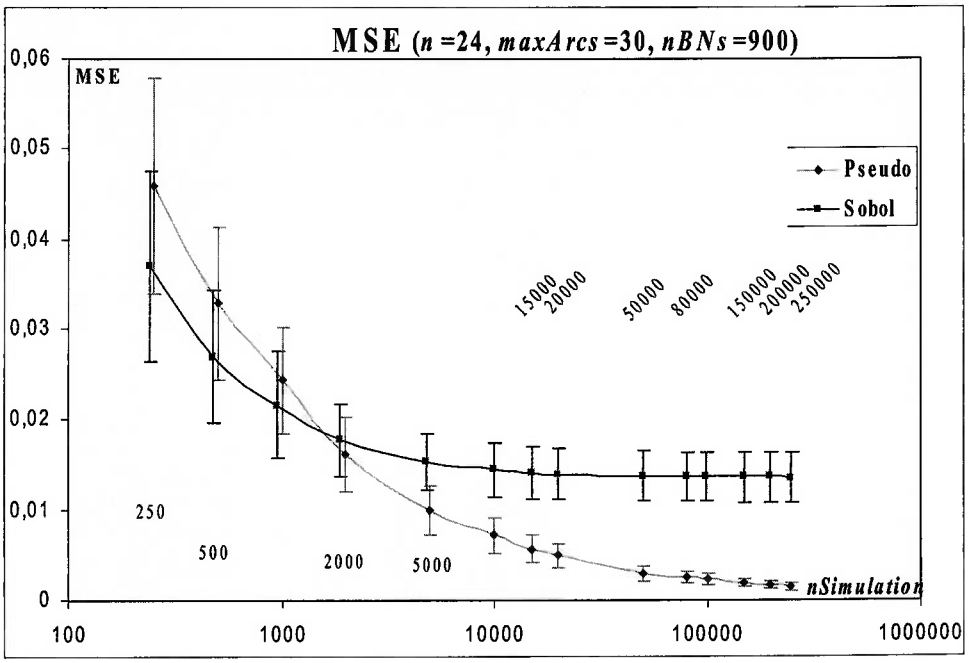


Figura 5.13. Média dos MSE em função de *nSimulation* para redes da amostra A_9. Redes desta amostra possuem 24 nós e 30 arcos.

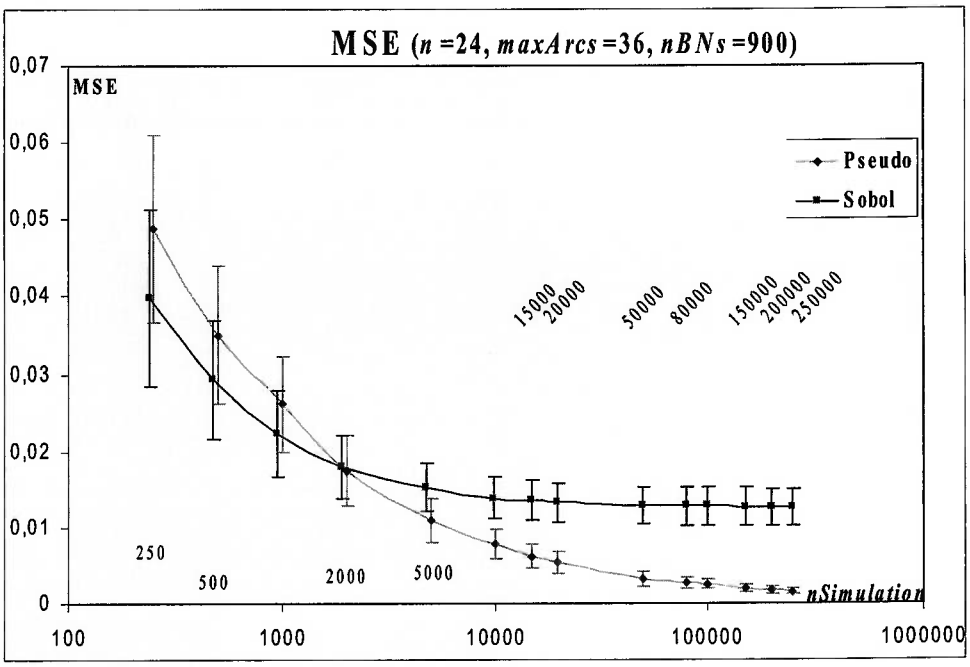


Figura 5.14. Média dos MSE em função de *nSimulation* para redes da amostra A_{10}. Redes desta amostra possuem 24 nós e 36 arcos.

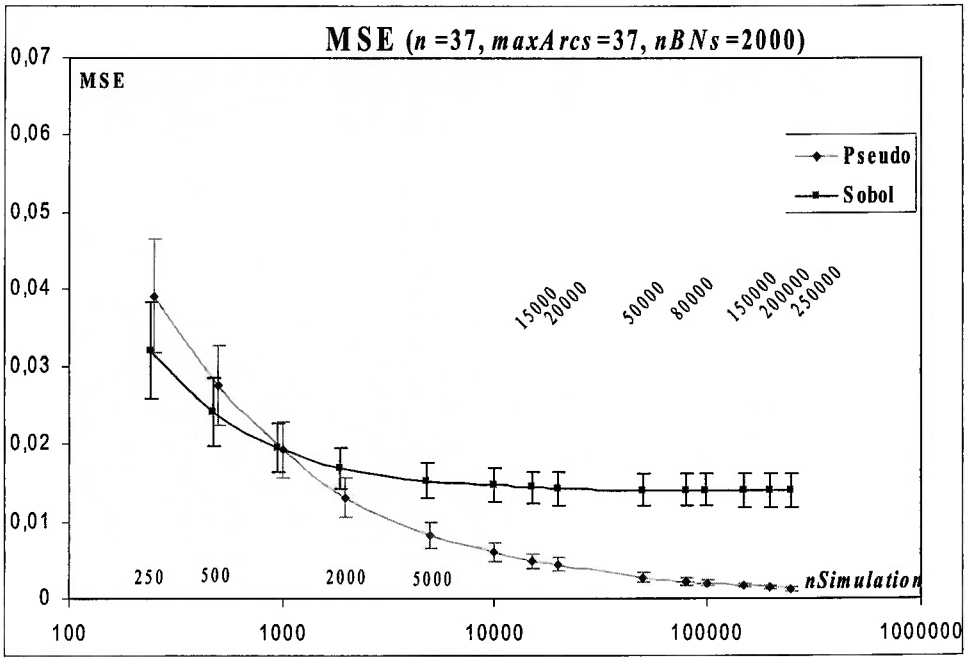


Figura 5.15. Média dos MSE em função de $nSimulation$ para redes da amostra A_{11}. Redes desta amostra possuem 37 nós e 37 arcos.

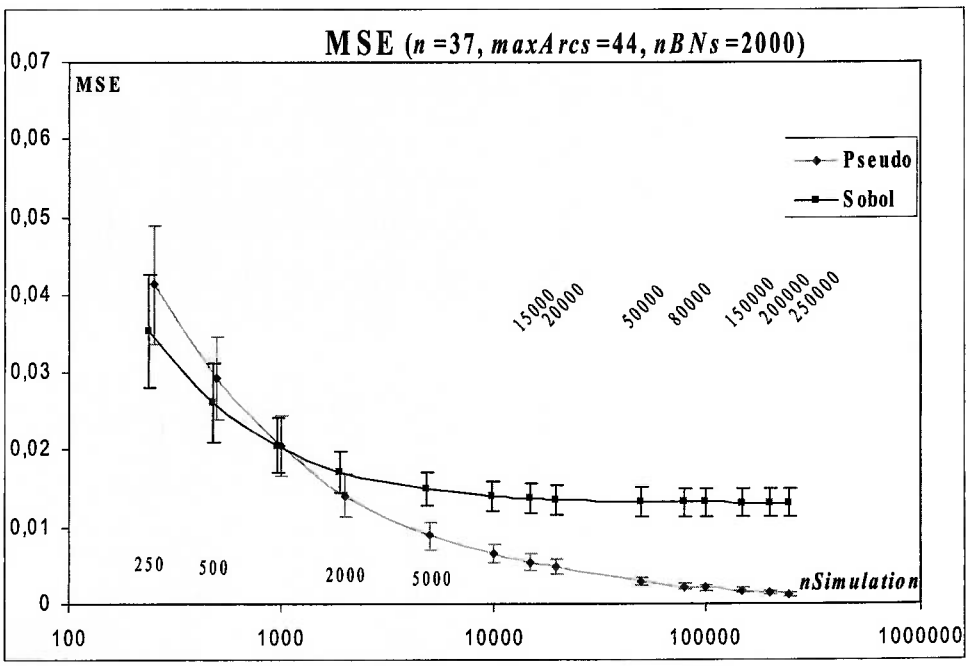


Figura 5.16. Média dos MSE em função de $nSimulation$ para redes da amostra A_{12}. Redes desta amostra possuem 37 nós e 44 arcos.

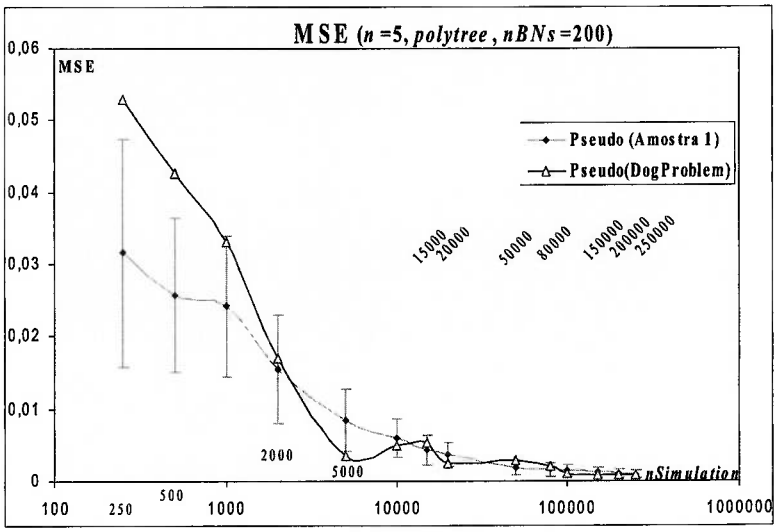


Figura 5.17. Comparação da média dos MSE em função de *nSimulation* para redes da amostra A_1, e valores MSE da rede *DogProblem*. Ambos com seqüência pseudo-aleatória.

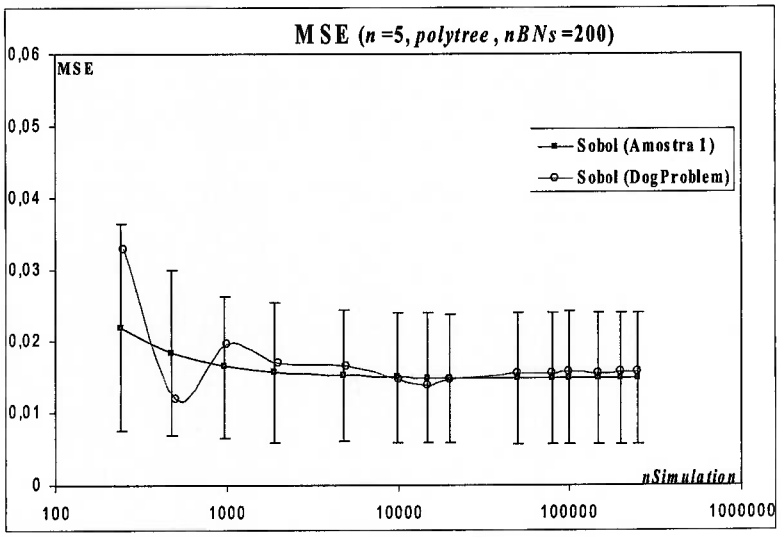


Figura 5.18. Comparação da média dos MSE em função de *nSimulation* para redes da amostra A_1, e valores MSE da rede *DogProblem*. Ambos com seqüência quasi-aleatória.

Nas Figuras 5.17 a 5.22, apresentam-se as médias dos valores MSE, assim como valores do MSE de uma rede Bayesiana em particular, com o objetivo de verificar como um resultado particular se enquadra dentro do resultado médio.

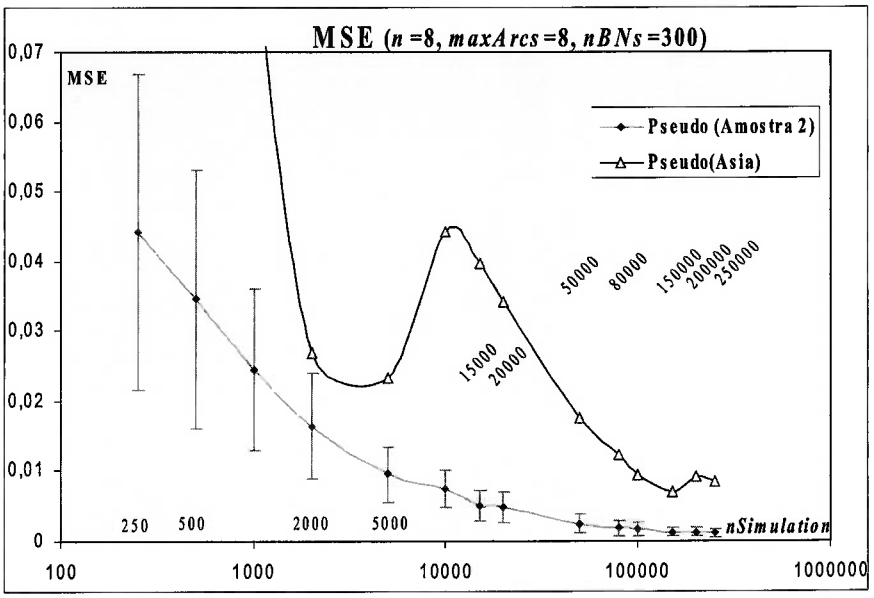


Figura 5.19. Comparação da média dos MSE em função de *nSimulation* para redes da amostra A_2, e valores MSE da rede *Asia*. Ambos com seqüência pseudo-aleatória.

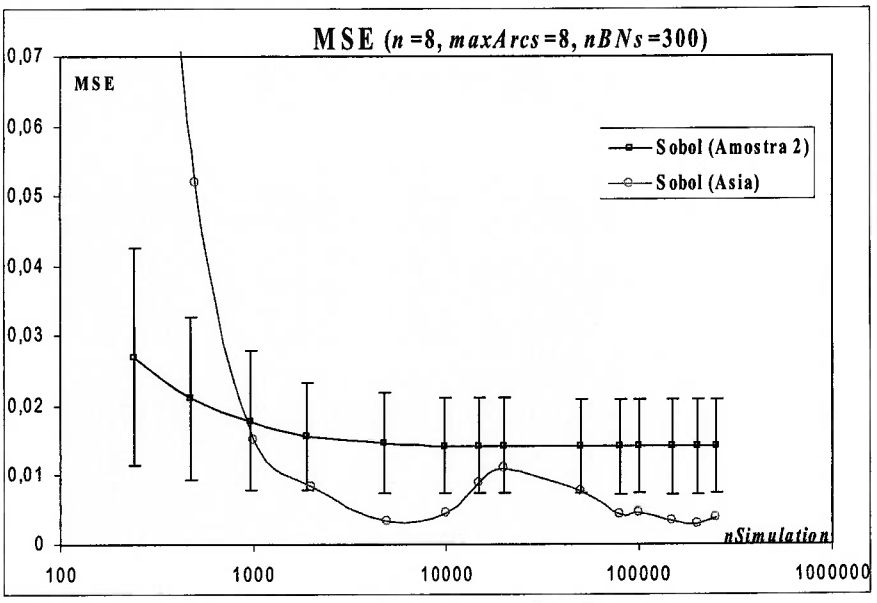


Figura 5.20. Comparação da média dos MSE em função de *nSimulation* para redes da amostra A_2, e valores MSE da rede *Asia*. Ambos com seqüência quasi-aleatória (*Sobol*).

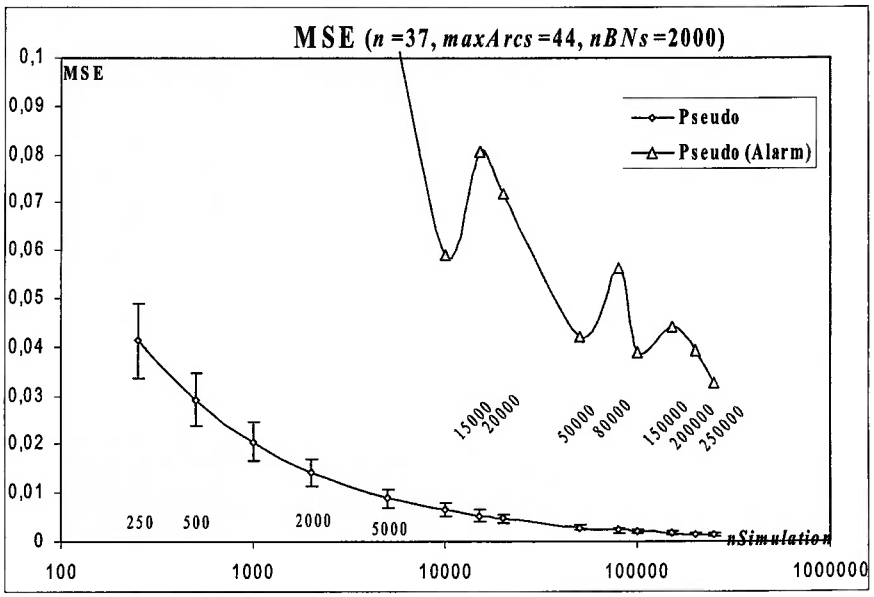


Figura 5.21. Comparação da média dos MSE em função de *nSimulation* para redes da amostra A_{12}, e valores MSE da rede *Alarm*. Ambos com seqüência pseudo-aleatória.

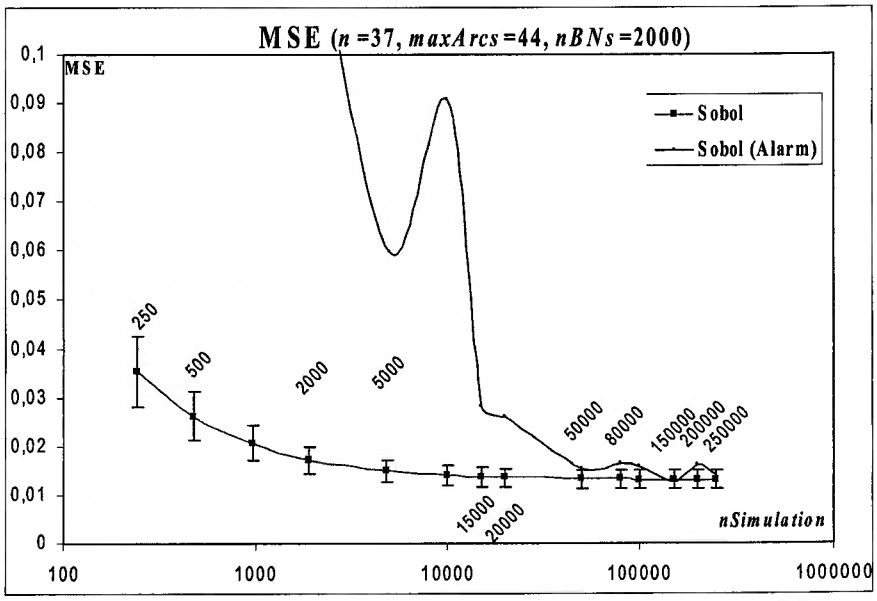


Figura 5.22. Comparação da média dos MSE em função de *nSimulation* para redes da amostra A_{12}, e valores MSE da rede *Alarm*. Ambos com seqüência quasi-aleatória (*Sobol*).

5.5 Discussão dos resultados

Nesta seção, o termo "resultados Pseudo" será empregado para significar resultados obtidos pela aplicação direta de métodos MCMC (Gibbs sampling), com seqüências pseudo-aleatórias. O termo "resultados Sobol", indica resultados obtidos pelo emprego de seqüências quasi-aleatórias em Gibbs sampling.

Pelas Figuras 5.5 a 5.16, concluem-se que os resultados Sobol são piores que os de Pseudo a partir de $nSimulation = 2000$, isto é, para tamanho da amostra de 2000 redes. Pelas Figuras 5.17 a 5.22, nas quais se procuram comparar valores médios MSE, obtidos por amostras de redes Bayesianas, com valores MSE obtidos por testes em redes particulares, conclui-se que testes isolados estão sujeitos à grandes variações para pequenos tamanhos de amostra ($nSimulation$). Pelos testes nas redes *Asia* e *Alarm* (Figuras 6.3 e 6.4), concluir-se-ia que o desempenho de seqüências quasi-aleatórias (Sobol) é superior a pseudo-aleatórias, entretanto, observa-se que (5.5 a 5.16), em geral (na média), o desempenho de seqüências pseudo-aleatórias é superior, e a conclusão seria equivocada.

Pode-se concluir ainda que, resultados Sobol são melhores quando as probabilidades das distribuições condicionais associadas a cada variável assumem valores extremos, por exemplo, distribuições do tipo [0,01 0,99], como é o caso das redes *Asia* e *Alarm*. Entretanto, os resultados gerais (5.5 a 5.16) levam a concluir que métodos quasi-Monte Carlo (resultados Sobol) estão sujeitos a valores de erros MSE constantes, ou seja, não convergem para a distribuição correta. Métodos quasi-Monte Carlo devem ser utilizados com bastante cuidado.

Capítulo 6

CONCLUSÕES FINAIS

Os resultados obtidos neste trabalho são:

1. Foram introduzidos novos algoritmos para geração de redes Bayesianas aleatórias distribuídas uniformemente, tanto redes multi-conectadas, como poly-trees. Os algoritmos desenvolvidos são flexíveis, permitindo a especificação do número total de arcos e de graus; além disso, características adicionais de redes Bayesianas podem ser incorporadas. A desvantagem dos algoritmos é que muitas redes devem ser geradas antes que uma amostra seja retirada (isto é, deve-se esperar pela convergência da cadeia de Markov). No programa implementado, BNGenerator, observou-se que os algoritmos são rápidos, e portanto, um número alto de iterações são realizadas em instantes.

2. Desenvolveu-se um programa, o BNGenerator, para geração aleatória de redes Bayesianas. O programa, codificado em *Java*, pode ser obtido na Internet[1], e ser rodado no computador do usuário, através de uma interface em linha de comando. O código fonte do programa é também disponibilizado para eventuais modificações que o usuário deseje efetuar.

3. Obtiveram-se novos resultados experimentais com o objetivo de ilustrar a utilidade de um gerador aleatório de redes Bayesianas para análise de eficiência de algoritmos de inferência. Foi possível, graças a testes em um número grande de redes Bayesianas, detectar casos em que métodos quasi-Monte Carlo, aplicados

[1]Programa BNGenerator disponível em :<http://www.pmr.poli.usp.br/ltd/Software/BNGenerator>

em Gibbs samping, possuem bom desempenho. Observou-se também, que de modo geral (isto é, para uma amostra representativa de redes Bayesianas), o performance de métodos quasi-Monte Carlo não é satisfatório.

Este trabalho abre caminho para inúmeras pesquisas futuras. Observou-se que, a determinação do número de transições necessárias para que uma cadeia de Markov (Capítulo 3) convirja para uma distribuição estacionária é uma questão ainda em aberto, pois depende de cada caso em particular. Sinclair [37] dá diretrizes para tal aprofundamento. Uma outra linha de desenvolvimento e continuidade deste trabalho é estudar redes Bayesianas com aplicações reais, extrair características peculiares destas redes, e orientar a geração de redes aleatórias segundo estas características [65]. Uma terceira linha de pesquisa é aprofundar no estudo da relação de dependência que existe entre seqüências quasi-aleatórias e a cadeia de Markov (gerada em Gibbs sampling), pois esta dependência origina o erro constante de convergência [62]. Uma possibilidade de se quebrar esta dependência é aplicar *processos de renovação* [66].

Referências Bibliográficas

[1] DURKIN, J., **Expert Systems: Design and Development**. New York: Macmillan, 1994.

[2] RUSSELL, S. J.; NORVIG, P., **Artificial Intelligence. A Modern Approach**. Englewood Cliffs: Prentice-Hall, 1995.

[3] THRUN, S., *Probabilistic Algorithms in Robotics*, em **AI Magazine**, v. 21, p. 93–109, 2000.

[4] JENSEN, F. V., **An Introduction to Bayesian Networks**. New York: Springer-Verlag, 1996.

[5] AUAI-COMPANIES. Uma lista de diversas empresas e grupos de pesquisa que desenvolvem aplicações empregando redes Bayesianas está disponível em: <http://www.auai.org/auai-companies.html.>.

[6] PEARL, J., **Probabilistic Reasoning in Intelligent Systems**. Morgan-Kaufman, 1988.

[7] CHARTRAND, G.; OELLERMANN, O. R., **Applied and Algorithmic Graph Theory**. New York: McGraw-Hill, 1993.

[8] CASTILLO, E.; GUTIERREZ, J. M.; HADI, A. S., **Expert Systems and Probabilistic Network Models**. New York: Springer, 1 ed., 1997.

[9] CANNINGS, C.; THOMPSON, E. A.; SKOLNICK, M. H., *Probability functions on complex pedigrees*, **Advances in Applied Probability**, v. 10, p. 26–61, 1978.

[10] SHENOY, P. P.; SHAFER, G., *Axioms of probability and belief-function propagation*, em **Uncertainty in Artificial Intelligence 4** (SHACHTER, R. D.;

LEVITT, T. S.; KANAL, L. N.; LEMMER, J. F., eds.), (Amsterdam), p. 169–198, North-Holland, 1990.

[11] ZHANG, N. L.; POOLE, D., *Exploiting Causal Independence in Bayesian network Inference*, em **Journal of Artificial Intelligence Research**, p. 301–328, Nov. 1996.

[12] COZMAN, F. G., *Generalizing Variable Elimination in Bayesian Networks*, em **Proceedings of the IBERAMIA/SBIA 2000 Workshops**, (São Paulo), p. 27–32, Tec Art Editora, 2000.

[13] DECHTER, R., *Bucket elimination: A unifying framework for probabilistic inference*, em **Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI-96)** (HORVITZ, E.; JENSEN, F., eds.), (San Francisco), p. 211–219, Morgan Kaufmann Publishers, Aug. 1–4 1996.

[14] IDE, J. S.; COZMAN, F. G., *Raciocínio Probabilístico em Sistemas Embarcados*, em **Anais do XXI Congresso da Sociedade Brasileira de Computação**, (Fortaleza), : Ed. Ana Teresa Martins, 2001.

[15] RAMOS, F. T.; COZMAN, F. G.; IDE, J. S., *Embedded Bayesian Networks: Anyspace, Anytime Probabilistic Inference*, em **AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems**, AAAI Press, 2002.

[16] SOBOL, I. M., **A Primer for Monte Carlo methods**. Florida: CRC Press, Inc., 1994.

[17] HENRION, M., *Propagation of uncertainty in Bayesian networks by probabilistic logic sampling*, em **Uncertainty in Artificial Intelligence 2** (LEMMER, J. F.; KANAL, L. N., eds.), (Amsterdam, London, New York), p. 149–163, Elsevier/North-Holland, 1988.

[18] FUNG, R.; CHANG, K. C., *Weighting and integrating evidence for stochastic simulation in Bayesian networks*, em **Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence (UAI-89)**, (Windsor, Ontario), Morgan Kaufmann, 1989.

[19] SHACHTER, R. D.; PEOT, M. A., *Simulation approaches to general probabilistic inference on belief networks*, em **Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence (UAI-89)**, (Windsor, Ontario), p. 221–231, Morgan Kaufmann, 1989.

[20] GENTLE, J. E., **Random Number Generation and Monte Carlo Methods**. Statistics and Computing, New York NY: Springer, 1998.

[21] NEAL, R. M., *Probabilistic Inference Using Markov Chain Monte Carlo Methods*, Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, Sept. 1993.

[22] GILKS, W. R.; RICHARDSON, S.; SPIEGELHALTER, D. J., **Markov Chain Monte Carlo in Practice.** Chapman&Hall, 1996.

[23] GAMERMAN, D., **Markov chain Monte Carlo: Stochastic Simulation for Bayesian Inference**. New York: Chapman&Hall, 1997.

[24] METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M. N.; TELLER, A. H.; TELLER, E., *Equation of State Calculations by Fast Computing Machines*, **Journal of Chemical Physics**, v. 21, p. 1087–1092, 1953.

[25] GEMAN, D.; GEMAN, S., *Stochastic relaxation, Gibbs distribution and Bayesian restoration of images*, **IEEE Trans. PAMI**, v. 6, n. 6, p. 721–741, 1984.

[26] GELFAND, A. E.; SMITH, A. F. M., *Sampling-Based Approaches to Calculating Marginal Densities*, **Journal of the American Statistical Association**, v. 85, p. 398–409, June 1990.

[27] CASELLA, G.; GEORGE, E. I., *Explaining the Gibbs Sampler*, **The American Statistician**, v. 46, p. 167–174, Aug. 1992.

[28] DEGROOT, M. H., **Optimal Statistical Decisions**. New York: McGraw-Hill, 1970.

[29] UAI-MAILING LIST. Lista de discussões da UAI (Uncertainty Artificial Intelligence). Arquivo disponível em: <http://cs.oregonstate.edu/~ dambrosi/~ uai-archive-00-01/0871.html.>.

[30] RIPLEY, B. D., **Stochastic Simulation**. Wiley series in probability and mathematical statistics, New York, NY, USA; London, UK; Sydney, Australia: John Wiley and Sons, 1987.

[31] CAPRILE, B., *Uniformly Generating Distribution Functions for Discrete Random Variables*, Technical Report MATH-0011025, Alamos e-print archive, Nov. 2000.

[32] COWELL, R. G.; DAWID, A. P.; LAURITZEN, S. L.; SPIEGELHALTER, D. J., **Probabilistic Networks and Expert Systems**. Statistics for Engineering and Information Science, New York: Springer-Verlag, 1999.

[33] ROBINSON, R. W., *Counting labeled acyclic digraphs*, em **New Directions in the Theory of Graphs** (HARARY, F., ed.), (Michigan), p. 28–43, Academic Press, 1973.

[34] TINHOFER, G., *Generating Graphs Uniformly at Random*, em **Computational graph theory** (TINHOFER, G.; ALBRECHT, R. R.; OTHERS, , eds.), v. 7 de **Computing. Supplementum**, (Wien / New York), p. 235–255, Springer, 1990.

[35] XIANG, Y.; MILLER, T., *A Well-Behaved Algorithms for Simulating Dependence Structure of Bayesian Networks*, em **International Journal of Applied Mathematics**, v. 1, p. 923–932, 1999.

[36] MELANÇON, G.; BOUSQUE-MELOU, M., *Random Generation of Dags for Graph Drawing*, Technical Report technical report INS-R0005, Dutch Research Center for Mathematical and Computer Science-CWI, 2000.

[37] SINCLAIR, A., **Algorithms for Random Generation and Counting, a Markov Chain Approach**. Birkhaüser, 1992.

[38] ROSS, S. M., **Stochastic Processes**. John Wiley and Sons; New York, NY, 1983.

[39] RESNICK, S. I., **Adventures in Stochastic Processes**. Cambridge, MA, USA; Berlin, Germany; Basel, Switzerland: Birkhäuser, 1992.

[40] IDE, J. S.; COZMAN, F. G., *Random generation of Bayesian networks*, em **Proc. of XVI Brazilian Symposium on Artificial Intelligence**, Springer-Verlag, 2002.

[41] SPIEGEL, M. R., **Estatística**, cap. 12, p. 332. Coleção Schaum, São Paulo: Editora McGRAW-HILL, 7a. ed., 1974.

[42] CERN-Centro Europeu de Pesquisas Nucleares. Biblioteca de classes em Java disponíveis na Internet em: <http://tilde-hoschek.home.cern.ch/~hoschek/colt/>.

[43] MATSUMOTO, ; NISHIMURA, , *Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*, **ACMTMCS: ACM Transactions on Modeling and Computer Simulation**, v. 8, 1998.

[44] MELANÇON, G.; HERMAN, I., *DAG drawing from an information visualization perspective*, Technical Report INS-R9915, CWI - Centrum voor Wiskunde en Informatica, Nov. 1999.

[45] FANG, K. T.; WANG, Y., **Number Theoretic Methods in Statistics**. New York: Chapman & Hall, 1994.

[46] NIEDERREITER, H., **Random Number Generation and Quasi-Monte Carlo Methods**, v. 63 de **CBMS-NSF regional conference series in Appl. Math.** Philadelphia: SIAM, 1992.

[47] HALTON, J. H., *On the efficiency of certain quasirandom sequences of points in evaluating multidimensional integrals*, em **Numerische Mathmatik**, v. 2, p. 84–90, 1960.

[48] SOBOL, I. M., *The distribution of points in a cube and the approximate evaluation of integrals*, **USSR Comp. Math. Math. Phys.**, v. 7, n. 4, p. 86–112, 1967.

[49] FAURE, H., *Discrepance de suites associees a un systeme de numeration en dimension s*, em **Acta Arithmetica**, v. 41, p. 337–351, 1982.

[50] FOX, B. L., *Algorithm 647: Implementation and Relative Efficiency of Quasirandom Sequence Generators*, **ACM Trans. Math. Softw.**, v. 12, p. 362–376, Dec. 1986.

[51] ANTONOV, I. A.; SALEEV, V. M., *An economic method of computing lp_t-sequences*, **U.S.S.R. Computational Mathematics and Mathematical Physics**, v. 19, p. 252–256, Dec. 1979.

[52] BRATLEY, P.; FOX, B. L., *Implementing Sobol's Quasirandom Sequence Generator*, **ACM Trans. Math. Softw.**, v. 14, p. 88–100, Mar. 1988.

[53] NIEDERREITER, H., *Quasirandom sampling computer graphics*, em **Proc. of the 3rd International Seminar on Digital Image Processing in Medicine**, 1992.

[54] MOROKOFF, W. J.; CAFLISCH, R. E., *Quasi-Monte Carlo integration*, **J. Comp. Phys.**, v. 122, p. 218–230, 1995.

[55] PASKOV, S. H.; TRAUB, J. F., *Faster valuation of financial derivatives*, **J. Portfolio Management**, v. 22, p. 113–120, Fall 1995.

[56] PAPAGEORGIOU, A. F.; TRAUB, J. F., *Faster evaluation of multidimensional integrals*, **Comp. Phys.**, v. 11, p. 574–578, Nov. 1997.

[57] BRATLEY, P.; FOX, B. L.; NIEDERREITER, H., *Implementation and Tests of Low Discrepancy Sequences*, **ACM Transactions on Modeling and Computer Simulation**, v. 2, p. 195–213, July 1992.

[58] MOROKOFF, W. J.; CAFLISCH, R. E., *Quasi-Random Sequences and Their Discrepancies*, **SIAM Journal on Scientific Computing**, v. 15, p. 1251–1279, Nov. 1994.

[59] PASKOV, S. H., *New methodologies for valuing derivatives*, em **Mathematics of Derivative Securities** (PLISKA, S.; DEMPSTER, M., eds.), p. 545–582, Cambridge: Cambridge University Press, 1997.

[60] CHENG, J.; Druzdzel, Marek, J., *Computational Investigation of Low-Discrepancy Sequences in Simulation Algorithms for Bayesian Networks*, em

Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00) (BOUTILIER, C.; GOLDSZMIDT, M., eds.), (SF, CA), p. 72–81, Morgan Kaufmann Publishers, June 30– July 3 2000.

[61] SHAW, J. E. H., *A Quasirandom Approach to Integration in Bayesian Statistics*, em **The Annals of Statistics**, v. 16, p. 895–914, 1988.

[62] LIAO, J. G., *Variance Reduction in Gibbs Sampler Using Quasi Random Numbers*, **Journal of Computational and Graphical Statistics**, v. 7, p. 253–266, Sept. 1998.

[63] GEWEKE, J., *Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments*, em **Bayesian Statistics 4**, (New York), p. 169–193, Oxford University Press, 1992.

[64] GELMAN, A.; RUBIN, D. B., *Inference from iterative simulation using multiple sequences*, **Statistical Science**, v. 7, p. 457–472, 1992.

[65] CHICKERING, D. M.; MEEK, C., *Finding Optimal Bayesian Networks*, em **Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-2002)** (DARWICHE, A.; FRIEDMAN, N., eds.), (San Francisco), p. 94–102, Morgan Kaufmann, Aug. 1–4 2002.

[66] MYKLAND, P.; TIERNEY, L.; YU, B., *Regeneration in Markov Chain Samplers*, **Journal of the American Statistical Association**, v. 90, p. 233–241, Mar. 1995.