Felipe Leal Valentim

ESTUDO E IMPLEMENTAÇÃO DE ALGORITMOS DE INFERÊNCIA E APRENDIZADO EM REDES *BAYESIANAS*

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS - BRASIL
2007

Felipe Leal Valentim

ESTUDO E IMPLEMENTAÇÃO DE ALGORITMOS DE INFERÊNCIA E APRENDIZADO EM REDES *BAYESIANAS*

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:
Inteligência Artificial

Orientador:
Prof. Rudini Menezes Sampaio

Co-Orientador:
Prof.Ricardo Martins de Abreu Silva

LAVRAS
MINAS GERAIS – BRASIL
2007

Ficha Catalográfica preparada pela Divisão de Processos Técnico da Biblioteca Central da UFLA

Valentim, Felipe Leal

Estudo e implementação de algoritmos de inferência e aprendizado em redes *bayesianas* / Felipe Leal Valentim. Lavras – Minas Gerais, 2007. 93p.: il.

Monografia de Graduação –Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Informática. 2. Inteligência Artificial. 3. Redes *Bayesianas*. I. VALENTIM, F. L. II. Universidade Federal de Lavras. III. Título.

Felipe Leal Valentim

ESTUDO E IMPLEMENTAÇÃO DE ALGORITMOS DE INFERÊNCIA E APRENDIZADO EM REDES *BAYESIANAS*

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 19/03/2007

_____
Prof. Thelma Safadi

_____
Prof. Wilian Soares Lacerda

_____
Prof. Rudini Menezes Sampaio
(Orientador)

_____
Prof. Ricardo Martins de Abreu Silva
(Co-Orientador)

LAVRAS
MINAS GERAIS – BRASIL

“Repita por pura alegria de viver: A salvação é pelo risco, sem o qual a vida não vale a pena!”
Clarice Lispector

“Um país sem pesquisa é um país sem futuro”.
César Freire Carvalho

Aos meus pais, Valter Álvares Valentim e Vânia Mara Leal Valentim pelo apoio incontestável, dedicação, amor, carinho, educação e por todos os outro motivos que me fazem sentir orgulho de ser seu filho. A meus irmãos Ivan e Júlia por fazerem parte da minha vida e serem a minha família. A querida Daniela, por ter cruzado meu caminho;
Dedico.

AGRADECIMENTOS

Aos meus pais, Valter Valentim e Vânia Mara, que não pouparam esforços para minha formação, seja de caráter ou profissional, a quem devo tudo e expresso meu eterno amor e gratidão.

A Universidade Federal de Lavras, pela oportunidade de realização de um curso de graduação.

Ao professor Rudini Menezes Sampaio pela orientação, por me inserir e despertar a ambição pela pesquisa e por as idéias que acabaram se transformando nessa monografia.

Ao professor Ricardo Martins de Abreu Silva pelas e valiosas dicas para a realização desse trabalho.

A todos os professores do DCC por transmitirem seus conhecimentos.

Aos todos os funcionários do DCC, em especial a Ângela e ao Deivson, por sempre me ajudarem, pelo bom humor e pela amizade.

A toda 2º turma do curso de Ciência da Computação do ano de 2002, pela amizade, apoio e por todos os ótimos momentos vividos nesses anos.

Aos amigos Ouropretanos e Ipatinguenses pela amizade, apoio e incentivo.

Aos amigos republicanos, Cajuru, Piccolo, Saulo, Haroldo, Rodolfo, Carijó e Pitta, por fazerem papel de família nesses anos vividos em Lavras, e por todas as aventuras que passamos juntos.

RESUMO

Redes *bayesianas* são grafos acíclicos dirigidos que representam dependências entre variáveis em um modelo probabilístico. Esta abordagem representa uma boa estratégia para lidar com problemas que tratam incertezas. A tarefa mais comum que desejamos efetuar numa rede *bayesiana* é determinar as várias probabilidades de interesse condicionadas a certos eventos que observamos. Estas probabilidades não são armazenadas diretamente no modelo, e consequentemente precisam ser computadas por algoritmos de inferência. Outra questão importante nessa abordagem é que a construção manual de uma rede *bayesiana* pode ser um processo bastante trabalhoso e caro para grandes aplicações e em domínios complexos. Por esse motivo, esforços têm sido dirigidos para o desenvolvimento de algoritmos de aprendizado que possam construir redes *bayesianas* diretamente de uma base de dados, ao invés do discernimento de especialistas humanos. Nesse trabalho são abordados aspectos fundamentais para a construção de um sistema capaz de realizar a inferência e o aprendizado em redes *bayesianas*, além disso, é apresentado o sistema *UFLABayes*, que permite uma análise dos algoritmos estudados e por fim são apresentadas as análises de eficiência para os algoritmos de inferência e aprendizado implementados.

Palavras-chave: Rede *bayesiana*, algoritmos de inferência, algoritmos de aprendizado**.**

ABSTRACT

Bayesian networks are directed acyclics graphs that represent dependences between variables in probabilistic models. This approach represents a good strategy to solve problems that treat uncertainties. The commonest task that we wish to make in a bayesian network is to determine the several interest probabilities conditioned to some events we observe. These probabilities are not stored directly in the model, and consequently need to be computed by inference algorithms. Another important quastion in this approach is that the manual construction of a bayesian net can be very laborious and expensive process for great applications and in complex domains. Therefore, efforts have been driven to development of learning algorithms that can build bayesian nets directly of a database, instead of the specialists' human discernment. In this work are explained fundamental aspects necessary for the construction of a system able to perform the inference and learning in bayesian networks, it is introduced the system UFLABayes, that allows analysis of the studied algorithms, and are introduced the analyses of the inference and learning algorithms implemented.

Keywords: *Bayesian networks, inference algorithms, learning algorithms.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Segundo [JENSEN, 2001], os estudos em Inteligência Artificial podem ser divididos em duas grandes áreas: o desenvolvimento de sistemas que agem como humanos (robôs) e o de sistemas que agem racionalmente. Dentro do contexto dos sistemas que agem racionalmente, duas abordagens principais são utilizadas: raciocínio lógico e raciocínio probabilístico. Quando se conhece fatos suficientes acerca de um problema, o raciocínio lógico permite a construção de um sistema inteligente capaz de derivar soluções que ofereçam a garantia de um bom funcionamento. Porém, nem sempre se dispõem de todas as verdades de um ambiente, e devemos construir sistemas capazes de agir sob incerteza, e para estes casos, o raciocínio probabilístico surge como uma boa opção [RUSSEL & NORVIG, 2004].

Um sistema que possa atuar em situações de incerteza deve ser capaz de atribuir níveis de confiabilidade para todas as sentenças em sua base de conhecimento, e ainda, estabelecer relações entre as sentenças [RUSSEL & NORVIG, 2004]. Uma maneira de estruturar um problema para raciocinar sobre incerteza é construir um grafo representando as relações causais entre as sentenças [JENSEN, 2001]. Redes *bayesianas* oferecem uma abordagem para o raciocínio probabilístico que engloba teoria dos grafos, para o estabelecimento das relações entre sentenças, e ainda, teoria da probabilidade, para atribuição de níveis de confiabilidade, contemplando as necessidades de se tratar a incerteza.

A representação visual das redes *bayesianas*, permite uma melhor comunicação com o especialista do domínio dos dados, favorecendo maior rapidez na definição, desenvolvimento ou modificação do modelo, o que por sua vez, ajuda a compreender melhor o problema. A topologia[1] da rede especifica os relacionamentos de independência condicional que são válidas no domínio.

Com ajuda de um especialista num certo domínio de dados, define-se um modelo de redes *bayesianas*: determina-se a estrutura e suas probabilidades condicionais associadas. Mas, em situações em que o especialista não está disponível, ou no caso de um grande domínio de dados em que fica difícil se especializar, são úteis métodos automáticos para aprendizado de estruturas e probabilidades (também denominados parâmetros) a partir de dados disponíveis [HECKERMAN, 1995].

[1] Conjunto de nós e vínculos

A tarefa mais comum é utilizar a rede *bayesiana* para inferência *byesiana*, que consiste em obter conclusões à medida que novas informações ou *evidências* são conhecidas. Por exemplo, na área médica, pode-se concluir por um diagnóstico baseando-se em sintomas que são evidências. Ou também se faz inferência para estimar estados de variáveis que não foram observadas [LUNA, 2004].

Neste trabalho, abordam-se os temas da inferência probabilística e aprendizado em redes *bayesianas*, objetivando a construção de um sistema capaz de realizar essas operações em redes de topologias diversas.

1.1 Motivação

Nos últimos anos, diversas áreas de pesquisa têm aplicado o raciocínio *bayesiano* para resolver problemas que lidam com incerteza. Uma lista exaustiva de aplicações baseadas em redes *bayesianas* reunida por [NEAPOLITAN, 1990] ilustra a diversidade de problemas que podem ser resolvidos utilizando tais técnicas. Diversos campos são beneficiados pela eficiência das redes *bayesianas,* como Medicina, Reconhecimento de fala, Jogos Computacionais, Diagnóstico de mau funcionamento de veículos automotores, Visão Computacional, Mineração de Dados, Processamento da fala natural, Biologia, Finanças, entre outras diversas áreas.

A inferência probabilística e o aprendizado *bayesiano* são tarefas muitas vezes complexas. Muitos esforços têm sido dedicados ao desenvolvimento de algoritmos eficientes para resolvê-los. Compreender tais algoritmos e técnicas é o primeiro passo para melhorá-los e aumentar o campo de aplicações de redes *bayesianas*.

1.2 Objetivos

Os objetivos deste trabalho são:

- Abordar diversos conceitos relacionados a redes *bayesianas*.
- Apresentar e implementar algoritmos para inferência *bayesiana*.
- Apresentar e implementar algoritmos de aprendizado em redes *bayesiana*.
- E por fim, juntar todo o conhecimento adquirido e construir um sistema capaz de realizar inferência e aprendizado em redes *bayesianas*, permitindo uma análise dos algoritmos implementados.

A fim de alcançar os objetivos citados, inicialmente realizou-se um amplo

levantamento bibliográfico sobre redes *bayesianas*, que permitiu estabelecer os tópicos relevantes para este trabalho. Assim, identificou-se a importância dos fundamentos da teoria da probabilidade, dos conceitos básicos de redes *bayesianas*, e dos algoritmos de inferência e aprendizado.

1.3 Estrutura do Trabalho

Quanto à organização do texto, no capítulo 2 são abordados os fundamentos de teoria da probabilidade, necessários para o estudo de redes *bayesianas*. No capítulo 3 são abordados aspectos como o raciocínio *bayesiano*, definição e construção de redes *bayesianas* e um critério de análise da estrutura gráfica do modelo. No capítulo 4 são apresentados algoritmos de inferência bayesiana. No capítulo 5 são discutidos métodos e algoritmos para o aprendizado de parâmetros e estrutura em tais redes. No capítulo 6 é apresentada a metodologia usada para o desenvolvimento dessa monografia. No capítulo 7 é apresentado o sistema *UFLABayes* e no capítulo 8 são apresentados os resultados de análises dos algoritmos implementados no sistema. E por fim, são feitas as considerações finais no capítulo 9.

2 FUNDAMENTOS DA TEORIA DA PROBABILIDADE

O objetivo deste capítulo é introduzir conceitos de teoria da probabilidade, suas definições, axiomas e distribuições de probabilidade, relevantes para o entendimento dessa monografia.

2.1 Experimento aleatório, espaço amostral e eventos

Segundo [LACERDA & BRAGA, 2004], um experimento aleatório (ξ) é aquele no qual o resultado varia de modo imprevisível, quando repetido em iguais condições.

Espaço amostral ($\hat{\mathbf{S}}$) é o conjunto de todos os resultados possíveis de um experimento aleatório [LACERDA & BRAGA, 2004].

Os exemplos abaixo ilustram o conceito de experimento aleatório e espaço amostral:

- Exemplo$_1$:

ξ_1 = Jogue um dado e observe o número na face de cima.

$\hat{\mathbf{S}}_1$ = {*1, 2, 3, 4, 5, 6* }.

- Exemplo$_2$:

ξ_2 = Lançar uma moeda quatro vezes e observar a seqüência de caras (C) e coroas (K)

$\hat{\mathbf{S}}_2$ = {*CCCC, CCCK, CCKC, CCKK, CKCC, CKCK, CKKC, CKKK, KCCC, KCCK, KCKC, KCKK, KKCC, KKCK, KKKC, KKKK*}.

- Exemplo$_3$:

ξ_3 = Um paciente chega ao médico, observe os valores dos atributos: *Coriza, Tosse, Espirro.*

$\hat{\mathbf{S}}_3$ = {(*não, não, não*), (*não, não, sim*), (*não, sim, não*), (*não, sim, sim*), (*sim, não, não*), (*sim, não, sim*), (*sim, sim, não*), (*sim, sim, sim*)};

Um experimento aleatório pode ser repetido indefinidamente sob condições inalteradas; e em cada um dos experimentos não se sabe, *a priori*, qual resultado individual

ocorrerá. Embora se possa definir o conjunto de todos os possíveis resultados, cada resultado individual parece ocorrer de forma acidental. Com a repetição em larga escala surge uma regularidade que permite construir um modelo matemático para analisar o experimento.

Por exemplo, as proporções de caras e coroas, após lançar uma moeda um grande número de vezes, são aproximadamente iguais. Isto faz com que se crie o modelo que para cara atribui probabilidade 1/2 e para coroa atribui probabilidade 1/2.

Um evento é um subconjunto dos resultados possíveis de um experimento [LACERDA & BRAGA, 2004]. O espaço amostral é em si um evento. Se $\hat{\mathbf{S}}$ é um espaço amostral e $a \subseteq \hat{\mathbf{S}}$ então *a* é um evento. Um evento ocorre se um de seus elementos ocorre. Um evento atômico é uma especificação completa dos estados observados de um experimento aleatório. Para ilustrar esse conceito, voltemos ao Exemplo$_2$:

ξ_2 = Lançar uma moeda três vezes e observar a seqüência de caras (C) e coroas (K);

São exemplos de eventos atômicos do experimento ξ_2:

- Evento$_1$: CCKC
- Evento$_2$: CKCC
- Evento$_3$: CCKK

2.2 Probabilidade

Para cada evento definido em um espaço amostral $\hat{\mathbf{S}}$, deseja-se atribuir um número não negativo chamado probabilidade. Probabilidade é uma função dos eventos definidos [LACERDA & BRAGA, 2004]. Usa-se a notação P(*A*) para designar a probabilidade de ocorrência do evento *A*.

As probabilidades podem ser expressas de diversas maneiras, inclusive decimais, frações e percentagens. Por exemplo, a chance de ocorrência de um determinado evento pode ser expressa como 20%; 2 em 10; 0,20 ou 1/5.

Em termos gerais, a probabilidade de ocorrência de um evento *A*, P(*A*), será igual ao número de ocorrência do evento *A* (n_a) dividido pelo número de ocorrência total *n*. Então, P(A) = n_a / *n*.

2.3 Axiomas da probabilidade

[RUSSEL & NORVIG, 2004] apresentam os axiomas de probabilidade considerando primeiramente os mais básicos, que servem para definir a escala de probabilidade e seus pontos extremos:

1. Todas as probabilidades estão entre 0 e 1, para qualquer proposição *a*.

$$0 \leq P(a) \leq 1.$$

2. Proposições necessariamente verdadeiras têm probabilidade 1, e proposições necessariamente falsas têm probabilidade 0.

$$P(verdadeiro) = 1 \text{ e } P(falso) = 0.$$

3. A probabilidade de uma disjunção é dada por:

$$P(a \cup b) = P(a) + P(b) - P(a \cap b).$$

2.4 Variável aleatória

Uma variável aleatória (v.a) é uma função que associa um número real $X(\xi)$ a cada aparecimento ξ no espaço amostral do experimento [LACERDA & BRAGA, 2004]. É comum representar uma variável aleatória por uma letra maiúscula (como *X, Y, Z*) e qualquer valor particular da variável aleatória por uma letra minúscula (tal como *x, y, z*).

Segundo [LACERDA & BRAGA, 2004], uma v.a pode ser considerada uma função que mapeia todos os elementos do espaço amostral (coisas) nos pontos da linha real (números) ou alguma parte dela. [RUSSEL & NORVIG, 2004] interpretam as v.a's como algo que se refere a "parte" do mundo cujo "status" é inicialmente desconhecido. Por exemplo, *DorDeCabeça* poderia se referir ao fato de uma pessoa estar sentindo uma dor de cabeça, e inicialmente não se sabe se o "status" seria *verdadeiro* ou *falso*.

Cada variável aleatória tem um domínio de valores que ela pode assumir. Por exemplo, o domínio de *DorDeCabeça* poderia ser <*verdadeiro, falso*>.

[RUSSEL & NORVIG, 2004] classificam as v.a's em três espécies, dependendo do tipo do domínio:

- Variáveis aleatórias *booleanas*: quando o domínio contem apenas dois elementos, por exemplo, o domínio de *DorDeCabeça* poderia ser <*verdadeiro, falso*>.

- Variáveis aleatórias discretas: admitem valores de um domínio enumerável. Incluem as variáveis *booleanas* como um caso especial. Por exemplo, o domínio de uma v.a *Tempo* poderia ser <*ensolarado*, *chuvoso*, *nublado*, *nevoento*>.
- Variáveis aleatórias contínuas: assumem valores a partir dos números reais. O domínio pode ser a linha real inteira ou algum subconjunto como o intervalo [0,1].

2.5 Probabilidade incondicional

A probabilidade incondicional, também chamada de probabilidade *a priori* ou marginal, pode ser representada pela notação *P(A)*, significando a probabilidade que a proposição *A* seja verdadeira na ausência de qualquer outra informação relevante [RUSSEL & NORVIG, 2004]. É importante salientar que *P(A)* somente pode ser usada quando não há outra informação relacionada. Logo, quando uma nova informação relevante *B* torna-se conhecida, é necessário raciocinar com a probabilidade condicional de *A* dado *B*, ao invés de raciocinar com a probabilidade *a priori* [RUSSEL & NORVIG, 2004].

A medida $P(A = a)$, para todas as instâncias *a* no domínio de *A*, é denominada distribuição de probabilidade de *A*. Por exemplo, o resultado do arremesso de uma moeda poderia ser representado por uma única variável aleatória *A*, cujo valor, *a*, poderia ser um dos valores *C* (*cara*) ou *K* (*coroa*).

2.6 Distribuição conjunta de probabilidade

Considerando *n* variáveis aleatórias, a probabilidade conjunta de que os valores de $X_1, X_2,..., X_n$ são, respectivamente, $x_1, x_2,..., x_n$, é representada por $P(X_1 = x_1, X_2 = x_2,..., X_n = x_n)$. Uma distribuição de probabilidade conjunta sobre as variáveis $X_1, X_2,..., X_n$ é uma expressão da forma $P(X_1, X_2,..., X_n)$, que mapeia o conjunto de variáveis em um valor real no intervalo [0,1] [RUSSEL & NORVIG, 2004].

Como exemplo, suponha o arremesso de uma moeda, em que $P(C) = 1/2$ e $P(K) = 1/2$ (onde *C* representa o estado *cara* e *K* o estado *coroa*). Arremessando esta moeda quatro vezes, seria possível ter $P(C,K,K,C) = 1/16$. A expressão $P(C,K,K,C)$ significa a probabilidade conjunta de o primeiro arremesso resultar em *cara*, de o segundo resultar em

coroa, do terceiro resultar em *coroa* e do quarto resultar em *cara*.

Uma propriedade que as funções de probabilidade conjunta tem que satisfazer é que:

$$\sum P(X_1, X_2, ..., X_n) = 1 \tag{2.1}$$

Na equação 2.1, o somatório abrange todos os possíveis valores das variáveis.

Um exemplo envolvendo apenas duas variáveis *booleanas* pode ser apresentado montando uma tabela que relaciona as probabilidades de uma pessoa ter Câncer(v.a *Cancer*) condicionada à evidência de uma pessoa fumar ou não (v.a *Fumante).*

Tabela 2.1 – Distribuição conjunta de *Câncer* e *Fumante*

	Cancer		Marginal de ***Fumante***
Fumante	*presente*	*ausente*	
sim	0,30	0,15	0,45
não	0,10	0,45	0,55
Marginal de ***Câncer***	0,40	0,60	1,00

Pode-se observar na Tabela 2.1 as probabilidades conjuntas:

P(Cancer=presente,Fumante=sim)=0,30,

P(Cancer=presente,Fumante=não)=0,10,

P(Cancer=ausente,Fumante=sim)=0,15,

P(Cancer=ausente,Fumante=não)=0,45.

A distribuição marginal da variável *Fumante* é dada pela última coluna da Tabela 2.1, enquanto que a distribuição marginal de *Cancer* é fornecida pela última linha. Portanto, em relação a *Cancer* tem-se:

P(Cancer=presente)=0,40 e *P(Cancer=ausente)=0,60*,

enquanto que em relação a *Fumante* tem-se:

P(Fumante=sim)=0,45 e *P(Fumante = não)=0,55*.

Pode-se notar que o somatório das probabilidades conjuntas, bem como o somatório das probabilidades marginais de cada variável, resulta em 1.

.

2.7 Probabilidade condicional

Para todo valor *x* (*presente* ou *ausente*) da variável aleatória *Cancer*, *P(Cancer = x|Fumante = y) = P(X|y)* significa a distribuição de probabilidade de *Cancer* condicionada a *Y = y* (*y* pode ser *sim* ou *não*), onde as demais informações conhecidas são irrelevantes para *X*. Considerando o exemplo da seção 2.6, cuja distribuição conjunta é dada pela Tabela 2.1, utiliza-se a expressão *Cacer|Fumante* para denotar que a variável *Cancer* está condicionada ao conhecimento de um valor para *Fumante*.

Quando o especialista do domínio obtém alguma evidência anteriormente desconhecida, as probabilidades *a priori* devem ser revistas, pois podem perder a sua utilidade. Ao invés delas, devem ser usadas probabilidades condicionais (também denominadas probabilidades *a posteriori*), com a notação *P(X|Y)* [RUSSEL & NORVIG, 2004]. Essa espressão é lida como "a probabilidade de *X*, dado que tudo o que sabemos é y".

Surgindo uma nova evidência *Z*, deve-se calcular *P(X|Y,Z)* ao invés de *P(X|Y)*.

Probabilidades condicionais podem ser definidas em termos de probabilidades incondicionais, como na equação fundamental [RUSSEL & NORVIG, 2004]:

$$P(X \mid Y) = \frac{P(X,Y)}{P(Y)} \tag{2.2}$$

A equação (2.2) é válida para *P(Y) > 0*. Ela também pode ser escrita como:

$$P(X,Y) = P(X \mid Y)P(Y) \tag{2.3}$$

A equação (2.3) é chamada de regra do produto. Também é válida a seguinte equação:

$$P(Y,X) = P(Y \mid X)P(X) \tag{2.4}$$

A regra do produto pode ser generalizada a fim de se obter a fórmula da regra da cadeia, que corresponde a:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i \mid X_{i-1}, ..., X_1) \tag{2.5}$$

Assumindo que $V = \{X_1, X_2, ..., X_n\}$ é um conjunto de variáveis aleatórias, então a equação (2.5) fornece a distribuição conjunta de *X*.

2.8 Tabela de probabilidade condicional

Considerando as equações apresentadas na seção 2.7 e as informações contidas na Tabela 2.1, da seção 2.6, podemos montar uma tabela de probabilidade condicional, ou TPC**,** formada pelas probabilidades condicionais da variável *Cancer*:

$$P(Cancer = presente \mid Fumante = sim) = \frac{P(Cancer = presente, Fumante = sim)}{P(Fumante = sim)}$$

$$= \frac{0,30}{0,45} = \mathbf{0,667}$$

$$P(Cancer = ausente \mid Fumante = sim) = \frac{P(Cancer = ausente, Fumante = sim)}{P(Fumante = sim)}$$

$$= \frac{0,15}{0,45} = \mathbf{0,333}$$

$$P(Cancer = presente \mid Fumante = não) = \frac{P(Cancer = presente, Fumante = não)}{P(Fumante = não)}$$

$$= \frac{0,10}{0,55} = \mathbf{0,1818}$$

$$P(Cancer = ausente \mid Fumante = não) = \frac{P(Cancer = ausente, Fumante = não)}{P(Fumante = não)}$$

$$= \frac{0,45}{0,55} = \mathbf{0,8182}$$

A Tabela 2.2 representa a TPC de *Cancer* que é condicionada à v.a *Fumante*.

Tabela 2.2 – TPC de *Cancer*

	Câncer	
Fumante	***presente***	***ausente***
sim	*0,667*	*0,333*
não	*0,1818*	*0,8182*

2.9 Independência condicional

Diz-se que duas variáveis *X* e *Y* são independentes se:

$$P(x|y) = P(x) \tag{2.6}$$

sempre que P(*y*) > 0, $\forall x \in D_x$ e $y \in D_y$, onde D_x e D_y denotam os domínios de X e Y, respectivamente **.** Se *X* e *Y* são independentes, então *Y* não é informativa para *X*. Significa que conhecer *Y* não altera a probabilidade de *X*.

Dados dois conjuntos disjuntos de variáveis, V_i e V_j, e uma variável *V*, diz-se que a variável *V* é condicionalmente independente do conjunto V_i, dado V_j, se $P(V|V_i,V_j) = P(V|V_j)$. Neste caso, a notação $I(V,V_i|V_j)$ é empregada para indicar este fato. A interpretação para a independência condicional é que se $I(V,V_i|V_j)$, então V_i não acrescenta informação relevante para *V*, quando já se dispõe de V_j.

Agora, considere um caso com duas variáveis aleatórias, *X* e *Y*, e seus respectivos domínios, D_X e D_Y. Se $P(x|y) = P(x)$ sempre que $P(y) > 0$, $\forall x \in D_X$ e $y \in D_Y$, então diz-se que *X* e *Y* são condicionalmente independentes. É possível representar esta independência através da equação $P(X,Y) = P(X)P(Y)$, que origina-se da equação (2.3). Na Tabela 2.1, *X* e *Y* são condicionalmente independentes.

Existe uma forma alternativa para determinar se variáveis são independentes ou não. Esta forma é denominada medida de informação mútua condicional, sendo dada pela seguinte equação:

$$I(X,Y) = \sum_x \sum_y P(x,y) \log\left(\frac{P(x,y)}{P(x)P(y)}\right) \tag{2.7}$$

A equação (2.7) é uma relação reflexiva; isto é, $I(X,Y) = I(Y,X)$. Se $I(X,Y) > 0$, *X* e *Y* são condicionalmente dependentes. Caso contrário, *X* e *Y* apresentam independência condicional. A explicação é que se *X* e *Y* são independentes, o argumento da função logarítmica, na equação (2.7), sempre tem valor 1, fazendo com que o somatório tenha valor 0.

[CHENG *et al*, 1997b] apresenta uma outra maneira de se utilizar a medida de informação mútua para verificar se *X* e *Y* são condicionalmente independentes quando é dado um subconjunto de variáveis *V* que não contém *X* nem *Y*. A seguinte equação deve ser empregada:

$$I(X,Y|V) = \sum_{v}\sum_{x}\sum_{y} P(x,y,v)\log(\frac{P(x,y|v)}{P(x|v)P(y|v)}) \quad (2.8)$$

As variáveis *X* e *Y* apresentam dependência condicional se *I(X,Y|V) = I(Y,X|V) > 0*; caso contrário, as duas variáveis são independentes.

3 REDES *BAYESIANAS*

Neste capítulo são abordados diversos aspectos fundamentais de redes Bayesianas. Assim, na seção 3.1 aborda-se o tema do raciocínio *bayesiano* e discute-se a utilização de probabilidade para representar incerteza; na seção 3.2 é apresentado o Teorema de Bayes, na seção 3.3 definem-se redes Bayesianas; em 3.4 aborda-se a questão da construção de tais redes, em 3.5 é apresentado um importante critério para verificação de dependência baseado na análise da estrutura da rede *bayesiana*.

3.1 Raciocínio *bayesiano*

O raciocínio bayesiano pode ser explicado com um exemplo médico extraído de [YUDKOWSKY, 2003]. Considere o problema:

"1% das mulheres com mais de 40 anos que participam de exames de rotina são portadoras de câncer de mama. 80% das mulheres com câncer terão resultados positivos de mamografias. 9,6% das mulheres sem a doença também terão resultado positivo nas mamografias. Uma mulher dessa idade se depara com um resultado positivo de mamografia, qual a probabilidade dela portar câncer de mama?"

Segundo [YUDKOWSKY, 2003], a maioria dos médicos estimaria que a probabilidade da mulher em questão, ter câncer de mama estaria entre 70% e 80%.

Montaremos o exemplo de maneira *bayesiana* para chegar ao resultado correto.

Em primeiro lugar, em uma mulher com mais de 40 anos o câncer de mama (*Cancer*) pode estar *presente* ou *ausente*. Essas alternativas, mutuamente excludentes, podem ser colocadas em uma tabela, como na Tabela 3.1. Podemos iniciar o raciocínio pela probabilidade de cada alternativa 'antes de fazer qualquer teste'. É a chamada probabilidade *a priori* – *Câncer=presente* ou *Câncer=ausente*. Como 1% das mulheres com mais de 40 anos têm câncer de mama, a probabilidade *a priori* de *Câncer* estar *presente* é de 0,01 e de estar *ausente* é de 0,99.

Tabela 3.1 – Probabilidade *a priori* de *Cancer*

	P(Câncer)	
	presente	***ausente***
Probabilidade *a priori*	*0,01*	*0,99*

Agora vamos incorporar o resultado da mamografia. Se *Câncer* está *presente*, a probabilidade condicional de *Mamografia* ser *positiva* é de 0,80 (80%), e se *Câncer* está *ausente* esta probabilidade é de 0,096 (9,6%). Podemos reunir essas informações numa tabela de probabilidade condicional (TPC) de *Mamografia*, como na Tabela 3.2.

Tabela 3.2 – TPC de *Mamografia* – P(Mamografia | Cancer)

	Mamografia	
Cancer	***positiva***	***negativa***
presente	*0,8*	*0,2*
ausente	*0,096*	*0,904*

Multiplicando a probabilidade *a priori* pela condicional – equação (2.3) – obtemos a probabilidade conjunta de *Câncer* e *Mamografia*, mostrada na Tabela 3.3:

Tabela 3.3 – Probabilidade conjunta de *Câncer* e *Mamografia* – P(Mamografia , Câncer)

	Mamografia	
Câncer	***positiva***	***negativa***
presente	*0,01×0,8 = 0,008*	*0,01×0,2 =0,002*
ausente	*0,99×0,096=0,09504*	*0,99×0,904 = 0,89496*

Para fazer com que a soma de cada linha da probabilidade conjunta se torne 1, é preciso usar uma *normalização*: multiplicando cada probabilidade pela *constante de normalização*, que é dada por 1 dividido pelo somatório de cada linha da tabela de probabilidade conjunta. Obtendo assim a chamada probabilidade *a posteriori*, mostrada na Tabela 3.4:

Tabela 3.4 – Probabilidades *a posteriori* – P(Câncer | Mamografia)

	Mamografia	
Mamografia	***presente***	***ausente***
positiva	0,008/(0.008+0.09504) = 0,07764	*0,92236*
negativa	*0,00223*	*0,99777*

Portanto, o raciocínio *bayesiano* nos levou a concluir que a probabilidade *a posteriori*, ou seja, após os testes, de uma mulher com mais de 40 anos de posse de um exame de mamografia cujo resultado é positivo ter câncer de mama, ou seja, a P(*Câncer=presente | Mamografia=positiva*) é de apenas 0,07764 (7,764%).

[PENA, 2006] relata que quando esse problema foi apresentado a vários médicos e estudantes de medicina, observou-se uma tendência a superestimar a probabilidade *a posteriori* da doença, e segundo [YUDKOWSKY, 2003], apenas 46% dos entrevistados estimaram uma probabilidade condizente com a resposta correta. Isso revela que o raciocínio *bayesiano* não é intuitivo. Parece haver uma tendência geral de ignorar o fato de que a probabilidade *a priori* de doença é pequena.

No exemplo acima, o raciocínio *bayesiano* permitiu quantificar o grau em que o resultado positivo de mamografia ajustou uma estimativa inicial da chance de uma mulher ter câncer de mama. Sob esse ponto de vista, um teste médico (ou *evidência)* funciona como um 'modificador de opinião', atualizando uma hipótese inicial (probabilidade *a priori*) para gerar outra (probabilidade *a posteriori*). Essa última engloba tanto a crença anterior (probabilidade *a priori*) quanto o resultado do teste. A probabilidade *a posteriori* torna-se automaticamente a probabilidade *a priori* para testes subseqüentes [PENA, 2006].

3.2 O Teorema de Bayes

Para chegar ao Teorema de Bayes, parte-se de princípios básicos da probabilidade. Assim a probabilidade de que se observe simultaneamente um evento *X* e um evento *Y* é dada pela equação (2.3). Por outro lado, a mesma probabilidade pode ser dada por (2.4). Combinando as equações (2.3) e (2.4), pode-se obter a seguinte igualdade: $P(X|Y)P(Y) = P(Y|X)P(X)$. Dividindo os dois lados desta igualdade por $P(X)$, obtém-se:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)} \tag{3.1}$$

A equação (3.1), inicialmente formulada pelo reverendo Thomas Bayes, é conhecida como Teorema de Bayes (ou ainda Regra de Bayes ou Lei de Bayes). Nesta equação, $P(X|Y)$ é a probabilidade da evidência *X* ser observada, dado que *Y* é verdadeiro. Ou seja, $P(X|Y)$ é a possibilidade da ocorrência de *Y* causar a evidência *X* [PENA, 2006].

Em algumas situações é necessário utilizar uma versão mais geral do Teorema de Bayes, condicionada sobre alguma evidência *E* conhecida. Esta versão mais geral é representada pela equação (3.2).

$$P(Y \mid X, E) = \frac{P(X \mid Y, E)P(Y \mid E)}{P(X \mid E)} \tag{3.2}$$

O Teorema de Bayes é uma equação de grande importância, pois permite o cálculo ou a atualização de probabilidades condicionais como uma função de probabilidades *a priori* e possibilidades.

3.3 Definição de redes *bayesianas*

Uma rede *bayesiana*, segundo [CASTILHO & GUTIERREZ, 1997], é um formalismo que mistura a teoria dos grafos e a teoria da probabilidade. Nesse sentido, uma rede *bayesiana* tem dois componentes principais:

a) uma estrutura, **S**, que define relacionamento qualitativo causal entre os nós;

b) parâmetros numéricos **Θ** , que quantificam a relação probabilística causal entre os nós da estrutura.

Segundo [RUSSEL & NORVIG, 2004] uma rede *bayesiana* é um grafo direcionado acíclico onde os nós representam variáveis aleatórias e os arcos direcionados representam relacionamentos causais diretos entre os nós que conectam. A especificação completa de [RUSSEL & NORVIG, 2004] para redes *bayesianas* é dada a seguir:

- Um conjunto de variáveis aleatórias constitui os nós da rede. As variáveis podem ser discretas ou contínuas.
- Um conjunto de vínculos orientados ou setas conecta pares de nós. Se houver uma seta do nó *X* até o nó *Y*, *X* será denominado *pai* de *Y*.
- Cada nó X_i tem uma distribuição de probabilidades condicional $P(X_i|pa(X_i))$ que quantifica o efeito dos pais sobre o nó, onde $pa(X_i)$ denotam os pais do nó X_i.
- O grafo não tem nenhum ciclo orientado (e consequentemente é um grafo acíclico orientado, ou GAO).

Cada nó *Xi* está associado a uma TPC, que quantifica os efeitos dos nós *pais* sobre o nó. As TPC´s de todos os nós representam os parâmetros numéricos **Θ.**

A direção dos arcos, em geral, representa relações de causa-efeito entre as variáveis

do domínio. Por exemplo, se houver um arco indo de um nó *X* para um nó *Y*, denota-se que *X* representa uma causa de *Y*, e assume-se que *X* é um dos *pais* de *Y*.

Em uma rede *bayesiana*, cada nó, ou variável aleatória, X_i é condicionalmente independente de qualquer subconjunto de nós que não são seus descendentes, conhecidos os nós pais de X_i (representados por $pa(X_i)$). Assim, considerando-se $X_1, X_2,..., X_n$ como os nós de uma rede *bayesiana* e tomando-se da estrutura desta rede as situações onde se tem independência condicional, a equação (3.3) permite determinar a probabilidade conjunta de todos os nós.

$$P(X_1, X_2,..., X_n) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)) \tag{3.3}$$

Considere o domínio1,extraído de [RUSSEL & NORVIG, 2004], como exemplo: "Você possui um novo alarme contra ladrões em casa. Este alarme é muito confiável na detecção de ladrões, entretanto, ele também pode disparar caso ocorra um terremoto. Você tem dois vizinhos, João e Maria, os quais prometeram telefonar-lhe no trabalho caso o alarme dispare. João sempre liga quando ouve o alarme, entretanto, algumas vezes confunde o alarme com o telefone e também liga nestes casos. Maria, por outro lado, gosta de ouvir música alta e às vezes não escuta o alarme." Este domínio pode ser representado como apresenta a Figura 3.1:

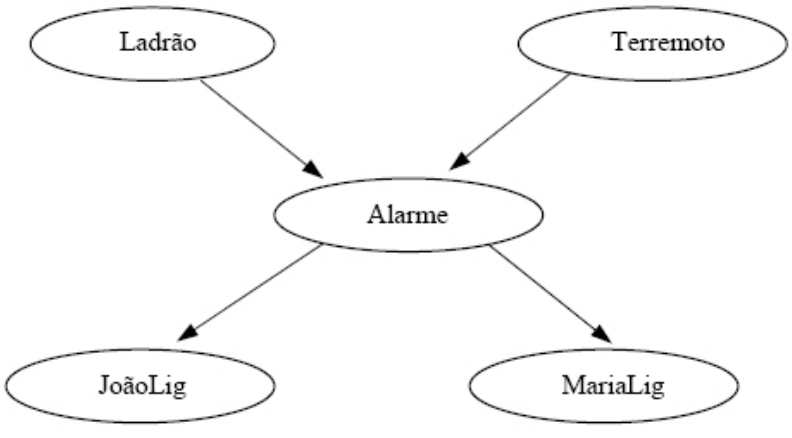


Figura 3.1 – Rede *bayesiana Alarme* para o domínio1

Uma vez definida a topologia, é necessário se definir a TPC para cada nó. Cada

linha na tabela contém a probabilidade condicional para cada caso condicional dos nós pais. Um caso condicional é uma possível combinação dos valores para os nós pais.

Por exemplo, para a variável aleatória *Alarme* temos:

Tabela 3.5 – TPC de *Alarme*

		P(*Alarme*|*Ladrão,Terremoto*)	
Ladrão	***Terremoto***	*verdadeiro*	*Falso*
Verdadeiro	*verdadeiro*	*0,950*	*0,050*
Verdadeiro	*falso*	*0,940*	*0,060*
Falso	*verdadeiro*	*0,290*	*0,710*
Falso	*falso*	*0,001*	*0,999*

Como *Terremoto* e *Ladrão* não tem pais, suas tabelas de probabilidades condicionais tem apenas as coluna representando seus estados, na qual estão as probabilidades *a priori* da variável. As Tabelas 3.6 e 3.7 apresentam as TPC´s de *Terremoto* e *Ladrão* respectivamente:

Tabela 3.6 – TPC de *Terremoto*

P(*Terremoto*)	
verdadeiro	*falso*
0,002	*0,998*

Tabela 3.7 – TPC de *Ladrão*

P(*Ladrão*)	
verdadeiro	*falso*
0,001	*0,999*

E por fim, as TPC´s de *JoãoLig* e *MariaLig*, que contêm apenas um pais, representado nas Tabelas 3.8 e 3.9:

Tabela 3.8 – TPC de *JoãoLig*

	P(*JoãoLig*|*Alarme*)	
Alarme	*verdadeiro*	*falso*
verdadeiro	*0,9*	*0,1*
falso	*0,05*	*0,95*

Tabela 3.9 – TPC de *MariaLig*

	P(*Maria,Lig*|*Alarme*)	
Alarme	*verdadeiro*	*falso*
verdadeiro	*0,7*	*0,3*
falso	*0,01*	*0,99*

3.4 Construção de redes *bayesianas*

A principal preocupação relacionada à estrutura de uma rede *bayesiana* é com a representação das dependências e independências condicionais. Como um grafo é utilizado, apenas as variáveis ligadas por arcos direcionados manifestam relações de dependência.

De acordo com [PEARL, 1986a], pode-se definir um procedimento para a construção de uma rede *bayesiana*. Assim, dada a distribuição conjunta $P(X_1,X_2,...,X_n)$ e uma determinada ordenação das variáveis, inicia-se a construção do grafo escolhendo o nó raiz X_1 e especificando sua probabilidade *a priori* $P(X_1)$. A seguir, acrescenta-se ao grafo o nó X_2. Se X_2 for dependente de X_1, deve-se inserir um arco direcionado com ponto inicial X_1 e ponto final X_2; feito isso, o arco deve ser quantificado com $P(X_2|X_1)$. Caso X_2 seja independente de X_1, deve-se atribuir ao nó X_2 a probabilidade *a priori* $P(X_2)$ e deixar os dois nós desconectados. Repetindo a operação para as demais variáveis, obtém-se a rede.

Uma rede *bayesiana* representa adequadamente um domínio se cada nó de sua estrutura é independente daqueles que o precedem na ordenação dos nós, dados os nós pais. Assim, é fundamental especificar corretamente quem são os pais de cada nó, a fim de construir a estrutura correta de uma rede.

Uma vez que se define corretamente a ordem de inserção na rede, ao inserir um nó X_i sabe-se que seus pais já foram inseridos e que são os nós em $X_1,...,X_{i-1}$ que influenciam diretamente X_i. O procedimento geral para a construção de uma rede *bayesiana*, definido por [RUSSEL & NORVIG, 2004], é apresentado na Figura 3.2.

1. Escolha o conjunto de variáveis relevantes X_i que descrevam o domínio;
2. Defina uma ordenação para as variáveis;
3. Enquanto restarem variáveis no conjunto:
    a. Selecione uma variável X_i e adicione um nó para ela à rede;
    b. Defina os pais de X_i ($pa(X_i)$) com algum conjunto mínimo de nós que já estão na rede, tal que a propriedade de independência condicional seja satisfeita;
    c. Defina a tabela de probabilidades condicionais de X_i;

Figura 3.2 – Procedimento para construção de uma rede *bayesiana*
Fonte: [RUSSEL & NORVIG, 2004]

Este procedimento permite notar que todo arco com ponto final X_i tem como ponto inicial algum nó que foi inserido na rede antes de X_i. Logo, tal método de construção garante a obtenção de uma rede acíclica.

As redes *bayesianas* também são livres de valores probabilísticos redundantes, o que exclui qualquer possibilidade do especialista do domínio definir uma rede que infringe os axiomas da Teoria da Probabilidade [RUSSEL & NORVIG, 2004].

Mesmo em um domínio localmente estruturado, construir uma rede *bayesiana* não é uma tarefa trivial. A estrutura da rede precisa representar, sem qualquer tipo de falha, todos os agrupamentos locais de nós. A ordem correta para adicionar nós à rede é colocar primeiro as causas que não são influenciadas, depois os nós que elas influenciam, e assim sucessivamente, até chegar aos nós que não exercem influência causal direta sobre os demais. Para esse problema são implementados algoritmos de aprendizado de estrutura, que “aprendem” diretamente de uma base de dados a topologia da rede.

Após obter a topologia de uma rede *bayesiana*, é necessário especificar a tabela de probabilidades condicionais para cada nó da rede. Em uma tabela deste tipo, cada linha contém as probabilidades condicionais de um estado do nó para todos os casos condicionantes. Chama-se de caso condicionante uma combinação possível de valores para os nós pais.

Mesmo que um nó possua poucos pais, sua tabela de probabilidades condicionais ainda requer uma grande quantidade de números. Geralmente, preencher completamente uma tabela deste tipo é uma tarefa demorada. Para solucionar este problema, é possível implementar o aprendizado automático dos parâmetros numéricos de uma rede *bayesiana*,

desde que se conheça a sua estrutura.

As tabelas dadas na Seção 3.4, são exemplos de tabelas de probabilidades condicionais. Nelas, pode-se notar que a soma dos valores em cada linha é 1, já que as colunas expressam um conjunto exaustivo de casos para a variável. No caso de um nó que não tem pais, sua tabela de probabilidades condicionais possui apenas uma linha e um número de colunas igual ao número de estados que a v.a pode assumir, na qual estão as probabilidades *a priori* para os estados da variável..

3.5 Verificação de dependência

Na Seção 2.9 mostrou-se como é possível verificar a independência condicional utilizando-se distribuições probabilísticas. Nesta seção é apresentado um importante critério para verificação de dependência baseado na análise da estrutura da rede *bayesiana*. Este critério é denominado *d-separação*.

A *d-separação* foi proposta por [PEARL, 1986b]. A idéia básica deste critério é: dois conjuntos de nós *X* e *Y* são *d-separados* por um conjunto de nós *C* se todo caminho não direcionado de um nó em *X* para um nó em *Y* é *d-separado* (bloqueado) por nós de *C*; neste caso, *X* e *Y* são condicionalmente independentes, dado *C*.

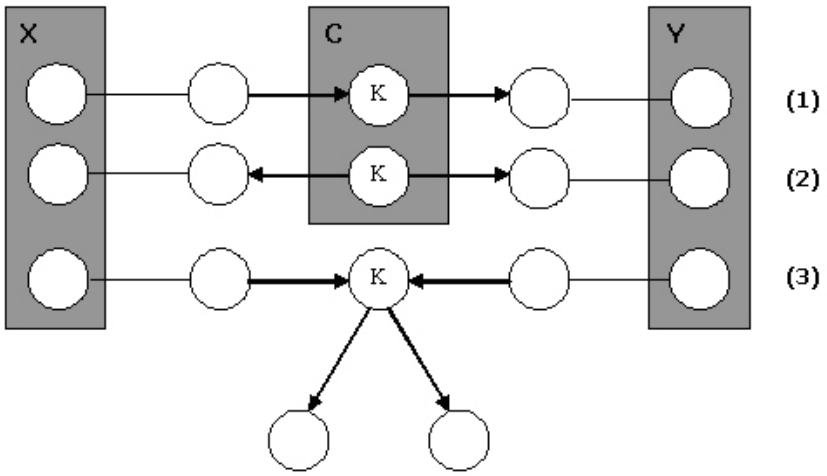


Figura 3.3 – Três maneiras de se bloquear um caminho de *X* para *Y*, dado *C*.

Assim, dado um conjunto *C*, pode-se dizer que um caminho é bloqueado se existe um nó *K* neste caminho para o qual uma das seguintes condições é verificada [SILVA & LADEIRA, 2002]:

- *K* pertence ao conjunto *C*, e *K* é ponto final para um arco orientado do caminho e ponto inicial para um outro arco orientado do caminho;
- *K* pertence ao conjunto *C*, e *K* é ponto inicial para dois arcos orientados do caminho;
- *K* e todos os seus descendentes não pertencem ao conjunto *C*, e *K* é ponto final para dois arcos orientados do caminho.

Acima, a Figura 3.3 ilustra essas três condições.

3.6 Considerações finais

Pode-se notar que as redes *bayesianas*, por representarem a incerteza através do uso de fundamentos da Teoria da Probabilidade, são muito úteis para a representação de domínios de conhecimento incerto e para a busca de respostas para questões relevantes destes domínios. Além disso, estas redes também são importantes porque possibilitam modelar o conhecimento do especialista intuitivamente e permitem que o cálculo da distribuição de probabilidade conjunta global seja evitado.

Mas, apesar de todos esses aspectos positivos associados ao formalismo rede *bayesiana*, deve-se conhecer bem o domínio antes de utilizá-lo. Isto porque se no domínio os relacionamentos entre as variáveis não são do tipo causal, então não se pode garantir que uma rede *bayesiana* é a estrutura mais apropriada para a representação das relações de dependência entre as variáveis.

4 INFERÊNCIA *BAYESIANA*

A tarefa básica de um sistema de redes bayesianas é computar a distribuição da probabilidade condicional para um conjunto de variáveis de consulta, dado os valores de um conjunto de variáveis de evidência, ou seja, computar a P(variável_consulta|variáveis_evidência).

Essa tarefa é denominada inferência *bayesiana* e permite responder a uma série de "consultas" sobre um domínio de dados. Por exemplo, na área médica, a principal tarefa consiste em obter um diagnóstico para um determinado paciente apresentando certos sintomas (evidências). Esta tarefa consiste em atualizar as probabilidades das variáveis em função das evidências. No caso do diagnóstico médico, tenta-se conhecer as probabilidades de cada uma das possíveis doenças, dados os sintomas observados no paciente. Essas são probabilidades *a posteriori*.

Segundo [CASTILHO & GUTIERREZ, 1997] há três tipos distintos de algoritmos de inferência: exatos, aproximados e simbólicos. Um algoritmo de inferência denomina-se exato se as probabilidades dos nós são calculadas sem outro erro senão o de arredondamento, inerente a limitações de cálculo dos computadores. Os algoritmos aproximados utilizam distintas técnicas de simulação para obter valores aproximados das probabilidades. Em geral, estes algoritmos são utilizados em casos em que os algoritmos exatos não são aplicáveis, ou o custo computacional é elevado. Já os algoritmos simbólicos podem operar tanto com parâmetros numéricos quanto com parâmetros simbólicos, obtendo probabilidades na forma simbólica, em função dos parâmetros.

A seguir são descritos cada um desses métodos, contudo a ênfase será dada aos métodos exatos e aproximados, que apresentam boas soluções para casos mais gerais.

4.1 Métodos exatos

Um método é denominado exato se realiza o cálculo das probabilidades *a posteriori* através de somatórios e combinações de valores, sem outro erro que não seja de arredondamento no cálculo [CASTILHO & GUTIERREZ, 1997]. Na Seção 4.1.1 introduz-se o algoritmo de Enumeração[2], e na Seção 4.1.2 um algoritmo mais eficiente, o de Eliminação de Variáveis.

[2] O algoritmo de Enumeração também é denominado de algoritmo de força bruta por alguns autores

4.1.1 Algoritmo de Enumeração

Pode-se derivar uma variedade de fatos úteis a partir dos axiomas básicos apresentados na seção 2.3. Por exemplo, a regra para negação segue-se pela substituição de b por $\neg a$ no axioma 3, resultando em:

$$P(a \vee \neg a) = P(a) + P(\neg a) - P(a \wedge \neg a) \quad \text{(pelo axioma 3 com } b = \neg a)$$
$$P(\textit{verdadeira}) = P(a) + P(\neg a) - P(\textit{falsa}) \quad \text{(por equivalência lógica)}$$
$$1 = P(a) + P(\neg a) \quad \text{(pelo axioma 2)}$$
$$P(\neg a) = 1 - P(a) \quad \text{(por álgebra)}$$

A terceira linha dessa derivação pode ser estendida do caso *booleano* até o caso discreto geral. Seja D a variável discreta que tem domínio $<d_1, ..., d_n>$. Então, é fácil demonstrar a igualdade apresentada na equação (4.1).

$$\sum_{i=1}^{n} P(D = d_i) = 1 \tag{4.1}$$

A equação (4.1) significa que qualquer distribuição de probabilidades sobre uma única variável discreta deve somar 1. Também que qualquer distribuição de probabilidade conjunta sobre qualquer conjunto de variáveis deve somar 1: isso pode ser verificado criando-se uma única *megavariável* cujo domínio é o produto cruzado das variáveis originais [RUSSEL & NORVIG, 2004].

Sabendo que qualquer proposição a é equivalente à disjunção de todos os eventos atômicos em que a é válida, denota-se tal conjunto por **e(a).** Sabendo também que os eventos atômicos são mutuamente exclusivos e, portanto a probabilidade de qualquer conjunção de eventos atômicos é 0, pelo axioma 2 apresentado na seção 2.3. Consequentemente, a partir do axioma 3, pode-se derivar o relacionamento a seguir: " *a probabilidade de uma proposição é igual à soma das probabilidades dos eventos atômicos em que ela é válida*" (equação (4.2)):

$$P(a) = \sum_{e_i \in e(a)} P(e_i) \tag{4.2}$$

A equação (4.2) fornece um método para calcular a probabilidade de qualquer proposição, dada uma distribuição conjunta total que especifique as probabilidades de todos os eventos atômicos.

Pode-se extrair das equações (4.1) e (4.2) um procedimento geral de inferência *bayesiana*. Denota-se *X* uma variável de consulta, **E** o conjunto de variáveis de evidência, **e** o conjunto de valores observados para **E,** e **Y** as variáveis restantes não-observadas (denominadas variáveis ocultas). Então, uma consulta P(*X*|**e**) pode ser avaliada como mostra a equação (4.3):

$$P(X \mid e) = \alpha(X, e) = \alpha \sum_{y} P(X, e, y), \tag{4.3}$$

onde o somatório é efetuado sobre todos os valores **y** possíveis (isto é, todas as combinações possíveis de valores das variáveis ocultas **Y**) e α é a constante de normalização, que garante que o somatório da distribuição resultante será igual a 1.

A equação (4.3), portanto, permite responder a qualquer consulta P(*X*|**e**) a partir da distribuição conjunta total da rede *bayesiana.*

Porém, não é interessante ter quer construir explicitamente uma distribuição conjunta total, pois uma consulta teria complexidade de tempo O($n2^n$) e, complexidade de espaço O(2^n), para uma rede com *n* nós *booleanos* [RUSSEL & NORVIG, 2004].

A idéia básica do algoritmo de Enumeração, apresentado na Figura 4.1, é avaliar a equação (4.3) sem ter que montar explicitamente a tabela de probabilidade conjunta total. Apenas, percorrem-se os nós da rede propagando as evidências e extraindo as probabilidades para que sejam feitos os somatórios e multiplicações necessárias.

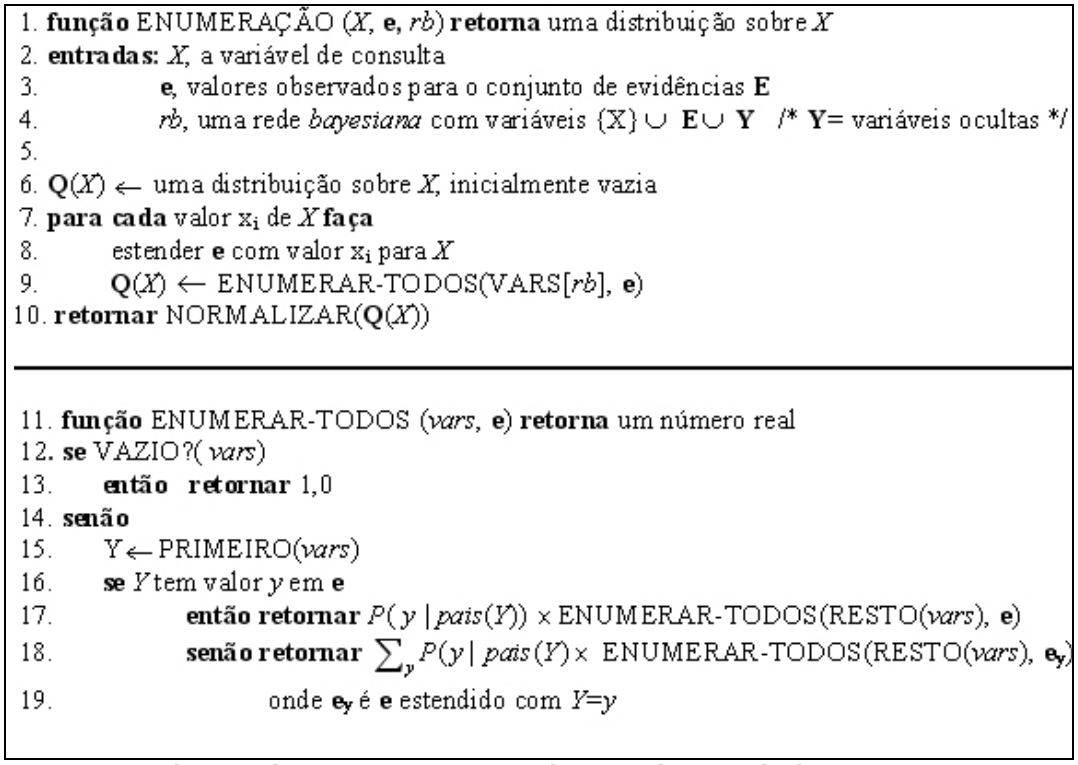


Figura 4.1 – O algoritmo de Enumeração para responder a consultas em redes *bayesianas*
Fonte : Adaptado de [RUSSEL & NORVIG, 2004] pelo autor

Desse modo, a complexidade de espaço do algoritmo de Enumeração é linear em relação ao número de variáveis, e sua complexidade de tempo para uma rede com *n* variáveis *booleanas* é $O(2^n)$.

Na linha 3, é feita uma declaração de uma estrutura que irá armazenar uma distribuição sobre uma variável *X*. **Q**(*X*) representa um vetor com tamanho igual ao número de estados que a variável *X* pode assumir, no caso de uma variável *booleana*, seria um vetor com 2 posições.

A linha 7 inicia um laço de repetição que varia também de acordo com o número de estados que a variável pode assumir. "Estender **e** com valor x_i para *X*", indicado na linha 8, significa acrescentar ao conjunto de valores **e** o valor x_i e ao conjunto de variáveis **E** acrescentar a variável *X*. A cada iteração chamada pela linha 7, o valor de **E** e **e** deve retornar ao estado inicial, contendo apenas as variáveis de evidência observadas inicialmente. A linha 9 armazena em **Q**(*X*), no índice i do estado *x* de *X* da variável em questão o resultado retornado da função ENUMERAR-TODOS. Na linha 10 é feita a normalização de **Q**(*X*), que garante que o somatório de todos os valores contidos no vetor seja igual a 1, e o resultado dessa operação é retornado como resposta.

A função ENUMERAR-TODOS é uma função recursiva que se inicia na linha 11 e recebe um conjunto de variáveis *vars* e um conjunto de valores observados para as variáveis de evidência **e**. Na linha 12 verifica-se se *vars* é um conjunto vazio que finalizará a recursão retornando o valor 1,0.

Na linha 15, é declarada uma variável **Y**, que recebe a primeira variável de *vars,* e na linha 16 é verificado se **Y** pertence ao conjunto **E,** passado implicitamente juntamente com **e** para a função. Se **Y** pertence a **E,** é retornada uma multiplicação da probabilidade P(*y* | pais(Y)), pelos outros valores achados chamando a própria função ENUMERAR-TODOS, passando agora o conjunto de variáveis *vars* menos a variável **Y** (operação indicada pela função RESTO(*vars*)), indicado na linha 17. Se **Y** não pertence a **E** é retornado o somatório das probabilidades de P(*y* | pais(Y)) para todo valor *y* de **Y,** multiplicado pelos outros valores, também chamados pela própria função ENUMERAR-TODOS, passando novamente o conjunto de variáveis *vars* menos a variável **Y.**

Um exemplo interessante para ilustrar a computação do algoritmo em questão, é considerar o domínio1, com a rede *bayesiana Alarme* e suas respectivas TPC´s apresentadas na seção 2.3. Considere a consulta: “qual a probabilidade de ter entrado um ladrão na casa, dado que *João* e *Maria* ligaram para a polícia?”, ou seja, qual o valor de **P**(*Ladrão*|*JoãoLig=verdadeiro, MariaLig=verdadeiro*) ?

As variáveis ocultas para essa consulta são *Terremoto* e *Alarme*. Da equação (4.3) (usando as letras *B, E, A, J* e *M* para indicar as variáveis *Ladrão, Terremoto, Alarme*, *JoãoLig* e *MariaLig*, respectivamente), tem-se que:

$$P(B \mid j,m) = \alpha P(B, j,m) = \alpha = \sum_{e} \sum_{a} P(B,e,a, j,m)$$

O processo de avaliação para essa equação é mostrado como uma árvore de expressões na Figura 4. O algoritmo ENUMERAÇÃO avalia tais árvores usando a recursão primeiro na profundidade.

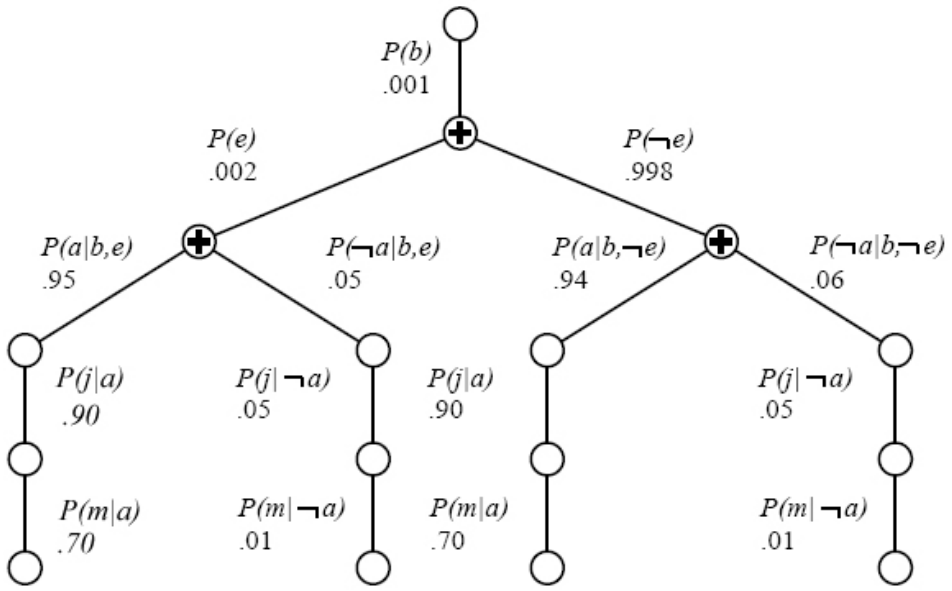


Figura 4.2 – A estrutura da expressão mostrada na equação (4.3)

Nota-se que a Figura 4.2 torna explícita as subexpressões repetidas que são avaliadas pelo algoritmo. Os produtos P(j | a)P(m | a) e P(j | ¬ a)P(m | ¬ a) são calculados duas vezes, um para cada valore de ***e.*** A próxima seção descreve um algoritmo que evita esse desperdício de computações: o Eliminação de Variáveis.

4.1.2 Algoritmo de Eliminação de Variáveis

O algoritmo de Enumeração, descrito na seção anterior, pode ser substancialmente melhorado eliminando-se cálculos repetidos, do tipo ilustrado na figura 4.2. A idéia é efetuar os cálculos apenas uma vez e guardar os resultados para uso posterior. Essa é uma forma de programação dinâmica. Existem várias versões desse algoritmo; essa seção apresenta uma versão baseada em [RUSSEL & NORVIG, 2004] e em [COZMAN, 2000].

Dada uma rede *bayesiana* sobre um conjunto de variáveis *vars*, um conjunto de evidência **E** com seus respectivos valores observados em **e,** e uma variável de consulta $X_{\mathbf{q}}$, a computação de P($X_{\mathbf{q}}$|**e)** envolve normalmente apenas um subconjunto de distribuições de probabilidade associadas à rede. Se a distribuição de probabilidades **P**(X_i|*pa*(X_i)) é necessária para o cálculo da consulta, então [DECHTER, 1999] diz que X_i é uma *variável de requisito*. O conjunto de variáveis de requisito é denotado por $X_{\mathbf{R}}$**.** Variáveis de consulta

$X_{\mathbf{q}}$ pertencem necessariamente a $X_{\mathbf{R}}$, mas nem todas as variáveis **E** pertencem a $X_{\mathbf{R}}$ (apenas as variáveis de consulta que têm pais não-evidência em $X_{\mathbf{R}}$ pertencem a $X_{\mathbf{R}}$).

Dada essa notação, pode-se transformar a equação (4.3) na equação(4.4):

$$\mathbf{P}(X|\mathbf{e}) = \mathbf{P}(X_{\boldsymbol{q}},\mathbf{e}) = \sum_{X_R \setminus \{\mathrm{Xq}, E\}} \left(\prod_{X_i \in X_R} P(X_i \mid pa(X_i)) \right), \qquad (4.4)$$

onde as distribuições de probabilidades devem ser restritas aos domínios que não contêm nenhuma evidência.

Denota-se por N o número de variáveis de requisito que não foram observadas e que não são variáveis de consulta (variáveis ocultas). Essas variáveis devem ser ordenadas de maneira especial $\{X_1, X_2, X_3, ..., X_n\}$. A qualidade da ordenação e a construção de boas ordenações serão discutidas adiante. Aplicando a Equação (4.4) a essa ordenação temos que:

$$\mathbf{P}(X_{\mathbf{q}},\mathbf{e}) = \sum_{X_n} \cdots \sum_{X_1} \mathrm{P}(X_N|pa(X_N)) \times ... \times \mathrm{P}(X_1|pa(X_1)) \qquad (4.5)$$

Pelo fato de X_1 poder aparecer somente em distribuições $\mathrm{P}(X_j \mid pais(X_j))$ para $X_j \in \{X_1, filhos(X_1)\}$, pode-se mover o somatório para X_1, resultando na equação (4.6):

$$\mathbf{P}(X_{\boldsymbol{q}},\mathbf{e}) = \sum_{X_n} \cdots \sum_{X_2} \left(\prod_{X_1 \in X_R \setminus \{X_1, filhos(X_1)\}} P(X_i \mid pa(X_i)) \right) \times \left(\sum_{X_1} \prod_{X_j \in \{X, filhos(X_1)\}_{1,}} P(X_j \mid pa(X_j)) \right) \qquad (4.6)$$

Até esse ponto, foram usadas as distribuições para $X_J \in \{X_1, filhos(X_1)\}$. Para visualizar as operações mais claramente, pode-se definir a equação (4.7), que retorna uma distribuição não normalizada:

$$\mathbf{P}(X_{\boldsymbol{q}},\mathbf{e}) = \mathbf{P}(filhos(X_1) \mid pa(X_1), tios(X_1)) = \sum_{X1} \left(\prod_{X_1, filhos(X_1)} P(X_j \mid pais(X_j)) \right). \qquad (4.7)$$

(onde $tios(X_i)$ denota os *pais* dos *filhos* de X_i que não são *filhos* de X_i).

O algoritmo de eliminação de variáveis coloca as várias distribuições num vetor de distribuições. Essas distribuições são chamadas fatores. Coletam-se todos os fatores que contêm a variável X_1, retira-os do vetor de distribuições, constrói-se uma nova distribuição

não-normalizado **P**(*filhos*(X_1)|*pais*(X_1) , *tios*(X_1)) e adiciona essa distribuição ao vetor de distribuição.

O resultado dessa operação é que X_1 foi "eliminado". Depois, sobre a variável de X_2, coletam-se todos os fatores que contêm X_2, retira-os do vetor de distribuições, multiplica as distribuições e elimina-se também X_2. O resultado dessa operação é novamente um fator, que é adicionado ao vetor de distribuições. Continua-se essa operação até se eliminar todas as variáveis possíveis.

No final, restará pelo menos um fator para a variável de consulta $X_{\mathbf{q}}$. Multiplicando esses fatores juntos e normalizando o resultado, tem-se **P**($X_{\boldsymbol{q}}$ |**e**).

A idéia é multiplicar os membros do vetor de fatores na seqüência dada pela ordenação. O algoritmo tenta eliminar as variáveis o mais rápido possível, para armazenar os produtos intermediários num tamanho razoável.

A ordenação das variáveis é arbitrária, mas diferentes ordenações fazem diferença na computação. Infelizmente, a construção de uma ordenação ótima é um problema *NP-Completo*. Porém, nesse trabalho usaremos uma ordenação predeterminada: das folhas para a raiz.

O corpo principal do algoritmo de eliminação de variáveis é mostrado na Figura 4.3, e as funções auxiliares nas Figuras 4.4, 4.5, 4.6, 4.7 e 4.8.

```
1. função ELIMINAÇÃO (X, e, rb) retorna uma distribuição sobre X
2. entradas: X, a variável de consulta
3.           e, valores observados para o conjunto de evidências E
4.           rb, uma rede bayesiana com variáveis {X} ∪ E ∪ Y   /* Y= variáveis ocultas */
5.
6. fatores ← [ ];
7. vars ← REVERTER(VARS[rb])
8. para cada var em vars faça
9.       fatores ← [CRIAR-FATOR(var, e) fatores]
10.      se var é uma variável oculta então fatores ← SOMAR(var, fatores)
11. retornar NORMALIZAR(PRODUTO-PONTUAL(X, fatores))
```

Figura 4.3 – O algoritmo de Eliminação de variáveis para responder a consultas em redes *bayesianas*
Fonte: Adaptado de [RUSSEL & NORVIG, 2004] pelo autor

```
1. função CRIAR-FATOR(var, e) retorna um fator para a variável var
2. fator ← TPC de var
3. se var tem valor x em e
4.       retirar de fator todos os valores inconsistentes com x de var
5. para cada pai em pais(var) faça
6.       se pai tem valor p em e
7.             retirar de fator todos os valores inconsistentes com p de Pai
8. retornar fator
```

Figura 4.4 – A função CRIAR-FATOR
Fonte: Elaborada pelo autor

```
1. função SOMAR(var,fatores) retorna um fator
2  fatoresRequisito ← todos os fatores que referenciam var
3. fator ← PRODUTO-PONTUAL(var,fatoresRequisito)
4. fatores ← [SUM-OUT(var, fator) fatores]
5. retornar fator
```

Figura 4.5 – A função SOMAR
Fonte: Elaborada pelo autor

```
1.  função PRODUTO-PONTUAL (var, fatores) retorna um fator
2.  se TAMANHO(fatores) = 1
3.        então retornar único fator em fatores
4.        senão
5.              fator1 ← PRIMEIRO-ELEMENTO(fatores)
6.              fator2 ← SEGUNDO-ELEMENTO(fatores)
7.              REMOVE-ELEMENTOS(fator1, fator2) de fatores
8.              para cada valor x de var faça
9.                    fatorResultado ← multiplica entradas de fator1 e fator2 onde var=x
10.             fatores ← [fatores, fatorResultado]
11.             PRODUTO-PONTUAL[var, fatores]
```

Figura 4.6 – A função PRODUTO-PONTUAL
Fonte : autor

```
1. função SUM-OUT(var,fator) retorna um fator
2. para cada estado x de var faça
3.       resultado ← somar todas as entradas de fator que contêm a entrada x de var
4. retornar resultado
```

Figura 4.7 – A função SUM-OUT
Fonte : Elaborada pelo autor

4.2 Métodos Aproximados

Os algoritmos considerados dentro do grupo de métodos aproximados utilizam distintas técnicas de simulação para obter valores aproximados das probabilidades. Estes métodos podem ser classificados em: algoritmos de simulação estocástica, métodos de simplificação de modelos e métodos baseados em busca e propagação de crença em ciclos [CASTILHO & GUTIERREZ, 1997]. Neste trabalho, abordamos algoritmos inseridos no grupo de simulação estocástica.

Simulação estocástica

Estes algoritmos também são conhecidos como algoritmos de amostragem estocástica. A idéia principal deste método aproximado é usar o modelo da rede *bayesiana* para simular o fluxo do impacto ou influência da evidência sobre o resto das variáveis [JENSEN, 2001]. Neste tipo de algoritmo, de acordo com as tabelas de probabilidade condicional da rede, gera-se um conjunto de amostras selecionadas aleatoriamente, então se realiza inferência, isto é, aproximam-se probabilidades de variáveis de "consulta" pela freqüência da suas aparições na amostra. A exatidão dos resultados vai depender do tamanho das amostras (do número de iterações que geram as amostras) e, diferentemente dos métodos exatos, a estrutura da rede não é relevante no cálculo da inferência, sendo essa uma de suas vantagens principais.

4.2.1 Algoritmo *Forward Sampling*

O elemento primitivo em qualquer algoritmo de amostragem é a geração de amostras a partir de probabilidade conhecida. Por exemplo, uma moeda imparcial pode ser considerada uma variável aleatória *Moeda* com valores <*cara*,*coroa*> e uma distribuição *a priori* **P**(*Moeda*) = <0,5,0,5>. A amostragem a partir dessa distribuição é exatamente igual ao lançamento da moda: com probabilidade 0,5 ela retornará *cara*, e com probabilidade 0,5 retornará *coroa*. Dada uma fonte de números aleatórios no intervalo [0,1], é uma questão simples a amostragem de qualquer distribuição sobre uma única variável [RUSSEL & NORVIG, 2004].

A espécie mais simples de processo de amostragem aleatória para redes *bayesianas* gera eventos a partir de uma rede que não tem nenhuma evidência associada a ela. A idéia é fazer a amostragem uma variável de cada vez, em ordem topológica. A distribuição de probabilidade a partir da qual se obtêm uma amostra do valor está condicionada aos

valores já atribuídos aos pais da variável. Esse algoritmo é apresentado na Figura 4.8.

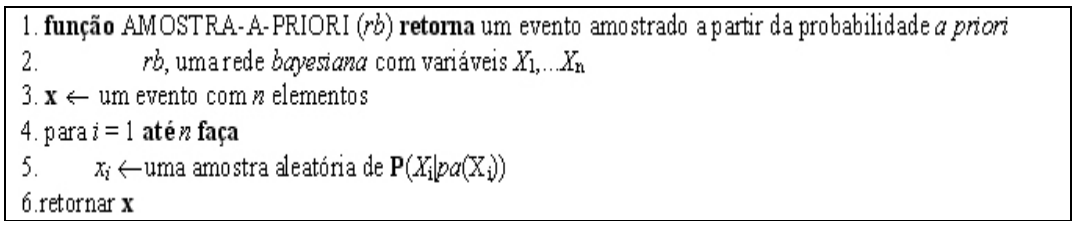


Figura 4.8 – O algoritmo AMOSTRA-A-PRIORI
Fonte : Adaptado de [RUSSEL & NORVIG, 2004] pelo autor

Pode-se construir um algoritmo para o processo de *gerar uma amostra aleatória* (chamado na linha 5 do algoritmo AMOSTRA-A-PRIORI) a partir da distribuição de **P**(*X*|*pa*(*X*)). Esse algoritmo é mostrado na Figura 4.9.

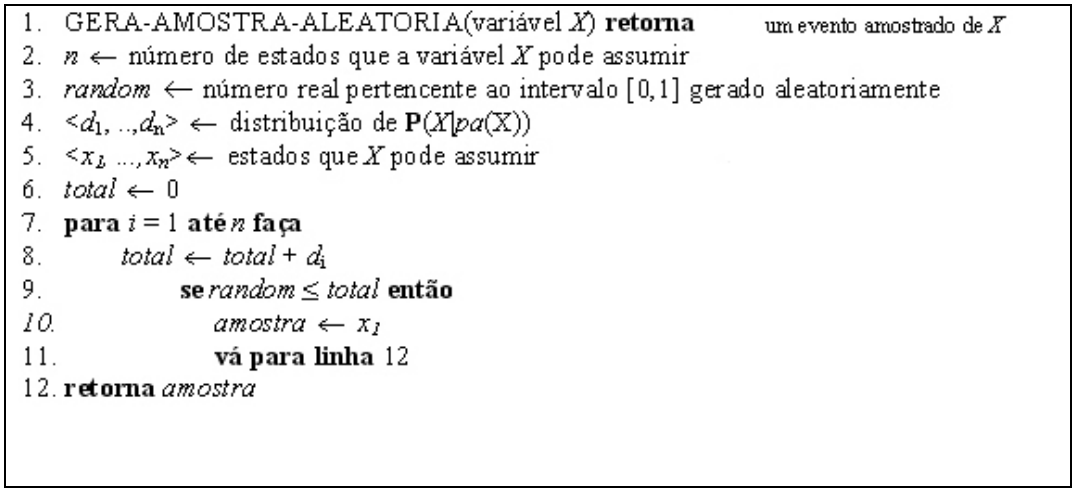


Figura 4.9 – A função GERA-AMOSTRA-ALEATORIA
Fonte : Adaptado de [RUSSEL & NORVIG, 2004] pelo autor

Pode-se observar que o algoritmo AMOSTA-A-PRIORI gera amostras a partir da distribuição conjunta *a priori* especificada pela rede. Primeiro, seja $S_{PS}(x_1, ..., x_n)$ a probabilidade de um evento específico ser gerado pelo algoritmo da Figura 4.8. A partir de observações do processo de amostragem temos que:

$$S_{PS}(x_1, ..., x_n) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)) \tag{4.8}$$

Porque cada etapa de amostragem depende apenas dos valores dos pais. Essa expressão também é a probabilidade do evento de acordo com a representação da rede *bayesiana* na distribuição conjunta, conforme a Equação (4.9):

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i \mid pa(X_i)) \quad (4.9)$$

Isto é $S_{PS}(x_1, ..., x_n) = P(x_1, ..., x_n)$. Esse fato permite responder a "consultas" utilizando amostras.

Em qualquer algoritmo de amostragem, as respostas são calculadas efetuando-se a contagem das amostras reais geradas. Supondo que existam *N* amostras ao todo, e seja $N(x_1, ..., x_n)$ a freqüência do evento específico $x_1, ..., x_n$. Espera-se que essa freqüência venha a convergir, no limite, para seu valor esperado de acordo com a probabilidade da amostragem [RUSSEL & NORVIG, 2004]:

$$\lim_{x \to \infty} \frac{N_{PS}(x_1, ..., x_n)}{N} = S_{PS}(x_1, ..., x_n) = P(x_1, ..., x_n) \quad (4.10)$$

Conseqüentemente, no limite de *N* muito grande, espera-se que a resposta convirja para o resultado exato. Sempre que se usa uma igualdade aproximada ("≈") no que se segue, quer-se indicar exatamente que a probabilidade estimada se torna exata no limite de uma amostra grande. Tal estimativa é chamada consistente. Por exemplo, pode-se produzir uma estimativa consistente da probabilidade de qualquer evento parcialmente especificado $x_1, ..., x_m$, onde m≤n, como a seguir:

$$P(x_1, ..., x_m) \approx \mathbf{N_{PS}}(x_1, ..., x_m)/N \quad (4.11)$$

O *Forward Sampling* é um algoritmo para produzir amostras a partir de uma distribuição difícil de amostrar, dada uma distribuição fácil de amostrar. O algoritmo pode ser usado para calcular probabilidade condicionais, isto é, para determinar $\mathbf{P}(X|\mathbf{e})$.

O algoritmo completo é mostrado na Figura 4.10. Primeiro ele gera amostras a partir da distribuição *a priori* especificada pela rede. Em seguida, rejeita todas as que não correspondem à evidência. Finalmente, a estimativa $\hat{P}(X=x|\mathbf{e})$ é obtida pela contagem da freqüência com que $X=x$ ocorre nas amostras restantes.

Seja $\hat{P}(X=x|\mathbf{e})$ a distribuição estimada que o algoritmo retorna. A partir da definição do algoritmo temos:

$$\hat{P}(X|\mathbf{e}) = \alpha \mathbf{N}_{PS}(X,\mathbf{e}) = \frac{N_{PS}(X,e)}{N_{PS}(e)} \quad (4.12)$$

A partir da equação (4.11), pode-se transformar a equação (4.12) em:

$$\hat{P}(X|\mathbf{e}) \approx \frac{P(X,e)}{P(e)} = P(X \mid e) \tag{4.13}$$

Ou seja, Forward Sampling produz uma estimativa consistente da probabilidade *a posteriori*.

```
1. função FORWARD-SAMPLING (X, e, rb, N) retorna uma estimativa de P(X|e)
2. entradas: X, a variável de consulta
3.           e, evidência especificada como um evento
4.           rb, uma rede bayesiana
5.           N, o número total de amostras a serem geradas
6. variáveis locais: N, um vetor de contagens sobre X, inicialmente zero
7. para j=1 até N faça
9.       x ← AMOSTRA-A-PRIORI(rb)
10.      se x é consistente com então
11.              N[x] ← N[x] +1, onde x é o valor de X em x
12.retornar NORMALIZAR(N[x])
```

Figura 4.10 – O algoritmo *Forward Sampling* para responder consultas em redes *bayesianas*
Fonte : [RUSSEL & NORVIG, 2004]

À medida que mais amostras são coletadas, a estimativa convergirá pra a resposta verdadeira. O desvio-padrão do erro em cada probabilidade será proporcional a $1/\sqrt{n}$, onde *n* é o número de amostras usadas na estimativa [RUSSEL & NORVIG, 2004].

O maior problema com esse algoritmo é que ele rejeita muitas amostras. A fração de amostras consistentes com a evidência **e** cai exponencialmente conforme o número de variáveis de evidência cresce, e assim o procedimento é simplesmente inútil para problemas complexos.

4.2.2 Algoritmo *Likelihood Weighting*

O algoritmo *Likelihood Weighting* [FUNG & CHANG, 1990] é o método de simulação estocástica mais implementado para inferência em redes *bayesianas*, em parte por causa da sua fácil implementação e rápido tempo de convergência comparado com o algoritmo *Forward Sampling*

O algoritmo fixa os valores para as variáveis de evidência **E** e efetua a amostragem apenas das amostragens restantes *X* e **Y**. Garantindo que cada evento gerado será consistente com a evidência. Porém, nem todos os eventos são iguais. Antes de efetuar as contas na distribuição para a variável de consulta, cada evento é ponderado pela probabilidade de que o evento concorde com a evidência, medida pelo produto das

probabilidades condicionais para cada variável de evidência, dados os seus pais. Intuitivamente, eventos em que a evidência real parece improvável devem receber menor peso.

Para entender por que a ponderação de probabilidades funciona, precisa-se examinar a distribuição de amostragem S_{WS} para o algoritmo. Continua-se com a notação em que o conjunto de variáveis de evidência **E** possui um conjunto de valores observados **e.** Denominam-se as outras variáveis de **Z**, isto é, **Z** = {X} $\cup$ **Y.** O algoritmo realiza a amostragem de cada variável em **Z,** dados os valores de seus pais:

$$S_{WS}(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{l} P(z_i \mid pa(Z_i)) \tag{4.14}$$

Pode-se observar que $pa(Z_i)$ pode incluir ao mesmo tempo variáveis de consulta e variáveis de evidência. Diferente da distribuição *a priori* **P**(**z**), a distribuição S_{WS} dedica alguma atenção à evidência: os valores amostrados para cada Z_i serão influenciados pela evidência entre os ancestrais de Z_i. Por outro lado S_{WS} dedica menor atenção à evidência do que a distribuição posterior verdadeira **P**(**z**|**e**).

O peso de probabilidade *w* constitui a diferença entre as distribuições de amostragem real e desejada [RUSSEL & NORVIG, 2004]. O peso para uma dada amostra **x**, composta de **z** e **e,** é o produto das probabilidades para cada variável de evidência, dados seus pais:

$$w(z,e) = \prod_{i=1}^{m} P(e_i \mid pa(E_i)) \tag{4.15}$$

Multiplicando as equações (4.14) e (4.15), pode-se observar que a probabilidade *ponderada* de uma amostragem tem uma forma particularmente conveniente:

$$S_{ws}(z,e) \times w(z,e) = \prod_{i=1}^{l} P(z_i \mid pa(Z_i)) \times \prod_{i=1}^{m} P(e_i \mid pa(E_i)) = \mathbf{P(y,e)} \tag{4.16}$$

Porque os dois produtos abrangem as variáveis de toda a rede, permitindo utilizar-se da probabilidade conjunta.

Pode-se mostrar que as estimativas de ponderação de probabilidades são consistentes. Para quaisquer valores específicos *x* de *X*, a probabilidade posterior pode ser calculada como:

$$\hat{P}(X|\mathbf{e}) = \alpha \sum_{y} \mathrm{N}_{\mathrm{WS}}(\mathrm{x,y,e}) \times \mathrm{w(x,y,e)} \quad \text{a partir do algoritmo}$$

$$\approx \alpha' \sum_{y} S_{ws}(x,y,e) \times w(x,y,e) \quad \text{para } \boldsymbol{N} \text{ grande}$$

$$\approx \alpha' \sum_{y} P(x,y,e) \quad \text{pela equação (4.16)}$$

$$= \alpha' \, \mathbf{P}(x,\mathbf{e}) = \mathrm{P}(x,\mathbf{e}).$$

Consequentemente, a ponderação de probabilidades retorna estimativas consistentes. Tendo em vista que a ponderação de probabilidade utiliza todas as amostras geradas, ela pode ser muito mais eficiente que o algoritmo apresentado na seção anterior. Entretanto, ele sofrerá uma degradação de desempenho à medida que o número de variáveis de evidência aumentar. Como muitas amostras terão pesos muito baixos, consequentemente a estimativa ponderada será dominada pela minúscula fração de amostras que concordam em uma proporção maior que uma probabilidade infinitesimal com a evidência. A Figura 4.11 apresenta o algoritmo *Likelihood Weighting*:

```
1. função LIKELIHOOD-WEIGHTING (X, e, rb, N) retorna uma estimativa de P(X|e)
2. entradas: X, a variável de consulta
3.           e, evidência especificada como um evento
4.           rb, uma rede bayesiana
5.           N, o número total de amostras a serem geradas
6. variáveis locais: W, um vetor de contagens ponderadas sobre X, inicialmente zero
7. para j=1 até N faça
9.       x,w ← AMOSTRA-PONDERADA(rb)
10.      W[x] ← W[x] +w, onde x é o valor de X em x
11.retornar NORMALIZAR(W[x])
-----------------------------------------------------------------------
12. função AMOSTRA-PONDERADA(rb, e) retorna um evento e um peso
13. x← um evento com n elementos
14. w← 1
15. para i=1 até n faça
16.     se Xi tem valor xi em e
17.             então w← w ×P(Xi=xi|pa(Xi))
18.             senão xi ← uma amostra aleatória a partir de P(Xi|pa(Xi))
19. retornar x,w
```

Figura 4.11 – O algoritmo *Likelihood Weighting* para responder consultas em redes *bayesianas*
Fonte: Adaptado de [RUSSEL & NORVIG, 2004] pelo autor

4.2.3 Algoritmo *Gibbs Sampling*

Nesta seção, será apresentado o algoritmo *Gibbs Sampling* para inferência em redes *bayesianas*. Primeiro, será definido o conceito de Cobertura de *Markov*, fundamental para o entendimento dessa seção, posteriormente será apresentado o algoritmo e depois a explicação de como e porque funciona.

Pode-se dizer que um nó é condicionalmente independente de todos os outros nós de uma rede dados seus pais, filhos e pais de seus filhos. Isto é, dada sua **Cobertura de *Markov***. Essa afirmação é equivalente à especificação de que um nó é independente de seus não-descendentes, dados seus pais. A Figura 4.12 ilustra essa cobertura:

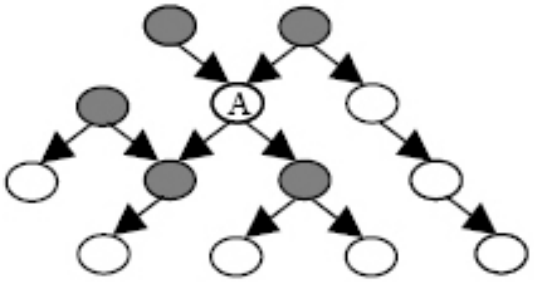


Figura 4.12 – O conjunto de nós em cinza representa a Cobertura de *Markov* de *A*

Em matemática, física ou estatística, *Gibbs Sampling* é um método usado para gerar uma seqüência de amostras da distribuição de probabilidade conjunta das variáveis aleatórias. O propósito dessa seqüência é aproximar a distribuição conjunta ou computar uma integral [CASELLA & GEORGE, 1992]. O algoritmo foi nomeado depois que o físico J.W. Gibbs, em referência a analogia entre o algoritmo de amostragem e a física estatística. O algoritmo foi inventado por [GEMAN & GEMAN, 1984] cerca de oito décadas depois da passagem de Gibbs.

Diferentemente dos outros algoritmos de amostragem, que geram cada evento a partir do nada, o *Gibbs Sampling* gera cada evento fazendo uma mudança aleatória no evento precedente. Portanto, deve-se pensar que a rede se encontra em um determinado estado atual especificando um valor para uma das variáveis não de evidência *X*, condicionadas sobre os valores atuais das variáveis na cobertura de *Markov*. Então, o algoritmo vagueia ao acaso pelo espaço de estados – o espaço de atribuições completas possíveis, invertendo uma variável de cada vez, mas mantendo fixas as variáveis de evidência.

O algoritmo começa com uma configuração das variáveis consistente com as

evidências, e então troca aleatoriamente o estado das outras variáveis condicionadas à sua cobertura de *Markov*. Depois é usada essa nova configuração gerada pra trocar os valores das outras variáveis. O algoritmo completo é mostrado na Figura 4.13.

```
1. função GIBBS-SAMPLING (X, e, rb, N) retorna uma estimativa de P(X|e)
2. entradas: X, a variável de consulta
3.           e, valores observados para o conjunto de variáveis de evidência E
4.           rb, uma rede bayesiana
5.           N, o número total de amostras a serem geradas
6. variáveis locais: N[X], um vetor de contagens sobre X, inicialmente zero
7.                   Z, as variáveis não de evidência em rb
8.                   x, o estado inicial da rede, inicialmente copiado de e
9. adicionar em x valores aleatórios para as variáveis em Z
10. para j=1 até N faça
11.      N[x] ← N[x] +1, onde x é o valor de X em x
12.      para cada Z_i em Z faça
13.           x ←GERA-AMOSTRA-COBERTURA-MARKOV(x,X)
14.retornar NORMALIZAR(N[X])
```

Figura 4.13 – O algoritmo Gibbs Sampling para inferência em redes *bayesianas*
Fonte: Adaptado de [RUSSEL & NORVIG, 2004] pelo autor.

A função GERA-AMOSTRA-COBERTURA-MARKOV faz uma amostragem do valor Z_i em x a partir de $P(Z_i(mb(z_i))$ dados os valores de $MB(Z_i)$ em **x,** e é mostrado na Figrua 4.14:

```
1.  função GERA-AMOSTRA-COBERTURA-MARKOV (estado_atual , variável X) retorna um novo_estado
2.  n ←número de estado que X pode assumir
3.  random ←número real pertencente ao intervalo [0,1] gerado aleatoriamente
4.  distribuição_Markoviana← <d_1,...,d_n>
5.  remover a instância de X de estado_atual
6.  para i=1 até n faça
7.       adicionar ao estado_atual o valor de X = x_i
8.       d_i ← P(X|pa(X) considerando valores em estado_atual
9.       para cada pai_j de X em pa(X)
10.           d_i ← d_i * P(pai_j |pa(pai_j))
11. remover a instância de X de estado_atual
12. NORMALIZAR(<d_1,...,d_n>)
13. novo_estado ← estado_atual
14. total ←0
15. para i=1 até n faça
16.    total ←total + d_i
17.    se random ≤total
18.       insere em novo_estado a instância de X=x_i
19. retorna novo_estado
```

Figura 4.14 – A função GERA-AMOSTRA-COBERTURA-MARKOV
Fonte: Elaborada pelo autor

O processo de amostragem se fundamenta em um equilíbrio dinâmico no qual a fração ao longo do prazo do tempo gasto em cada estado é exatamente proporcional à sua probabilidade posterior [RUSSEL & NORVIG, 2004]. Essa propriedade decorre da probabilidade de transição específica com que o processo passa de um estado para o outro, definida pela distribuição dada pela cobertura de *Markov* da variável cuja amostra está sendo coletada.

Seja $q(\mathbf{x} \rightarrow \mathbf{x'})$ a probabilidade de que o processo faça uma transição do estado **x** para o estado **x'.** Essa probabilidade define o que se denomina cadeia de *Markov* sobre o espaço de estados. Agora, supondo que executemos a cadeia de *Markov* para *t* etapas e seja $\pi_t(\mathbf{x})$ a probabilidade de que o sistema esteja no estado **x** no tempo *t*. De modo semelhante, seja $\pi_{t+1}(\mathbf{x'})$ a probabilidade de o sistema se encontrar no estado **x'** no tempo $t+1$. Dado $\pi_t(\mathbf{x})$ pode-se calcular $\pi_{t+1}(\mathbf{x'})$ efetuando o somatório da probabilidade de estar em um estado multiplicada pela probabilidade de fazer a transição para **x',** para todos os estados em que o sistema poderia se encontrar no tempo *t:*

$$\pi_{t+1}(\mathbf{x'}) = \sum_x \pi_t(x) q(x \rightarrow x') \tag{4.17}$$

Diz-se que a cadeia alcançou sua distribuição estacionária se $\pi_{t+1} = \pi_t$. Essa distribuição estacionária é denotada por π; e sua definição é:

$$\pi(\mathbf{x'}) = \sum_x \pi(x) q(x \rightarrow x') \quad \text{para todo } \mathbf{x} \tag{4.18}$$

Sob certas suposições-padrão sobre a distribuição de probabilidade de transição *q*, existe exatamente uma distribuição π que satisfaz a essa equação para qualquer *q* dado.

A Equação (4.18) pode ser interpretada com o significado de que o "fluxo de saída" esperado a partir de cada estado é igual ao "fluxo de entrada" de todos os estados. Um modo de satisfazer a esse relacionamento ocorre se o fluxo esperado entre qualquer par de estados é o mesmo em ambos os sentidos. Essa é a propriedade de equilíbrio detalhado [JENSEN, 2001]:

$$\pi(x) q(x \rightarrow x') = \pi(x') q(x' \rightarrow x) \text{ para todo } \mathbf{x}, \mathbf{x'} \tag{4.19}$$

Pode-se mostrar que o equilíbrio detalhado implica imutabilidade efetuando o somatório sobre **x** na Equação 4.19. Tem-se que:

$$\sum_x \pi(x) q(x \rightarrow x') = \sum_x \pi(x') q(x' \rightarrow x) = \pi(x') \tag{4.20}$$

Onde a última etapa se segue porque tem-se a garantia de que irá ocorrer uma transição a partir de **x'**.

Agora, pode ser demonstrado que a probabilidade de transição $q(x \to x')$ definida pela etapa de amostragem em *Gibbs Sampling* satisfaz à equação de equilíbrio detalhado com uma distribuição estacionária igual a **P**(**x**|**e**).

Para tal demonstração, primeiro define-se uma cadeia de *Markov* no qual que se efetuam amostras condicionais de cada variável sobre os valores atuais de todas as outras variáveis, e mostra-se que isso satisfaz ao equilíbrio detalhado. Em seguida, observa-se que, para redes *bayesianas*, isso é equivalente a fazer a amostragem condicional sobre a cobertura de *Markov* da variável.

Seja X_i a variável a ser amostrada, e seja $\overline{X}_i$ todas as variáveis ocultas diferentes de X_i. Seus valores no estado atual são x_i e $\overline{x}_i$. Se for feita a amostragem de um novo valor x'_i para X_i condicionalmente sobre todas as outras variáveis, inclusive a evidência tem-se que:

$$q(x \to x') = q((x_i, \overline{x}) \to (x'_i, \overline{x}_i)) = P(x'_i \mid \overline{x}_i, e)$$

Essa probabilidade de transição é chamada amostragem de *Gibbs*. Agora pode ser demonstrado que a amostragem de *Gibbs* está em equilíbrio detalhado com a distribuição posterior verdadeira:

$$\pi(x)q(x \to x') = P(x \mid e)P(x'_i, e) = P(x_i, \overline{x}_i \mid e)P(x'_i \mid \overline{x}_i, e)$$

$$= P(x_i, \overline{x}_i \mid e)P(\overline{x}_i \mid e)P(x'_i \mid \overline{x}_i, e) \text{ (usando a regra da cadeia no primeiro termo)}$$

$$= P(x_i, \overline{x}_i, e)P(x'_i \mid \overline{x}_i \mid e) \text{ (usando a regra da cadeia no sentido inverso)}$$

$$= \pi(x')q(x' \to x).$$

Como declarado no início dessa seção, uma variável é independente de todas as outras variáveis, dada sua cobertura de *Markov*; então:

$$P(x_i, \overline{x}_i, e) = P(x'_i \mid mb(X_i))$$

Onde $mb(X_i)$ denota os valores das variáveis na cobertura de *Markov* de X_i. A probabilidade de uma variável dada sua cobertura de *Markov* é proporcional à probabilidade da variável dados seus pais, multiplicada pela probabilidade de cada filho dados seus respectivos pais:

$$P(x'_i \mid mb(X_i)) = \alpha P(x'_i \mid pa(X_i)) \times \prod_{Y \in Filhos(X_i)} P(y_j \mid pa(Y_j)) \quad (4.21)$$

Consequentemente, para inverter cada variável X_i, o número de multiplicações necessárias é igual ao número de filhos de X_i.

[JENSEN, 2001] apresenta alguns problemas comuns encontrados no uso do método de *Gibbs Sampling* e propõe soluções para cada problema:

- **Problema 1:** Se a configuração inicial é pouco provável, as primeiras serão representativas. ***Solução***: *Burn-in* (Descartar as primeiras 5-10% das configurações)
- **Problema 2:** As configurações podem ficar restritas a certas configurações. ***Solução***: Para alcançar as configurações mais prováveis, uma variável poderia alterar seu estado um estado altamente improvável.
- **Problema 3:** Pode ser muito difícil obter uma configuração inicial (NP-Difícil).

  Solução: Usar heurísticas para determinar o estado inicial.

4.3 Métodos Simbólicos

Os métodos apresentados nas seções anteriores requerem que a função de probabilidade conjunta do modelo seja especificada numericamente, isto é, que sejam atribuídos valores numéricos fixos a todos os parâmetros [CASTILHO & GUTIERREZ, 1997]. Em algumas situações, a especificação numérica destes parâmetros não é desejada ou possível. Neste caso, os métodos numéricos devem ser substituídos por métodos simbólicos que sejam capazes de lidar com os parâmetros, sem precisar atribuir-lhes nenhum valor.

Os métodos de propagação simbólica conduzem a soluções que se expressam como funções dos parâmetros. As respostas a questões gerais podem ser dadas em forma simbólica em função dos parâmetros, e as perguntas específicas podem ser obtidas fazendo a substituição dos valores dos parâmetros na solução simbólica, sem precisar refazer a propagação. A propagação simbólica pode ser útil nos casos seguintes:

1. Quando não está disponível a especificação numérica dos parâmetros do modelo probabilístico.

2. Quando os especialistas somente são capazes de especificar intervalos dos parâmetros ao invés de valores exatos. Neste caso, os métodos de propagação simbólica podem ser utilizados para obter cotas inferiores e superiores das probabilidades para todos os valores possíveis dos parâmetros dos intervalos dados.

3. Quando é requerida uma análise de sensibilidade. Uma das questões que surgem normalmente no contexto é: Quão sensíveis são os resultados a mudanças nos parâmetros e aos valores de evidência?.

4.4 Considerações finais

Na literatura encontram-se várias propostas para a resolução deste problema de grande importância prática. As propostas atuais de implementações dos algoritmos exatos de inferência oferecem soluções em tempo razoável para modelos com poucas variáveis, porém, em redes maiores a inferência exata se torna intratável.

Técnicas de simulação estocástica, como o *Gibbs Sampling*, *Likelihood Weighting* e *Forward Sampling* podem fornecer estimativas razoáveis das probabilidades *a posteriori* em uma rede e podem lidar com redes muito maiores do que os algoritmos exatos.

5 APRENDIZADO

Originalmente o conceito das redes *bayesianas* foi desenvolvido supondo-se uma dependência de especialistas humanos para a definição do grafo, ou seja, da estrutura ou topologia da rede e para a estimação das probabilidades condicionais [NEAPOLITAN, 1990], mas elas podem ser construídas tanto a partir do conhecimento de especialistas humanos quanto a partir de bases de dados, com a utilização de algoritmos de aprendizado *bayesianos*.

A construção manual de uma rede *bayesiana* pode ser um processo bastante trabalhoso e caro para grandes aplicações, e em domínios complexos sua especificação além de consumir bastante tempo está propensa a erros.

Por esse motivo, os esforços dirigidos para o desenvolvimento de métodos que possam construir redes *bayesianas* diretamente de um banco de dados, ao invés do discernimento de especialistas humanos, vem crescendo constantemente [NEAPOLITAN, 1990].

O aprender pode ser interpretado como o processo de aquisição de uma representação interna efetiva para as restrições existentes no mundo, fatos e regras [PEARL, 1986] ou, em outras palavras, a aquisição de conceitos e de conhecimentos estruturados.

O aprendizado de redes *bayesianas* consiste em induzir, a partir de uma amostra de dados, as distribuições de probabilidades simples e condicionais e/ou identificar as relações de interdependência entre as variáveis de um domínio de dados, que se constitui na população de interesse. Esse processo de aprendizado indutivo pode ser de dois tipos: aprendizado de estrutura (seção 5.2), quando não se tem a estrutura e aprendizado de parâmetros (seção 5.1) quando não se tem as TPC´s de cada variável envolvida no problema.

5.1 Aprendizado de parâmetros

Considere um conjunto de variáveis $X = \{X_1, X_2,..., X_n\}$. Uma rede *bayesiana* para o conjunto X possui n famílias locais, sendo cada família do tipo $X_i|pa_i$. Cada variável X_i possui r_i possíveis estados, representados por $x_i^1,..., x_i^{ri}$.

A probabilidade $P(X_i = x_i^k | pa_i^j, \mathbf{S}) = \boldsymbol{\Theta}_{ijk}$ especifica a probabilidade de X_i estar no estado x_i^k, conhecidos o j-ésimo estado de seus pais e a estrutura **S** da rede. As

probabilidades associadas à variável X_i, estando seus pais no estado j, são fornecidas pelo vetor de parâmetros $\boldsymbol{\Theta}_{ij} = (\boldsymbol{\Theta}_{ij1},..., \boldsymbol{\Theta}_{ijri})$. As probabilidades da família $X_i|pa_i$ são fornecidas pelo vetor de parâmetros $\boldsymbol{\Theta}_i = (\boldsymbol{\Theta}_{i1}, \boldsymbol{\Theta}_{i2},..., \boldsymbol{\Theta}_{iqi})$, ao passo que as probabilidades de todas as famílias que se encontram na rede são fornecidas pelo vetor $\boldsymbol{\Theta} = (\boldsymbol{\Theta}_1,..., \boldsymbol{\Theta}_n)$.

A notação q_i, utilizada no parágrafo anterior, representa o número de configurações distintas que os elementos de pa_i podem formar. Este número de configurações é determinado pelo produtório das cardinalidades de domínio das variáveis componentes de pa_i. As q_i configurações de pa_i podem ser representadas por $pa_i^1,..., pa_i^{qi}$.

Para se fazer uma estimativa dos parâmetros de uma rede *b*ayesiana, deve-se computar a distribuição $P(\boldsymbol{\Theta}|D,\mathbf{S})$, onde $\boldsymbol{\Theta}$ é uma variável que tem $\boldsymbol{\theta}$ como instância, D é uma amostra aleatória de tamanho apropriado para a realização da estimativa e **S** é a estrutura da rede. Assim, caso **S** seja conhecida e uma amostra D esteja disponível, pode-se computar a probabilidade condicional $P(\boldsymbol{\Theta}|D, \mathbf{S})$ com certa facilidade.

De acordo com [DRUZDZEL & VAN DER GAAG, 2000], para realizar a computação de $P(\boldsymbol{\Theta}|D, \mathbf{S})$ devem ser feitas algumas suposições. Se uma rede *bayesiana* é discreta, então todas as suas variáveis são discretas e têm domínios finitos. Conseqüentemente, podem ser feitas as seguintes suposições:

- A amostra aleatória D é completa; ou seja, não faltam valores para as variáveis que compõem a amostra;
- Os parâmetros $\boldsymbol{\Theta}_{ij}$ são independentes; logo, a equação (5.1) mostra que é possível fatorar a distribuição conjunta deles.

$$P(\theta \mid S) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\theta_j \mid S) \tag{5.1}$$

Dada a amostra D, os parâmetros permanecem independentes; logo, a equação (5.2) mostra que é possível fatorar a distribuição conjunta condicionada.

$$P(\theta \mid D, S_S) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\theta_j \mid D, S) \tag{5.2}$$

Com isso, estando disponíveis a amostra D e a estrutura **S**, é possível atualizar o conhecimento sobre a distribuição de $\boldsymbol{\Theta}_{ij}$.

Considerando N_{ijk} a medida de freqüência, na amostra D, com que a variável X_i tem

o k-ésimo estado quando condicionada à j-ésima configuração dos seus pais; considerando $N_{ij} = N_{ij1} + ... + N_{ijri}$ e considerando, ainda, r_i como sendo a quantidade de estados da variável X_i, determina-se a distribuição probabilística $P(X_i = x_i^k | pa_i^j, D, \mathbf{S})$ utilizando-se a equação (5.2).

5.1.1 Algoritmo *AprendeParametros*

Nesta seção são apresentados algoritmos propostos por [SILVA & LADEIRA, 2002] para o aprendizado dos parâmetros numéricos de uma rede Bayesiana.

Na Figura 5.1 é apresentado o primeiro algoritmo, chamado de *AprendeParametros*, que corresponde ao procedimento principal do algoritmo de aprendizado.

Na primeira linha de *AprendeParametros* são declaradas as variáveis globais *X*, *Dx*, *Dados*, *Freq*, *Pa*, *Nijk* e *P*. Com isso, estas variáveis podem ser utilizadas por qualquer outro procedimento chamado de *AprendeParametros* sem a necessidade de declará-las.

O procedimento *LeAmostra* é invocado na linha 2. Ele é responsável pela leitura de arquivos textos que contêm amostras condensadas. Em um arquivo com amostra, a primeira linha deve conter os nomes das variáveis e o campo de freqüência, enquanto que as demais linhas devem conter as observações e as freqüências das observações. O procedimento *LeAmostra* efetua as seguintes operações: instancia *X* com os nomes das variáveis; instancia *Dx* com os domínios das variáveis de *X*; instancia a matriz *Dados* com as observações (tal que cada estado de variável é representado pelo índice desse estado no domínio da variável); instancia o vetor *Freq* com as freqüências de todas as instâncias; define *ND* como o número total de observações distintas presentes na amostra condensada; e também define *Nt* como o número total de observações na amostra (que é o somatório das freqüências armazenadas em *Freq*).

```
Procedimento AprendeParametros();
1.  declare X, Dx, Dados, Freq, Pa, Nijk, P;
2.  { LeAmostra;
3.  n ← |X| ;
4.    DefineEstrutra ;
5.    Para i=0 até n-1, faça
6.       Nijk[i] ← ContaNijk(i, Pa[i]) ;
7.    Prob ;
8.  GravaRede } ;
```

Figura 5.1 – Algoritmo A*prendeParametros* para aprendizado de parâmetros
Fonte: [SILVA & LADEIRA, 2002]

O valor de *n*, que corresponde ao número total de variáveis da rede Bayesiana, é calculado na linha 3.

O procedimento *DefineEstrutura*, responsável pela leitura da estrutura da rede, é chamado na linha 4. A estrutura pode ser definida pelo usuário (especialista ou não) ou também pode ser lida de um arquivo. A execução do procedimento *DefineEstrutura* faz com que sejam determinados os pais de cada variável da rede.

O procedimento *ContaNijk(i,Pa[i])* é invocado iterativamente nas linhas 5 e 6, com *i* variando de 0 a *n-1*. Cada valor de *i* está associado a uma família *X[i]|Pa[i]*, para a qual os contadores N_{ijk} são determinados com base nas observações em *Dados* e nas freqüências em *Freq*. O resultado de cada chamada de *ContaNijk(i,Pa[i])* é o retorno de uma matriz *Njk* bidimensional, que armazena o número total de ocorrências dos estados de *X[i]* e dos seus pais *Pa[i]*. O vetor *Nijk* armazena, na i-ésima posição, *Njk* como elemento.

O procedimento *Prob* é chamado na linha 7. Ele utiliza a equação (5.2) para calcular as probabilidades condicionais de cada família *X[i]|Pa[i]*.

Por fim, o procedimento *GravaRede*, responsável por gravar a estrutura e os parâmetros numéricos da rede em um formato que permita sua recuperação posterior, é invocado na linha 8. A seguir, na Figura 5.2, é apresentado o procedimento *LeAmostra*, que é chamado pelo procedimento principal *AprendeParametros* (ver Figura 5.1).

A declaração das variáveis *X*, *Dx*, *Dados*, *ND* e *Nt* como globais ocorre na primeira linha.

```
Procedimento LeAmostra()
1. Globais: X, Dx, Dados, ND, Nt;
2.      X ← Componentes(Leia arq) ; %  Lê a linha das variáveis
3.      X ← ExcetoUltimo(X);  n ← |X|; % elimina campo freq das variáveis
4.      Para i=0 até n-1 faça Dx[i] ← nil; % inicializa os domínios das variáveis
5.      ic ← -1;
6.      Enquanto not(eof.arq) faça
7       { caso ← componentes(Leia arq) ;
8.        ic ← ic + 1; %  Define o índice do atual caso
9.        Para i=0 até n-1 faça   % converte estados em índices de estados
10.         { Se not(membro(caso[i], Dx[i])) então
16.                 pos ← Enfilera(caso[i], Dx[i]) % inclui novo estado
12.           Senão  pos ← posição(caso[i], Dx[i]); % Acha o índice do estado
13.           D[ic,i] ← pos } % armazena o índice de estado de X[i] em Dx[i]
14.       Se caso[n] ≠nil, então Freq[ic] ← caso[n] Senão Freq[ic] ← 1;
15.       Nt ← Nt + Freq[ic] };
16.     ND ← ic +1;
```

Figura 5.2 – Método *LeAmostra*
Fonte: [SILVA & LADEIRA, 2002]

O procedimento *Componentes(Leia arq)*, chamado na linha 2, é responsável por ler a primeira linha de um arquivo de amostra condensada e retornar os componentes desta linha na forma de variáveis, incluindo o rótulo da coluna de freqüências, que é o último componente. Este rótulo, que não é uma variável da rede Bayesiana, é eliminado na linha 3. Nesta mesma linha, *n* recebe o número de variáveis contidas em *X*.

O vetor de domínios, *Dx*, é iniciado na linha 4. Para isto, cada posição deste vetor recebe *nil*. *Dx* é um vetor de listas; ou seja, cada posição *i* aponta para uma lista de estados da variável *X[i]*.

O índice de casos *ic* é iniciado com -1 na linha 5.

Na linha 6 ocorre iteração com escopo das linhas 7 a 15, enquanto existirem observações no arquivo de amostra condensada. Em cada iteração, ocorre o que é descrito para as linhas 7 a 15.

Na linha 7, é executada a leitura de uma observação e os estados das variáveis da rede são armazenados no vetor *caso*.

Na linha 8, incrementa-se de uma unidade a variável índice *ic*, para que esta fique consistente com o atual índice da observação lida.

Na linha 9, ocorre a iteração da variável *i* de *0* até *n-1*, com escopo das linhas 10 a 13. Com esta iteração, todos os estados das *n* variáveis são colocados na matriz *Dados*, representados como índices de suas posições nos seus respectivos domínios. Cada valor de *i* está associado ao estado *caso[i]* da variável *X[i]* que está na observação atual *ic*. Na linha 10, se a instância *caso[i]* ainda não pertence à lista de estados apontada por *Dx[i]*, então, na linha 11, ela é inserida em *Dx[i]* e a posição *pos*, onde ela foi inserida na lista de estados, é retornada. Se a instância *caso[i]* já está no domínio *Dx[i]*, então, na linha 12, sua posição na lista de estados é retornada. Na linha 13, *Dados[ic,i]* recebe o índice *pos* do estado *caso[i]*.

A verificação quanto à existência de contador de freqüência para a observação *ic* acontece na linha 14. Se existir, coloca-se este contador em *Freq[ic]*; caso contrário, *Freq[ic]* recebe 1.

Na linha 15, o contador *Nt* é incrementado com a freqüência da observação corrente *ic*.

Na linha 16, a iteração foi finalizada e todas as observações do arquivo de amostra já foram lidas e armazenadas em *Dados*. A variável *ND*, que dá a quantidade de linhas efetivas em *Dados*, recebe *ic + 1*.

Deve-se observar que, nos algoritmos desta seção, os vetores e as matrizes têm índices com limite inferior igual a 0.

Na Figura 5.3, é apresentado o algoritmo *ContaNijk*, que também é chamado no procedimento principal *AprendeParametros*.

```
Função ContaNijk(i, Pai)
0.  Global X, Dx; Dados, ND;
1.  qi ← calculaqi(Pai) ;
2.  ri ← |Dx[i]| ;
3.  mult ← achaMultiplicadores(Pai) ;
4.  Njk ← Crie um arranjo qi x ri; com valor inicial 0.
5.  n ← |X|  %  numero de variaveis ;
6.  Para ic = 0 até ND-1, faça
7.    { j  ← 0 ;
8       im ← 0 ;
        % calcula a j-ésima instância de Pa[i]
9.      Para ip em Pai, faça   % ip = indice de uma variável em Pai
10.         { j ← j + Dados[ic, ip] * mult[im] ; %  j-ésima instância em
Pai
11.         im ←  im + 1 } ;
12.   % acha a k-esima instancia da variavel X[i]
13.   k ← Dados[ic, i] ;
14.   fc ← Freq[ic]  ;; recupera Freq Observações iguais
15.   Njk[j, k] ← Njk[ j, k] +  fc ; } ;
16. Retorna (Njk) ;
```

Figura 5.3 – Método ContaNijik, chamado pelo algoritmo A*prendeParametros*
Fonte: [SILVA & LADEIRA, 2002]

A função *ContaNijk* é responsável por contar a ocorrência em *Dados* de cada estado *k* de *X[i]*, para cada configuração *j* das variáveis de *Pa[i]*; também é responsável por retornar esses valores como um arranjo de dimensões $q_i \, x \, r_i$.

A declaração das variáveis *X*, *Dx*, *Dados* e *ND* como globais ocorre na linha 0.

Na linha 1, *qi* recebe a quantidade de estados na variável *Pai* (que representa *Pa[i]*), determinada pela função *Calculaqi(Pai)*.

Na linha 2, *ri* recebe a quantidade de estados armazenados em *Dx[i]*.

O vetor *mult* recebe, na linha 3, o resultado da chamada de *achaMultiplicadores(Pai)*. Tais multiplicadores são empregados no cálculo da j-ésima configuração dos pais em *Pai*. O que se faz é tratar as configurações dos pais como

elementos de uma matriz de dimensão |*Pai*| (tem-se uma dimensão para cada variável pai). A matriz tem ordem $n_1 \; x \; ... x \; n_k$, tal que n_i é a cardinalidade do domínio da i-ésima variável pai. A função *achaMultiplicadores* é responsável por retornar um vetor de multiplicadores com |*Pai*| elementos. Caso *Pai* seja um conjunto vazio, o vetor possuirá somente um elemento.

A matriz *Njk* recebe, na linha 4, um arranjo *qi x ri*, onde todos os elementos estão instanciados com zero.

A variável *n* recebe, na linha 5, a quantidade de variáveis armazenadas no vetor *X*.

Na linha 6 é aberta uma iteração, cujo escopo é das linhas 7 a 15, que varia o índice *ic* de 0 a *ND-1*. Para cada valor deste índice, a linha *Dados[ic,*]* é acessada, a j-ésima configuração de *Pai* é determinada, o k-ésimo estado da variável *X[i]* é computado, e a freqüência *fc* da instância corrente é somada a *Njk[j,k]*.

As variáveis *j* e *im* recebem zero nas linhas 7 e 8, respectivamente.

A j-ésima instância de *Pai*, presente na instância corrente da amostra em *Dados[ic,*]*, é calculada nas linhas 9 a 11.

O k-ésimo estado de *X[i]*, fornecido por *Dados[ic,i]*, é computado na linha 13.

A variável *fc* recebe, na linha 14, a freqüência da instância corrente, dada por *Freq[ic]*.

A posição *Njk[j,k]* é incrementada, na linha 15, com o valor de *fc*.

Com o fim da iteração, na linha 16 ocorre o retorno do arranjo *Njk*.

5.2 Aprendizado de estrutura

Comparado com o aprendizado de parâmetros numéricos, o aprendizado de estrutura de redes *bayesianas* costuma ser um assunto bem mais complexo, que ainda não está bem resolvido.

Na seção 5.1, mostrou-se que é possível estimar uma distribuição de probabilidade conjunta **Θ** sobre as variáveis de uma rede a partir dos dados de uma amostra aleatória. Obviamente, a estrutura **S** que pode ser obtida a partir dos dados deve ser compatível com essa distribuição **Θ**.

Um grafo direcionado acíclico **S** é compatível com uma distribuição conjunta **Θ** se a distribuição **Θ** é dada pela equação (3.3), onde cada valor de X_i no grafo **S** é independente dos nós que não são seus descendentes, dados seus pais $pa(X_i)$.

Mesmo que **Θ** seja compatível com o grafo direcionado acíclico **S**, isto não quer dizer que as assertivas de independência e dependência condicionais verificadas em **Θ** sejam verificadas também em **S**, e vice-versa. Deve-se lembrar que é possível verificar tais assertivas em **S** por meio do critério *d-separação*.

É possível pensar nessas correspondências entre as assertivas de independência e dependência do grafo **S** e da distribuição conjunta **Θ** como mapeamentos. Os mapeamentos relevantes são:

- Mapa de dependência (*D-map*): para a distribuição **Θ**, o grafo **S** é um mapa de dependência de **Θ** se toda independência em **Θ** pode ser expressa em **S**;
- Mapa de independência (*I-map*): para a distribuição **Θ**, o grafo **S** é um mapa de independência de **Θ** se toda independência em **S** é verdadeira em **Θ**;
- Mapa perfeito (*P-map*): **S** é um mapa perfeito de **Θ** se **S** for simultaneamente um *D-map* e um *I-map* de **Θ**.

No aprendizado de uma estrutura **S** de rede *bayesiana* a partir de dados, o desejável é que a estrutura obtida seja compatível com a distribuição **Θ** e seja também um mapa perfeito de **Θ**.

Atualmente, existem dois enfoques relevantes para o aprendizado de estruturas de redes *bayesianas*: busca e pontuação, e análise de dependência (também denominado independência condicional).

Na abordagem busca e pontuação, a idéia fundamental é a escolha de uma métrica para pontuar a aderência de cada possível rede aos dados e a escolha de um algoritmo que selecione, dentre as possíveis redes, aquelas cujas chances de sucesso são maiores.

Já na abordagem análise de dependência, procura-se uma rede que represente da melhor maneira possível a distribuição conjunta que surge da amostra aleatória. É fundamental que esta rede represente todas as relações de independência e dependência da distribuição conjunta induzida pela amostra.

Nessa monografia serão apresentados algoritmos de ambos os enfoques: o de busca e pontuação (seção 5.2.1) e análise de dependência (seção 5.2.2). Contudo, será dada ênfase ao algoritmo *K2*, que pertence ao enfoque de busca e pontuação.

5.2.1 Métodos baseados em busca e pontuação

Neste enfoque de aprendizado de estrutura, o problema de aprendizado é tratado pelos algoritmos como um problema onde se deve buscar a estrutura que se encaixe melhor aos dados.

Um algoritmo deste enfoque inicia com uma estrutura de grafo sem arcos e faz uso de algum método de busca para incluir um arco no grafo. Em seguida, usa um método de pontuação para definir se a nova estrutura é melhor do que a antiga. Caso a nova estrutura seja melhor, o novo arco inserido é mantido, e o algoritmo tenta inserir outro. Este processo se repete até que nenhuma estrutura nova seja melhor do que a estrutura atual. Uma grande parte dos algoritmos do enfoque busca e pontuação requer ordenação das variáveis (para reduzir o espaço de busca) e faz uso de métodos de busca heurística (pelo fato do aprendizado usando métodos de busca e pontuação ser *NP-Difícil*) [CHENG *et al,* 1997b].

Na etapa de avaliação da estrutura, é possível aplicar diferentes métodos de pontuação. Merecem destaque os seguintes:

- Pontuação *bayesiana* [COOPER & HERSKOVITS, 1992]: a idéia básica é obter uma medida de qualidade de uma certa estrutura de rede calculando a probabilidade relativa de tal estrutura dada uma base de dados de casos. Os métodos de pontuação *bayesiana* começam pela adoção de uma distribuição de probabilidade *a priori P* sobre todo o espaço de estruturas. Fornecido um banco de dados de casos, *P* é atualizada, gerando uma distribuição *a posteriori* sobre o espaço de estruturas. Os algoritmos que empregam o método de pontuação *bayesiana* exploram tal espaço, retornando a estrutura que maximiza a distribuição *a posteriori*;
- Métodos com base na entropia [HERSKOVITS & COOPER, 1990]: pode-se considerar entropia uma medida não negativa de informação em uma distribuição. Assim, quanto maior a entropia, menos informativa é a distribuição. Nestes métodos o aprendizado baseia-se na quantificação de uma rede *bayesiana* selecionada, a qual representa uma distribuição com baixa entropia: seleciona-se a rede através da escolha de uma estrutura e da estimativa de suas probabilidades condicionais a partir de um banco de dados. A distância entre duas distribuições *P* e *P'* é dada por uma medida

relativa denominada entropia cruzada. Assim, quanto maior o valor da entropia cruzada, maior a diferença entre *P* e *P'*.

5.2.1.1 Algoritmo *K2*

Para representar o aprendizado de estrutura *bayesiana* foi escolhido o algoritmo *K2* [COOPER & HERSKOVITS, 1992] para ser implementado, que é o mais representativo entre os baseados em busca e pontuação e se tornou bastante popular devido aos resultados obtidos quando aplicado ao conjunto de dados da rede ALARM, um "*benchmark*" amplamente aceito para os algoritmos de aprendizagem *bayesiana* [KOEHLER & NASSAR , 2002].

O algoritmo *K2* avalia as possíveis topologias de uma rede bayesiana calculando a probabilidade dessa topologia gerar a base de dados em questão. O algoritmo começa assumindo que um nó não tem antecessores e incrementa o número de antecessores adicionando o antecessor que resulta no maior aumento de probabilidade da estrutura gerar a base de dados. Quando a adição de mais um antecessor ao nó não aumenta mais a probabilidade, o nó para de receber antecessores e o algoritmo faz o mesmo para o nó seguinte.

Este algoritmo requer uma ordenação das variáveis e faz uso de um método de pontuação *bayesiana* para alcançar seu objetivo, que é encontrar a estrutura de rede *bayesiana* **S** mais provável, dado um conjunto de dados *D*. Logo, o que este algoritmo busca é o maior valor possível para a probabilidade $P(\mathbf{S}|D)$.

Admitindo que **R** é uma rede Bayesiana que tem estrutura gráfica **S** e probabilidades condicionais (associadas à estrutura) **Θ**, [COOPER & HERSKOVITS, 1992] determinaram uma fórmula para calcular a probabilidade $P(\mathbf{S}|D)$ supondo que nenhum conjunto de probabilidades condicionais **Θ** é preferido para uma estrutura **S** antes de se analisar a base de dados.

A idéia do algoritmo é representar um problema com *n* variáveis através da existência de $2^{n(n-1)/2}$ possíveis estruturas, podendo escolher aquela estrutura que melhor representa o problema por meio da equação (5.3) [HRUSCHKA, 1997]:

$$P(S \mid D) = c \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{5.3}$$

Em (5.3), n é o número de variáveis, r_i é a quantidade total de possíveis valores que a variável X_i *(i = 1,..., n)* pode assumir, D é uma base de dados com m observações (casos). O conjunto de pais da i-ésima variável X_i é representado por π_i. A j-ésima configuração dos pais de X_i é representada por w_{ij}, e o número total de possíveis configurações dos pais π_i é representado por q_i. O valor de N_{ijk} representa a quantidade total de observações em D onde a variável X_i está no k-ésimo estado e os seus pais apresentam a j-ésima configuração. A constante c é chamada de constante de probabilidade *a priori* $P(\mathbf{S})$, para cada $\mathbf{S}$. Já N_{ij} é o número total de observações em D onde se tem X_i com qualquer um de seus possíveis valores e π_i com a j-ésima configuração; isto é:

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk} \tag{5.4}$$

A melhor estrutura é aquela que maximiza o valor da equação (5.5).

Segundo [HRUSCHKA, 1997], o algoritmo para construção de estruturas *K2* deve ser iniciado admitindo que um nó não possui pais, para então serem adicionados à rede os pais que maximizam a probabilidade da estrutura como um todo. A partir do momento em que a adição de qualquer nó não aumenta mais a probabilidade da rede, deve-se parar de inserir pais para o nó atual. Para se saber quando tal probabilidade não está mais sofrendo aumento, deve-se utilizar a seguinte função:

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{5.5}$$

Na Figura 5.4, é apresentado o *K2*. Neste algoritmo, a função *Pred*(x_i) retorna os nós que precedem o nó X_i na ordenação de variáveis fornecida.

```
Entrada: Um conjunto de n nós, uma ordenação destes nós, um limite máximo u
         para o número de pais que uma variável pode ter e um banco de dados contendo m cas

Saída: Para cada nó, são dados os seus pais.

Início
   Para i de 1 até n faça
   Início
      π_i := ϕ;
      P_old := g(i, π_i);
      Oktoproceed := verdadeiro;
      Enquanto Oktoproceed E |π_i| < u faça
      Início
         Seja z o nó em (Pred(x_i) - π_i) que maximiza g(i, π_i ∪ (z));
         P_new := g(i, π_i ∪ (z));
         Se P_new > P_old Então
            P_old := P_new;
            π_i := π_i ∪ (z)
         Senão
            Oktoproceed := falso;
       Fim do Enquanto;
    Escreva('Nó: ', x_i, ' Pais deste nó: ', π_i);
  Fim do Para;
Fim do Procedimento;
```

Figura 5.4 – O algoritmo *K2* para aprendizado de estrutura
Fonte: Adaptado de [COOPER & HERSKOVITS, 1992] pelo autor

Para a execução deste método deve-se assumir que:

- as variáveis do banco de dados são discretas;
- os casos no banco de dados ocorrem independentemente uns dos outros;
- todas as possíveis instanciações das variáveis ocorrem no banco de dados (amostra completa);
- todas as variáveis são inicialmente ordenadas de forma que, se a variável x_i precede x_j na ordenação, então quando da construção da rede, não haverá nenhum arco que vá de xj até xi, ou seja, xj não pode ser pai de xi (xj não pode ser causa de xi).

A maior desvantagem deste algoritmo é que a qualidade da rede gerada é influenciada pela ordenação das variáveis.

5.2.2 Métodos baseados em análise de dependência

Considere uma rede *bayesiana R* qualquer. A independência condicional na distribuição de probabilidade representada por *R* é codificada na estrutura **S** desta rede, podendo ser encontrada a partir de tal estrutura com a utilização do critério *d-separação* (ver seção 3.5).

Para estudar os algoritmos de aprendizado baseados em análise de dependência, assume-se que as dependências e independências no domínio são representadas perfeitamente pela estrutura gráfica da rede. Assim, uma afirmação de independência é dada por uma estrutura se e somente se ela é uma independência válida para o domínio considerado. A idéia geral dos algoritmos desta classe é a seguinte:

- Iniciar com um grafo não direcionado sobre o conjunto de variáveis *V*;
- Remover o arco entre dois nós *a* e *b* para os quais um conjunto de variáveis $S \subseteq V\text{-}\{a,b\}$ pode ser encontrado, tal que *a* e *b* são condicionalmente independentes, dado *S*;
- Selecionar arcos e nós, e associar uma direção aos arcos para formar um *v-node* na estrutura;
- Associar direções aos arcos restantes, tal que um grafo direcionado acíclico seja formado.

É importante observar que as principais diferenças entre os algoritmos do enfoque análise de dependência são: a maneira como os conjuntos de variáveis *S* são determinados e as regras para associar direções aos arcos.

5.2.1.2 Algoritmo *PC*

O *PC* é um algoritmo de aprendizado de redes *bayesianas* que não requisita uma ordenação de variáveis. Através de testes de independência condicional, ele é capaz de orientar arcos automaticamente.

Este algoritmo baseia-se na teoria probabilística da causalidade. Tal teoria afirma que qualquer processo causal que não envolve *feedback* pode ser representado perfeitamente por uma estrutura de rede onde as direções dos arcos são interpretadas como influências causais [SPIRTES *et al*, 1990].

O problema com o algoritmo *PC* é a necessidade de um número exponencial de

testes de independência condicional.

5.2.1.3 Algoritmo *CBL*

O termo *CBL* não é utilizado para nomear um algoritmo, mas sim para nomear uma abordagem composta por dois algoritmos, que são comumente referenciados como *CBL-A* e *CBL-B*.

Ambos os algoritmos foram desenvolvidos por **C**heng, **B**ell e **L**iu: em 1997 foi proposto o *CBL-A*; e no ano seguinte, foi apresentado o *CBL-B*.

O algoritmo *CBL-A* utiliza informações fornecidas pelo usuário a fim de determinar as orientações dos arcos. Ele assume que não existem variáveis com valores faltantes na base de dados. Por outro lado, *CBL-B* aceita a existência de variáveis com valores ausentes.

CBL-A

Segundo [CHENG et al, 1997a, 1997b], no algoritmo *CBL-A* o aprendizado da rede é efetuado com base na ordenação de todas as variáveis (que devem ser discretas), na distribuição probabilística que deriva da amostra, e na informação *a priori*, mútua ou condicional disponível.

Quando se sabe da existência de um relacionamento entre duas variáveis, também se sabe a orientação do arco entre elas. Isto é possível porque se dispõe da ordenação total das variáveis, que define uma seqüência das variáveis $X = \{X_1,..., X_n\}$ que obedece à seguinte regra: se uma variável X_i tiver pais, eles têm que ser escolhidos como um subconjunto dos predecessores de X_i; isto é, na ordenação total, cada variável necessariamente aparece depois de todos os seus pais.

O algoritmo *CBL-A* determina quais as variáveis do domínio que estão conectadas. No entanto, este algoritmo não tem como atribuição determinar as orientações dos arcos. Isto porque esta última tarefa torna-se bastante simples, como foi explicado acima, quando a ordenação completa das variáveis é conhecida.

CBL-B

O algoritmo *CBL-B* é uma versão um pouco modificada do *CBL-A*. Para ser exato, a diferença entre os dois algoritmos consiste no fato do *CBL-B* não utilizar uma ordenação das variáveis do problema, enquanto o *CBL-A* necessita de uma ordenação completa.

No processo de aprendizado de uma rede *bayesiana R*, o algoritmo *CBL-B* baseia-se

na distribuição de probabilidade **Θ** extraída da amostra e utiliza, para calcular a dependência entre variáveis, a informação *a priori*, condicional e mútua.

5.3 Considerações finais

Com relação aos aprendizados de estrutura gráfica e de parâmetros numéricos em redes *bayesianas*, deve-se destacar que são operações de grande importância, já que a construção destas redes não é uma atividade trivial.

O aprendizado da estrutura de uma rede *bayesiana* geralmente é uma atividade bem mais complexa que o aprendizado dos parâmetros, que em geral, é um assunto já bem resolvido. Os algoritmos de aprendizado de estrutura abordados neste trabalho recebem grande destaque na literatura, sendo o algoritmo *K2* o mais importante dentre aqueles que utilizam o enfoque busca e pontuação, e os algoritmos *CBL-A* e *CBL-B* são os mais importantes dentro do enfoque de análise de dependência.

As propostas atuais de implementação desses algoritmos oferecem soluções razoáveis, descobrindo distribuições de probabilidades e a estrutura gráfica da rede para futuras inferências e análises de relações causais entre variáveis, mas ainda sujeitas a melhorias. Além disso, atualmente não existem soluções que possam ser consideradas padrões. Portanto, há necessidade de investimentos em pesquisas neste campo.

6 METODOLOGIA

Neste capítulo, será esclarecido o tipo de pesquisa utilizada para a fundamentação dos conceitos ilustrados nos capítulos anteriores e a descrição de como o estudo foi realizado.

6.1 Tipo de pesquisa

De acordo com [JUNG, 2004] pode-se definir pesquisa básica como aquela que objetiva a obtenção de conhecimentos elementares, como por exemplo: novas propriedades de materiais e fenômenos associados a estes, novas fontes de energia, descoberta de elementos físico-químicos, reações químicas, efeitos eletromagnéticos, etc. O conhecimento resultante deste tipo de pesquisa pode em um primeiro momento apresentar-se desagregado do contexto cotidiano, mas, posteriormente, tornar-se-á vital para a aplicação em pesquisas tecnológicas.

Ainda conforme [JUNG, 2004], diz-se que a pesquisa exploratória visa o aprimoramento de idéias ou a descoberta de intuições, ou seja, fornecer ao pesquisador um maior conhecimento sobre o tema ou problema de pesquisa em questão.

A partir destas definições pode-se classificar este trabalho como sendo de pesquisa básica e exploratória, levando em conta a sua natureza e os seus procedimentos, pois com a construção de um sistema para aprendizado e inferência em redes *bayesianas,* objetiva-se gerar conhecimentos básicos ou fundamentais para o entendimento ou descoberta de novos fenômenos e propor alternativas ou teorias que poderão modificar as existentes.

6.2 Procedimentos metodológicos

A pesquisa foi realizada no período de maio de 2006 a fevereiro de 2007.

Inicialmente, foi realizada uma revisão bibliográfica sobre princípios das redes *bayesianas*, estrutura e formatos utilizados em sistemas de redes *bayesianas*, os algoritmos de inferência, e técnicas computacionais para o desenvolvimento de um sistema que resolva os algoritmos estudados. Foram consultados livros, monografias, teses e dissertações disponibilizadas na *Internet* e na literatura de modo geral. Posteriormente, foi implementado um sistema capaz de realizar tal operação de maneira gráfica, intuitiva e sistêmica.

Depois, foram estudados técnicas, algoritmos de aprendizado, da mesma forma que

foram estudados os algoritmos de inferências. Foi feita uma escolha dos algoritmos de aprendizado a serem implementados e os algoritmos *AprendeParâmetros* e *K2* foram implementados no sistema *UFLABayes*, tanto por suas importâncias práticas quanto pela facilidade de implementação.

Ao final do trabalho, foi feita uma análise dos resultados dos algoritmos implementados com o intuito de mostrar eficiência de cada um e demonstrar o funcionamento do sistema construído.

7 O SISTEMA *UFLABAYES*

Durante a implementação dos algoritmos de inferência e aprendizado, também foi construído um sistema, livremente distribuído, que tem como objetivo a execução dos algoritmos implementados, de forma prática e sistêmica. O sistema, denominado *UFLABayes*, foi implementado utilizando a linguagem de programação Java.

O *UFLABayes* foi construído dividindo as funcionalidades do sistema entre diversas camadas (*Packages* Java), de modo que se pode modificar uma camada sem afetar as outras ou substituir uma camada completamente, garantindo certo grau de portabilidade, extensibilidade e manutenibilidade ao software (Figura 7.1).

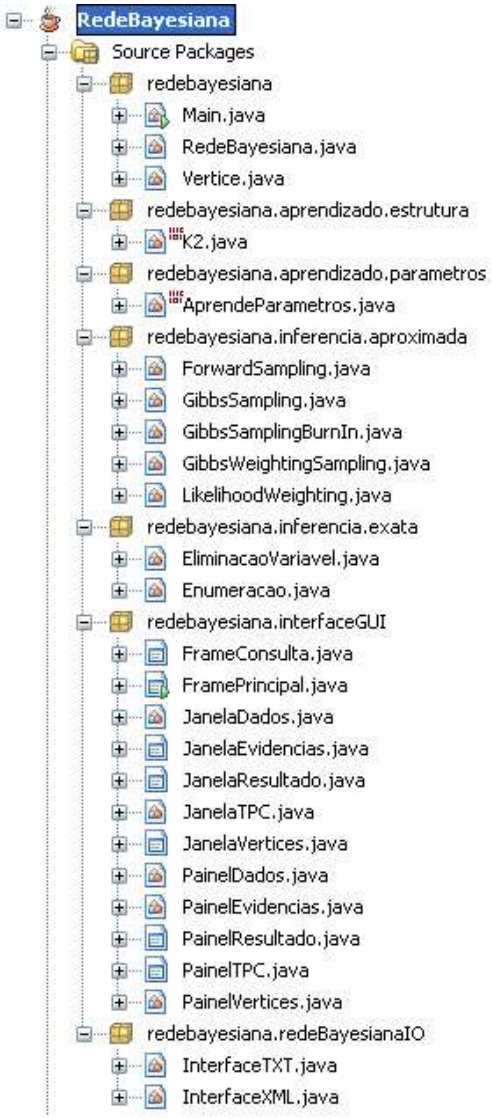


Figura 7.1 – Divisão de classes entre as camadas do sistema

O sistema *UFLABayes* (Figura 7.2), constitui-se de uma ferramenta computacional que permite o aprendizado da estrutura de uma rede *bayesiana* e seus parâmetros; e a inferência probabilística através dos diferentes algoritmos implementados, permitindo uma análise e comparação de cada método. Apesar de ser bastante genérico, o sistema foi projetado de modo que possa atender problemas reais e ser utilizado como um sistema especialista. Buscou-se, portanto, facilidade na entrada dos dados, uma interface visual intuitiva, e uma linguagem menos técnica. Assim, construiu-se uma ferramenta que auxilia na indicação de diagnósticos em domínios diversos, através do conhecimento adquirido junto a especialistas, ou por métodos automáticos implementados no próprio sistema.

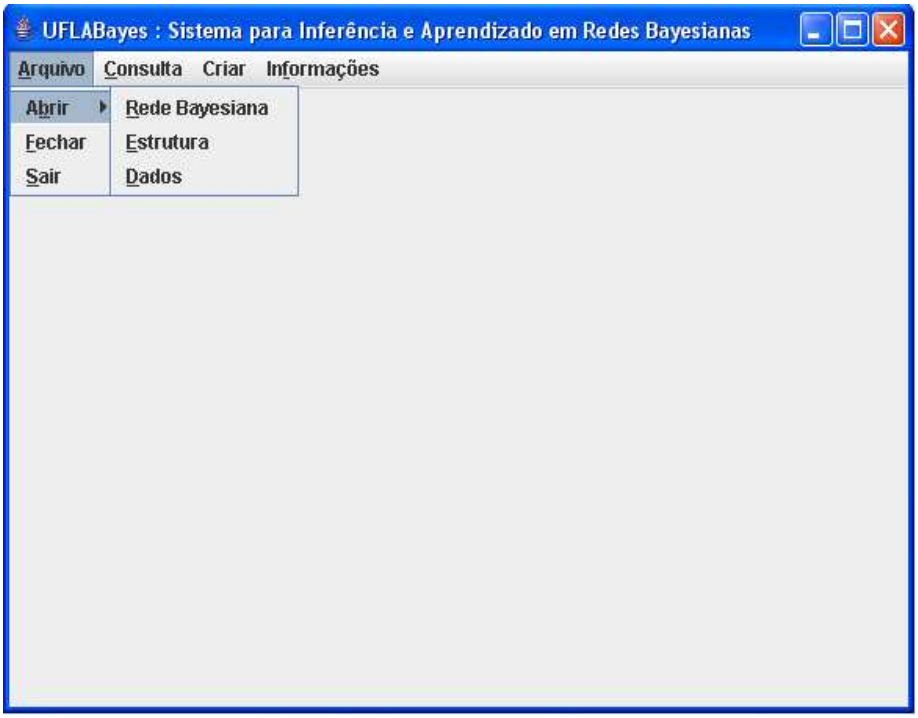


Figura 7.2 – Janela Principal do *UFLABayes*

O *UFLABayes* foi construído para interpretar dados de redes *bayesianas* armazenados em arquivos XML, que representa um formato compacto e eficiente para armazenar informações de grafos. E para a construção automática de redes *bayesianas,* escolheu-se o formato TXT para armazenar os dados, por apresentar uma portabilidade inerente ao resto do sistema. *Templates* dos arquivos de entradas de redes *bayesianas* no formato XML e de dados no formato TXT são apresentados nos Anexos A e B respectivamente.

7.1 Módulo de inferência

Para se realizar a inferência *bayesiana* utilizando o *UFLABayes*, é necessário que três elementos estejam disponíveis: 1) a rede bayesiana na qual será efetuado a inferência; 2) as evidências observadas para a consulta; e 3) qual o algoritmo irá realizar o cálculo da inferência probabilística (Figura 7.3).

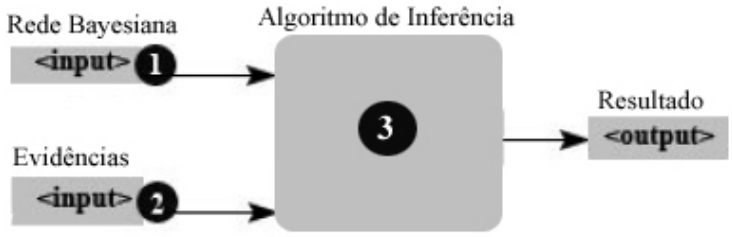


Figura 7.3 – Diagrama da operação de inferência no *UFLABayes*

O sistema procura apresentar uma interface intuitiva para facilitar a inferência. Há uma janela específica (Figura 7.4) para a apresentação da estrutura **S** da rede *bayesiana*; uma janela mostrando as tabelas de probabilidades condicionais de cada variável (Figura 7.5), representando os parâmetros numéricos **Θ;** e uma em que se pode selecionar quais variáveis foram observadas como evidência e quais os valores dessas observações (Figura 7.6).

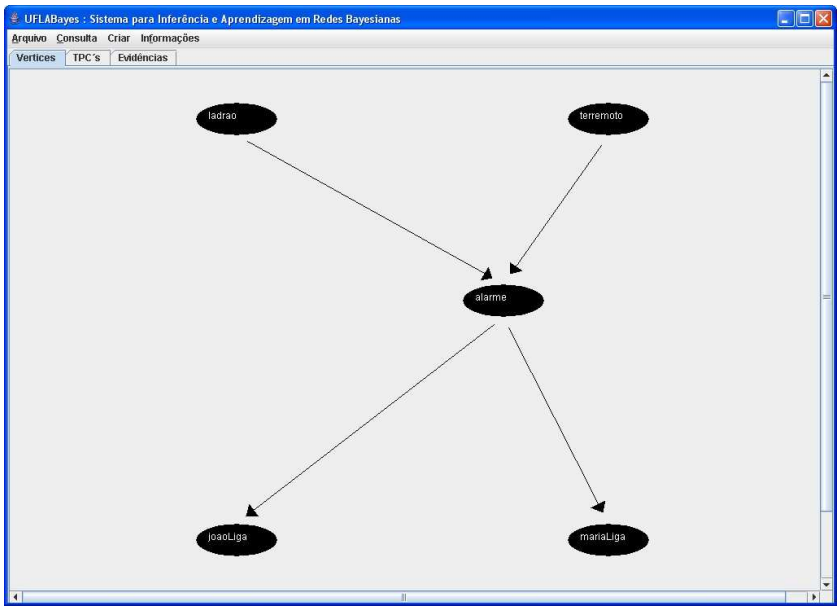


Figura 7.4 – Janela para visualização gráfica das variáveis de uma rede no *UFLABayes*

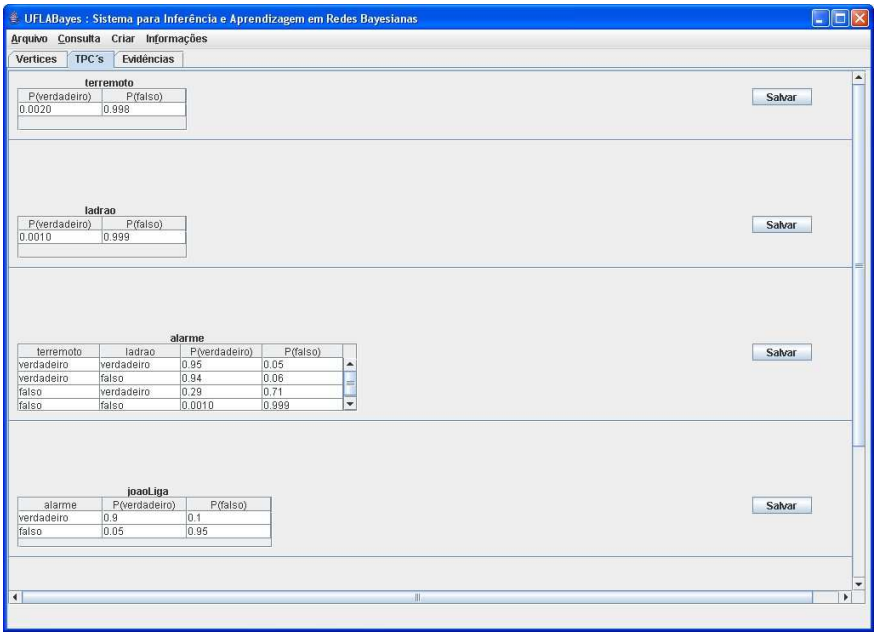


Figura 7.5 – Janela para visualização das TPC´s das variáveis no *UFLABayes*

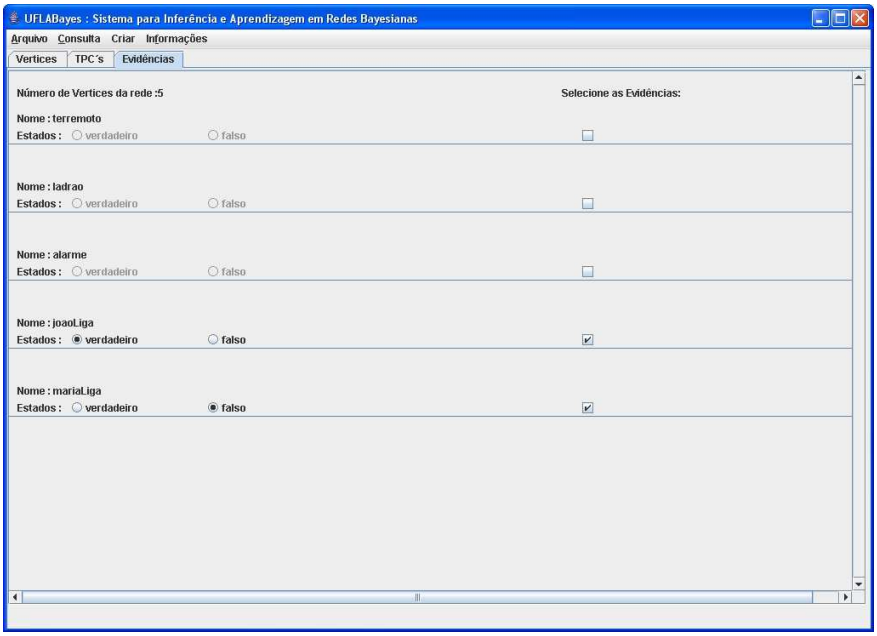


Figura 7.6 – Janela para seleção das variáveis de evidência no *UFLABayes*

Após a validação visual da rede, das TPC´s e das evidências, deve-se então, selecionar qual algoritmo de inferência será utilizado e informar, quando necessário, os parâmetros utilizados para cada algoritmo (Figura 7.7).

Figura 7.7 – Janela para escolha do algoritmo

Depois da execução do algoritmo, é mostrada uma janela que informa qual o algoritmo utilizado, o tempo de execução do algoritmo, a carga de memória utilizada pelo algoritmo para o cálculo, e as probabilidades *a posteriori*, resultantes do cálculo da inferência (Figura 7.8). Esse resultado pode ser salvo em arquivo selecionando a opção correspondente na própria janela.

Dessa forma, os três elementos necessários para a realização da inferência (itens 1, 2 e 3 da Figura 7.3) e a saída do sistema são informados através da interface gráfica do sistema.

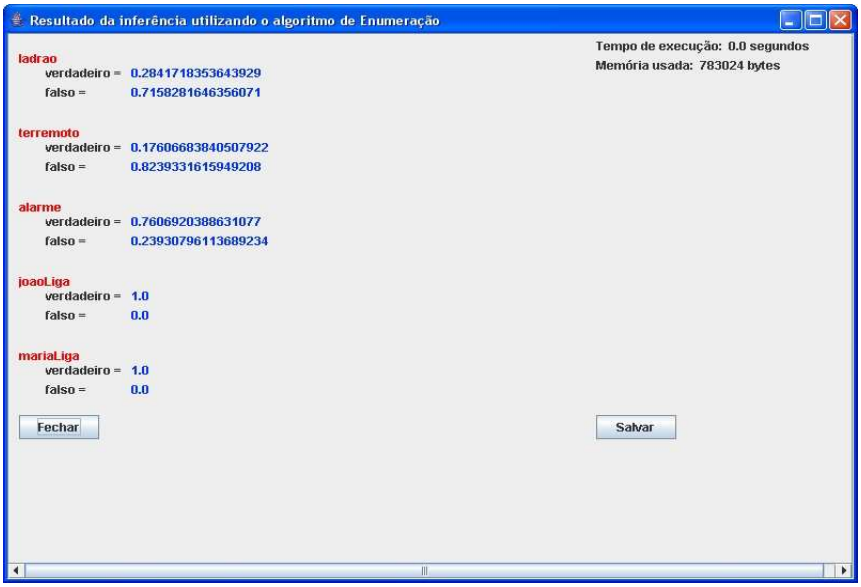


Figura 7.8 – Janela apresentando os resultados da inferência

7.2 Módulo de aprendizado de parâmetros

O aprendizado de parâmetros no *UFLABayes* pode ser realizado quando se tem disponível a estrutura, e deseja-se encontrar as tabelas de probabilidades condicionais de cada variável a partir de uma base de dados contendo eventos atômicos das variáveis no domínio da rede. Para realizar o aprendizado de parâmetros devem estar disponíveis três elementos: 1) a estrutura da rede bayesiana, 2) a base de dados a partir de qual serão calculados os parâmetros, e 3) o algoritmo aprendizado de parâmetros (Figura 7.9).

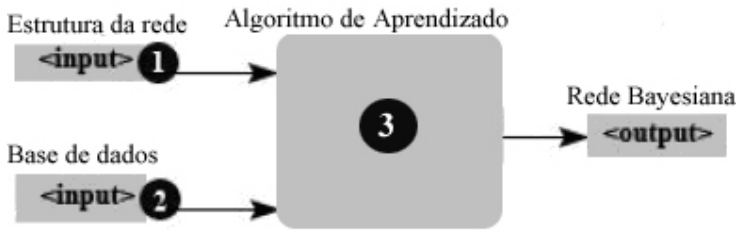


Figura 7.9 – Diagrama da operação de aprendizado de parâmetros no *UFLABayes*

A estrutura da rede *bayesiana* é armazenada de forma similar a uma rede *bayesiana* completa, porém, os valores das tabelas de probabilidades condicionais não são informados. Essa estrutura é selecionada para o aprendizado através da interface gráfica, utilizando o *menu* do sistema (Figura 7.10).

Figura 7.10 – Opção para abrir estrutura no *UFLABayes*

O algoritmo *AprendeParametros*, necessário para a realização do aprendizado, é automaticamente selecionado pelo sistema ao se optar pela opção Criar Parâmetros (Figura 7.11).

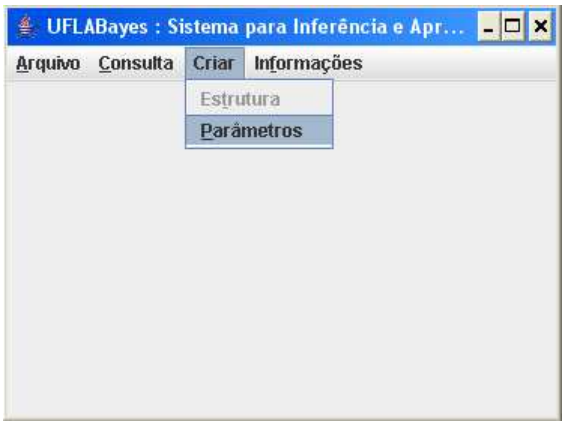


Figura 7.11 – Opção para criar parâmetros no *UFLABayes*

A base de dados a partir da qual os parâmetros são calculados (no formato TXT conforme Anexo B) é selecionada também por uma janela que permite a navegação entre os arquivos do computador.

Dessa forma, todos os elementos necessários para o aprendizado de parâmetros são informados de maneira gráfica no sistema. A saída de um cálculo de aprendizado de parâmetros é gravada em um arquivo XML e apresentada graficamente através da janela TPC (Figura 7.12).

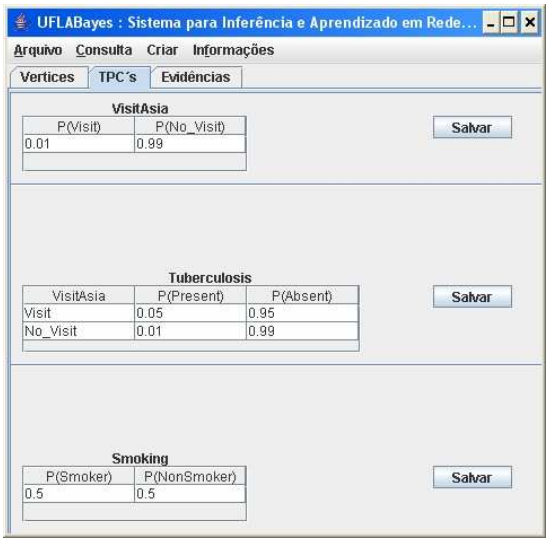


Figura 7.12 – Janela TPC - Resultado visual do algoritmo *AprendeParâmetros*

Os parâmetros aprendidos podem ser calibrados, modificando as probabilidades diretamente na tela de resultados escolhendo a opção salvar, que irá re-gravar no arquivo XML a nova TPC para a variável correspondente.

Dessa forma, o sistema está apto a fazer o cálculo da inferência na rede com os parâmetros aprendidos.

7.3 Módulo de aprendizado de estrutura

O aprendizado de estrutura no *UFLABayes* pode ser realizado quando não se tem disponível a estrutura e deseja-se encontrar uma que melhor condiz com os dados armazenados numa base de dados contendo eventos atômicos das variáveis no domínio da rede. Para realizar o aprendizado de estrutura devem estar disponíveis três elementos : 1) a base de dados a partir de qual será calculada a melhor estrutura, e 2) o algoritmo aprendizado de estrutura e 3) uma ordenação para as variáveis contidas na base de dados (Figura 7.13).

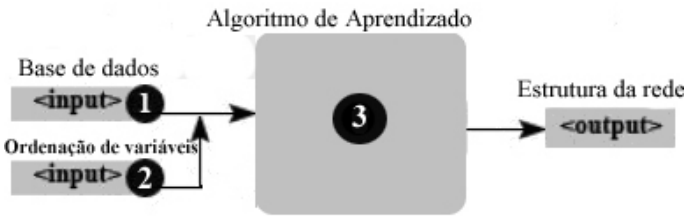


Figura 7.13 – Diagrama da operação de aprendizado de estrutura no *UFLABayes*

A ordenação das variáveis é implícita no arquivo da base dados; a primeira linha do arquivo contem o nome das variáveis, e a ordenação das colunas representam a ordenação das variáveis.

A base de dados a partir de qual será calculada a melhor estrutura (no formato TXT conforme Anexo B) é indicada através da interface gráfica, utilizando o *menu* do sistema (Figura 7.14).

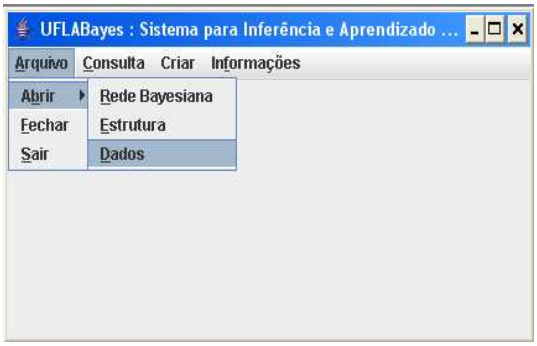


Figura 7.14 – Opção para abrir dados no *UFLABayes*

O algoritmo necessário para a realização do aprendizado de estrutura, é automaticamente selecionado pelo sistema ao se optar pela opção Criar Estrutura (Figura 7.15), pois apenas o algoritmo *K2* está implementado no sistema para esse fim.

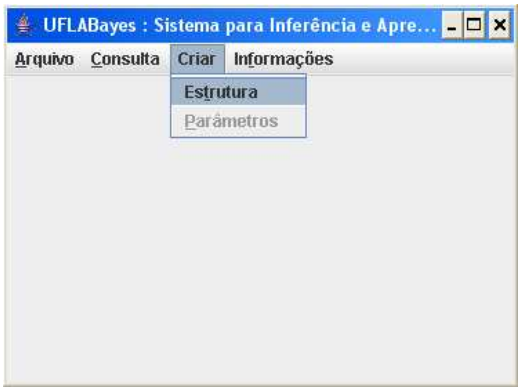


Figura 7.15 – Opção para criar estrutura no *UFLABAyes*

Dessa forma, todos os elementos necessários para o aprendizado de estrutura também são informados de maneira gráfica no sistema. A saída de um cálculo de aprendizado de estrutura é gravada em um arquivo XML e apresentada graficamente através da janela Vértices (Figura 7.16).

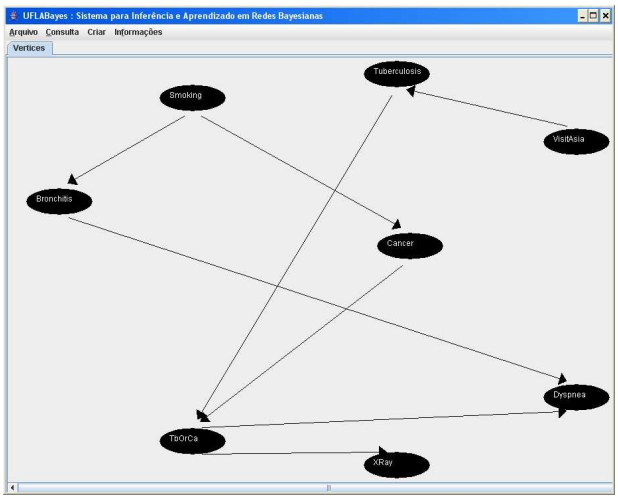


Figura 7.16 – Janela Vértices – Resultado visual do algoritmo *K2*

Dessa forma, o sistema está apto a efetuar o cálculo do aprendizado de parâmetros na nova rede com a estrutura aprendida.

8 RESULTADOS E DISCUSSÃO

Nessa seção serão apresentados os resultados das execuções de cada algoritmo implementado. Os resultados estão divididos em duas seções: inferência (seção 8.1), aprendizado de parâmetros (seção 8.2) e aprendizado de estrutura (seção 8.3). Todos os testes foram realizados numa máquina com Processador Pentium III 1.2GH$_Z$ e 512MB de memória RAM.

Conforme relatado na seção 4.2.3, o algoritmo de inferência *Gibbs Sampling* apresenta alguns problemas inerentes a suas características; e algumas sugestões de soluções foram implementadas. O método *Burn-in*, que descarta entre 5 a 10% das amostras iniciais geradas pelo algoritmo, foi implementado no sistema *UFLABayes*. O algoritmo *Gibbs Sampling* com esse método implementado será identificado por *Gibbs Sampling Burn-in* nas tabelas e figuras que apresentam os resultados, para avaliar o ganho de eficiência proporcionado pelo método.

Ao entrar em contato com as técnicas de inferência aproximada, surgiu a idéia de mesclar características de dois algoritmos aproximados que apresentaram soluções mais interessantes, e a título de validação, os resultados desse algoritmo, batizado de *Gibbs Weighting*, serão incluídos nas comparações. O *Gibbs Weighting* gera amostra dos eventos exatamente como o *Gibbs Sampling*, porém faz a atualização da probabilidade *a posteriori* de acordo com a função de ponderação do algoritmo *Likelihood Weighting*.

Portanto, os algoritmos de inferência avaliados foram: *Forward Sampling, Likelihood Weighting, Gibbs Sampling, Gibbs Sampling Burn-in, Gibbs Weighting, Enumeração e Eliminação de Variáveis.*

Para o aprendizado de parâmetros e estrutura foram implementados os algoritmos *AprendeParâmetros* e *K2*, respectivamente.

Os algoritmos implementados no sistema, foram testados em redes de topologias diversas. Entretanto, as redes *Ásia*, *DogProblem* e *CarDiagnostic* foram escolhidas para demonstrarem os resultados encontrados (Apêndices A, B e C respectivamente).

A rede *Asia* é uma rede multi-conectada associada a oito variáveis *booleanas*. A rede *DogProblem* é exemplo clássico encontrado em [CHARNIAK, 1991], mais simples, com cinco variáveis *booleanas*. E a rede *CarDiagnostic* é uma rede mais complexa, com 20 variáveis discretas, com diversas camadas, que pode ser aplicada ao diagnóstico de mau funcionamento de veículos automotores.

8.1 Algoritmos de inferência

Para comparar os algoritmos de inferência implementados, foram feitas análises de precisão dos resultados dependendo do número de iterações e do tempo para avaliar a eficiência de cada um deles.

Os resultados exatos, que serviram de base de comparação, foram obtidos a partir dos algoritmos de Enumeração e Eliminação de Variáveis, que apresentaram resultados idênticos (ignorando os erros de arredondamento).

As Figuras 8.1, 8.2 e 8.3 mostram os resultados obtidos pelos algoritmos para as redes de teste propostas, variando o número de iterações.

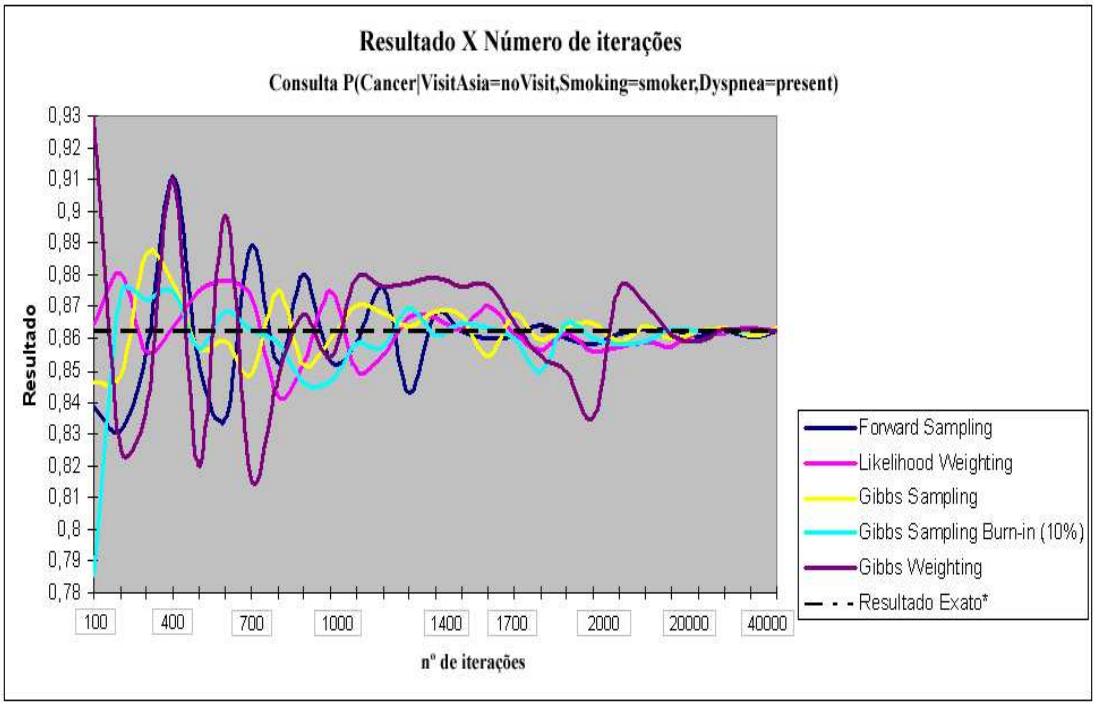


Figura 8.1 – Gráfico "Resultado X Nº de iterações" para uma consulta na rede *Ásia*

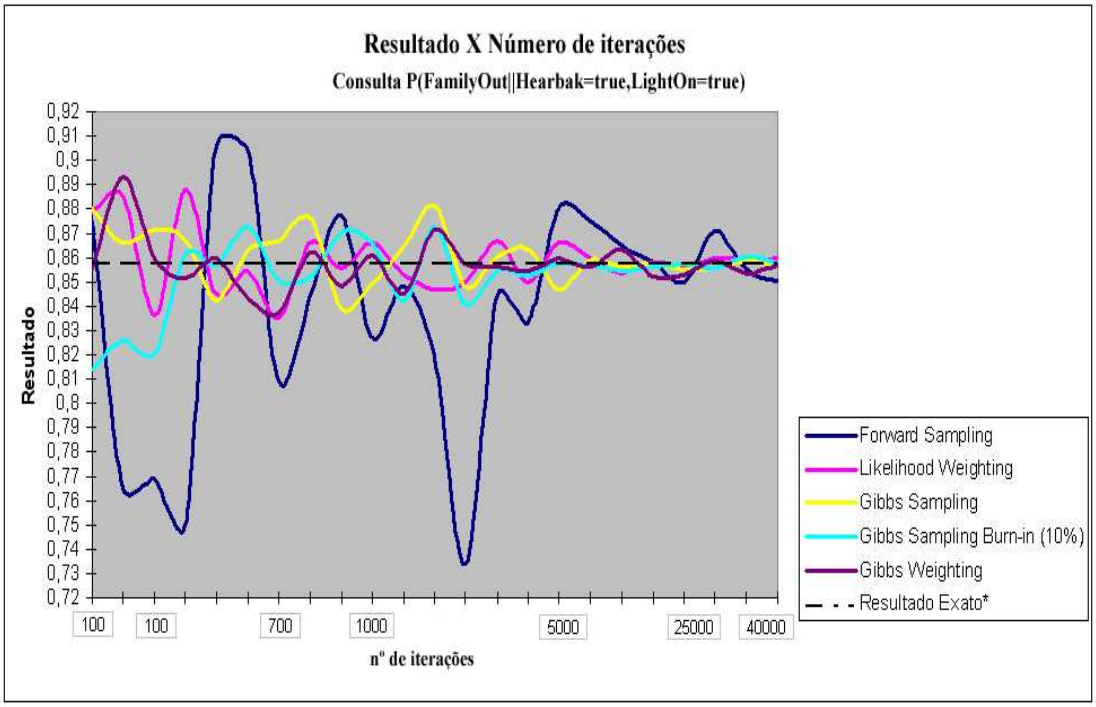


Figura 8.2 – Gráfico "Resultado X Nº de iterações" para uma consulta na rede *DogProblem*

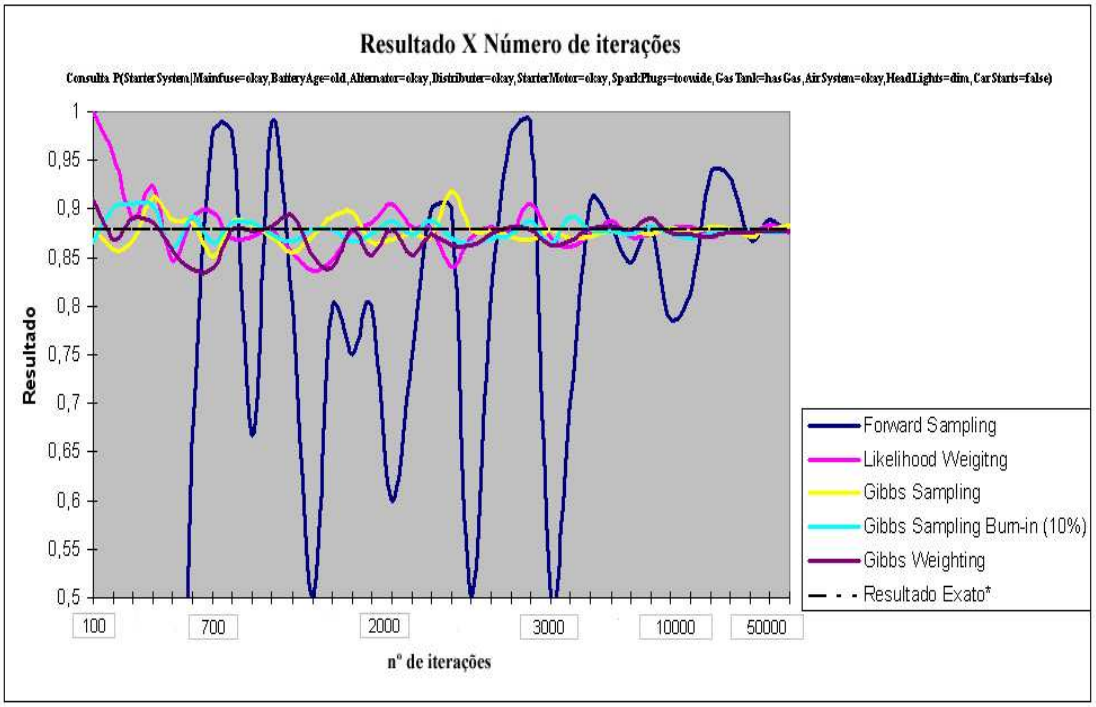


Figura 8.3 – Gráfico "Resultado X Nº de iterações" para uma consulta na rede *CarDiagnostic*

Pode-se observar que a precisão dos resultados gerados pelos algoritmos cresce aumentando-se o número de iterações.

Os algoritmos *Gibbs Sampling* e sua variação com o método *Burn-in* implementado se mostraram os mais eficientes, necessitando de um menor número de iterações para gerar um bom resultado exato. Pode-se notar que, em redes pequenas que necessitam de poucas iterações para garantir um resultado preciso, o descarte das amostras iniciais influencia negativamente no resultado, porém para um número maior de iterações o *Burn-in* fornece uma ajuda valiosa para a precisão do resultado.

O algoritmo *Likelihood Weighting* mostra-se um método poderoso considerando o número de iteração. Pode-se notar que para alguns casos das redes testadas o comportamento desse algoritmo se mostrou similar ao do *Gibbs Sampling*. Deve-se destacar a facilidade de implementação desse algoritmo em comparação com os métodos mais eficientes.

O algoritmo *Forward Sampling* apresentou um bom comportamento quando efetuada uma consulta na rede *Ásia*, porém para as consultas nas redes *DogProblem* e *CarDiagnostic*, esse algoritmo parece inviável. Este fato pode ser explicado pelo grande número de amostras rejeitadas. Segundo [RUSSEL & NORVIG, 2004], a fração de amostras rejeitadas cresce exponencialmente conforme o número de variáveis de evidência cresce, e assim o algoritmo é simplesmente inútil para problemas complexos. Esse fato pode ser comprovado experimentalmente, observando o comportamento do algoritmo na Figura 8.3.

O algoritmo *Gibbs Weighting*, proposto pelo autor, demonstrou comportamento similar ao de seus precursores (*Gibbs Sampling* e *Likelihood Weighting*), superando o algoritmo *Forward Sampling* nas análises de precisão variando o número de iterações nas redes testadas.

O número de iterações necessárias para um que o algoritmo gere uma boa estimativa é um bom indicador de eficiência dos algoritmos. Porém, cada um tem uma velocidade média de execução por iteração. Para comparar qual algoritmo converge mais rapidamente, estipulamos tempo para a execução ao invés de número de iterações. As Figuras 8.4, 8.5 e 8.6 ilustram o comportamento dos algoritmos para essa análise. Pode-se observar que não foram feitas análises para o algoritmo *Gibbs Sampling Burn-in*, pois não é possível prever quantas amostras seriam geradas e descartadas pelas iterações.

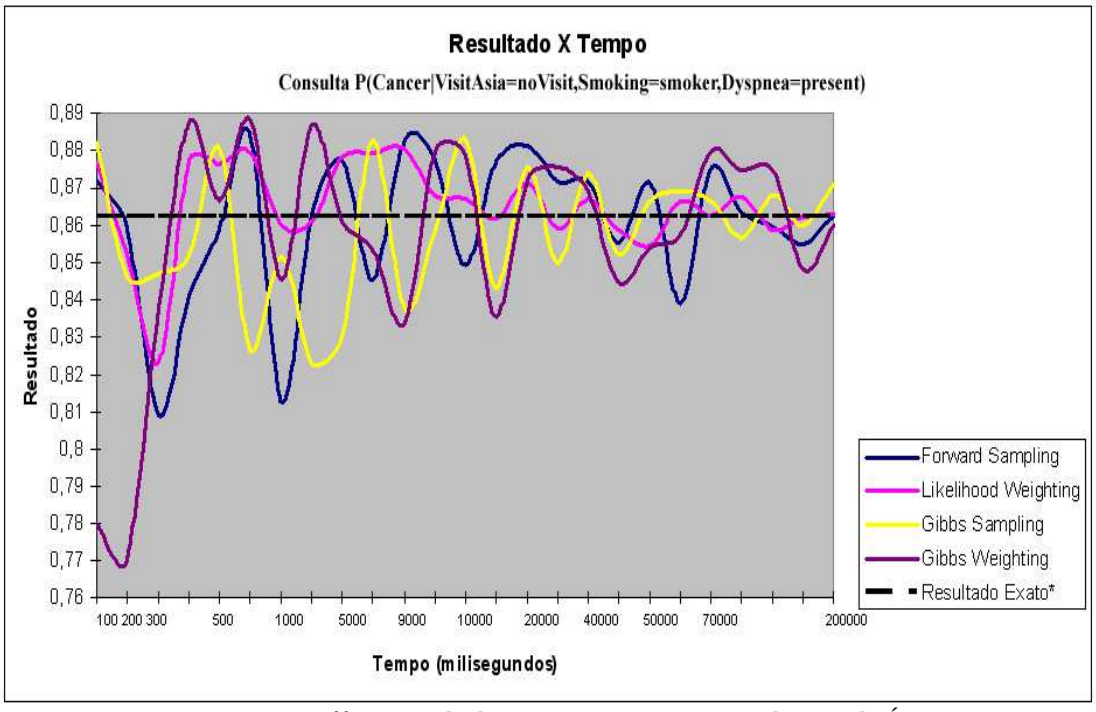


Figura 8.4 – Gráfico "Resultado X Tempo" para uma consulta na rede *Ásia*

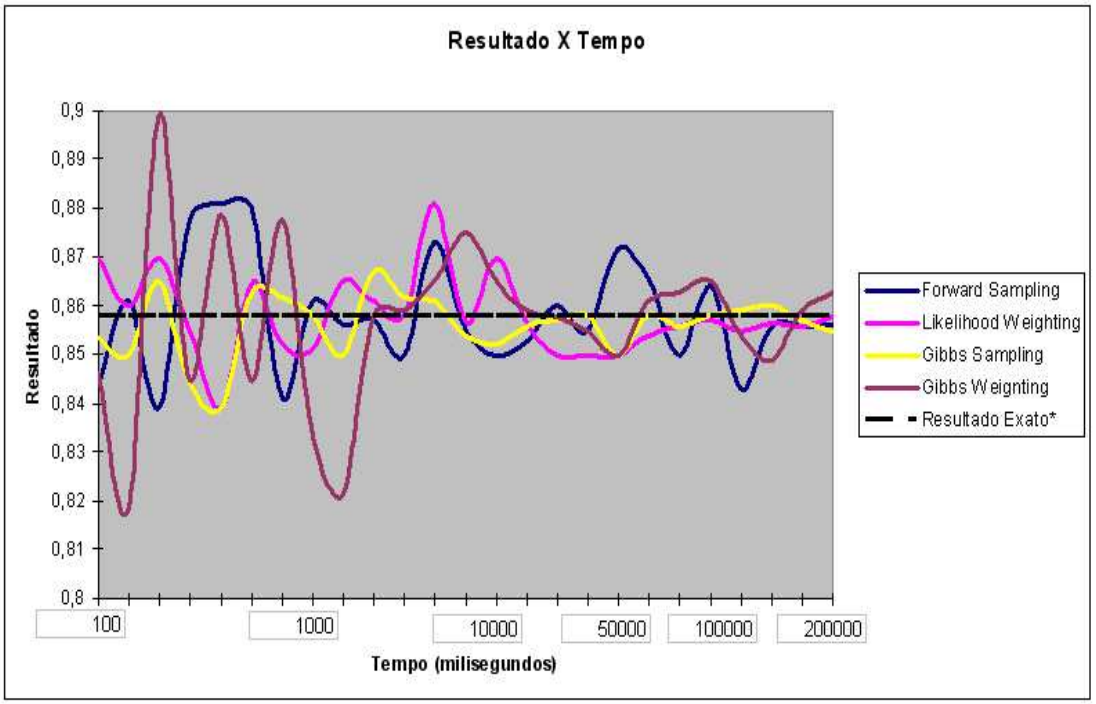


Figura 8.5 – Gráfico "Resultado X Tempo" para uma consulta na rede *DogProblem*

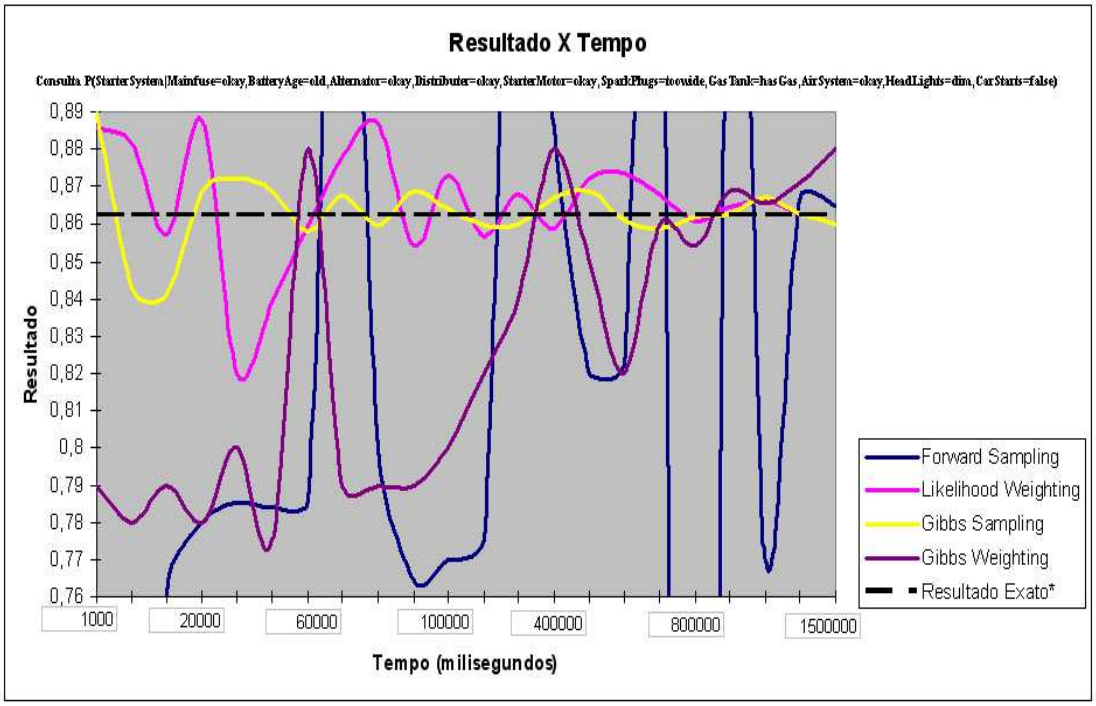


Figura 8.6 – Gráfico "Resultado X Tempo" para uma consulta na rede *CarDiagnostic*

Pode-se observar que o *Gibbs Sampling* retorna um resultado preciso mais rapidamente, seguido do *Likelihood Weighting*. Esse comportamento é apenas evidenciado em redes maiores (Figura 8.6).

A grande vantagem do algoritmo *Gibbs Weighting* é o pequeno número de iterações necessárias que o resultado convirja para o resultado exato, pois analisando os gráficos 8.4, 8.5 e 8.6 pode-se observar que a eficiência medida pelo tempo necessário para a convergência não supera a de seus precursores.

Observando as Figuras de 8.1 a 8.6 pode-se concluir que o algoritmo mais poderoso é o *Gibbs Sampling*, considerando o número de iterações ou o tempo necessário para atingir o resultado exato.

A boa qualidade dos números aleatórios gerados constitui o segredo para a eficiência e boa convergência dos algoritmos de inferência aproximada. Nos algoritmos implementados, empregou-se um eficiente gerador de números aleatórios, o *MersenneTwister* [MATSUMOTO & NISHIMURA, 1998], que se encontra disponível na Internet.

Para comparar a eficiência dos algoritmos exatos, foram geradas redes aleatórias com diferentes números de variáveis (sempre booleanas) e mediu-se o tempo de execução. A Tabela 8.1 ilustra os tempos medidos.

Tabela 8.1 – Tempo de Execução dos algoritmos exatos

Tempo de Execução (milisegundos)		**Número de variáveis**
Enumeração	Eliminação de Variáreis	
15	15	**3**
21	20	**4**
38	32	**5**
47	45	**7**
67	59	**8**
15485	4875	**20**

O aumento do tamanho da rede implica num aumento do tempo de execução de forma não-linear. Sabe-se que a complexidade de tempo de ambos os algoritmo exatos implementados é exponencial em relação ao tamanho da *rede* [RUSSEL & NORVIG, 2004]. Contudo, o algoritmo de Eliminação de Variáveis, como previsto e relatado na seção 4.1.2, elimina cálculos repetidos, proporcionando um ganho na eficiência desse algoritmo em relação ao algoritmo de Enumeração.

Deve-se destacar que a ordenação das variáveis influencia no tempo de execução do algoritmo *Eliminação de Variáveis* [COZMAN, 2000; RUSSEL & NORVIG, 2004], e que neste trabalho a ordenação utilizada foi sempre das folhas para a raiz.

8.2 Algoritmo de aprendizado de parâmetros

Nessa seção são especificados os testes e resultados da aplicação do algoritmo *AprendeParâmetros* aos conjuntos de dados considerados.

A rede considerada para os testes do algoritmo de aprendizado de parâmetros foi a redes *Ásia*. A eficiência desse algoritmo pode ser avaliada levando em conta a qualidade dos parâmetros aprendidos, variando o número de eventos presentes na base de dados. Essa qualidade foi medida calculando a "Divergência de Kullback-Leibler" [MACKKAY, 2003], também conhecida como entropia relativa, ou divergência *KL*.

A divergência *KL* mede a qualidade de quaisquer probabilidades estimadas (seja pelos algoritmos de inferência, seja pelos algoritmos de aprendizado ou qualquer outro método que retorne uma distribuição de probabilidades). Essa divergência mede a diferença entre duas probabilidades (as probabilidades exatas e as probabilidades estimadas pelos algoritmos neste caso) usando a equação (8.1), onde P denota a probabilidade exata e $\hat{P}$ denota a estimativa:

$$\sum_{x \in X} P(x|e) \log \left(\frac{P(x|e)}{\hat{P}(x|e)} \right), \tag{8.1}$$

Onde quanto menor for a divergência *KL*, menor a distância entre as probabilidades em questão[3]. Essa medida pode ser interpretada como a distância entre as entropias[4] de duas probabilidades.

A principal vantagem desse método é que ele permite considerar todas as probabilidades envolvidas de uma só vez e garante uma comparação precisa entre diferentes probabilidades [MACKKAY, 2003].

A Figura 8.7 apresenta um gráfico com as divergências *KL* medidas na TPC da variável aleatória TbOrCa aprendida pelo algoritmo *AprendeParametros* aplicada a rede *Ásia*, em relação a TPC original da variável, com diferentes números de eventos na base de dados. Observe que quanto menor a divergência *KL*, melhor a qualidade dos parâmetros aprendidos.

[3] A medida da divergência *KL* é sempre um número não-negativo, satisfazendo muitas condições importantes. Detalhes desse e outros métodos para comparação entre probabilidades são apresentados em [MACKKAY, 2003].

[4] Entropia é medida da quantidade de informação que uma probabilidade carrega.

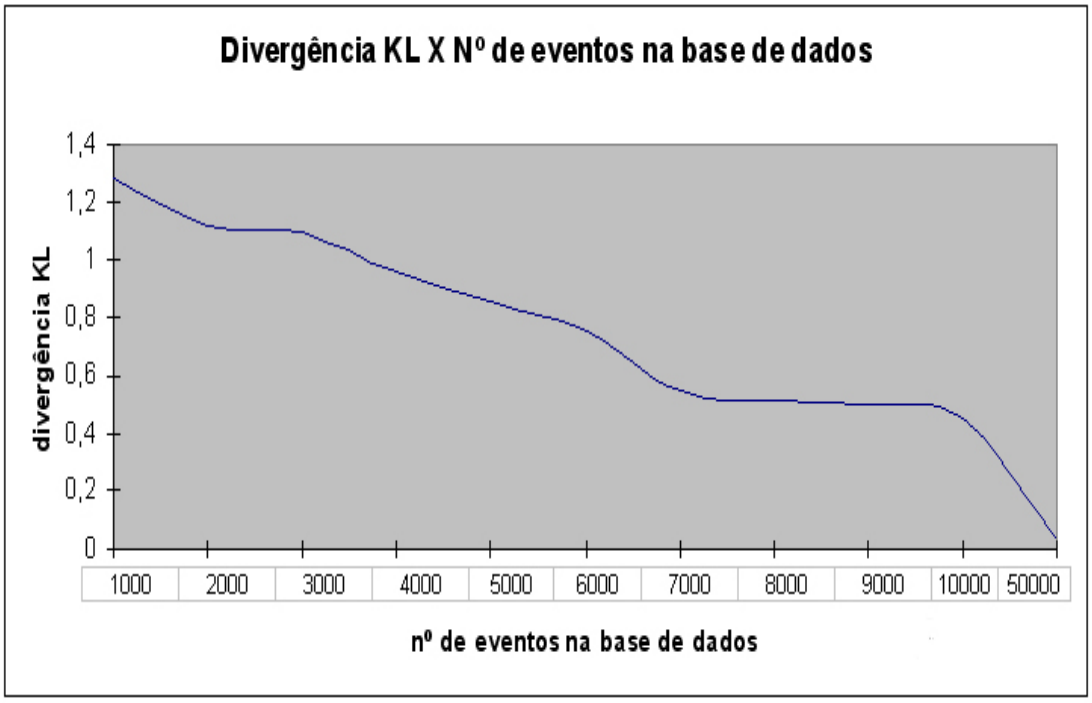


Figura 8.7 – Gráfico "Divergência *KL* medida na TPC de *TbOrCa* gerada pelo algoritmo *Aprende Parâmetros X* Número de casos na base de dados"

A Figura 8.7 destaca uma queda na divergência *KL* com o aumento do número de eventos na base de dados, significando um aumento na qualidade dos parâmetros gerados. Assim, pode-se concluir que a implementação do algoritmo *AprendeParâmetros* alcançou seu objetivo principal.

Deve-se destacar que a rede *Ásia* testada possui apenas oito vértices e a divergência *KL* se aproximou de zero (parâmetros gerados próximos aos parâmetros ideais) apenas quando o número na base de dados se aproximou de 50000. Em problemas reais não é uma tarefa barata, nem mesmo intuitiva coletar muitas informações para a base de dados.

8.3 Algoritmo de aprendizado de estrutura

Nessa seção são especificados os testes e resultados da aplicação do algoritmo de aprendizado de estrutura *K2* aos conjuntos de dados considerados.

Para esse algoritmo, foram realizados testes tendo a rede *Asia* como parâmetro. [HECKERMAN *et al*, 1994] sugerem a utilização de *entropia cruzada* e da *análise estrutural* [MACKAY, 2003] para medir a qualidade das redes aprendidas. Porém, na utilização da medida da entropia são necessários os seguintes passos: criar uma estrutura comum para a rede original e a rede aprendida, atribuir probabilidades e aplicar a equação de Kulllback-Leibler. Contudo, o enfoque usado neste trabalho foi apenas medir a diferença estrutural entre as redes original e aprendida, especificamente a relação entre arcos adicionados e faltosos.

A Figura 8.8 mostra as redes aprendidas pelo algoritmo em comparação com a rede ótima.

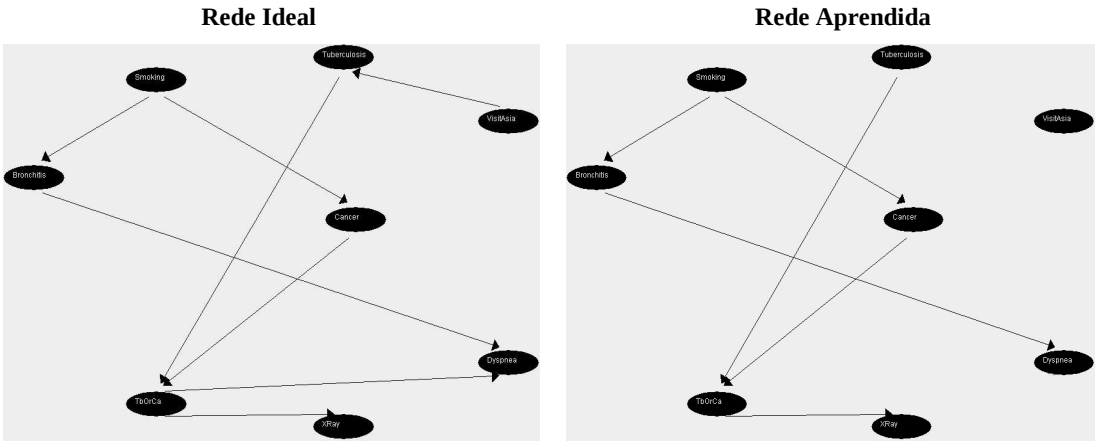


Figura 8.8 – Rede Ásia ideal X Rede Ásia aprendida pelo algoritmo *K2* com 10000 eventos na base de dados

O algoritmo *K2* aprendeu a rede *Ásia* a partir da base de dados com 10000 eventos na base de dados com dois arcos faltosos (*VisitAsia* $\rightarrow$ *Tubeculosis* e *TbOrCa* $\rightarrow$ *Dyspnea*) e nenhum arco adicionado sem necessidade. Assim, pode-se concluir que o algoritmo *K2* está apto a alcançar seu objetivo.

Novamente, deve-se destacar que a rede *Ásia* testada possui apenas oito variáveis e a rede obtida ainda possuía dois nós faltosos em relação à rede ótima quando se tinha 10.000 eventos na base de dados. Em problemas reais não é uma tarefa barata, nem mesmo intuitiva coletar muitos informações para a base de dados. Esse fato pode ser considerado mais grave do que a estimação de parâmetros “distantes” dos parâmetros ideais gerados

pelos algoritmos de aprendido de parâmetros, pois podem resultar em consultas totalmente inconsistente com a realidade.

Vale lembrar a principal desvantagem desse algoritmo, que é a necessidade de uma boa ordenação entre as variáveis para garantir um bom resultado da rede gerada.

9 CONSIDERAÇÕES FINAIS

9.1 Conclusões

O objetivo deste trabalho foi a construção de um sistema capaz de realizar inferência e aprendizado em redes *bayesianas*. Assim, pode-se observar que o objetivo principal foi alcançado com sucesso. A escolha pela implementação de cada algoritmo estudado, ao invés de usar pacotes e ferramentas prontas, permitiu um grande entendimento das técnicas aplicadas à resolução dos problemas de inferência e aprendizado em redes *bayesianas*.

Sobre os algoritmos de inferência, essa monografia permitiu concluir que:

- Na literatura encontram-se várias propostas para a resolução deste problema de grande importância prática. As propostas atuais de implementações dos algoritmos de inferência oferecem boas soluções, podendo ser consideravelmente melhorados para instâncias específicas de redes *bayesianas*.
- A escolha do algoritmo de inferência ideal depende, essencialmente, do tamanho da rede *bayesiana* e dos recursos que se deseja otimizar. A inferência exata consome processamento exageradamente em redes pequenas e consome tempo demais em redes muito grandes. Tornando-se inviável para certos problemas reais, principalmente se levarmos em conta que os algoritmos poderiam ser destinados a resolver problemas em tempo real ou ser implementada em sistemas móveis e/ou embarcados com restrição de processamento e memória.
- Dentre os algoritmos aproximados, o *Gibbs Sampling* mostrou-se, em geral, o método mais poderoso para realizar a inferência, merecendo atenção especial para possíveis melhorias e adaptações que tornem a inferência ainda mais eficiente. O algoritmo *Likelihood Weighting* também se mostrou um método poderoso (tendo desempenho similar em muitos casos ao do *Gibbs Sampling*) principalmente se levarmos em conta sua fácil implementação. A grande vantagem do algoritmo *Gibbs Weighting* é o pequeno número de iterações necessárias para retornar um resultado preciso, pois pode-se observar avaliando os tempos de execução medidos que o algoritmo demora mais para a convergência.

Em relação aos algoritmos de aprendizado, pode-se concluir que:

- O aprendizado de parâmetros e de estrutura são operações de grande importância, já que a construção de redes *bayesianas* não é uma atividade trivial. Muitas vezes o domínio do problema não apresenta de forma intuitiva as relações causa-efeito, podendo confundir o especialista ao modelar a rede.
- Enquanto o aprendizado de parâmetros é relativamente simples, o aprendizado de estrutura é um assunto em geral complexo e ainda não está bem resolvido.
- Em problemas reais, não é aconselhado construir uma rede *bayesiana* sem assistência do especialista para validar ou recalibrar os resultados gerados pelos algoritmos de aprendizado. Apesar de os algoritmos oferecerem uma ajuda valiosa, não se pode estimar quantas amostras são necessárias para que o algoritmo possa convergir para o resultado exato, sem saber de antemão, qual é o resultado exato. E em problemas reais, não é uma tarefa barata coletar muitos casos para a base de dados.

9.2 Trabalhos futuros

A implementação deste trabalho despertou grande interesse em pesquisar diversos aspectos relacionados a redes *bayesianas* com diferentes enfoques. A seguir, são apresentadas algumas sugestões para trabalhos futuros.

Sobre os algoritmos de inferência exata foram observadas as seguintes necessidades:

- Analisar o impacto da ordenação de variáveis no tempo de execução do algoritmo *Eliminação de Variáveis;*
- Estudar heurísticas para a ordenação de variáveis para o algoritmo *Eliminação de Variáveis* (encontrar a ordenação ótima é um problema *NP-Completo*).

Sobre os algoritmos de inferência aproximada foram identificadas as seguintes necessidades:

- Aprofundar os estudos nas propriedades dos métodos de amostragem;
- Fazer análises mais sofisticadas nos algoritmos, buscando identificar outras propriedades que permitam uma maior contribuição para esta área;
- Analisar possíveis melhorias e adaptações no algoritmo Gibbs Sampling;

- Determinar procedimentos para a escolha do algoritmo a ser implementado, objetivando otimizar a operação de inferência.

Em relação à aprendizagem, são sugestões de trabalhos futuros:

- Estudar e implementar outros algoritmos de aprendizado de estrutura, baseados em diferentes métodos permitindo uma comparação entre eles;
- Estudar as abordagens de aprendizagem de estrutura que trabalhem quando a base de dados contêm dados faltosos;
- Estudar a fundo as propriedades e características dos algoritmos de aprendizado de estrutura que permitam uma maior contribuição para a área;
- Determinar procedimentos que otimizem a construção de uma rede *bayesiana*, quantificando, por exemplo, o número mínimo de dados que precisam ser coletados para os algoritmos de aprendizado gerarem resultados aceitáveis.

E no geral, objetiva-se pesquisar toda a abordagem dessa monografia para redes *bayesianas* com variáveis contínuas.

ANEXO A – EXEMPLO DO ARQUIVO DE ENTRADA NO SISTEMA *UFLABAYES* PARA REPRESENTAR A ESTRUTURA DE UMA REDE *BAYESIANA* NO FORMATO XML

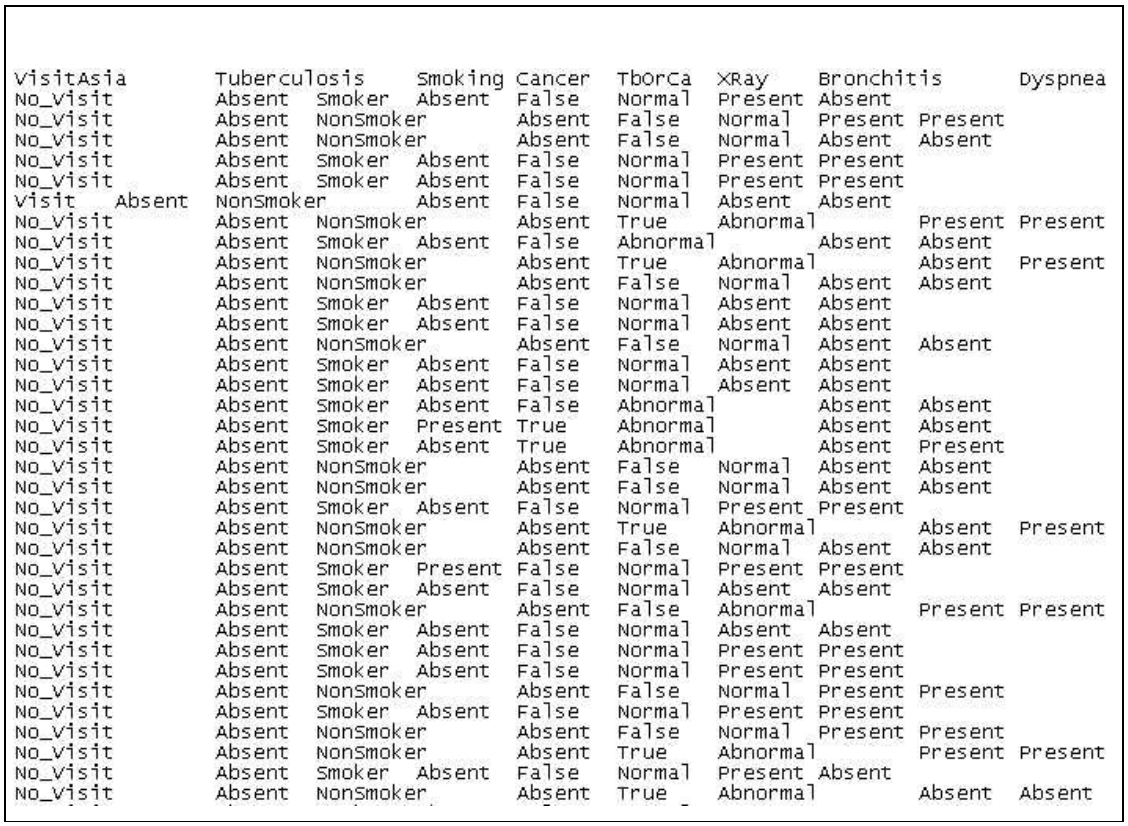
O arquivo Xml deve se conter informações sobre cada variável como mostrado na figura. Se o arquixo for destinado à construção automática de parâmetros o campo <tpc> </tpc> deve estar vazio. Se conter alguma probabilidade, esta será descartada pelo algoritmo.

```
<bayes_classifier>
    <vertice>
        <nome>nublado</nome>
        <nPais>0</nPais>
        <nFilhos>2</nFilhos>
        <estados>verdadeiro,falso</estados>
        <pais></pais>
        <filhos>regador,chuva</filhos>
        <tpc>0.5,0.5</tpc>
    </vertice>

    <vertice>
        <nome>regador</nome>
        <nPais>1</nPais>
        <nFilhos>1</nFilhos>
        <estados>verdadeiro,falso</estados>
        <pais>nublado</pais>
        <filhos>grama_molhada</filhos>
        <tpc>0.1,0.9,0.5,0.5</tpc>
    </vertice>

    <vertice>
        <nome>chuva</nome>
        <nPais>1</nPais>
        <nFilhos>1</nFilhos>
        <estados>verdadeiro,falso</estados>
        <pais>nublado</pais>
        <filhos>grama_molhada</filhos>
        <tpc>0.8,0.2,0.2,0.8</tpc>
    </vertice>

    <vertice>
        <nome>grama_molhada</nome>
        <nPais>2</nPais>
        <nFilhos>0</nFilhos>
        <estados>verdadeiro,falso</estados>
        <pais>regador,chuva</pais>
        <filhos></filhos>
        <tpc>0.99,0.01,0.9,0.1,0.9,0.1,0.0,1.0</tpc>
    </vertice>
</bayes_classifier>
```

ANEXO B – *TEMPLATE* DO ARQUIVO DE ENTRADA NO SISTEMA *UFLABAYES* PARA ARMAZENAR A BASE DE DADOS NO FORMATO TXT

A primeira linha da coluna deve conter o nome das variáveis e as colunas devem conter o estado observado para a variável.

O algoritmo de aprendizado de estrutura utiliza a ordenação apresentada nas colunas da primeira linha do arquivo. Cada palavra no arquivo de ser separada pelo caractere de tabulação.



VisitAsia	Tuberculosis	Smoking	Cancer	TbOrCa	XRay	Bronchitis	Dyspnea
No_Visit	Absent	Smoker	Absent	False	Normal	Present	Absent
No_Visit	Absent	NonSmoker	Absent	False	Normal	Present	Present
No_Visit	Absent	NonSmoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Absent	False	Normal	Present	Present
No_Visit	Absent	Smoker	Absent	False	Normal	Present	Present
Visit	Absent	NonSmoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	NonSmoker	Absent	True	Abnormal	Present	Present
No_Visit	Absent	Smoker	Absent	False	Abnormal	Absent	Absent
No_Visit	Absent	NonSmoker	Absent	True	Abnormal	Absent	Present
No_Visit	Absent	NonSmoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	NonSmoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Absent	False	Abnormal	Absent	Absent
No_Visit	Absent	Smoker	Present	True	Abnormal	Absent	Absent
No_Visit	Absent	Smoker	Absent	True	Abnormal	Absent	Present
No_Visit	Absent	NonSmoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	NonSmoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Absent	False	Normal	Present	Present
No_Visit	Absent	NonSmoker	Absent	True	Abnormal	Absent	Present
No_Visit	Absent	NonSmoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Present	False	Normal	Present	Present
No_Visit	Absent	Smoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	NonSmoker	Absent	False	Abnormal	Present	Present
No_Visit	Absent	Smoker	Absent	False	Normal	Absent	Absent
No_Visit	Absent	Smoker	Absent	False	Normal	Present	Present
No_Visit	Absent	Smoker	Absent	False	Normal	Present	Present
No_Visit	Absent	NonSmoker	Absent	False	Normal	Present	Present
No_Visit	Absent	Smoker	Absent	False	Normal	Present	Present
No_Visit	Absent	NonSmoker	Absent	False	Normal	Present	Present
No_Visit	Absent	NonSmoker	Absent	True	Abnormal	Present	Present
No_Visit	Absent	Smoker	Absent	False	Normal	Present	Absent
No_Visit	Absent	NonSmoker	Absent	True	Abnormal	Absent	Absent

REFERÊNCIAS BIBLIOGRÁFICAS

[CASELLA & GEROGE, 1992] CASELLA G., GEROGE E. I.. "Explaining the Gibbs sampler". Am. Stat. 46: p.167–174, 1992.

[CASTILHO & GUTIERREZ, 1997] CASTILHO E., GUTIERREZ J.. "Expert Systems and Probabilistic Network Models". Ed. Springer, 1997.

[CHARNIAK,1991] CHARNIAK E. K.. "Bayesian networks without tears", AI magazine, pp.50- 63, 1991.

[CHENG et al, 1997a] J. CHENG, D. BELL, W. LIU.. “An algorithm for Bayesian belief networks construction from data”. In: AI & STAT’97, 1997a. Proceedings... Florida. p. 83-90.

[CHENG et al, 1997b] J. CHENG, D. BELL, W. LIU.. “Learning belief networks from data: An information theory based approach”. In: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 6., 1997b. Proceedings. p. 325-331.

[COOPER & HERSKOVITS, 1992] COOPER G. F., HERSKOVITS E.. "A Bayesian method for the induction of probabilistic networks from data". Machine Learning, v. 9, p. 309-347, 1992.

[COZMAN, 2000] COZMAN F. G.. "Generalizing Variable Elimination in Bayesian Netorks". To appears: Workshop on Probabilistic Reasoning in Artificial Intelligence, Atibaia, 2000.

[DECHTER, 1999] R. DECHTER R.. "Bucket elimination: A unifying framework for probabilistic inference". In M. I. Jordan, editor, *Learning in Graphical Models*, p.75–104. MIT Press, 1999.

[DRUZDZEL & VAN DER GAAG , 2000] DRUZDZEL, M. J., VAN DER GAAG, L. C.. "Building probabilistic networks: “Where do the numbers come from?”" IEEE Transaction on knowledge and data engineering, v. 12, n. 4, p. 481-486, july/aug. 2000.

[FUNG & CHANG, 1990] FUNG R., CHANG K.C.. "Weighing and integrating evidence for stochastic simulation in Bayesian networks". In M. Henrion, R. Shachter, L. Kanal, & J. Lemmer (Eds.), Uncertainty in Artificial Intelligence 5. Amsterdam: Elsevier.

[GEMAN & GEMAN, 1984] GEMAN S., GEMAN D.. "Stochastic relaxation, Gibbs distribution and Bayesian restoration of images". IEE Transactions on Pattern Analysis and Machine Intelligence 6: p.721–741, 1984.

[HECKERMAN et al, 1994] HECKERMAN D., GEIGER D., CHICKERING D. M.. "Learning Bayesian networks: The combination of knowledge and statistical data". Microsoft Research technical report (MSR-TR-94-09). 1994.

[HECKERMAN, 1995] HECKERMAN D.. "A tutorial on learning with bayesian networks". Microsoft Research technical report (MSR-TR-95-06). 1995.

[HERSKOVITS & COOPER, 1990] E. H. HERSKOVITS, F. F. COOPER.. Kutato: "An entropy-driven system for construction of probabilistic expert systems from databases". In: CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, 6., 1990. Proceedings... Cambridge: Piero Bonissone Ed., 1990. p. 54-62.

[HRUSCHKA, 1997] HRUSCHKA E. R.. "Propagação de evidências em redes Bayesianas: Diagnóstico sobre doenças pulmonares".132 p. Dissertação (Mestrado em Ciência da Computação) – Universidade de Brasília, Brasília, 1997.

[JENSEN, 2001] JENSEN F.V.. "Bayesian Networks and Decision Graphs". Ed. Springer, 2001.

[JUNG, 2004] JUNG, C. F.. "Metodologia para Pesquisa & Desenvolvimento". Ed.Axcel Books do Brasil Editora, 2004.

[KOEHLER & NASSAR , 2002] KOEHLER C., NASSAR S.. M. "Modelagem de redes Bayesianas a partir de bases de dados médicas". In: Jornadas argentinas de informática e investigacion operativa, 31., 2002. Anais... [S.l.n.], p.164-176, 2002.

[LACERDA & BRAGA, 2004] LACERDA W.S., BRAGA A.P.. "Experimento de um Classificador de Padrões Baseado na Regra Naive de Bayes". INFOCOMP Journal of Computer Science, Lavras, v.3, n.1, p.30-35, 2004.

[LUNA, 2004] LUNA J.E.O.. "Algoritmos EM para aprendizado de redes bayesianas a partir de dados incompletos". Dissertação de mestrado.Universidade Federal de Mato Grosso do Sul (DCT-UFMS), 2004.

[MACKKAY, 2003] MACKAY D. J.C.. "Information Theory, Inference, and Learning Algorithms". Ed.Cambridge University Press, 2003.

[MATSUMOTO & NISHIMURA, 1998] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator", ACM Transactions on Modeling and Computer Simulation,Vol.8, No.1, pp 3-30. <http://www.math.keio.ac.jp/matumoto/emt.html >.

[NEAPOLITAN, 1990] NEAPOLITAN R.E.. "Learning Bayesian Networks". Ed. Prentice Hall, 1990.

[PEARL, 1986a] PEARL J.. "Fusion, propagation, and structuring in belief networks". Journal of Artificial Intelligence, v.29 , p.241-288, 1986.

[PEARL, 1986b] PEARL J.. "Reasoning in Intelligent Systems". Ed.Morgan Kaufman, 1986.

[PENA, 2006] PENA, S.D.. "Thomas Bayes: O 'cara'!" Revista Ciência Hoje, v.38, n.228, p.22-29, 2006.

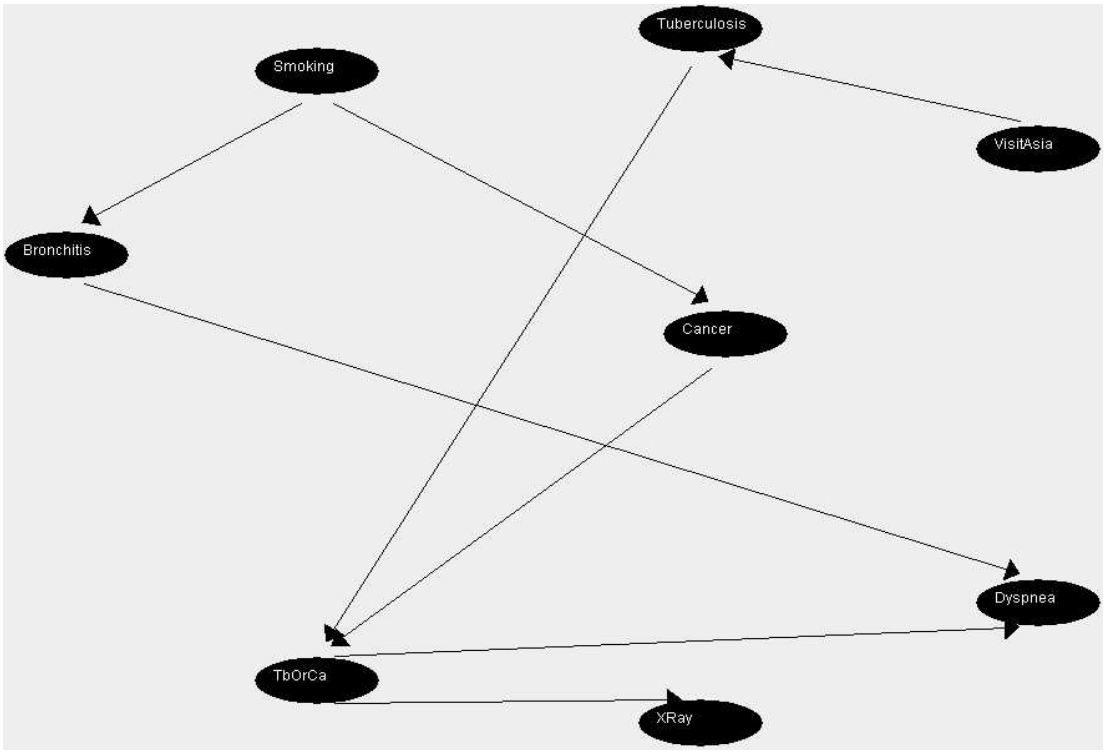
[RUSSEL & NORVIG, 2004] RUSSEL S. J., NORVIG P.; "Inteligência Artificial". Tradução da 2a Edição. Ed.Campus, 2004.

[SILVA & LADEIRA , 2002] SILVA, W. T. da., LADEIRA, M. "Mineração de dados em redes Bayesianas". In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 22., 2002, Florianópolis. Anais... Florianópolis: UFSC, 2002.

[SPIRTES et al, 1990] P. SPIRTES, C. GLYMOUR, R. SCHEINES.. “Causality from probability”. In: ADVANCED COMPUTING FOR THE SOCIAL SCIENCES, 1990, Williamsburgh. Proceedings.

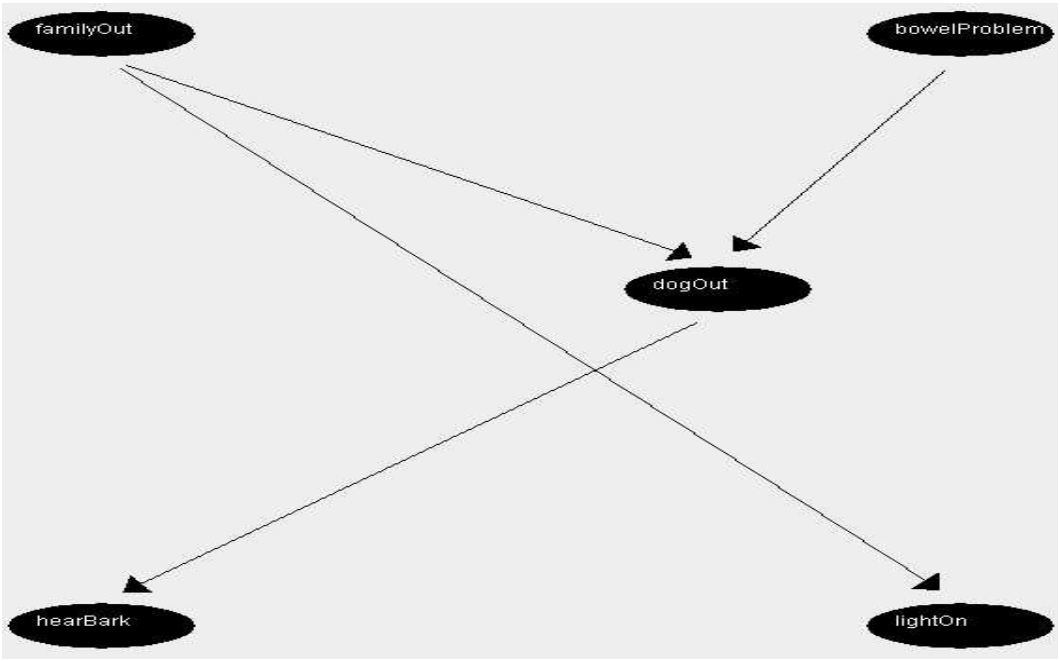
[YUDKOWSKY, 2003] YUDKOWSKY E.. "An Intuitive Explanation of Bayesian Reasoning", 2003 . http://yudkowsky.net/bayes/bayes.html. Fevereiro de 2007.

APÊNDICE A – REDE *ASIA*



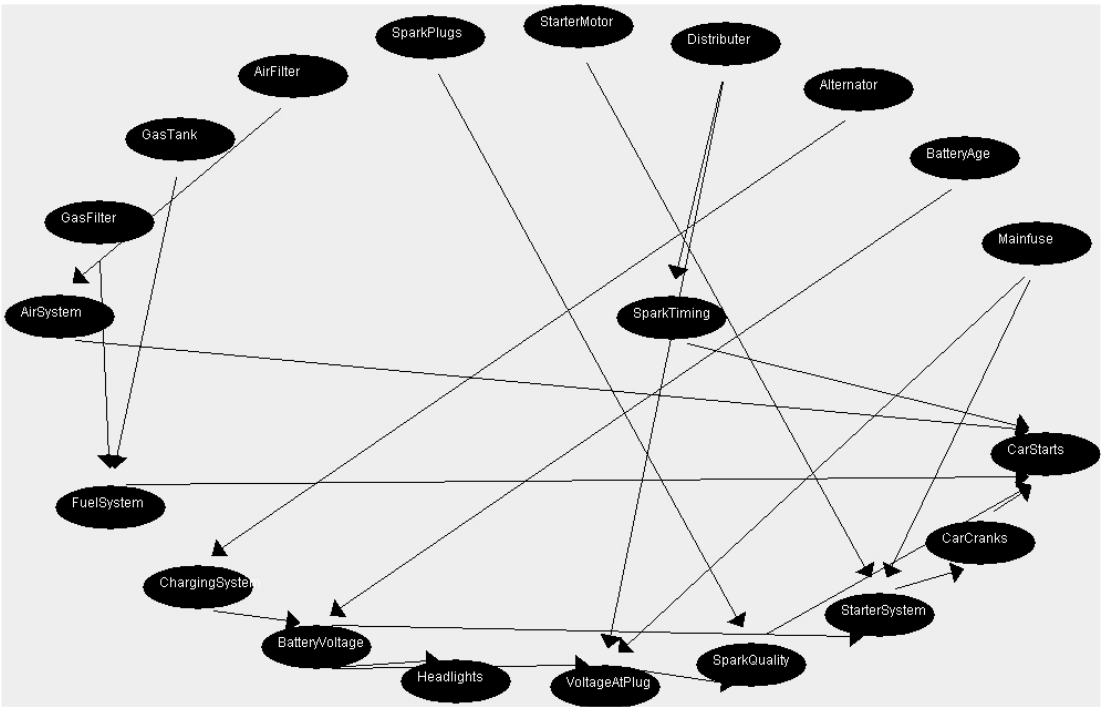
Variável	**Estados**	
VisitAsia	*visit*	*no_Visit*
Tuberculosis	*present*	*absent*
Smoking	*smoker*	*nonSmoker*
Bronchitis	*present*	*absent*
Cancer	*present*	*absent*
TbOrCa	*true*	*false*
XRay	*abnormal*	*normal*
Dyspnea	*present*	*absent*

APÊNDICE B – REDE *DOGPROBLEM*



Variáveis	**Estados**	
BowelProblem	*true*	*false*
FamilyOut	*true*	*false*
DogOut	*true*	*false*
HearBark	*true*	*false*
LightOn	*true*	*false*

APÊNDICE C – REDE *CARDIAGNOSTIC*



Variáveis	Estados		
MainFuse	*okay*	*blown*	
BatteryAge	*new*	*old*	*veryOld*
Alternator	*okay*	*faulty*	
Distributer	*okay*	*faulty*	
StarterMotor	*okay*	*faulty*	
SparkPlugs	*okay*	*tooWide*	*fouled*
AirFilter	*clean*	*dirty*	
GasTank	*hasGas*	*empty*	
GasFilter	*clean*	*dirty*	
AirSystem	*okay*	*faulty*	
SparTiming	*good*	*bad*	*veryBad*
FuelSystem	*good*	*poor*	*faulty*
ChargingSystem	*okay*	*faulty*	
BatteryVoltage	*strong*	*weak*	*dead*
HeadLights	*bright*	*dim*	*off*
VoltageAtPlug	*strong*	*weak*	*none*
SparkQuality	*good*	*bad*	*veryBad*
StarterSystem	*okay*	*faulty*	
CarCranks	*true*	*false*	
CarStarts	*true*	*false*	