Raciocínio Probabilístico em Sistemas Embarcados

Jaime Shinsuke Ide, Fabio Gagliardi Cozman

Escola Politécnica, Universidade de São Paulo
Av. Prof. Mello Moraes, 2231, 05508-900, São Paulo, SP – Brazil
jaime.ide@poli.usp.br, fgcozman@usp.br

Abstract: *This paper presents techniques for probabilistic reasoning (particularly reasoning based on Bayesian networks) in embedded systems located inside machines, vehicles, domestic and medical equipment, security and surveillance devices. The goal of the work is to create tools that empower any embedded device, however small, with the ability to deal with uncertainty. An analysis of existing inference algorithms lead to the use of approximate sampling methods, as these methods employ relatively little memory – an essential condition in small embedded devices. Inference algorithms for Bayesian networks have been implemented in a board; this paper describes these algorithms, the implementation and tests with the embedded board.*

Resumo: *Este artigo apresenta técnicas que permitem levar raciocínio probabilístico (em particular raciocínio realizado através de redes Bayesianas) a sistemas embarcados em máquinas, veículos, equipamentos domésticos, hospitalares. O objetivo é criar ferramentas que permitam qualquer computador embarcado manipular incertezas. Uma análise dos algoritmos existentes para inferências com redes Bayesianas levou à escolha de métodos aproximados por amostragem, já que estes métodos utilizam pouca memória – condição essencial em sistemas embarcados de pequeno porte. Algoritmos para inferência em redes Bayesianas foram implementados numa placa; o artigo descreve os algoritmos, a implementação e os testes com esse dispositivo.*

1. Introdução

Técnicas de inteligência artificial são aplicadas nos mais diversos domínios, indo desde a construção de sistemas voltados a realizar uma única tarefa repetitiva até complexas tentativas de simular inteligência humana [Russell 1995]. Entre estes extremos estão vários sistemas de decisão que usam técnicas de busca e raciocínio lógico para analisar alternativas. Em geral, sistemas que usam inteligência artificial requerem considerável poder de manipulação algébrica e simbólica. Sistemas de inteligência artificial são em geral baseados em computadores de boa capacidade computacional. Mesmo quando tais sistemas de decisão e controle são embarcados – por exemplo em robôs móveis – os computadores utilizados possuem elevada capacidade computacional, sendo equipados com sistemas operacionais complexos e pesados [Kortenkamp 1998].

Uma das atividades mais complexas realizadas por inteligências (artificiais ou não) é a manipulação de incertezas. Técnicas probabilísticas tem sido utilizadas com crescente sucesso em inteligência artificial, para lidar com todas as situações em que medidas de sensores são imprecisas ou informações são vagas ou incompletas [Thrun 2000]. Um dos aspectos importantes do raciocínio probabilístico é a complexidade dos métodos de inferência. Problemas de inferência probabilística são em geral NP-

completos [Cooper 1997], o que evidencia a dificuldade das possíveis soluções. Além desse aspecto teórico, a manipulação de modelos probabilísticos geralmente leva a grande consumo de memória. Mesmo modelos que visam reduzir o esforço computacional de inferência, como redes Bayesianas, ainda possuem elevada complexidade.

Devido à complexidade envolvida em raciocínio probabilístico, é ainda difícil imaginar a manipulação inteligente de incertezas em sistemas embarcados genéricos. Uma das dificuldades básicas em sistemas embarcados é a limitação de memória (tanto memória permanente quanto memória de trabalho) [Ramos-Mikami 2000]. No entanto, há razões para considerar que esta situação deve mudar em um futuro relativamente próximo [Muslim 1995]. A crescente miniaturização de componentes tem levado a um avanço constante na capacidade de processamento de sistemas embarcados. O mercado de sistemas embarcados e ferramentas relacionadas têm crescido de forma explosiva e levado a grandes investimentos no setor. Mas ao mesmo tempo em que esse avanço tem ocorrido, ainda persiste uma falta de ferramentas de inteligência artificial voltadas a sistemas embarcados – em especial, ferramentas que possam manipular incertezas.

O trabalho descrito nesse artigo visa preencher essa lacuna através do desenvolvimento de técnicas que permitam levar raciocínio probabilístico a sistemas embarcados de pequeno porte. O modelo de redes Bayesianas é adotado como representação eficiente para modelos probabilísticos (Seção 2). O principal desafio enfrentado é: como realizar inferência de forma a utilizar pouca memória? A Seção 3 faz uma análise dos algoritmos existentes e indica que algoritmos aproximados baseados em amostragem são apropriados aos objetivos do trabalho. A Seção 4 descreve a implementação do algoritmo de Gibbs sampling como um pacote de código em linguagem Java destinado a uma placa embarcada. Esta placa possui todas as características associadas a sistemas embarcados: tamanho reduzido, baixo consumo, pouca memória, ligação por rede, e recursos para controle em tempo real. A Seção 4 também descreve testes realizados com os algoritmos desenvolvidos e com a placa embarcada.

Este artigo descreve uma etapa dentro de um objetivo maior, o objetivo de levar técnicas de inteligência artificial a dispositivos que estejam próximos ao quotidiano da população. Este objetivo só será alcançado quando as principais técnicas de inteligência artificial forem adaptadas a sistemas com baixa memória. Este artigo apresenta uma solução nesse sentido, para o caso de raciocínio probabilístico.

2. Redes Bayesianas

Redes Bayesianas são representações compactas e eficientes para um conjunto de distribuições de probabilidade [Pearl 1988]. Uma rede Bayesiana representa um modelo estatístico complexo num pequeno número de valores probabilísticos. Para tanto, toda rede Bayesiana representa uma única distribuição conjunta. A regra de formação dessa distribuição é simples. Considere um grafo direcionado acíclico, onde cada nó é associado a uma variável X_i. Denote por pa$\{X_i\}$ o conjunto de variáveis que são pais diretos da variável X_i na rede. Cada variável X_i é associada a uma distribuição p(X_i | pa$\{X_i\}$) (uma variável sem pais é associada à distribuição marginal p(X_i)).

A propriedade básica em redes Bayesianas é que toda variável X_i é independente de todos os antecessores de X_i, condicional nos pais de X_i. Isto garante que a distribuição conjunta seja definida por [Pearl 1988]:

$$p(X) = \prod_i p(X_i \mid pa\{X_i\})$$

Essa expressão é importante por garantir que uma distribuição conjunta, potencialmente complexa, possa ser representada através do produto de blocos menores.

3. Algoritmos para inferências em redes Bayesianas

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

$$p(X_q|\mathbf{E}) = p(X_q,X_4, X_6) / p(X_4, X_6)$$

A expressão geral para inferências em redes Bayesianas é:

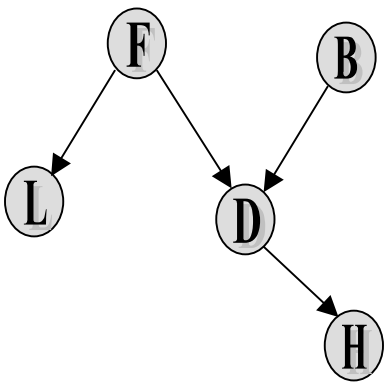
$$p(X_q|\mathbf{E}) = \sum_{\{X/X_q,\mathbf{E}\}} [\ \prod_i p(X_i/pa\{X_i\}\ ] \Big/ \sum_{\{X/\mathbf{E}\}} [\ \prod_i p(X_i/pa\{X_i\}\ ]. \qquad (1)$$

Existem algoritmos eficientes para realizar estas operações em redes Bayesianas genéricas (redes com representação por "poly-trees" admitem algoritmos específicos de alta eficiência [Pearl 1988]). Dois tipos de algoritmos são comumente utilizados: algoritmos baseados em "junction trees" [Jensen 90] ou algoritmos baseados em eliminação de variáveis [Cannings 1978], [Shafer-Shenoy 1990], [Zhang 1996]. Uma apresentação compacta de eliminação de variáveis está em [Cozman 2000]. Todos esses algoritmos são exatos e exigem muita memória, pois lidam com resultados intermediários que podem ser exponenciais no tamanho do maior clique existente no grafo da rede triangulado [Dechter 1996]. Isto faz com que algoritmos exatos sejam potencialmente intratáveis.

O caminho para realizar inferências em sistemas embarcados é utilizar aproximações. As aproximações que ocupam menos memória são baseadas em amostragem. A proposta do método Monte Carlo é que se calcule o valor das integrais da equação(1) numericamente, ou seja, cálculos aproximados, através de amostras geradas em seu domínio de integração [Sobol 1994]. Em redes Bayesianas, a probabilidade $p(X_q|\mathbf{E})$ é dada por uma somatória com n-dimensões, onde n é o número de nós. O problema do método de Monte Carlo é como gerar estas amostras de $p(X_q|E)$. Para tanto, uma idéia conhecida como "logic sampling" é a seguinte. Deve-se ordenar os nós da rede, de maneira que os nós sempre estejam precedidos de seus pais, e amostrá-los em seqüência. Assim, começando com X_1 (um nó sem pais), encontra-se uma amostra de X_1 a partir de $p(X_1)$. Depois, seguindo para X_2 e encontra-se uma amostra de X_2. E assim sucessivamente, obtêm-se uma amostra (X_1, X_2,..., X_n). O passo a seguir é verificar se a amostra é consistente com as evidências. Se a evidência não é representada na amostra, descarta-se esta amostra; caso a evidência seja consistente com a amostra, retém-se a amostra. Parte-se então para uma nova amostragem, até que se obtenha uma que seja consistente. O problema do "logic samplig" é que: quanto maior for o número de evidências, mais amostras devem ser geradas para que uma amostra

seja aceita. Se uma das evidências tem pouca probabilidade para ocorrer, muitas amostras terão que ser geradas até que consiga uma que seja consistente.

Este tipo de problema é solucionado por métodos aproximados conhecidos como métodos MCMC (Monte Carlo Markov Chain) [Gilks 1996], [Gamerman 1997]. Esses métodos se baseiam na aplicação de técnicas de Monte Carlo em uma cadeia de Markov especialmente construída para simular o modelo de interesse. Vários métodos utilizam essa abordagem; entre esses métodos Gibbs sampling é um método de implementação relativamente simples em redes Bayesianas. Gibbs sampling é um caso específico do algoritmo Metropolis-Hastings [Metropolis et al. 1953], e foi proposto inicialmente por Geman e Geman [Geman-Geman 1984]. A idéia básica do Gibbs sampling é utilizar a *probabilidade condicional total* (*pct*) de cada variável para gerar as amostras. O *pct* representa a probabilidade de uma variável da rede assumir um certo valor, dado o estado de todas as demais variáveis. Um exemplo clássico [Charniak 1991], o *Dog Problem*, pode ilustrar o algoritmo. A seguinte rede Bayesiana (*Figura 1*) representa uma situação do cotidiano: queremos visitar um amigo, e queremos saber qual a chance de sua família estar em casa. Se a família estiver em casa há uma certa possibilidade de que a luz esteja acesa e de que o cachorro esteja vigiando a casa. Mas, se o cachorro estiver doente, isto pode influenciar o fato do cachorro estar em vigia. E, se o cachorro está vigiando, existe uma possibilidade de que ele esteja latindo.



A *probabilidade condicional total* (*pct*) para cada variável é dada por:

- *F*: *family-out*: $pct_F = P(F).P(L/F).P(D/F,B)$
- *L*: *light-on*: $pct_L = P(L/F)$
- *D*: *dog-out*: $pct_D = P(D/F,B).P(H/D)$
- *B*: *bowel-problem*: $pct_B = P(B).P(D/F,B)$
- *H*: *hear-bark*: $pct_H = P(H/D)$

Figura 1. Representação do Dog Problem

Podemos querer saber qual é a probabilidade de que a família não esteja em casa (*F=true*) se a luz está apagada (*L=false*) e o cachorro está latindo (*H=true*). Para iniciar as simulações do Gibbs sampling, atribuímos um estado inicial qualquer para a rede Bayesiana constituída por F, L, D, B e H. Por exemplo, todas valendo *true*. As variáveis em evidências são fixas e, portanto, não se geram amostras de *L* e de *H*. Escolhendo por exemplo a variável D, calcula-se o seu pct_D, que é comparado com um número aleatório entre 0 e 1, gerando uma amostra para *D*. Assim, percorre-se todas as variáveis da rede, geram-se inúmeros estados para a rede. Esse processo é então repetido por várias iterações. Esta técnica não gera inicialmente a amostra correta, mas faz com que as várias iterações levem à convergência, para as distribuições corretas.

Uma das vantagens de algoritmos MCMC é que estes algoritmos podem ser codificados de forma bastante compacta e podem ocupar reduzidas porções de memória. Os sistemas existentes hoje para redes Bayesianas possuem versões sem interface gráfica, mas a porção de memória ocupada por tais sistemas em geral é alta e não se

adapta a sistemas embarcados. O desafio é portanto implementar Gibbs sampling para redes Bayesianas de uma forma que seja adequado para sistemas embarcados. O restante deste artigo descreve uma implementação especialmente feita com esse objetivo, e os testes realizados em uma placa de processamento embarcado.

4. Sistemas embarcados com inteligência artificial: Implementação e teste de Gibbs sampling para redes Bayesianas em uma placa processadora embarcada

4.1. Sistema Embarcado aJ-PC104

Com o objetivo de validar nossa proposta, o algoritmo Gibbs sampling foi implementado numa placa embarcada; o projeto do programa enfatizou a otimização no uso de memória. O algoritmo e implementação foram testados em uma placa embarcada, a aJ-PC104. O sistema embarcado produzido pela empresa aJile, o aJ-PC104 Ethernet-enabled, torna viável e real o emprego da linguagem Java nestes tipos de aplicações. Trazendo ao sistema embarcado todas as vantagens apresentadas pela linguagem Java. Baseado no microprocessador Java, JEM2 direct-execution, o aJ-PC104 proporciona uma eficiente plataforma para desenvolvimento de aplicações em tempo real inteiramente em Java. A placa inova ao processar diretamente as instruções de máquina da linguagem Java, sem necessidade de interpretadores. Este sistema embarcado suporta o popular protocolo de via de dados PC/104, permitindo ao projetista controlar uma larga variedade de periféricos com comunicação PC/104 [Ngoc-Hardin 1999].

A linguagem Java foi escolhida para este trabalho por ser uma linguagem com características apropriadas para sistemas embarcados. Nesses sistemas, modelos centralizados estão sendo substituídos por modelos envolvendo trabalho em rede com certo nível de inteligência e de autonomia embarcada [Ngoc-Hardin 1999]. Ou seja, existe uma tendência atual de descentralização do processamento, potencializado pelos padrões de comunicação da Internet [Wang 1998]. A portabilidade da linguagem Java e sua similaridade com a linguagem C, faz dela ideal para o uso em comunicação via Internet. E o atual desenvolvimento de hardware, como por exemplo a placa aJ-PC104, torna possível à linguagem Java avançar além das plataformas UNIX/WINDOWS e ser inserida em sistemas embarcados [Marsh 2001].

4.2. Implementação do Gibbs sampling

O algoritmo Gibbs sampling foi implementado, adicionando-se um método a um programa já existente, o JavaBayes, o qual está disponível na Internet em http://www.cs.cmu.edu/~Javabayes. JavaBayes é uma implementação de algoritmos de eliminação de variáveis em redes Bayesianas; o programa é distribuído livremente através de licença GNU.

O programa Gibbs sampling possui a seguinte estrutura*:*

0. Gerar um estado inicial qualquer para as variáveis da rede.

1. Entrar com as ***n*** *simulações a serem realizadas. Repetir* ***n*** *vezes os procedimentos abaixo.*

2. Percorrer cada uma das variáveis.

3. Se a variável não estiver observada, fazer os procedimentos abaixo.

*4. Calcular a **pct** da variável, resultante de uma produtória. Normalizar o vetor de probabilidades, ao final.*

*5. Gerar uma amostra para a variável, gerando-se um número aleatório (entre 0 e 1) e comparando-o com a **pct** normalizada, obtida anteriormente.*

6. Se a variável amostrada anteriormente assumir o valor desejado, adicionar ao contador $\boldsymbol{n_x}$.

7. A probabilidade desejada é dada pela razão: $\boldsymbol{n_x / n}$.

A boa qualidade dos números aleatórios gerados constitui o segredo para a eficiência destes algoritmos de simulação. No Gibbs sampling implementado, empregou-se um eficiente gerador de números aleatórios, o *MersenneTwister* [Matsumoto 98], que também se encontra disponível na Internet.

Maiores detalhes de implementação estão descritas na referência [Ide 2001].

4.3. Testes realizados

Os testes foram realizados no exemplo Dog Problem, apresentado na Seção 3. Este exemplo é um dos exemplos contidos no pacote do programa JavaBayes. Os resultados obtidos no programa Gibbs sampling(***Gibbs***) são apresentadas na *Tabela 1*, juntamente com os resultados obtidos no JavaBayes, que realiza cálculos exatos. A probabilidade P(*F*/*L*,*H*) é a chance de que a família esteja fora de casa, sendo que a luz está apagada e se ouvem latidos de cachorro.

F	**L**	**H**	**P(F=*true*/L=*false*,H=*true*) (*Java*Bayes)**	***Gibbs* n=100**	***Gibbs* n=1000**	***Gibbs* n=10.000**	***Gibbs* N=20.000**
true	*true*	*true*	0,8578	0,88	0,866	0,8572	0,86065
true	*true*	*false*	0,5006	0,67	0,513	0,5103	0,50864
true	*false*	*true*	0,1748	0,25	0,194	0,1798	0,17489
true	*false*	*false*	0,034	0,03	0,044	0,0368	0,03367

Tabela 1. Resultados do programa Gibbs sampling

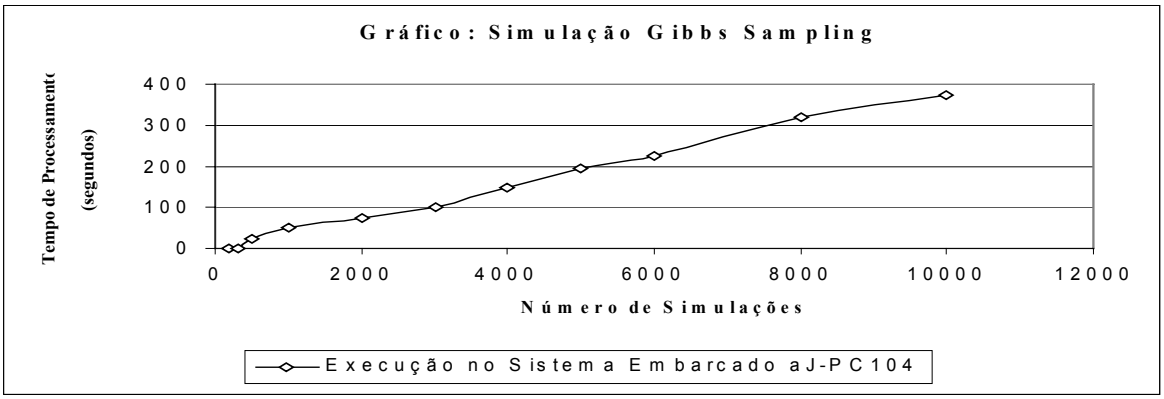


Gráfico 1. Desempenho do algoritmo no Sistema Embarcado aJ-PC104

No *Gráfico 1*, está plotado o tempo gasto no aJ-PC104 para se efetuar as simulações.

4.4. Análise dos resultados

Plotando-se alguns dos resultados da *Tabela 1*, temos o seguinte gráfico:

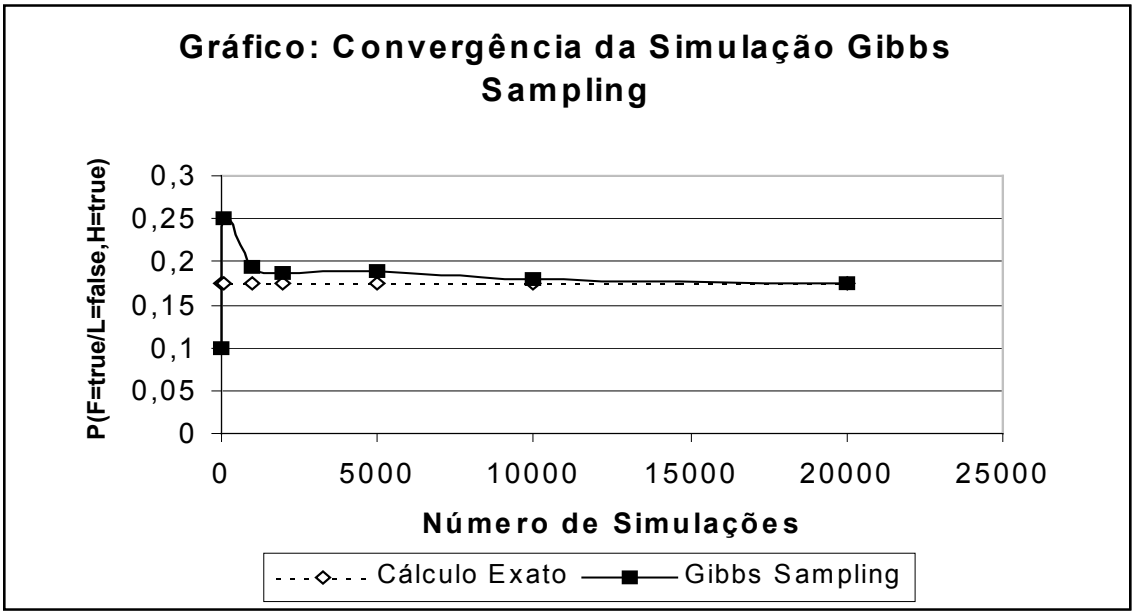


Gráfico 2. Comportamento da Convergência

Os resultados obtidos foram bastante satisfatórios. Para n=2000 simulações, tem-se um desvio de 6%. Para n>10.000 simulações, tem-se desvios menores que 3%. É possível concluir , comparando-se com outros testes realizados que o algoritmo *Gibbs sampling*, perde eficiência para probabilidades "limites", muito próximos de 0 ou de 1.

O tempo gasto para processamento no sistema embarcado (*Gráfico 1*:) é maior que num PC (100Mhz). Isto se deve aos recursos limitados do atual sistema (40Mhz de velocidade e 1Mbytes de memória RAM). É interessante notar ainda neste gráfico, que o tempo de processamento varia linearmente com o número de simulações. Ou seja, o crescimento do tempo necessário de processamento varia linearmente de acordo com o número de iterações e independentemente da estrutura da rede Bayesiana, diferentemente do crescimento exponencial dos cálculos exatos.

5. Conclusão

Para julgar os resultados obtidos, deve-se considerar que a implementação de Gibbs sampling que roda na placa aJ-PC104 ocupa apenas 4Kbytes de memória (código executável), mais um espaço fixo para as variáveis de programa. Toda a ocupação de memória pode ser prevista no início da execução, e portanto um sistema embarcado de baixa velocidade e pouca memória (como o sistema por nós utilizado) pode empregar o algoritmo implementado sem dificuldades. Esse resultado abre a possibilidade de utilizar raciocínio probabilístico em sistemas embarcados de pequeno porte, como discutido na Seção 1. Como trabalho futuro, planejamos otimizar o código para obter maior velocidade, testar o algoritmo em placas mais velozes, e realizar aplicações práticas em detecção e análise de falhas e confiabilidade.

Referências

[Cannings 1978] C. Cannings and E. A. Thompson and M. H. Skolnick. Probability Functions in Complex Pedigrees, in *Advances in Applied Probabilit,vol.10,* pp.26-61, 1978.

[Charniak 1991] E. Charniak K., Bayesian networks wituout tears, AI magazine, pp.50-63, 1991.

[Cooper 1997] G. F. Cooper. *The Computational Complexity of Probabilistic Inference using Bayesian Belief Networks*, AI journal, vol.42, pp. 393-405, Oct., 1997.

[Cozman 2000] F. G. Cozman. Generalizing Variable Elimination in Bayesian Networks, In *Proceedings of the IBERAMIA/SBIA 2000 Workshops*, pp. 27-32, Tec Art Editora, São Paulo, 2000.

[Dechter 1996] R. Dechter. Bucket Elimination: A unifying framework for probabilistic inference, in proceedings *XII Uncertainty in Artificial Intelligence Conference,* pp.*211-219,* Morgan Kaufmann, California, 1996.

[Gamerman 1997] Dani Gamerman. *Markov Chain Monte Carlo, Stochastic Simulation for Bayesian Inference*, Chapman and Hall, London, 1997.

[Geman-Geman 1984] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. Pattn. Anal. March. Intel., 6*, pp.721-741, 1984.

[Gilks 1996]. W.R. Gilks, S. Richardson and D.J. Spiegelhalter. *Introducing Markov Chain Monte Carlo*, pp. 1-16, Chapman and Hall, 1996.

[Ide 2001] J.S. Ide, *Implementação de algoritmos de Simulação Monte Carlo para processamento estatístico em sistemas embarcados*, Escola Politécnica, Universidade de São Paulo (publicação interna), 2001.

[Jensen 1990] F. V. Jensen. *An algebra of Bayesian belief universes for knowledge-based systems,* Networks 20, pp.637-659, 1990.

[Kortenkamp 1998] D. Kortenkamp, R. P. Bonasso and R. Murphy (editors), *AI-based Mobile Robots: Case Studies of Successful Robot Systems*, MIT Press, Cambridge, Massachussets, 1998.

[Marsh 2001] D. Marsh. Silicon variety vanquishes embedded-*Java* taboos, *EDN Europe, February, 2001, pp. 71-82.* www.ednmag.com. 2001.

[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, January 1998, pp 3-30. <http://www.math.keio.ac.jp/matumoto/emt.html >.

[Metropolis et al. 1953] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller. Equations of state calculations by fast computing machine. J. Chem. Phys.,21, pp.1087-1091, 1953.

[Musliner 1995] D. J. Musliner, J. A. Hendler, A. K. Agrawala, E. H. Durfee, J. K. Strosnider, and C. J. Paul, The Challenges of Real-Time AI, *IEEE Computer, Vol 28 #1, January 1995.*

[Ngoc-Hardin 1999] D. L. Ngoc e D. Hardin, aJile Systems Inc., www.ajile.com, *Low-power, Direct Java Execution for Real-Tme Networked Embedded Aplications,* 1999.

[Pearl 1988]. J.Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kauffman, San Mateo, California, 1998.

[Ramos-Mikami 2000] F. T. Ramos, F. Mikami, F. G. Cozman. Implementação de redes Bayesianas em Sistemas Embarcados. In *Proceedings of the IBERAMIA/SBIA 2000 Workshops*, pp. 65-69, Tec Art Editora, São Paulo, 2000.

[Russel 1995] S. J. Russell and P. Norvig. *Artificial Intelligence: a Modern Approach*, Upper Saddle River, Prentice Hall, New Jersey, 1995.

[Shafer-Shenoy 1990] Glenn Shafer and Prakash Pundalik Shenoy. Probability Propagation, *Annals of Mathematics and Artificial Intelligence, vol.2*, pp.327-352, 1990.

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

[Thrun2000] S. Thrun. *Probabilistic Algorithms in Robotics*. AI Magazine, 21(4):93--109, 2000.

[Wang 1998] C. B. Wang: Fundador da CA, *Techno Vision II,* MAKRON Books, 1998.

[Zhang 1996] N. L. Zhang and D. Poole. Exploiting Causal Independence in Bayesian network Inference, *Journal of Artificial Intelligence Research*, pp.301-328, Nov.,1996.