CENTRO UNIVERSITÁRIO FEEVALE

FELIPE SCUCIATTO DOS SANTOS

DESENVOLVIMENTO DE MÓDULO DE REDES BAYESIANAS PARA O AMPLIA

Novo Hamburgo

2009

FELIPE SCUCIATTO DOS SANTOS
scuciatto@feevale.br

DESENVOLVIMENTO DE MÓDULO DE REDES BAYESIANAS PARA O AMPLIA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pelo Centro Universitário Feevale

Orientador: Marta Rosecler Bez
Co-orientador: Cecília Dias Flores

Novo Hamburgo
2009

FELIPE SCUCIATTO DOS SANTOS

Trabalho de Conclusão do Curso de Ciência da Computação, com o título de DESENVOLVIMENTO DO MÓDULO DE REDES BAYESIANAS PARA O AMPLIA, submetido ao corpo docente do Centro Universitário Feevale, como requisito necessário para obtenção do Grau Bacharel em Ciência da Computação.

Aprovado por:

Marta Rosecler Bez

Fabian Viegas(Banca examinadora)

Gilda Aparecida de Assis(Banca examinadora)

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho, em especial:

Aos meus pais por todo apoio, incentivo e confiança que sempre me foi dado.

A minha namorada, por me dar o incentivo e compreensão necessários permanecendo ao meu lado durante todo este trabalho.

A minha orientadora e co-orientadora, por todo conhecimento que me foi passado.

RESUMO

O software AMPLIA foi desenvolvido pelo Grupo de Inteligência Artificial do Instituto de Informática da Universidade Federal do Rio Grande do Sul (II/UFRGS), em parceria com outras instituições. O AMPLIA é um Ambiente Multiagente Probabilístico Inteligente de Aprendizagem, que possibilita a aprendizagem colaborativa. Dentro do ambiente, o aluno demonstra sua hipótese médica através de um modelo gráfico, construindo uma Rede Bayesiana. Este modelo é baseado em casos clínicos reais e, posteriormente, comparado com modelos criados por especialistas. Visando sanar algumas limitações do projeto inicial, foi proposto um novo projeto, com objetivo de reescrever o AMPLIA, tendo agora a Internet como meio de colaboração, possibilitando assim a interação entre várias instituições. Com este trabalho, será desenvolvido o módulo para criação, inferência e avaliação das Redes Bayesianas, dentro do AMPLIA.

Palavras-chave: Inteligência Artificial, Redes Bayesianas, Ensino da Medicina.

ABSTRACT

The AMPLIA software was developed by the Group of Artificial Intelligence of the Institute of Informatics from the Universidade Federal do Rio Grande do Sul (II / UFRGS), in partnership with other institutions. AMPLIA is a Multi-agent Probabilistic Intelligent Learning Environment, which provides collaborative learning. Within the environment, the student demonstrates his medical hypothesis through a graphical model, building a Bayesian Network. This model is based on actual cases and thereafter compared with models created by specialists. Aiming to address some limitations of the initial project, new project was proposed, aiming to rewrite the AMPLIA taking now Internet as a way of collaboration, tenabling the collaboration between several institutions. In this work, will be developed the module for creation, inference and evaluation of Bayesian networks within the AMPLIA software.

Keywords: Artificial Intelligence, Bayesian Networks, Medicine Teaching.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AMPLIA	Ambiente Multiagente Probabilístico Inteligente de Aprendizagem
API	Application Programming Interface
FACIL	FIPA-ACL Interface Library
CVS	Concurrent Version System
FOL	First Order Logic
IDE	Integrated Development Environment
JADE	Java Agent Development Framework
LAN	Local Area Network
MEBN	Multi-Entity Bayesian Network
SVN	Subversion
UFCSPA	Universidade Federal de Ciências da Saúde de Porto Alegre
UFRGS	Universidade Federal do Rio Grande do Sul

SUMÁRIO

INTRODUÇÃO

O AMPLIA[1] é um ambiente computacional multiagente, que objetiva apoiar o desenvolvimento do raciocínio diagnóstico e a modelagem de hipóteses, como um recurso adicional para a formação de novos médicos. No Amplia, o conhecimento é modelado através de Redes Bayesianas, representando assim, de forma gráfica e probabilística, as hipóteses levantadas pelo estudante.

As Redes Bayesianas são uma forma gráfica de representação das relações entre variáveis e suas probabilidades dentro de um escopo. Elas são representadas por grafos acíclicos, nos quais cada nó é uma variável aleatória (CHARNIAK, 1991). Cada variável deve possuir um conjunto limitado de valores (estados) e a cada nó raiz da rede, deve ser atribuída uma probabilidade. As probabilidades dos nós não raiz devem levar em conta as probabilidades dos pais.

Segundo Flores (2003), a abordagem de Redes Bayesianas foi escolhida para tratar o conhecimento incerto por sua rigorosa fundamentação em princípios matemáticos. A abordagem Bayesiana também permite lidar com incertezas, representada por probabilidades, isto é, a probabilidade de variáveis assumir valores específicos, dadas as evidências disponíveis. Ainda segundo a mesma autora, outra razão importante para a escolha desta abordagem é que a mesma permite a modelagem qualitativa e quantitativa do domínio. O enfoque qualitativo é representado pelo conjunto de variáveis e seu relacionamento causal, enquanto o enfoque quantitativo expressa a intensidade desse relacionamento.

No AMPLIA, o aluno desenvolve uma rede bayesiana através do seu raciocínio lógico por meio de um caso clínico apresentado. A rede construída pelo aluno é então comparada com a rede construída pelo especialista, que deve identificar os prováveis conflitos. O resultado gerado é então utilizado nas estratégias pedagógicas mais convenientes para auxílio ao diagnóstico correto (FLORES, 2005).

Observando as limitações do projeto original, um novo projeto foi iniciado pelo grupo de pesquisas da UFCSPA, com intuito de utilizar tecnologias e recursos de Internet dentro do AMPLIA, visando principalmente a troca de conhecimento entre várias instituições. Esta nova versão mantém todas as funcionalidades já implementadas na versão inicial,

[1] Ambiente Multiagente Probabilístico Inteligente de Aprendizagem

porém, o software será reescrito utilizando a linguagem Java, facilitando assim, a comunicação através de Web e seu uso em diversas plataformas.

Este trabalho tem como principal objetivo implementar o módulo de desenvolvimento e avaliação de Redes Bayesianas dentro do AMPLIA. Para atingir tal objetivo, técnicas de criação, avaliação e inferência de Redes Bayesianas serão estudadas. Este trabalho utiliza como base o software UnBBayes[2], utilizado para modelar e avaliar Redes Bayesianas.

Este trabalho está dividido da seguinte forma: no capítulo um, é apresentada a teoria sobre Redes Bayesianas, bem como as fundamentações matemáticas utilizadas pela técnica. No segundo capítulo, os principais algoritmos de inferência sobre redes são apresentados. O capítulo três apresenta o software AMPLIA, mostrando as características e a arquitetura da versão atual. O capítulo quatro aborda o UnBBayes, que teve suas funcionalidades extendidas para adaptação ao AMPLIA. No capítulo 5 toda a implementação das modificações do editor UnBBayes é apresetada, seguido das conclusões.

[2]Disponível em: http://unbbayes.sourceforge.net/

1 REDES BAYESIANAS

As Redes Bayesianas são uma forma gráfica de representação das relações entre variáveis dentro de um escopo. É uma representação das probabilidades de um problema. As Redes Bayesianas são representadas por grafos acíclicos, onde cada nó é uma variável aleatória (CHARNIAK, 1991). Utilizando-se Redes Bayesianas, pode-se calcular a probabilidade condicional de cada variável em um problema, apenas observando valores de variáveis próximas.

1.1 Teoria da Probabilidade

A Teoria das Probabilidades é o estudo matemático das probabilidades. Por décadas, tem-se usado métodos probabilísticos como base de inferência em sistemas de Inteligência Artificial. Normalmente estes métodos são aplicados a problemas complexos, onde uma característica de determinada entidade tem influência direta sobre outra característica de outra entidade. Por exemplo, a presença ou ausência de uma determinada doença em um ser humano, tem influência direta em um exame que verifica a presença da mesma (NEAPOLITAN, 2003).

A principal vantagem de se trabalhar com métodos probabilísticos é poder lidar com incertezas, podendo assim tomar decisões mesmo com um número reduzido de evidências. Ao se trabalhar com dados incertos, precisa-se utilizar níveis de certeza, e não apenas valores absolutos. Por exemplo, a probabilidade de acertar um número ao se jogar um dado é de aproximadamente 0,2 (20%). Ou então, qual a probabilidade de um paciente fumante apresentar ou não câncer de pulmão?

Segundo Onishi (2003), a comunidade científica tem manifestado, grande aceitação e interesse, pelo uso da teoria da probabilidade como ferramenta para manipulação de incerteza, principalmente através do uso de Redes Probabilísticas.

Neste capítulo serão abordados os pilares da teoria da probabilidade, que servirão de

base para assuntos subsequentes. Nos capítulos seguintes estes conceitos serão aplicados no desenvolvimento de um sistema inteligente.

1.1.1 Cálculo de probabilidades: o axioma base, trabalhando com probabilidades incondicionais

O axioma base da teoria de probabilidades demonstra o cálculo de probabilidades incondicionais: para qualquer evento a dado que $0 \leq P(a) \geq 1P(a) = 1$ se e apenas se (a) ocorrer com certeza.

Simplificando o axioma acima, a probabilidade de um evento qualquer é representada por um número dentro do intervalo [0,1]. A probabilidade é igual a 1 apenas se o grau de certeza for igual a 100%.

Extendendo o axioma anterior, passa-se a trabalhar com mais eventos: para quaisquer eventos a e b, mutuamente exclusivos, a probabilidade de a ou b ocorrer é:

$$P(a \vee b) = P(a) + P(b) \tag{1.1}$$

Simplificando, a probabilidade de um evento a ou b ocorrer é igual a soma de suas probabilidades.

Exemplificando, considere que a variável aleatória *câncer* denota a possibilidade de um paciente qualquer apresentar câncer. Com isso ao dizer que P(câncer) = 0,01, informa-se que a probabilidade deste paciente ter câncer é de 1%, ignorando qualquer outra evidência que possa existir.

Nos exemplos anteriores foram demonstradas probabilidades de um evento, sem que nenhuma outra informação tenha sido observada. Casos como estes são conhecidos por *probabilidade a priori*. Estes casos só ocorrem quando não existem novas informações dentro do escopo, caso contrário, utiliza-se a probabilidade condicional.

1.1.2 Probabilidade Condicional

Sempre que uma nova evidência sobre alguma variável aleatória é obtida, a probabilidade atual é condicionada à essa nova evidência. A probabilidade condicional demonstra como um evento se comporta dada determina evidência. Exemplificando esta situação, pode-se pensar no seguinte exemplo: qual a probabilidade de um paciente possuir câncer

dado o fato que ele é fumante?

Segundo Moore (2005), a probabilidade condicional de um evento A ocorrer, sendo que P(A) $>$ 0 é definida por:

$$P(A|B) = \frac{P(A \wedge B)}{P(A)} \tag{1.2}$$

Os eventos A e B ocorrem de forma distinta, sendo A a probabilidade que está sendo calculada e B a informação repassada.

1.1.3 O Teorema de Bayes

O Teorema de Bayes foi criado pelo reverendo Thomas Bayes, sendo publicado póstumamente em 1764 (ALDRICH, 2008). O teorema é a base para a Inferência Bayesiana. No teorema, Bayes demonstra como alterar probabilidades considerando novas evidências, e assim, obtendo novas probabilidades. A seguir, o teorema é demonstrado (NEAPOLITAN, 2003)

Dados dois eventos E e F, sendo que P(E) $\neq$ 0 e P(F) $\neq$ 0, tem-se:

$$P(E|F) = \frac{P(F|E)P(E)}{P(F)} \tag{1.3}$$

Onde:

- P(E|F) significa a probabilidade do evento E ocorrer dado que F ocorreu, é chamada de probabilidade posterior.
- P(E) é a probabilidade *a priori* do evento E ocorrer. Esta probabilidade foi calculada previamente, antes da evidência (ou evento) F ocorrer.
- P(F) é a probabilidade marginal, ou seja, a probabilidade de testemunharmos o evento F. A probabilidade marginal pode ser calculada através da soma e produto de todas as probabilidades mutualmente exclusivas de um determinado conjunto: $P(F) = \sum i = 1 P(F|H_i) + P(H_i)$

1.1.4 Inferência Bayesiana

Inferência Bayesiana é um método estatístico de inferência. Este método é utilizado quando precisa-se atualizar ou inferir uma probabilidade de determinado evento. Como

citado anteriormente, a inferência Bayesiana faz uso constante do Teorema de Bayes.

Realizar este tipo de inferência consiste basicamente em aplicar o Teorema de Bayes para descobrir determinada probabilidade. Deve-se identificar possíveis evidências, e estimar suas probabilidades iniciais (*a priori*).

Exemplificando a inferência Bayesiana, observe o seguinte caso, extraído de Neapolitan (2003):

Suponha que Joe precisa fazer um exame de Raio-X rotineiro para admissão em um novo emprego. Ao ver o resultado, Joe entra em pânico ao descobrir que seu exame é positivo para câncer de pulmão. Mas qual será a real probabilidade de Joe possuir câncer?

Sabe-se que o exame não é totalmente absoluto. É conhecida uma taxa de 60% de acertos, também sabe-se que em 2% dos casos o exame aponta um falso positivo. Para descobrir qual a real probabilidade de Joe possuir câncer de pulmão, recorremos ao Teorema de Bayes.

Uma busca mais apurada de informações mostrou a Joe que apenas um em cada mil funcionários possui câncer de pulmão. Logo, a probabilidade de se ter cancer (independente de teste) é de 1%. Com essas informações apuradas, aplica-se o Teorema de Bayes:

$$P(cancer|positivo) = \frac{P(pos|cancer)P(cancer)}{P(pos|cancer)P(cancer) + P(pos|sadio)P(sadio)} \tag{1.4}$$

$$P(cancer|positivo) = \frac{P(0,6)P(0,001)}{P(0,6)P(0,001) + P(0,02)P(0,99)} \tag{1.5}$$

$$P(cancer|positivo) = 0,23 \tag{1.6}$$

- onde "pos" indica positivo.

Conclui-se, com isso, que a probabilidade de Joe possuir câncer, dado todos os fatos levantados, é de 23%

A inferência Bayesiana vem sendo utilizada em inteligência artificial e sistemas especialistas. A inferência Bayesiana foi aplicada com sucesso em sistemas para filtragem de e-mails indesejados (SAHAMI et al., 1998). Destaca-se, nesta área, o trabalho de Graham, chamado *A Plan for Spam* (GRAHAM, 2002)

1.2 As Redes Bayesianas

O uso simples do Teorema de Bayes torna-se difícil ao analisar-se problemas complexos. Problemas de maior complexidade não podem ser resolvidos com uma simples aplicação do Teorema de Bayes. Para estas situações, usa-se Redes Bayesianas, já que elas exploram dependências condicionais entre as variáveis.

Trabalhar com inferência Bayesiana é uma tarefa um tanto quanto simples quando o problema envolve poucas variáveis interrelacionadas. Porém, em situações em que tem-se um número grande de variáveis, cada uma afetando direta ou indiretamente a tomada de decisão, esta abordagem se torna impraticável.

Pode-se usar como exemplo de um problema maior, a clássica situação do ladrão / terremoto, exposta por Pearl (1988): *"Estou no trabalho e recebo uma ligação de meu vizinho 1, ele me avisa que meu alarme está disparando. Porém, meu vizinho 2 não liga avisando. Meu alarme pode ser acionado por um pequeno terremoto. Ou será que um ladrão entrou em minha casa"*. Neste exemplo, demonstrado graficamente na figura 1, um número maior de situações deve ser analisado. Um ladrão pode disparar o alarme; um pequeno terremoto pode disparar o alarme; o alarme disparando pode fazer o vizinho 1 telefonar; o alarme disparando pode fazer o vizinho 2 telefonar.

A situação anterior, apesar de ainda ser um caso simples, mostra uma relação entre vários acontecimentos. Um exemplo bem mais complexo seria um conjunto de variáveis para um diagnóstico médico, composto por várias evidências que podem ou não levar à uma doença.

Utilizando-se ainda do exemplo mostrado na sessão anterior, acrescenta-se a seguinte situação: o fato do paciente ter um histórico de fumo afeta (e quanto) ou não a presença de uma anomalia em seu exame. O fato do indivíduo ser fumante pode levar à presença de bronquite? Este histórico de bronquite pode favorecer a presença de câncer de pulmão?

No caso citado anteriormente, é preciso inferir sobre um grande número de variáveis, muitas das quais não estão relacionadas por influência direta. O uso de Inferência Bayesiana se torna extremamente complexo nestes casos, já que o número de cálculos a ser realizado tende a ser muito grande.

As Redes Bayesianas resolvem este problema, calculando as probabilidades conjuntas de um grande número de variáveis, e efetuando inferência sobre as mesmas. As Redes Bayesianas são representadas por grafos acíclicos dirigidos (*directed acyclic graph*, DAG),

onde cada variável do escopo é associada à um nó do gráfico e as relações entre as variáveis são representadas pelas arestas do grafo.

De acordo com Castillo et al (1998), a representação gráfica de modelos probabilísticos têm a vantagem de mostrar explicitamente as relações entre as variáveis e conservar estas relações de forma qualitativa. Os modelos gráficos são também mais intuitivos.

Russel e Norvig (2004) definiram uma Rede Bayesiana como um grafo orientado, onde cada nó é identificado com informações de probabilidades quantitativa. Neste grafo, um conjunto de variáveis aleatórias constituem os nós da rede. Elas podem ser discretas ou contínuas.

Os nós desta rede são conectados entre sí (em pares e sem ciclos). Se houver uma seta do nó X até o nó Y, diz-se que X é pai de Y.

A cada nó de uma Rede Bayesiana, é atribuída uma tabela de probabilidades condicionais. Esta tabela demonstra as probabilidades do evento ocorrer, dado que seus pais tenham ou não ocorrido. Caso o nó não possua um pai, sua tabela de probabilidades é reduziada à probabilidade incondicional daquele evento ocorrer. A figura 1 mostra um exemplo de Rede Bayesiana, destacando suas tabelas de probabilidades iniciais e condicionais.

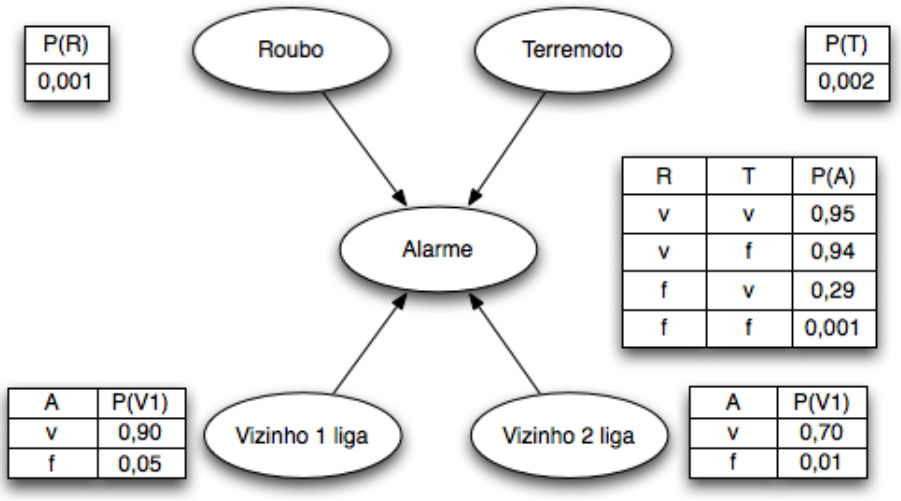


Figura 1: Rede Bayesiana do problema do terremoto/ladrão. Esta rede além da topologia, demonstra a tabela de probabilidades condicionais (RUSSEL; NORVIG, 2004)

1.2.1 Formalização Matemática

As Redes Bayesianas fornecem uma descrição total do domínio em estudo. Qualquer entrada na rede pode ser calculada considerando as outras entradas armazenadas na rede. Considerando estas características, chega-se a seguinte fórmula (RUSSEL; NORVIG, 2004):

$$P(X1, ..., Xn) = \prod_{i=1}^{n} P(Xi|pais(Xi)) \tag{1.7}$$

Visualizando a fórmula anterior, deduz-se que a probabilidade de cada entrada na rede é representada pelo produto de suas probabilidades dados seus antecessores. Como exemplo, observe a figura 2:

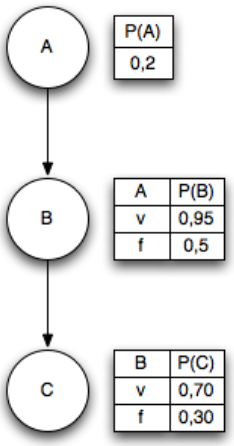


Figura 2: Rede Bayesiana com tabela de probabilidades condicionais

Na rede anterior, caso queira-se calcular a probabilidade do nó C ocorrer, dado que A e B ocorram, deve-se observar as distribuições de probabilidade de todos seus antecessores (pais). A seguir, uma demonstração do cálculo:

$$P(c \wedge b \wedge a) = P(c|b)(Pb|a)P(a) = 0,70 * 0,95 * 0,2 = 0,133 \tag{1.8}$$

1.2.2 Independência condicional nas Redes Bayesianas

Segundo Charniak (CHARNIAK, 1991), uma objeção ao uso da teoria da probabilidade é que a especificação completa de uma distribuição requer uma quantidade absurda de números. Por exemplo, se o problema possui n variáveis aleatórias, sua distribuição completa é representada por $2 \sup n$ probabilidades conjuntas. Ainda segundo Charniak,

as Redes Bayesianas incorporam a independência na sua própria estrutura, reduzindo assim, consideravelmente o número de variáveis.

Pode-se visualizar as Redes Bayesianas como distribuições compactas para uma série de independências condicionais assumidas em uma distribuição. Esta interpretação nos dá a visão de Rede Bayesiana como provedora de fatorização de uma distribuição. Em outras palavras, cada nó X é independente de seus não dependentes, dados seus pais (KOLLER et al., 2007).

A independência que está implícita em uma Rede Bayesiana indica que cada variável é independente de seus não descendentes, dados seus pais (ZHANG; POOLE, 1996). Considerando esta independência, o número de probabilidades requeridas é drasticamente reduzido.

Um procedimento chamado d-separação (separação direta) identifica relações de independência nas Redes Bayesianas. Dois nós em uma Rede Bayesiana estão d-separados sempre que as evidências precisam "passar através" de um nó intermediário, para afetar o destino final. Observe na figura 3 que o nó Y está d-separado do nó X.

Figura 3: D-separação entre X e Y

1.2.3 Redes Bayesianas Multiplas Secionadas

Redes Bayesianas múltiplas secionadas, foram propostas Xiang 1993, com o objetivo de explorar a localidade em grandes domínios de raciocínio. Pode-se dizer que as Redes Bayesianas múltiplas secionadas, ao invés de representar todo o domínio em uma grande

rede monolítica, divide o problema em redes menores. Segundo Onishi (2003), Um ser humano, ao raciocinar em domínios grandes, analisa de forma "natural"as informações que utiliza, concentrando-se em porções específicas do domínio e usando as conclusões oriundas desta análise para selecionar qual a próxima porção que analisará.

Ainda conforme Xiang (1993), domínios complexos apresenta uma característica de localidade, criando subdomínios naturais. Ao analisar o problema, a atenção é focada em apenas um subdomínio, obtendo ali as evidências e tomando decisões.

Uma Rede Bayesiana Múltipla Secionada, ou simplesmente, MSBN, é formada por um conjunto de pequenas Redes Bayesianas (subredes) que, juntas definem uma Rede Bayesiana. Cada subrede representa um subdomínio em um domínio maior, que por sua vez representa um problema. Cada subrede compartilha um conjunto de variáveis com ao menos uma outra subrede.

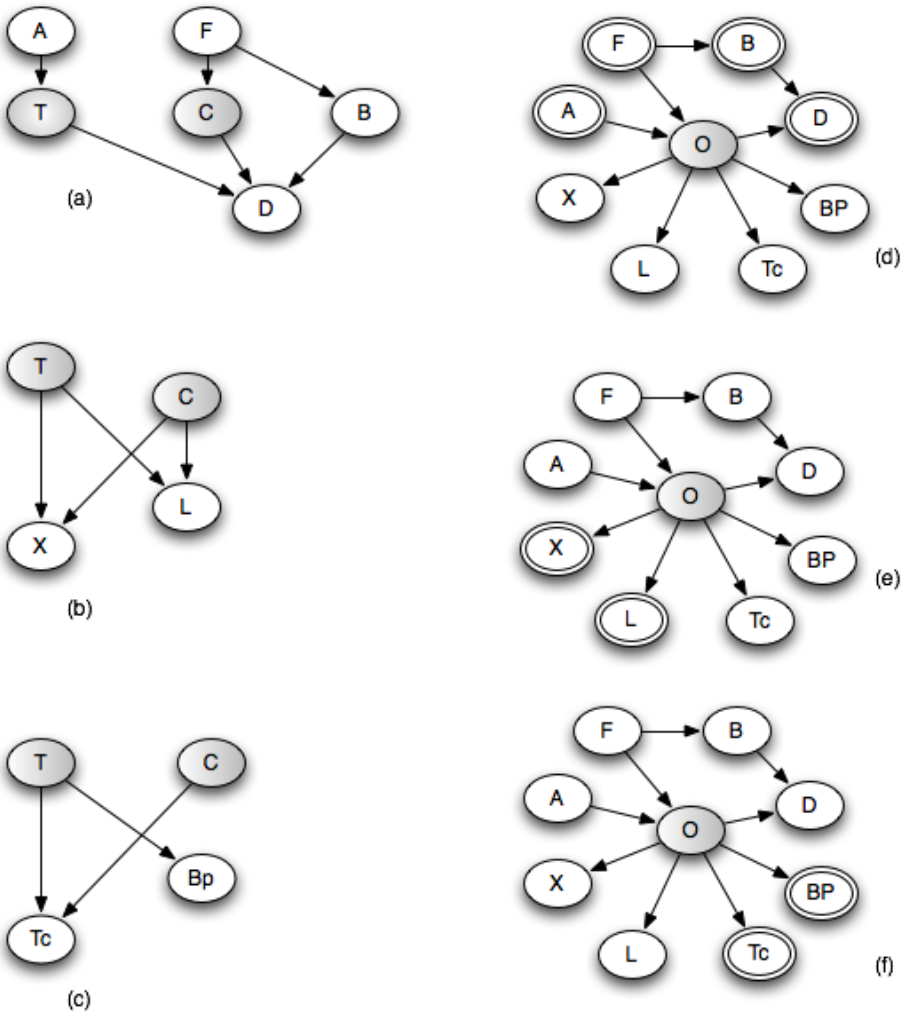


Figura 4: Asia extendida (ONISH; CARVALHO, 2003)

A figura 4 representa um modelo de Redes Bayesianas múltiplas secionadas para o

problema Asia. Este modelo demonstra que Dispnéia pode ser causada por tuberculose (T), câncer de pulmão (C) ou bronquite (B). Uma visita recente a Ásia (A) aumenta as chances de tuberculose. Fumar (F) aumenta a probabilidade de câncer de pulmão e bronquite. Dados indicam que o paciente com dispnéia, esteve na Ásia, mas não informam se ele é fumante ou se fez raio X do tórax. Qual a possibilidade do paciente ter tuberculose, câncer pulmonar ou bronquite? Após um diagnóstico baseado nestas informações, para melhor discriminar entre tuberculose e câncer, o médico pode requerer testes radiológicos ou biológicos. Para tais patologias, o laboratório de radiologia tem dois testes relevantes: raio X (X) e laminografia (L) enquanto no de biologia são relevantes os testes de catarro (Tc) e biópsia (Bp) (ONISH; CARVALHO, 2003).

1.2.4 Trabalhos Correlatos

As Redes Bayesianas têm sido utilizadas em várias áreas onde tomadas de decisão baseadas em informações incompletas precisam ser realizadas. A área que mais se utiliza destas técnicas é, sem dúvida, a medicina, principalmente para realização de diagnóstico. A seguir, alguns trabalhos utilizando Redes Bayesianas são expostos.

NasoNet (GALáN et al., 2001) é um sistema que realiza diagnóstico de câncer nasofaringeal. Ele utiliza uma Rede Bayesiana composta por mais de cem nós, que indicam hemorragias, infecções, sintomas, etc.

A Microsoft utiliza Redes Bayesianas no Office para detectar distrações do usuário e abrir o sistema de ajuda, observando o ambiente em que o usuário está trabalhando. O Projeto *Lumière*, como foi batizado pelos pesquisadores da Microsoft, transforma eventos do sistema em variáveis observáveis, representadas em um modelo Bayesiano. Este modelo contém variáveis como dificuldade da tarefa que está sendo realizada, acessos recentes à itens de menu, pausas após determinada atividade, entre outros (HORVITZ et al., 1996).

As Redes Bayesianas também são utilizadas em sistemas de descoberta de genes. Usa-se as Redes Bayesianas para representar dependência estatística entre os dados coletados. As redes representam um modelo de dependência entra expressões de diferentes genes. Uma outra vantagem das Redes Bayesianas, é a possibilidade de análise mesmo em amostras poluídas (FRIEDMAN et al., 2000).

O Instituto Suiço de Ciência Ambiental e Tecnologica utilizou Redes Bayesianas para determinar o declínio do número de peixes em seus rios. O projeto batizado de *Fischnetz* trabalhou com doze hipóteses investigadas em laboratório. O resultado destas investiga-

ções gerou uma Rede Bayesiana. Por lidar bem com escopos incompletos, as Redes Bayesianas se mostraram de grande utilidade para o projeto (BORSUK; BURKHARDT-HOLM; REICHERT, 2002)

Na área industrial, pesquisas utilizam Redes Bayesianas para gerenciamento de risco de rupturas em cadeias de suprimento. Os modelos determinísticos de gestão de estoques como, assumem premissas que não condizem com o contexto real em que as decisões de estoque devem ser tomadas. A modelagem bayesiana permite a inclusão de dados subjetivos de especialistas, no caso de insuficiência de informações passadas (SILVA; LADEIRA; OLIVEIRA, 2008).

Redes Bayesianas são utilizadas para fusão dos dados de sensores de sinais de várias fontes, integrados com o objetivo chegar a uma interpretação de uma situação. Mesmo com a falta de dados de um dos sensores, as Redes Bayesianas são capazes de tomar uma decisão fazendo inferência sobre os dados desconhecidos (MARTINS1 et al., 2007).

Neste capítulo foi abordada a teoria da probabilidade, o teorema de Bayes e as Redes Bayesianas. O conteúdo aqui apresentado fornece sustentação para compreensão da sequencia do trabalho, onde será apresentada a inferência nas Redes Bayesiana e exemplos de algoritmos para manipulação da rede.

2 REALIZANDO INFERÊNCIA NAS REDES BAYESIANAS

Segundo Russel et al (2004), a tarefa básica de inferência em uma rede Bayesiana consiste no cálculo da distribuição de probabilidades posteriores, para um conjunto de variávies, dado algum evento observado. Basicamente, calcular as probabilidades requer a avaliação de todos os nós da rede, considerando cada evidência que levou ao mesmo.

O principal problema que observa-se na avaliação de Redes Bayesianas é o tempo elevado que uma busca exata requer. Uma série de algoritmos de aproximação foram desenvolvidos para solucionar este problema. No decorrer deste capítulo, diferentes métodos de inferência serão demonstrados e avaliados.

2.1 A complexidade dos algoritmos de inferência

O maior desafio no uso de Redes Bayesianas é a complexidade dos algoritmos de inferência. Normalmente, a avaliação exata de uma rede requer um algoritmo de complexidade NP-Difícil (G.F.COOPER, 1991). Problemas de complexidade NP-Difícil (NP-Hard) não podem ser resolvidos em tempo polinomial (GAREY; JOHNSON, 1979).

De acordo com Charniak (1991), há apenas uma classe restrita de redes que podem ser resolvidas de forma exata em tempo aceitável. Estas redes não possuem mais do que um caminho entre dois nós. Estas redes são conhecidades como *singly connected network*. Existem algoritmos que resolvem Redes Bayesianas explorando a estrutura da rede em particular, ou então trabalhando com fatorização das probabilidades conjuntas (ZHANG; POOLE, 1994).

Para evitar o uso de algoritmos NP-Difícil, pode-se optar pelo uso de técnicas de inferência aproximada. Estas técnicas fornecem respostas próximas as exatas, mas evitam a necessidade de um algoritmo NP-Difícil. Na maioria das vezes, estes algoritmos definem valores aleatórios em alguns nós e, após esta definição, calculam probabilidades baseadas nestes valores.

2.2 Exemplificação da Inferência em uma Rede Bayesiana

Esta seção demonstra como se dá a inferência em uma Rede Bayesiana. Este conceito básico é aplicado à maioria dos algoritmos.

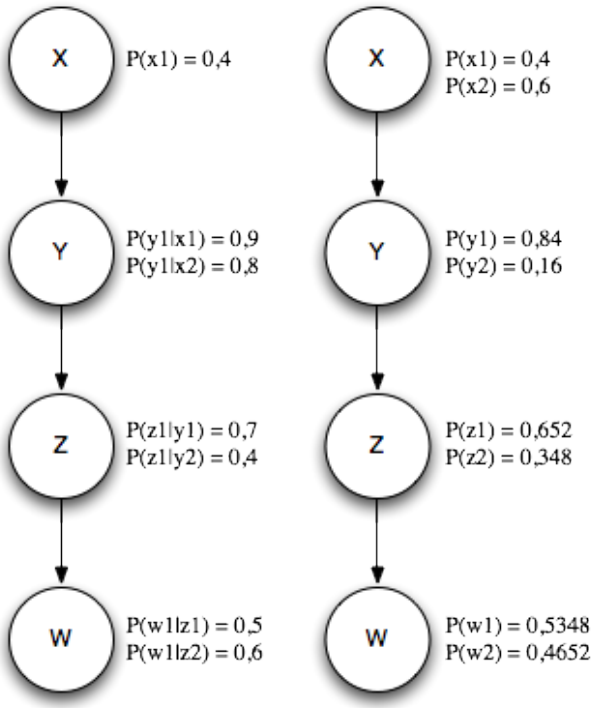


Figura 5: Rede Bayesiana com suas probabilidades a priori à esquerda (NEAPOLITAN, 2003)

Considerando a rede demonstrada na figura 5, para calcular as probabilidades de cada variável, necessita-se de informações determinadas por seus pais.

O cálculo das probabilidades do exemplo anterior se dá da seguinte forma:

$$P(y1) = P(y1|x1)P(x1)+P(y1|x2)P(x2) = 0{,}84$$
$$P(z1) = P(z1|y1)P(y1)+P(z1|y2)P(y2) = 0{,}652$$
$$P(w1) = P(w1|z1)P(z1)+P(w1|z2)P(z2) = 0{,}5348$$

2.3 Algoritmos de Inferência Exata

A inferência exata de uma Rede Bayesiana consiste no cálculo da distribuição de probabilidade para um conjunto de variáveis (RUSSEL; NORVIG, 2004). Como a inferência exata, no caso geral, é um problema intratável, apenas um algoritmo é demonstrado neste trabalho. Na sessão seguinte, algoritmos de aproximação serão estudados.

2.3.1 Inferência por enumeração

Citando novamente Russel (2004), qualquer probabilidade condicional pode ser calculada através do somatório da distribuição total. Uma consulta P(X|e) pode ser respondida através da equação 2.1.

$$P(X|e) = \alpha P(X, e) = \alpha \sum_{y} P(X, e, y) \tag{2.1}$$

Observando a equação anterior, conclui-se que uma consulta à uma Rede Bayesiana pode ser respondida calculando-se as somas dos produtos de probabilidades condicionais na rede. Considerando a Rede Bayesiana representada na figura 1, demonstrada no capítulo 1, caso queira-se fazer a seguinte busca na rede: P(Roubo|Vizinho 1 liga = verdadeiro, Vizinho 2 liga = verdadeiro), obtem-se a seguinte expressão:

$$P(R|v1, v2) = \alpha P(R, v1, v2) = \alpha \sum_{e} \sum_{a} P(R, e, a, v1, v2) \tag{2.2}$$

Onde:

- R = roubo
- v1 = vizinho 1
- v2 = vizinho 2

No algoritmo, demonstrado a seguir, a complexidade de processamento cresce linearmente de acordo com as variáveis acrescentadas à Rede Bayesiana, porém, a complexidade em tempo de processamento cresce na ordem de 2n, onde n é o número de variáveis.

A seguir, uma demonstração do algoritmo em linguagem estruturada (RUSSEL; NORVIG, 2004):

```
entradas: X, a variável de consulta
          e, valores observados para as variáveis
          rb, uma Rede Bayesiana

função ENUMERAÇÃO(X, e, rb) retorna uma distribuição sobre X
     Q(X) <- uma distribuição sobre X, inicialmente vazia
```

```
      para cada valor x1 em X faça
         estender e com valor x1 para X
         Q(x1) <- ENUMERAR_TOOS(VARS(rb,e))

      retornar NORMALIZAR(Q(X))

função ENUMERAR-TODOS(vars,e) retorna um número real
      se VAZIO(vars) então retornar 1
      Y <- PRIMEIRO(vars)
      se Y tem valor y em e então
         retornar (Py | pais(Y)) * ENUMERAR-TODOS(RESTO(vars),ey)
```

2.3.2 Algoritmo de eliminação de variáveis

Segundo Zhang e Poole (1996), a chave para uma inferência mais eficiente está no conceito de fatorização. A fatorização de uma probabilidade conjunta é a lista de fatores (funções), das quais as probabilidades conjuntas podem ser construidas.

Nesta sessão, será demonstrado o algoritmo de eliminação de variáveis (*Variable Elimination*) descrito em Zhang e Poole (1996). Este algoritmo deriva do algoritmo conhecido como *Bucket Elimination*, proposto por Dechter (1996). Este algoritmo tem esse nome pois ele soma as variáveis de uma lista de fatores um-a-um. Uma ordem de eliminação das variáveis deve ser informada, ela é chamada de ordem de eliminação (*elimination ordering*).

Russel e Russel (2004), apresentam o algoritmo de forma mais simplificada. De acordo com os autores, a eliminação de variáveis funciona avaliando expressões (equações) da direita para a esquerda. Os resultados intermediários são então armazenados, e os somatórios, são efetuados apenas para as partes da expressão que dependem de determinada variável. Além disso, toda variável que não é ancestral das variáveis de consulta ou evidência podem ser removidas, já que elas são irrelevantes à pesquisa.

A seguir, uma demonstração do algoritmo em linguagem estruturada (RUSSEL; NORVIG, 2004):

```
entradas: X, a variável de consulta
          e, vlores observados para variáveis E
```

```
            rb, uma Rede Bayesiana

função ELIMINAÇÃO(X, e, rb) retorna uma distribuição sobre X
    fatores <- []
    vars <- REVERTER(VARS[rb])
    para cada var em vars faça
        fatores <- CRIAR-FATOR(var,e)
        se var é uma variável oculta então
           fatores <- SOMAR(var,fatores)

    retornar NORMALIZAR(PRODUTO(fatores))
```

2.3.3 Inferência por árvores de junção

A inferência em Redes Bayesianas pode occorer de forma eficiente ao se trabalhar com estruturas computacionais auxiliares. Estas estruturas são conhecidas como árvores de junção, ou *junction trees*. Esta abordagem foi apresentada por Jensen e Jensen (1994).

Segundo Ladeira (1999), uma árvore de junção T de um grafo G é uma árvore cujos nós são agregados de G e:

1. para cada nó X de G, existem nós em T que contém X.
2. para cada par de agregados (cliques) Ci, Ck, todos os agregados entre eles contém $Ci \cap Ck$.
3. cada arco é rotulado com as variáveis comuns aos agregados que os une (separador).

2.3.4 Transformação topológica e construção da árvore de junção

Antes de aplicar-se o algoritmo de inferência, uma transformação topológica deve ser realizada na Rede Bayesiana, transformando-a em um árvore de junção. Esta transformação se dá em quatro passos: moralização, triangularização, identificação de cliques e construção da árvore.

A moralização consiste em adicionar arcos entre cada par de pais de cada nó do grafo original. Em seguida, deve-se eliminar a orientação dos arcos. A seguir, demonstração de um grafo antes e após a moralização.

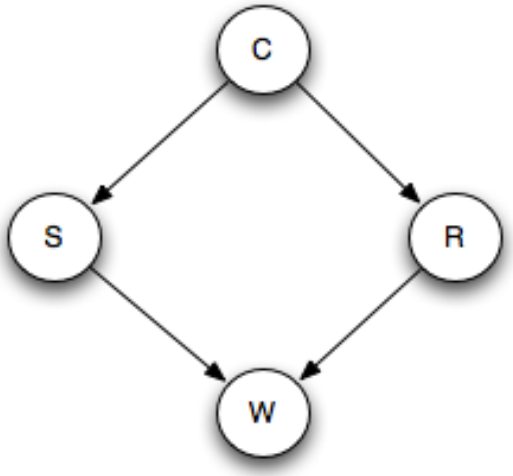


Figura 6: Grafo antes do processo de moralização

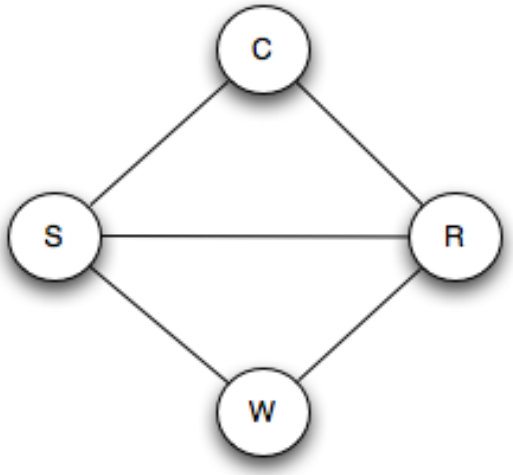


Figura 7: Grafo após o processo de moralização

O algoritmo em pseudo-linguagem para moralização de um grafo é mostrado abaixo. Este algoritmo foi proposto por Ladeira, em 1999. O grafo, após o processo de moralização, é demonstrado na figura 7.

```
moralização(N, pa, Adj, n) {            // Retorna Gm(N, Ajm,n)
 for (i=1; i<=n; i++) {                  // Inicializações
 no = N(i); Adjm(no) = 0;              // i-ésimo nó de N
 }
 for (i=1; i<=n; i++) {
     no = N(i); np = |pa(no)|;           // Cardinalidade pa(no)
     if (np == 0) continue;              // Inicia nova iteracao
```

```
      for (j=1; j<=np; j++) {
       p1 = pa(no) (j);                // p1: jésimo pai do no
       Adjm(no) = Adjm(no) U p1;       // p1: adjacente de no
       Adjm(p1) = Adjm(p1) U no;       // no: adjacente de p1
       for (k=j+1; k<=np; k++) {
       p2 = pa(no) (k);            // p2: k-ésimo pai do nó
       Adjm(p1) = Adjm(p1) U p2;   // p2: adjacente de p1
       Adjm(p2) = Adjm(p2) U p1;   // no adjacente de p2
       }
      }
  }
}
```

No algoritmo anterior, considere:

- N é o conjunto de nós;
- pa é o conjunto de pais;
- Adj é o conjunto de nós adjacentes;
- n é o número de nodos do grafo.

O passo seguinte é conhecido como triangularização. Ele consiste na inserção de arcos em *loops* com mais de três nós. Cada arco inserido decompõe o *loop* em dois *loops* menores (LADEIRA; VICARI, 1999).

Ainda, segundo os mesmos autores, um grafo não orientado é triangular se, e somente se, todos os seus nós podem ser eliminados, um a um, sem a adição de qualquer arco. A seguir, um exemplo de um grafo após o processo de triangularização. Os arcos pontilhados foram incluídos pelo processo.

O algoritmo de triangularização baseado na Heurística do Peso Mínimo é demonstrado no anexo A. Este algoritmo é proposto por (LADEIRA; VICARI, 1999), com base em (KJAERULFF, 1990).

Seja Gt, um grafo tringular, A1,...,An, uma seqüência de eliminação de seus nós, e Ci, o conjunto contendo Ai, e seus vizinhos no instante de sua eliminação (i.e os vizinhos do nó eliminado, com numeração A, maior do que a numeração A do nó sendo eliminado).

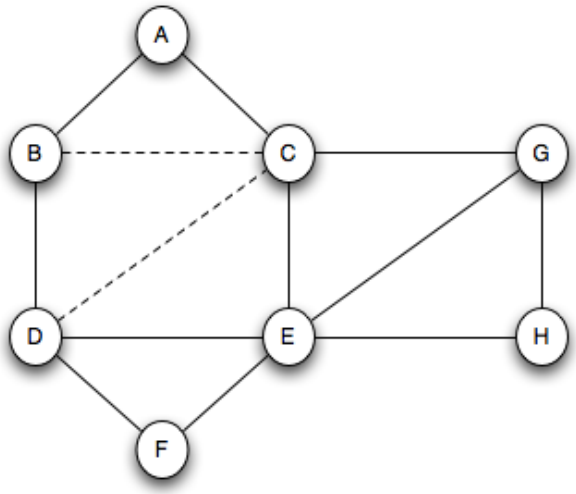


Figura 8: Grafo após o processo de triangularização

Então Ci é um clique de Gt se Ci é máximo, se não está contido em nenhum outro clique (LADEIRA; VICARI, 1999 apud JENSEN, 1996).

No algoritmo de construção da árvore de junção, deve-se definir um índice para cada vértice da rede. Este índice é o inverso da ordem de eliminação da fase de triangularização. Para cada conjunto de cliques no grafo, cria-se uma lista de vértices adjacentes. Se a lista de vértices adjacentes possui um vértice com índice menor do que o vértice de maior índice no clique, então este vértice recebe o valor do maior índice. O clique com índice 1, deve ser tornar a raiz da árvore.

Após a etapa anterior ser realizada, deve-se criar os separadores entre os cliques. Chama-se separador, o conjunto de vértices que contém a intersecção de vértices entre dois cliques. Caso um clique possua uma intersecção com dois cliques, o clique com menor índice é usado como separador.

Segundo Ladeira (1999), que cita Jensen (1994), o algoritmo para construção de uma árvore de junção segue os passos abaixo:

- Defina a numeração α **U** $\leftrightarrow$ {1,2,...,|U|} em função da inversa ordem de eliminação, tal que para u, v $\in$ **U**, se u foi eliminado após v então $\alpha(u) < \alpha(v)$;

- Seja v, a variável de C de maior numeração, tal que as w, com numeração menor que v possuam um vizinho comum, não em C e com numeração menor que v, isto é u $\notin$ C e $\alpha(u) < \alpha(v)$. Se existir, o índice de C é $\alpha(v)$, senão é 1. Após obter todos os índices, número os cliques em ordem crescente de índice, iniciando com 1;

- Considere o clique de número 1, isto é, C1, como a raiz da árvore;
- Ligue Ck a algum clique Cj, j<k>1, já na árvore, que contenha $Sk = Ck \cap \cup -1Ci$ Se existir mais de um, ligue ao de menor índice.

2.3.5 Inferência na árvore de junção

Após a construção da árvore de junção, o algoritmo de inferência segue basicamente 5 passos (KJAERULFF; MADSEN, 2005). São eles:

1. Cada item de evidência deve ser incorporado aos pontenciais da árvore de junção. Para cada ítem de evidência, uma função de evidência é multiplicada no clique apropriado.
2. O clique raíz deve ser selecionado. Ele é conhecido como o clique raiz da propagação.
3. Mensagens são passadas por todos os separadores da árvore de junção, até o clique raiz. Estas mensagens, atualizam os potenciais dos clique e dos separadores da árvore de junção.
4. Mensagens são passadas novamente, porém, na direção contrária. Esta fase é conhecida como distribuição de informação.
5. Neste momento, a árvore é dita em equilíbrio. A probabilidade P(X|e), pode ser computada à partir de qualquer clique ou separador que contenha X.

2.4 Inferência aproximada em Redes Bayesianas

Como foi tratado anteriormente, o uso de inferência exata pode levar à situações intratáveis. O alto tempo de execução destes algoritmos pode, muitas vezes, tornar a tarefa de inferência em uma Rede Bayesiana impossível.

Para reduzir o tempo de execução, pode-se utilizar algoritmos de inferência aproximada nas Redes. Estes algoritmos fazem uma aproximação das distribuições conjuntas de algumas variáveis da rede (KOLLER et al., 2007). Estas instâncias são conhecidas como amostras. Estas amostras representam parte das probabilidades do problema.

Segundo Castillo (1998), a idéia básica dos algoritmos de inferência aproximada é gerar uma amostra de tamanho N, a partir da função de probabilidade conjunta das variáveis.

Estas amostras são utilizadas para calcular valores aproximados das probabilidades de certos nós, dado a evidência.

Ainda, segundo o mesmo autor, os algoritmos de inferência aproximada podem ser classificados em dois tipos: métodos de simulação estocástica e métodos de busca determinística. Os métodos estocásticos geram as amostras tomando como base a função de probabilidade conjunta, e também, usam mecanismos aleatórios. Já os métodos determinísticos, geram as amostras de forma sistemática.

2.4.1 Inferência por simulação de cadeias de Markov

Este método foi proposto por Pearl (1987). Segundo Castillo (1998), este método consiste em designar aos nós de evidências seus respectivos valores e, logo após, simular estocasticamente a rede resultante.

O algoritmo gera cada evento fazendo uma mudança aleatória no evento precedente. Cada estado da rede é gerado por amostragem aleatória de um valor em variáveis que não são de evidência. O algoritmo trabalha invertendo uma variável de cada vez, mas mantém fixas as variáveis de evidência. O grafo triangular é demonstrado na figura (**??**).

Toma-se como exemplo a rede exposta por Russel em RUSSEL; NORVIG (2004), demonstrada na figura 9.

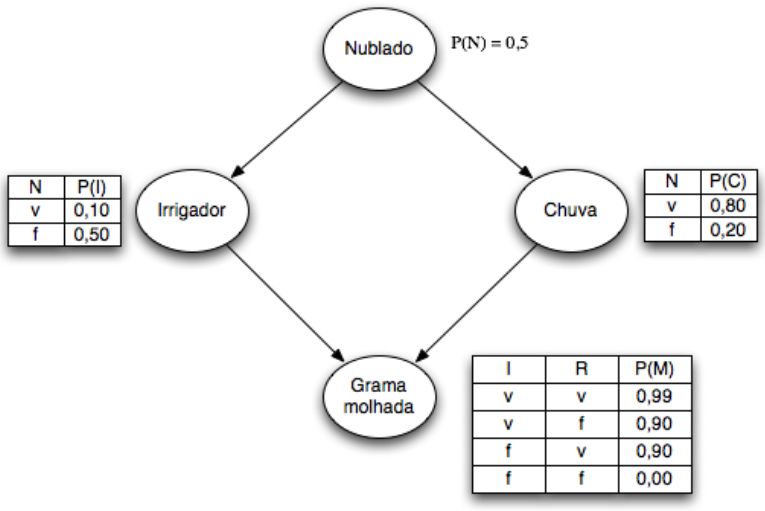


Figura 9: (RUSSEL; NORVIG, 2004)

Executa-se a seguinte consulta na rede: $Chuva/Irrigador = v$, $GramaMolhada = v$ aplicado a rede demonstrada anteriormente. As variáveis de evidência, neste caso *Irrigador* e *GramaMolhada*, são definidas com seus respectivos valores observados. As variáveis

ocultas, que neste caso são *Nublado* e *Chuva*, são inicializadas com valores aleatórios (para este exemplo considera-se os valores verdadeiro e falso, respectivamente). As seguintes etapas são então executadas repetidamente:

A variável *Nublado* é amostrada, levando em conta suas variáveis pais. Observando o exemplo anterior, a amostra é dada por *P(Nublado|Irrigador=v, Chuva=f)*. Suponha, para este exemplo que o resultado da amostragem seja *Nublado = falso*.

Cada amostra gerada contribui para o resultado da variável de consulta, neste caso *Chuva*. Se o algoritmo visitar X estados em que a variável possui valor verdadeiro e Y estados em que seu valor é falso, a resposta à consulta efetuada será a normalização dos valores de X e Y.

Castillo (1998) resume o algoritmo de simulação de cadeias de Markov em três passos básicos. São eles:

1. Deve-se atribuir a cada uma das variáveis não evidenciais um valor aleatório qualquer.
2. Deve-se criar uma ordenação aleatória para selecionar os nós não evidenciais. Para cada variável da lista, se gera um valor aleatório. Este valor é gerado à partir da distribuição de probabilidades conjuntas das variáveis. Normaliza-se então as probabilidades.
3. Repete-se a etapa dois para N extrações

A seguir, uma demonstração do algoritmo em linguagem estruturada (RUSSEL; NORVIG, 2004):

```
função MARKOV(X,e,rb,N) retorna uma estimativa de P(X|e)
    variáveis: N[X]: um vetor de contagens sobre X (inicia em zero)
               Z: as variáveis não de evidência em rb
               x: o estado atual da rede, copiado de e

    inicializar x com valores aleatórios para as variáveis em Z
    para j=1 até n faça
       N[x] <- N[x] + 1
       para cada variável em Z faça
          fazer a amostragem de Z em x
```

```
retornar NORMALIZAR(N[X])
```

A amostragem, citada no algoritmo demonstrado anteriormente se dá através da computação de todas as variáveis da *cobertura de Markov*, da variável em evidência naquele momento. A cobertura de Markov de uma variável, consiste em seus pais, filhos e pais dos filhos (RUSSEL; NORVIG, 2004).

Neste capítulo, o leitor conheceu os algoritmos usado para manipulação das Redes Bayesianas, fundamentais para o entendimento do projeto AMPLIA, uma vez que este utiliza-se deste recurso. No próximo capítulo, o AMPLIA será apresentado, de forma que o leitor possa situar-se no contexto de desenvolvimento deste trabalho de conclusão.

3 O PROJETO AMPLIA

Neste capítulo, o projeto AMPLIA será apresentado, mostrando suas características atuais e as características desejadas a nova versão do software. O AMPLIA é um um ambiente multiagente de aprendizagem que tem como principal objetivo, através de um ambiente simulado, colocar o aluno em contato com práticas reais do dia-a-dia médico.

3.1 Características do projeto AMPLIA

O projeto AMPLIA, cujo nome é sigla para *Ambiente Multiagente Probabilístico Inteligente de Aprendizagem* foi desenvolvido pelo Grupo de Inteligência Artificial do Instituto de Informática da UFRGS. O projeto propõe um ambiente de aprendizagem onde existe uma negociação entre aluno e especialista, construindo assim, o conhecimento em ambas as partes (FLORES, 2005).

Segundo Vicari (2003), o ambiente AMPLIA tem como objetivo apoiar o desenvolvimento do raciocínio diagnóstico, bem como a modelagem de hipóteses diagnósticas. Para atingir este objetivo, o software utiliza mecanismos probabilísticos para a representação do conhecimento. O AMPLIA utiliza as Redes Bayesianas, foco principal deste trabalho, para representar graficamente o raciocínio formado. Ainda entende-se como objetivo do AMPLIA, além de produzir um diagnóstico eficaz, entender como as diferentes variáveis (sintomas, sinais, dados laboratoriais) relacionam-se probabilisticamente (FLORES, 2005). A figura 10 demonstra a tela de edição de Redes Bayesianas da versão anterior do AMPLIA.

O AMPLIA também tem o objetivo de auxiliar as escolas de medicina a adaptarem seus currículos as mudanças de diretrizes curriculares para os cursos de graduação em Medicina, aprovadas pelo Conselho Nacional de Educação, que estão em implantação desde 2003 através do Programa de Incentivo às Mudanças Curriculares nas Escolas Médicas. É esperada a utilização de metodologias de ensino que favoreçam a participação mais ativa do aluno na construção do conhecimento (MACEDO, 2001), aproximando assim, as

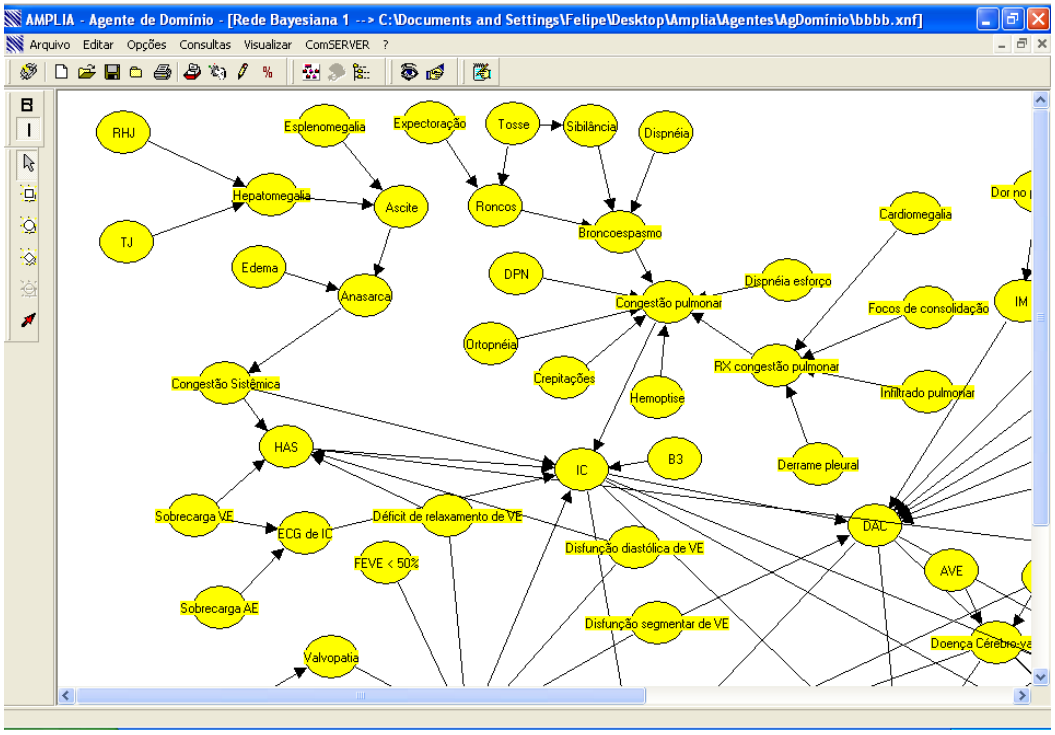


Figura 10: Tela da versão antiga do software AMPLIA mostrando uma Rede Bayesiana

práticas da medicina do ambiente de aprendizado.

O AMPLIA conta com dois estágios distintos. No primeiro, o especialista "alimenta"o sistema com uma caso clínico. Este caso é detalhado com informações clínicas do paciente. O especialista monta seu diagnóstico, estruturando seu raciocínio em uma Rede Bayesiana. Com base nos conhecimentos previamente adquiridos, e no princípio da negociação pedagógica, o aluno então monta seu diagnóstico, também baseado no caso clínico em estudo. Ao fim deste processo, o sistema analisa o diagnóstico apresentado pelo aluno, comparando o mesmo com o fornecido pelo especialista.

O processo de ensino aprendizagem é a base da negociação pedagógica dentro do AMPLIA. O sistema emprega um modelo de negociação baseada em argumentação aplicada à aprendizagem colaborativa, contemplando aspectos cognitivos como crenças, ações e níveis de confiança entre aprendiz e especialista. No AMPLIA, a negociação é empregada para alcançar um ponto de equilíbrio entre aluno e professor. Este objetivo é atingido através do uso intenso da argumentação nas estratégias pedagógicas (FLORES, 2005).O processo de negociação ocorre em ciclos. Cada ciclo é iniciado quando o aluno submete seu modelo para avaliação do especialista.

A negociação pedagógica dentro do AMPLIA tem como principal objetivo o estabelecimento e a confirmação de um alto grau de confiança entre os participantes do processo.

Esta tarefa é responsabilidade do chamado agente mediador (FLORES, 2005). Ainda citando Flores, a diferenças entre os diagnósticos (aqui representados através de uma Rede Bayesiana) são tratadas por meio de estratégias pedagógicas, baseadas na interação e na negociação entre os agentes inteligentes do sistema e o aluno.

Ao trabalhar com estes dois estágios, o AMPLIA parte do princípio que o conhecimento do especialista (professor) não é o único correto. O conhecimento do especialista é um dos possíveis para a resolução de um problema. O aluno pode apresentar argumentos relevantes, que ainda não foram observados, para encontrar a solução de um problema.

3.2 Arquitetura do AMPLIA

O AMPLIA está estruturado utilizando uma arquitetura baseada em agentes. Conforme definido por Russel (2004), agente é qualquer entidade que coleta informações no meio em que se encontra, utilizando, para isso, sensores. Ele então processa estas informações e promove alguma ação sobre o meio através de atuadores, tendo sempre como objetivo maximizar o seu desempenho. Em sistemas multi-agente, como o AMPLIA, cada agente apresenta características independentes, tendo um comportamento próprio em relação aos outros agentes do sistema. Apesar desta independência, todos os agentes conseguem se comunicar com os demais, utilizando protocolos de comunicação.

O sistema amplia conta com três agentes, todos mantendo comunicação entre si. Os três agentes são: agente Aprendiz, agente de Domínio e agente Mediador. A figura 11 mostra como os agentes estão organizados dentro do AMPLIA.

Cada agente atua em três níveis distintos: Interação; Operacional e Decisão. O nível de Interação, como o próprio nome diz, é responsável pelas atividades de comunicação com o meio. Já o nível Operacional, é responsável pela separação e classificação das informações que vem do nível de Interação, encaminhamento da informação ao próximo nível (Decisão) bem como a execução das instruções recebidas do nível de Decisão. O nível de Decisão é responsável pela avaliação do cenário. Esta avaliação se dá através dos dados recebidos dos demais níveis, retornando aos níveis inferiores quais atitudes tomar com relação a um determinado estado.

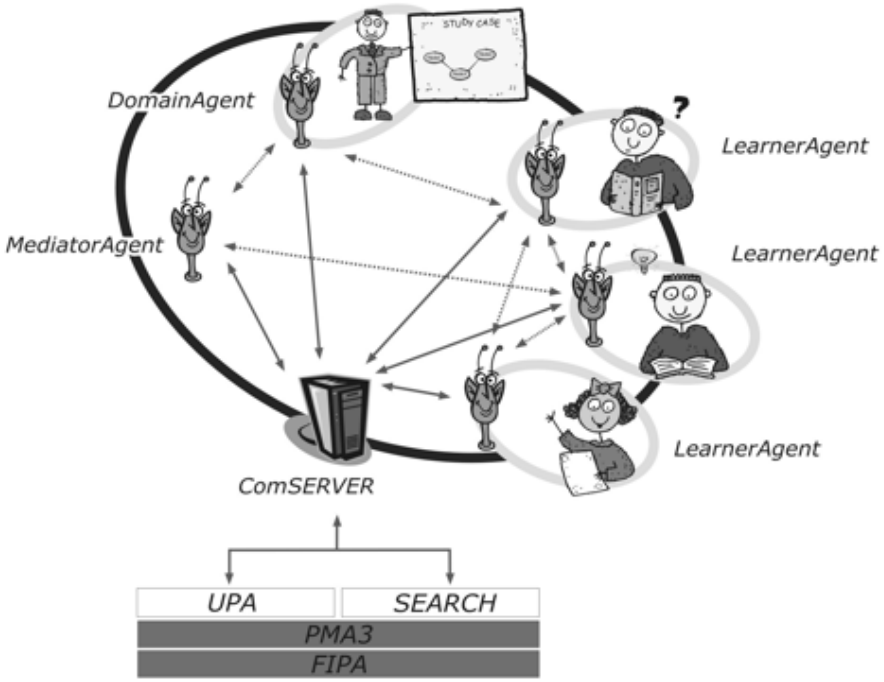


Figura 11: Estrutura multiagente do projeto AMPLIA (FLORES, 2005)

3.3 A divisão de tarefas entre os agentes do projeto AMPLIA

O agente aprendiz representa o papel do aluno durante a realização das atividades Este agente é responsável por manipular todas as informações referentes ao processo de aprendizagem, bem como, o grau de autoconfiança declarado pelo aluno. Este agente é responsável por deduzir o nível de autonomia que o aluno apresenta. Para deduzir este nível de autonomia, o agente analisa como o aluno escolheu as variáveis da rede, como as ligações entre elas foram realizadas, quantas vezes o alunos refez estes passos, etc. O agente classifica a autoconfiança e o nível de autonomia do aluno em baixo, médio ou alto.

O agente de Domínio é responsável pela Rede Bayesiana desenvolvida pelo especialista, bem como a base de dados com os casos reais. Este agente é responsável pela avaliação da Rede Bayesiana desenvolvida pelo aluno, utilizando verificações qualitativas e quantitativas.

Na avaliação qualitativa, o AMPLIA realiza uma simplificação do modelo Bayesiano desenvolvido pelo especialista, eliminando todos os nodos que não se aplicam à determinado caso clínico. Ainda durante a análise qualitativa, o AMPLIA procura por relações equivocadas ou omitidas, bem como nodos nestas mesmas situações. Esta avaliação se dá

na rede do aluno, em comparação com a produzida pelo especialista.

Durante a avaliação qualitativa, os nodos são classificados da seguinte forma: *Trigger* - Quando presente, seleciona o diagnóstico como solução potencial; Essencial - Deve estar presente para assegurar a identificação do diagnóstico; Complementar - Sua presença aumenta a probabilidade do diagnóstico; Excludente - Sua presença diminui a probabilidade de confirmação do diagnóstico - Desnecessário - Não é necessário para a confirmação do diagnóstico. Caso a avaliação qualitativa seja positiva, a rede do aluno é submetida à avaliação quantitativa (FLORES, 2005).

A avaliação quantitativa submete a rede do aluno a uma base de casos reais, com o objetivo de testar seu desempenho. Além da estrutura da rede, é avaliada a distribuição de probabilidades entre as variáveis, com o objetivo de verificar se a rede do aluno realmente pode fornecer como resposta o diagnóstico esperado pelo especialista.

Ao realizar as avaliações, que basicamente levam em conta a forma como o aluno estruturou seu modelo, as redes são classificadas da seguinte forma: Inviável - o modelo não representa uma Rede Bayesiana, contém ciclos ou nós não orientados; Incorreta - não apresenta diagnóstico, diagnóstico justifica as causas, presença de nós excludentes; Potencial - não possui alguns nós importantes ou possui nós desnecessários; Satisfatória - é diferente do modelo apresentado pelo especialista, mas satisfaz o estudo de caso proposto; Completa - idêntica ao modelo proposto pelo especialista (FLORES, 2005).

O agente Mediador implementa as táticas pedagógicas necessárias para o apoio do aluno durante o processo de aprendizagem. Para tomar decisões, leva em conta a avaliação da Rede Bayesiana montada pelo aluno, a autoconfiança declarada pelo aluno e a credibilidade analisada pelo agente Aprendiz. As táticas adotadas pelo Mediador podem ser a correção; indicação; sugestão; experimentação; busca; reflexão; exemplos; problematização; discussão; demonstração e hipóteses. Estas táticas se distribuem entre as seguintes estratégias: Orientação - utilizada quando problemas graves são identificados na rede do aluno, indicando os erros para correção; Contestação - indica que existem inconsistências na rede do aluno, pedindo uma reavaliação desta; Apoio - é utilizada quando a rede do aluno é classificada como potencial, são reapresentados os casos e é indicado o já está correto em sua avaliação; Ampliação - utilizada com redes potenciais ou satisfatórias e autoconfiança alta, sugere revisões que visam a melhoria da performance da rede e a elaboração de novas hipóteses; Comprovação - utilizado para aumentar a confiança dos alunos que chegaram a um resultado satisfatório.

3.4 Características desejadas à nova versão do AMPLIA

A forma como o projeto AMPLIA está estruturando atualmente, o limita ao uso em uma rede local (LAN), em função das tecnologias aplicadas. Esta limitação se dá pelo uso da biblioteca FACIL (GLUZ; VICCARI, 2003), utilizada para a comunicação entre agentes de software (FLORES, 2005).

Pensando nestas limitações, um novo projeto foi iniciado pelo grupo de pesquisa da UFCSPA, com o objetivo de utilizar tecnologias que comportem o uso de recursos de Internet. Ampliando o escopo do software para a Internet, aumenta-se a abrangência e o alcance do mesmo, possibilitando expandir o conhecimento à várias instituições, permitindo a criação de um repositório com diagnósticos e casos clínicos a disposição de acadêmicos da área da saúde.

O objetivo desta nova versão é manter as mesmas funcionalidades do projeto original, porém, adicionando recursos que permitam seu uso através da Internet. Para atingir estes objetivos, o software está sendo reescrito utilizando linguagem Java, com intuito de facilitar a comunicação via Internet, bem como ser uma linguagem multi-plataforma. A biblioteca de comunicação entre agentes será substituida pelo framework JADE (JADE..., 2009).

Além de mudanças de plataforma, a nova versão do software AMPLIA também prevê mudanças na interação com o usuário. Muitas vezes, a modelagem do conhecimento através de uma Rede Bayesiana "intimida" o usuário. Uma solução para este problema será a criação de um *front-end* para o editor de Redes Bayesianas. Este *front-end* trará uma interface mais amigável ao editor Bayesiana.

O foco deste trabalho é o editor de Redes Bayesianas. Ele é utilizado para a modelagem do domínio. Segundo Flores (FLORES et al., 2003), a abordagem Bayesiana foi escolhida por sua rigorosa fundamentação matemática. Ainda segundo a mesma autora, as Redes Bayesianas têm sido amplamente utilizadas, em todo mundo, para modelar domínios incertos. Para a criação do editor, foi utilizada uma versão modificada do software UnBBayes (UNBBAYES, 2009), desenvolvido pela Universidade de Brasília. Maiores detalhes sobre o UnBBayes e as modificações nele realizadas serão abordadas no próximo capítulo.

4 O UNBBAYES

Este capítulo apresenta UnBBayes (o logotipos do projeto é apresentado na figura 12), utilizado dentro da nova versão do AMPLIA como framework para o montagem e inferência em Redes Bayesianas. Nas próximas páginas, toda a arquitetura do software será apresentada, bem como as modificações realizadas na ferramenta serão disponibilizadas.

Figura 12: Logotipo do projeto UnBBayes

4.1 Visão geral do UnBBayes

O UnBBayes é um framework desenvolvido pelo Grupo de Inteligência Artificial da Universidade de Brasília (LADEIRA et al., 2003). Desenvolvido em Java, tem como principal objetivo dar suporte ao raciocínio probabilístico. O aplicativo conta com um editor visual que facilita o uso de Redes Bayesianas, Diagramas de Influência e Redes Bayesianas Multiplas Secionadas. O UnBBayes também suporta realização de inferência probabilística, propagação de evidências e aprendizagem de topologia em Redes Bayesianas. O software é distribuído sob licença GPL, tendo seu código fonte aberto.

Ainda segunda Ladeira (2003), o UnBBayes utiliza o método da árvore de junções para propagação de evidências e os algoritmos K2 B para a aprendizagem de Redes Bayesianas. O UnBBayes pode ser utilizado no desenvolvimento de aplicações para triagem e diagnósticos médico com interface gráfica baseada na metáfora dos procedimentos de diagnóstico médico (FLORES et al., 2001). A figura 13 apresenta a tela do UnBBayes.

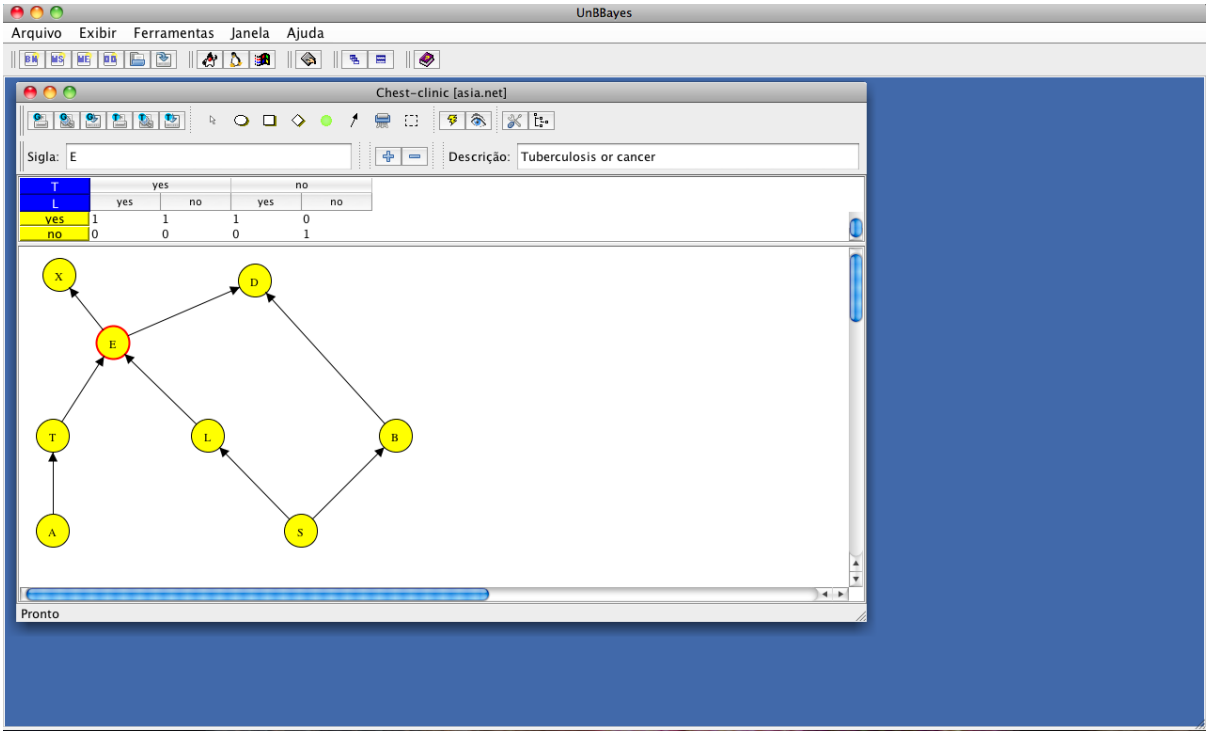


Figura 13: Tela da versão 3.7.29 do software UnBBayes, mostrando a Rede Bayesiana clássica conhecida como Asia

O método da árvore de junção foi amplamente discutido no capítulo dois. Para maiores referências sobre os algoritmos K2 e B, recomenda-se a leitura dos trabalhos de Cooper and Herskovits (1992) e Buntine (1991).

O software UnBBayes é constituído por sete pacotes principais, cada um deles cumprindo tarefa específica do aplicativo. São eles:

- **Aprendizagem**: implementa as classes relacionadas com o processo de aprendizagem em lote.
- **Controller**: controla as operações realizadas pelo usuários. Ele faz a ligação entre a interface gráfica e as funcionalidades.
- **Gui**: implementa a interface com o usuário.
- **Io**: responsável por exportar e importar as Redes criadas em arquivos. O UnBBayes trabalha com os formatos *net* e *xmlbif*.
- **Montecarlo**: utilizado para gerar amostras aleatórias das Redes Bayesianas.
- **Prs**: implementa as classes que definem a estrutura das Redes Bayesianas, bem como os algoritmos de inferência.

- **Util**: define as classes utilitárias.

4.2 Funcionalidades do UnBBayes

Esta sessão demonstra a arquitetura das principais funcionalidades do framework UnBBayes. Como as Redes Bayesianas e Diagramas de Influência são as funcionalidades mais presentes no AMPLIA, elas serão detalhadas na próxima seção. As demais funcionalidades serão tratadas de forma mais sucinta, apenas para fins de referência.

4.2.1 Redes Bayesianas e Diagramas de Influência

No UnBBayes, as funcionalidades de Redes Bayesianas e Diagramas de influência são representados por treze classes principais. Estas classes são detalhadas nos próximos parágrafos.

A classe *Node* implementa um nó genérico. Este nó possui uma posição x e y, um tamanho composto por altura e largura e um estado. Tanto o tamanho como a posição são implementados pela classe *SerializablePoint2D* (JAVA, 2009).

As variáveis que são visualizadas na árvore de nodos são representadas pela classe *TreeVariable*. Esta classe também armazena as probabilidades nas telas de controle e inferência em redes probabilísticas. As variáveis de decisão, variáveis probabilísticas, nodos utilidade, e arcos são representados, respectivamente, pelas classes *DecisionNode*, *ProbabilisticNode*, *UtilityNode* e *Edge*.

A classe *PotentialTable* implementa a tabela de potencial, armazenando números reais entre zero e um. Na classe *ProbabilisticTable* é implementada a tabela de potenciais de probabilidades. Já a classe *UtilityTable* implementa a tabela de potencial de utilidade.

Na interface *ITabledVariable* são definidas as variáveis com tabela para valores associados aos possíveis estados dos nodos. A classe *Network* representa um grafo. Já a classe *ProbabilisticNetwork* é responsável pela implementação de uma rede probabilística. O diagrama das classes responsáveis pela implementação das Redes Bayesianas e diagramas de influência, é exibido na figura 14.

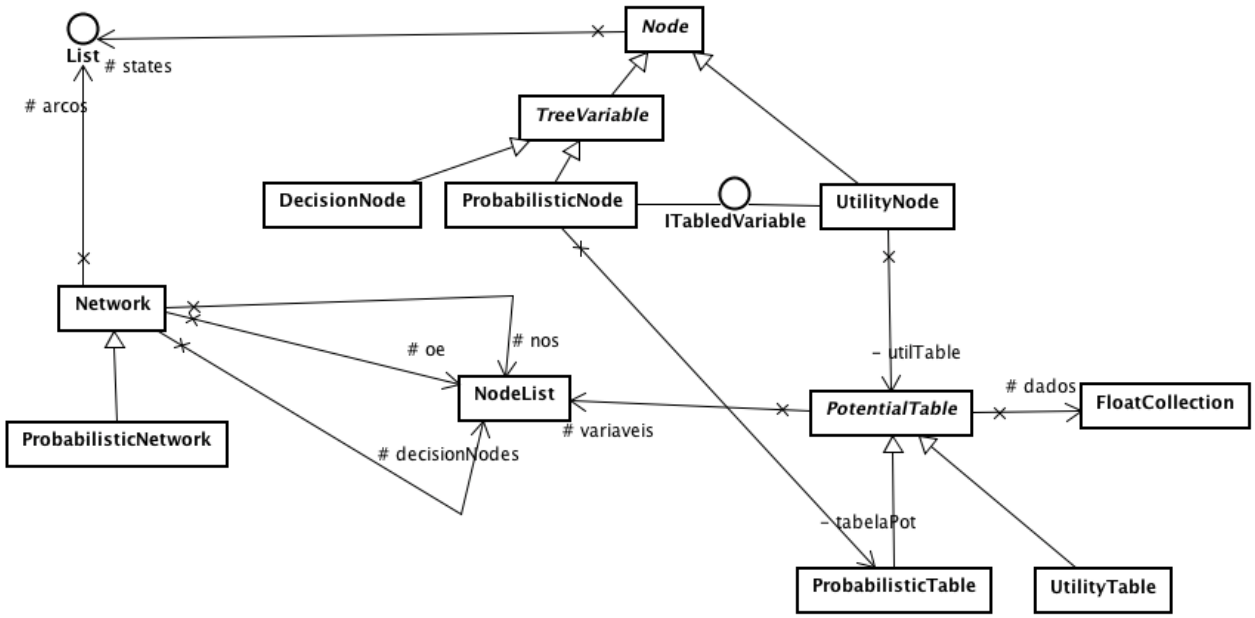


Figura 14: Modelagem das classes para Redes Bayesianas e Diagramas de influência (UNBBAYES, 2009)

4.2.2 Redes Bayesianas Multi-seccionadas

As Redes Bayesianas Multi-seccionadas são redes que têm seu domínio dividido em subredes. O resultado da inferência deve ser igual ao obtido com uma rede formada pela união destas subredes.

O UnBBayes utiliza a classe *SubNetwork* para implementar uma subrede. A classse *Linkage* possui os nodos responsáveis por ligar duas subredes. Os métodos utilizados para tratar a Rede Bayesiana Multi-seccionada são definidos na classe *AbstractMSBN*. O diagrama das classes responsáveis pela implementação das Redes Bayesianas Multi-seccionadas, é exibido na figura 15.

4.2.3 Outras funcionalidades do UnBBayes

Além de trabalhar com Diagramas de Influência e Redes Bayesianas, o UnBBayes é capaz de trabalhar com Redes Bayesianas Multi-entidades. Segundo Laskey (2006), Redes Bayesianas Multi-Entidades (MEBN) são redes que implementam Lógica de Primeira Ordem (FOL). O domínio de conhecimento em MEBN é expresso por MEBN Fragments (ou MFrags), organizadas em MEBN Theories (ou MTheories). Uma MTheory é um

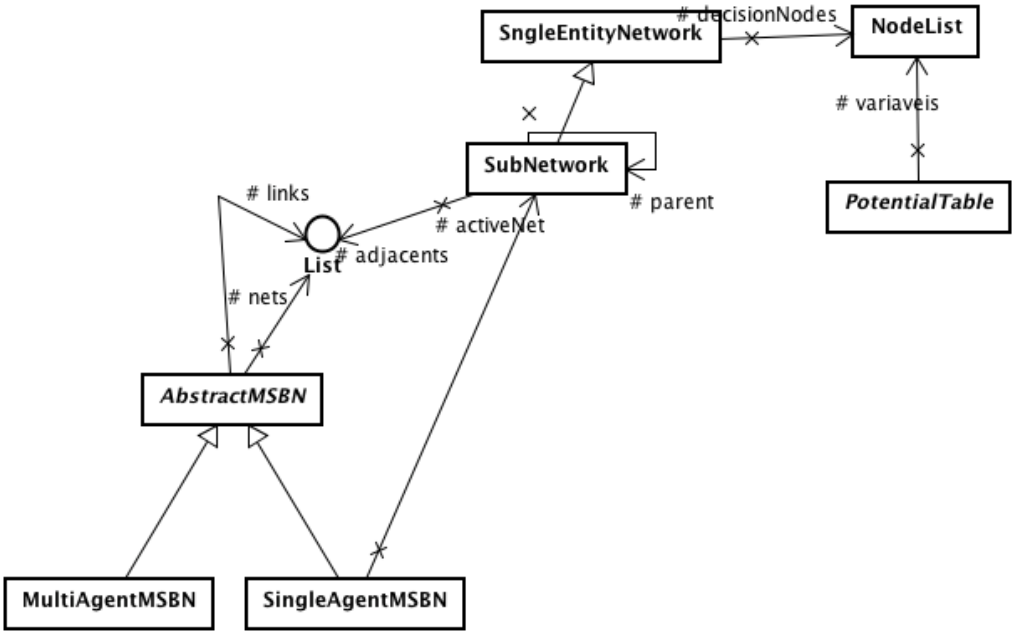


Figura 15: Modelagem das classes para Redes Bayesianas Multi-seccionadas (UNBBAYES, 2009)

conjunto d e MFrags que satisfaz determinadas condições, que garantem a existência de uma distribuição de probabilidade conjunta única sobre suas variáveis aleatórias.

5 MODIFICAÇÕES FEITAS NO UNBBAYES PARA ADAPTAÇÃO AO PROJETO AMPLIA

Para poder ser incorporado ao projeto AMPLIA, uma série de modificações foram realizadas na versão original do software UnBBayes. Este capítulo relata o desenvolvimento destas funcionalidades, bem como as dificuldades encontradas durante o processo e as necessidades futuras do projeto.

5.1 Ferramentas Utilizadas

Mantendo a linha de desenvolvimento do UnBBayes, toda a edição de código foi feita utilizando a IDE [1] Eclipse (ECLIPSE, 2009). Desenvolvido em Java, o Eclipse é um ambiente integrado de desenvolvimento, criado com o objetivo de facilitar a criação de aplicações Java. Através do uso de *plugins*[2], o Eclipse possibilita o desenvolvimento de aplicações também C, C++, Python, PHP, entre outros. O Eclipse é distribuido sobre os termos da *Eclipse Public License* (http://www.eclipse.org/legal/eplfaq.php), sendo software livre e tendo seu código fonte aberto. Neste projeto, foi utilizada a versão 3.4.1 do Eclipse IDE (*Ganymede*).

O Eclipse é uma das ferramentas de desenvolvimento de software mais populares para a tecnologia Java. O Eclipse é considerado uma das ferramentas essenciais ao se tratar de software livre.

O desenvolvimento do Eclipse teve início em 2001, sendo os principais membros na época a Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft e Webgain. Posteriormente, empresas como Oracle e JBos se juntaram ao projeto. Atualmente, o responsável pelo desenvolvimento da software é o Consórcio Eclipse.org (ECLIPSE, 2009), criado pela IBM, empresa responsável pelo desenvolvimento da ferramenta em sua fase inicial.

[1] IDE, do Inglês *Integrated Development Enviroment* ou Ambiente Integrado de Desenvolvimento.
[2] pequeno software que tem como objetivo adicionar funcionalidades a outros programas maiores.

Um dos maiores diferenciais do Eclipse é a flexibilidade proporcionada ao desenvolvedor. Durante o desenvolvimento em um workbench, isto é, um ambiente que pode ser configurado conforme suas necessidades com uso de perspectivas. Além de uma interface rica e de fácil entendimento (ver figura 16), o Eclipse faz uso de uma componente chamado *Method Completion*, que sugere e completa o código digitado.

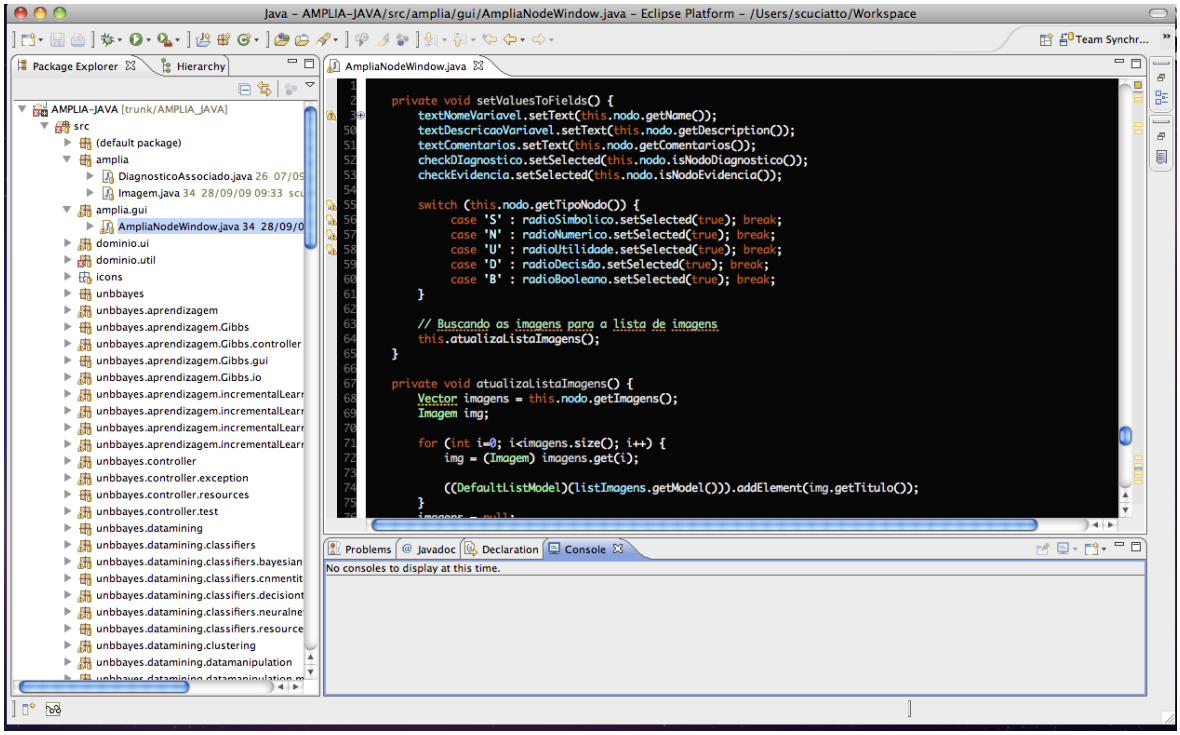


Figura 16: Tela do software Eclipse IDE

Para armazenamento de código e controle de versão, o projeto AMPLIA está sendo armazenado no *Google Code* (GOOGLE..., 2009), utilizando *subversion* como ferramenta de controle de versão. O *Google Code* é um site desenvolvido pelo *Google* para auxiliar desenvolvedores de software. O site provê uma série de recursos, como APIs, serviço de hospedagem e serviço de armazenamento de projetos.

O serviço de hospedagem de projetos *Google Code* implementa um sistema de controle de versão, oferecendo os sistemas *Subversion*[3] e *Mercurial*[4]. Além disso, o *Google Code* oferece um sistema de controle de problemas no software, um sistema de *wiki* para fins de documentação e um recurso para download de arquivos. O sistema é aberto para qualquer tipo de projeto, desde que o mesmo seja software livre e possua seu código-fonte aberto. A figura 17 demonstra o site Google Code, exibindo uma listagem de *commits* realizados.

[3]Subversion é um software livre para controle de versões (http://subversion.tigris.org/)

[4]Mercurial é um software livre para controle de versões, seu principal diferencial é a facilidade de uso(http://mercurial.selenic.com/)

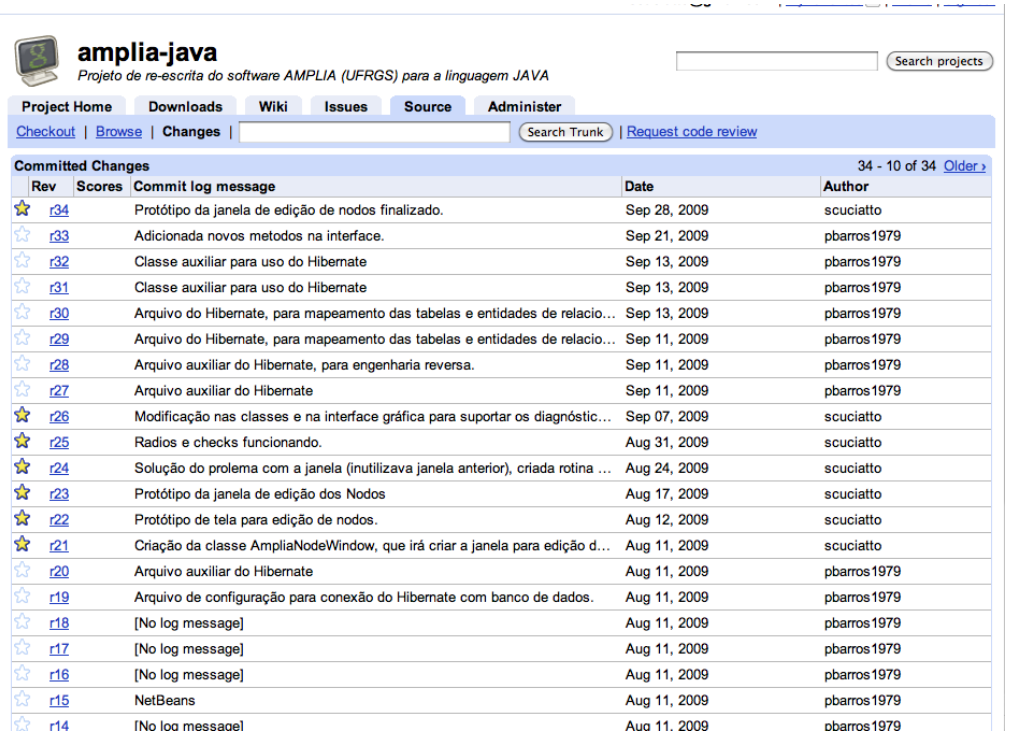


Figura 17: Tela do Google Code

O Eclipse já traz uma ferramenta para acesso à repositórios CVS. Porém, o projeto AMPLIA está hospedado no google code, que trabalha com SVN. Para manipulação do repositório SVN (hospedado pelo *Google Code*), está sendo utilizado o plugin *Subeclipse* (SUBCLIPSE, 2009). O plugin *Subeclipse* trabalha como uma interface gráfica para interação com repositórios SVN, dentro do ambiente do Eclipse. A figura 18 mostra o plugin *Subeclipse*, durante navegação pelo histórico do repositório.

Para montagem das telas, foi utilizado o plugin *Visual Editor* (VISUAL..., 2009). O *Visual Editor* é uma plataforma que disponibiliza uma série de componentes e ferramentas para a criação de interface gráfica. O plugin trabalha de forma integrada com o editor de códigos do Eclipse, desta forma, as implementações realizadas no ambiente gráfico são automaticamente visualizadas no editor de códigos. A figura 19 mostra o *Visual Editor* durante o desenvolvimento do editor de nós do AMPLIA. A tela está dividida entre o editor gráfico e o editor de código-fonte.

5.2 Funcionalidades desejadas ao editor de Redes Bayesianas

Além das características básicas de uma Rede Bayesiana, o projeto AMPLIA demanda uma série de novos componentes à Rede Bayesiana. A necessidade de armazenamento de

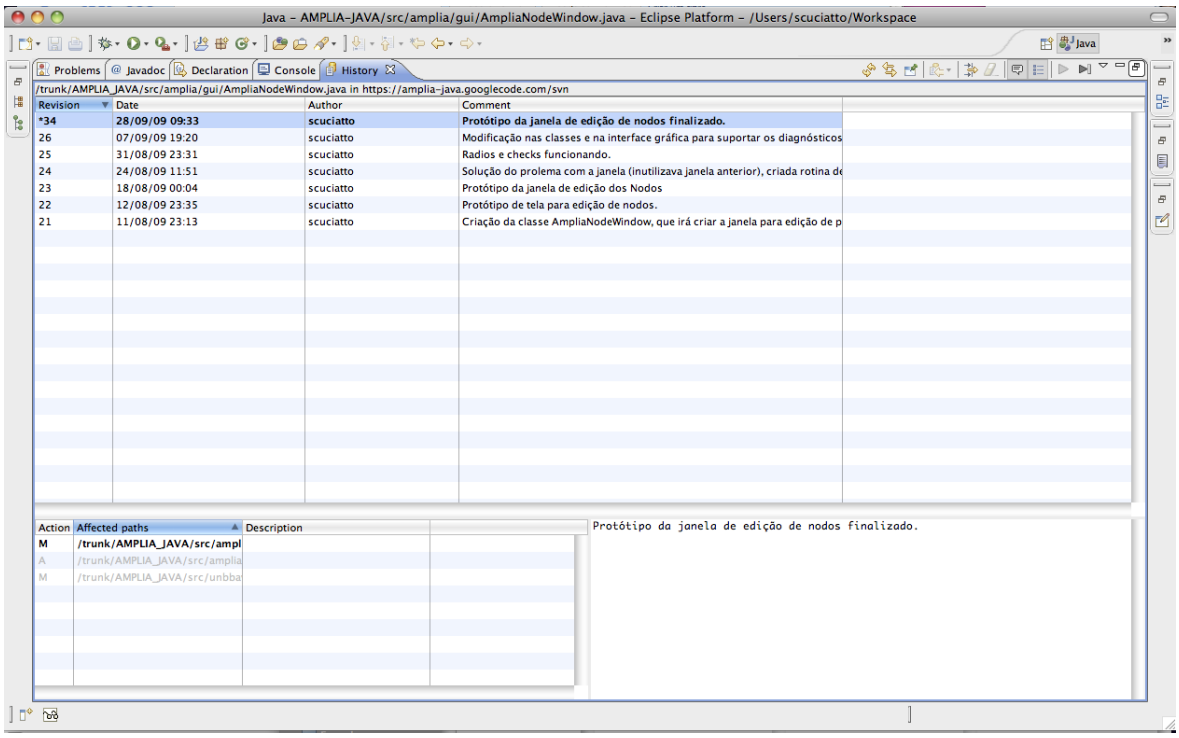


Figura 18: Captura de tela do plugin Subversion

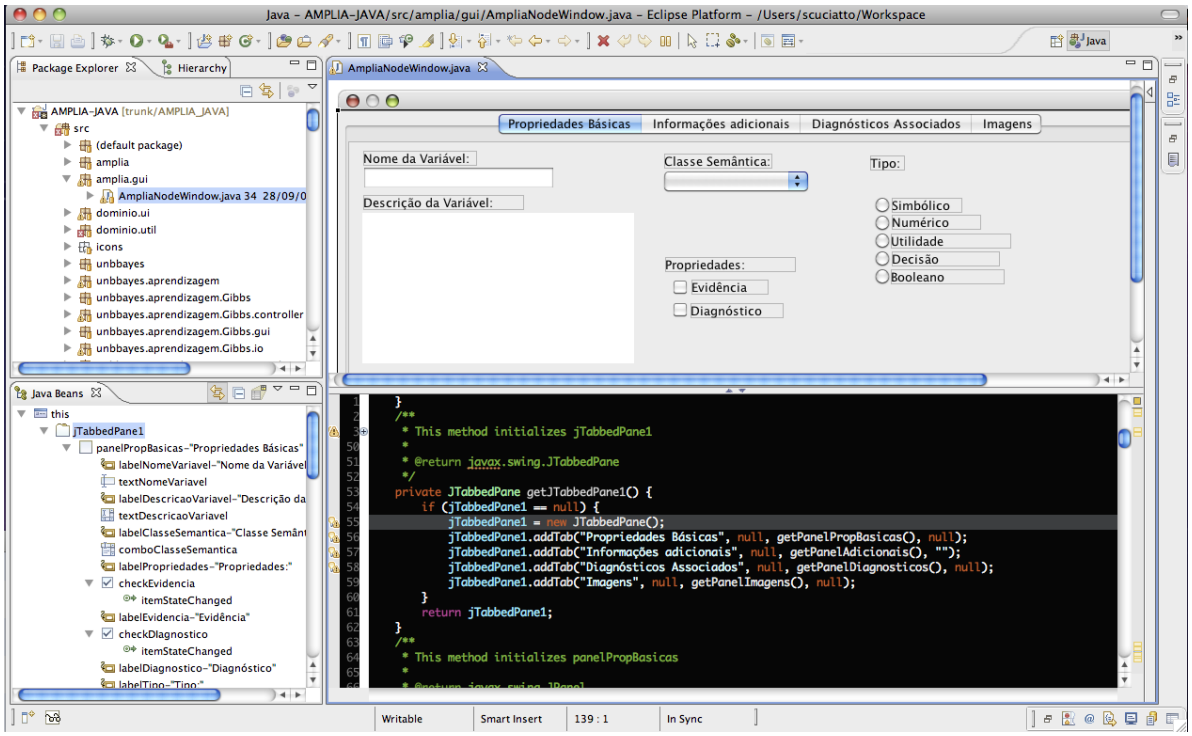


Figura 19: Captura de tela do plugin Visual Editor

inúmeras informações dentro dos nós, implica em uma série de mudanças nos algoritmos do UnBBayes.

5.2.1 Modificações nos nodos das Redes Bayesianas

Habitualmente, os nodos de uma Rede Bayesiana possuem como características um identificador (normalmente um nome e uma descrição), bem como a tabela de probabilidades condicionais ou incondicionais. Os nodos que possuem antecessores, possuem tabelas de probabilidades condicionais, que são as probabilidades afetadas por seus antecessores.

No AMPLIA, cada nodo armazena uma série de características específicas. Como é uma ferramenta voltada ao ensino, é necessário prover ao aluno o maior número de informações possíveis. As características específicas dos nodos são listadas abaixo.

- Classe Semântica: É utilizada para a classificação dos nós. Funciona como uma espécie de *tag* de identificação.
- Propriedades: Identifica se um nodo é do tipo evidência (nodo comum) ou diagnóstico. Nodos diagnóstico indicam um nodo que pode ser um resultado, um diagnóstico atingido. Estes nodos são utilizados pelo item diagnósticos associados.
- Tipo: Classifica um nodo como simbólico, numérico, utilidade, decisão e booleano.
- Comentários: Campo de texto livre, utilizado para a escrita de comentários diversos.
- Diagnósticos Associados: Diagnósticos associados ao nodo.
- Imagens: Imagens associadas ao nodo.

Todas as adaptações necessárias aos nodos foram desenvolvidas na classe *Node*, do pacote *unbbayes.prs*. Esta classe representa um nó genérico da rede e, como estas caracaterísticas são necessárias tanto para Redes Bayesianas como diagramas de influência, devem ser implementadas nesta classe.

As modificações se dão pela adição de atributos e métodos para manipulação dos mesmos. Os comentários e a classe semântica foram armazenados em atributos do tipo *String*. As imagens e os diagnósticos do nodo ficam armazenados em um objeto do tipo *Vector*. A implementação de imagens e diagnósticos demandou a criação de novas classes.

Além dos atributos citados anteriormente, foram criados dois atributos do tipo *boolean*, para identificar se um nodo é evidência ou diagnóstico. O tipo do nodo é implementado através de um atributo char. Este atributo tem seu valor atribuído através de um método que valida o mesmo. O atributo pode possuir valor "S" indicando que o nodo é simbólico,

"N" para nodos numéricos, "U" para nodos utilidade, "D" para decisão e "B" para nodos booleanos. As figuras 20 e 21 mostram parte do código-fonte da classe Node, responsável pela implementação dos nós das Redes Bayesianas e Diagramas de Influência.

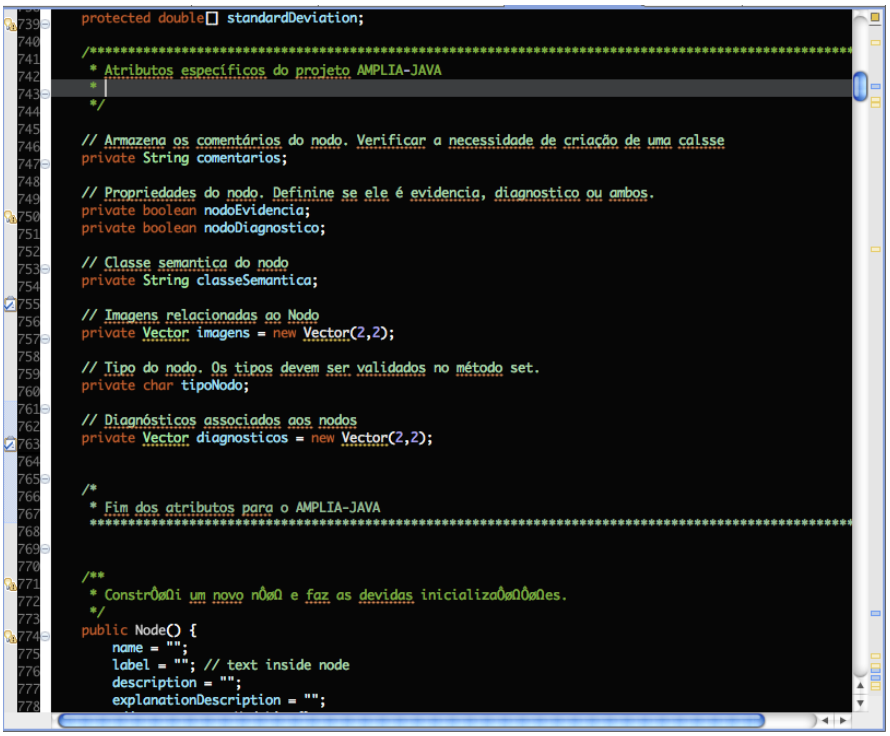
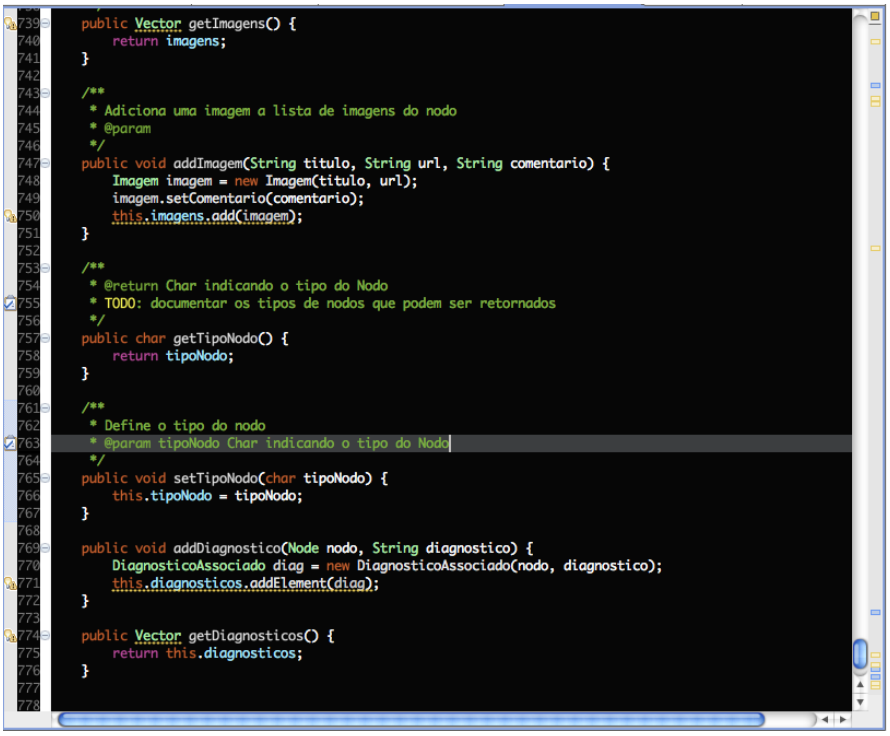


Figura 20: Captura de tela mostrando os atributos específicos do AMPLIA, na classe Node

Os nodos da Rede Bayesiana do AMPLIA podem possuir imagens associadas. Cada imagem é representada pela classe *Imagem*, dentro do pacote *amplia*. Esta classe, além do caminho para a imagem, armazena uma descrição e observações sobre a imagem. Como o nó pode possuir uma série de imagens associadas, a classe *Node* armazena todos objetos de imagem em um *Vector*. Para manipulação destas imagens, métodos para adição e remoção de imagens no *Vector* foram implementados. Uma captura de tela mostrando o código da classe *Imagem* é exibida na figura 22.

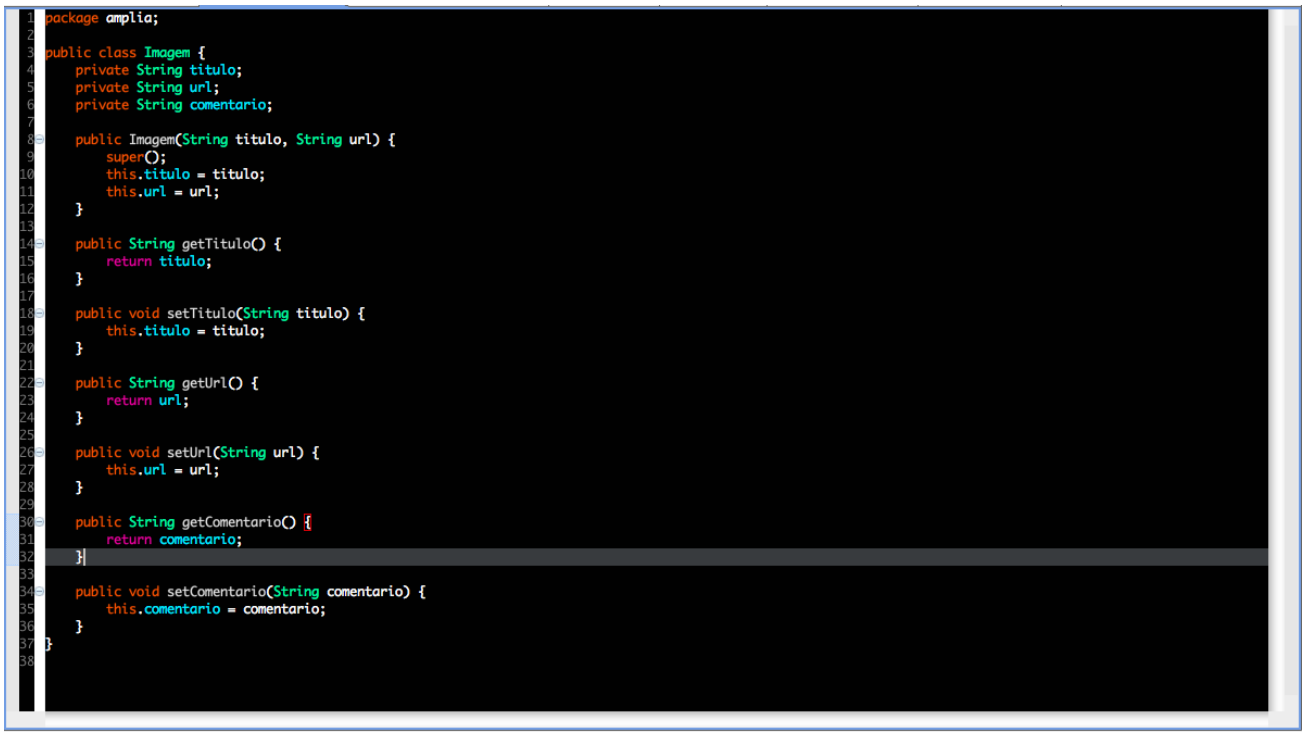
As imagens podem ser armazenadas localmente (na máquina em que o software está sendo executado) ou ser referenciada através de uma URL. Está sendo desenvolvido um projeto para criação de um banco de imagens, que futuramente será integrado ao AMPLIA. O projeto está sendo desenvolvido como trabalho de conclusão de curso pelo aluno Sandro Frazão Specht, nesta instituição, e as imagens serão obtidas diretamente deste banco.

Os diagnósticos associados seguem a mesma lógica utilizada nas imagens. Cada nó armazena seus diagnósticos em objeto do tipo *Vector*. Cada diagnóstico possui caracte-



```
    public Vector getImagens() {
        return imagens;
    }

    /**
     * Adiciona uma imagem a lista de imagens do nodo
     * @param
     */
    public void addImagem(String titulo, String url, String comentario) {
        Imagem imagem = new Imagem(titulo, url);
        imagem.setComentario(comentario);
        this.imagens.add(imagem);
    }

    /**
     * @return Char indicando o tipo do Nodo
     * TODO: documentar os tipos de nodos que podem ser retornados
     */
    public char getTipoNodo() {
        return tipoNodo;
    }

    /**
     * Define o tipo do nodo
     * @param tipoNodo Char indicando o tipo do Nodo
     */
    public void setTipoNodo(char tipoNodo) {
        this.tipoNodo = tipoNodo;
    }

    public void addDiagnostico(Node nodo, String diagnostico) {
        DiagnosticoAssociado diag = new DiagnosticoAssociado(nodo, diagnostico);
        this.diagnosticos.addElement(diag);
    }

    public Vector getDiagnosticos() {
        return this.diagnosticos;
    }
```

Figura 21: Captura de tela mostrando métodos específicos do AMPLIA, classe Node



```
package amplia;

public class Imagem {
    private String titulo;
    private String url;
    private String comentario;

    public Imagem(String titulo, String url) {
        super();
        this.titulo = titulo;
        this.url = url;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getComentario() {
        return comentario;
    }

    public void setComentario(String comentario) {
        this.comentario = comentario;
    }
}
```

Figura 22: Captura de tela mostrando o código de abertura da janela

rísticas específicas, como o nó ao qual ele está associado e seu diagnóstico. Estas características foram implementadas na classe *DiagnosticoAssociado*, dentro da classe *amplia*. A figura 23 mostra a implementação da classe.

```java
package amplia;

import unbbayes.prs.Node;

public final class DiagnosticoAssociado {
    private Node nodoAssociado;
    private String diagnostico;



    public DiagnosticoAssociado(Node nodoAssociado, String diagnostico) {
        this.nodoAssociado = nodoAssociado;
        this.diagnostico = diagnostico;
    }

    public Node getNodoAssociado() {
        return nodoAssociado;
    }

    public void setNodoAssociado(Node nodoAssociado) {
        this.nodoAssociado = nodoAssociado;
    }

    public String getDiagnostico() {
        return diagnostico;
    }

    public void setDiagnostico(String diagnostico) {
        this.diagnostico = diagnostico;
    }
}
```

Figura 23: Captura de tela mostrando a classe DiagnosticoAssociado

5.2.2 Modificações na interface gráfica do UnBBayes

O editor gráfico de Redes Bayesianas, implementado no UnBBayes, também precisou sofrer algumas alterações. Além da exibição e manipulação da tabela de probabilidades (iniciais e condicionais), sigla e descrição do nó, as nova propriedades precisam ser manipuladas via interface gráfica.

A classe *AmpliaNodeWindow* é responsável pela implementação do editor do nós. Seguindo o padrão de desenvolvimento do UnBBayes, a tela de edição de nós foi desenvolvida utilizando a API *Swing* (SWING..., 2009). A API *Swing*, que faz parte do pacote de classes fornecidas pela linguagem Java, emula a aparência de várias plataformas, também suportando uma aparência plugável, permitindo que as aplicações tenham uma aparência não vinculada a plataforma. A figura 24 demonstra parte do código JAVA/SWING que implementa a janela de edição de nós.

A tela de edição de nós foi dividida em quatro *Tabbed Panels*, que implementam abas na janela gráfica. Cada aba é responsável por agrupar determinados conjuntos de informações.

A aba propriedades básicas (figura 26) agrupa a maior parte das informações referentes ao nó. Nesta aba estão implementados os campos para nome e descrição do nó, bem

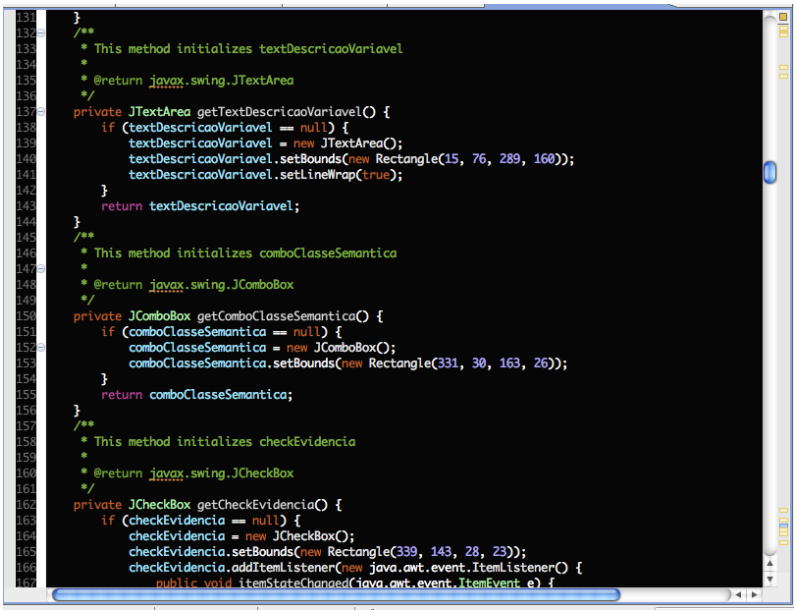


Figura 24: Trecho de codigo mostrando a implementação da AmpliaNodeWindow

como seletores para as propriedades do nó, tipo e classe semântica. Um exemplo de implementação das propriedades básicas é demonstrado na figura 25. O código em questão demonstra a implementação de dois *check-boxes*, onde o usuário seleciona se o nó é do tipo evidência ou diagnóstico.

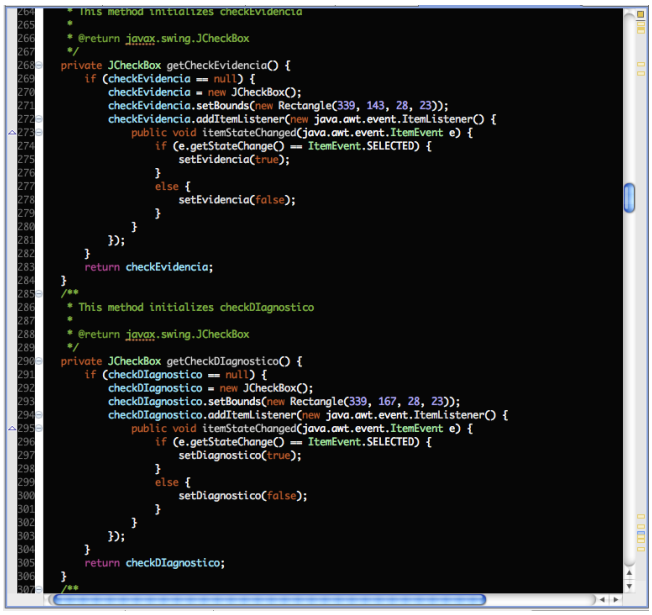


Figura 25: Parte da implementação das propriedades básicas

A aba informações adicionais (figura 27) guarda um campo de texto livre, para digitação de observações sobre aquela variável. Na aba diagnóstico associado (figura 28), pode-se selecionar qualquer nodo que tenha sua propriedade *diagnóstico* selecionada, e

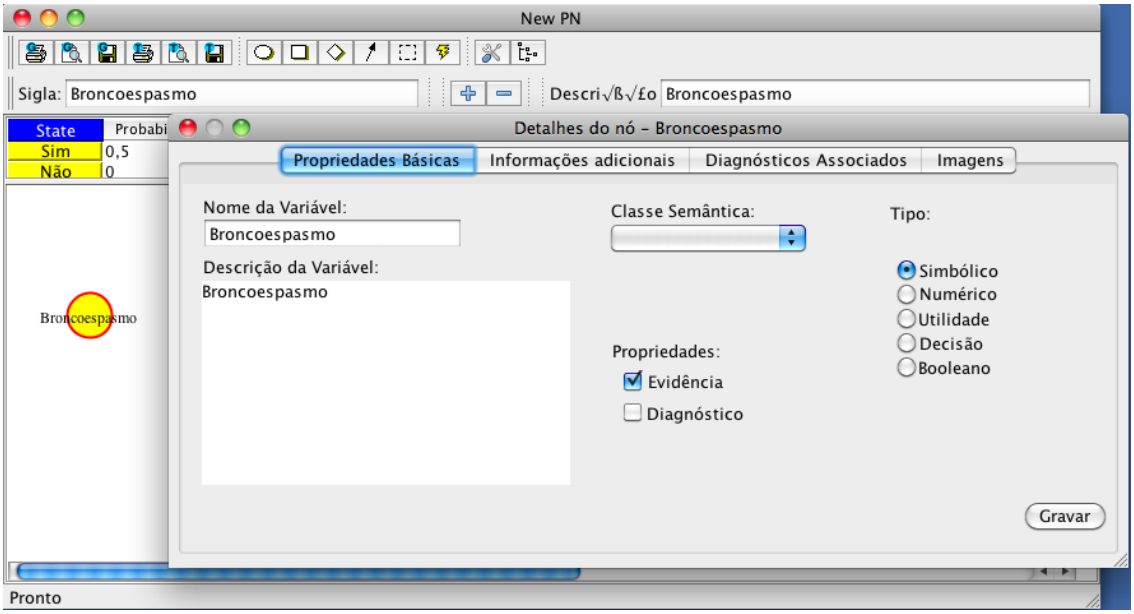


Figura 26: Captura de tela das propriedades básicas do nó

vinculá-lo ao nó. Este diagnóstico também possui um campo descritivo.

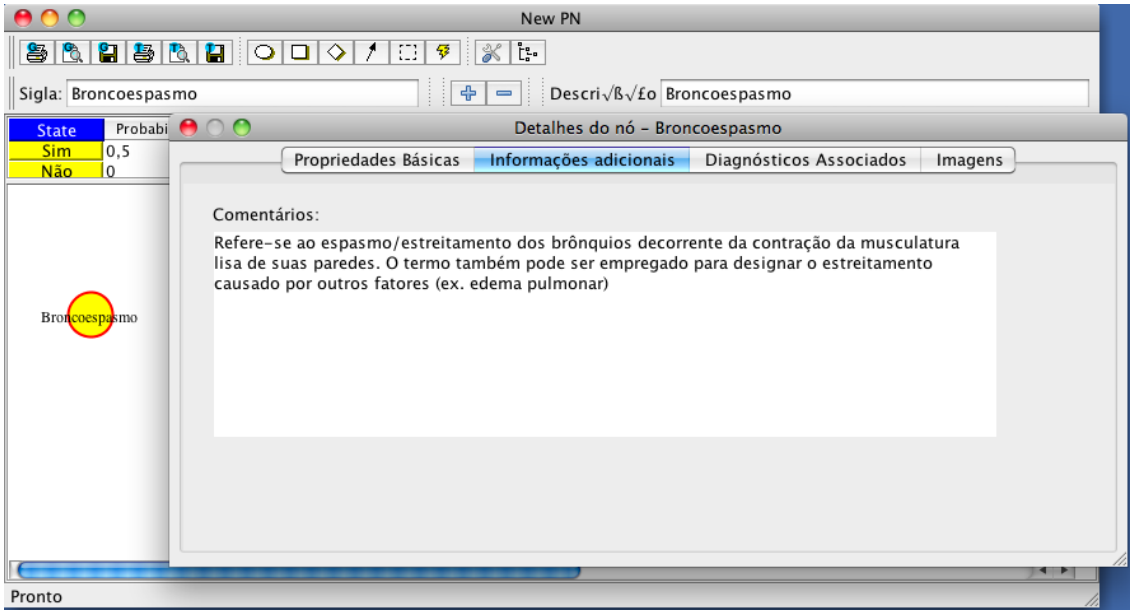


Figura 27: Captura de tela das propriedades adicionais do nó

A tela de seleção e descrição de diagnósticos associados (figura 28) é composta por uma caixa de seleção de nós e uma área de texto. A caixa de seleção de nós é populada com todos os nós que possuem a proprieadade *diagnóstico* selecionada. O código responsável pela varredura e preenchimento da caixa de seleção é demonstrado na figura 29.

A aba imagens (figura 30) agrupa as informações necessárias às imagens vinculadas ao nodo. Cada imagem pode possuir além de seu caminho, uma descrição e observações.

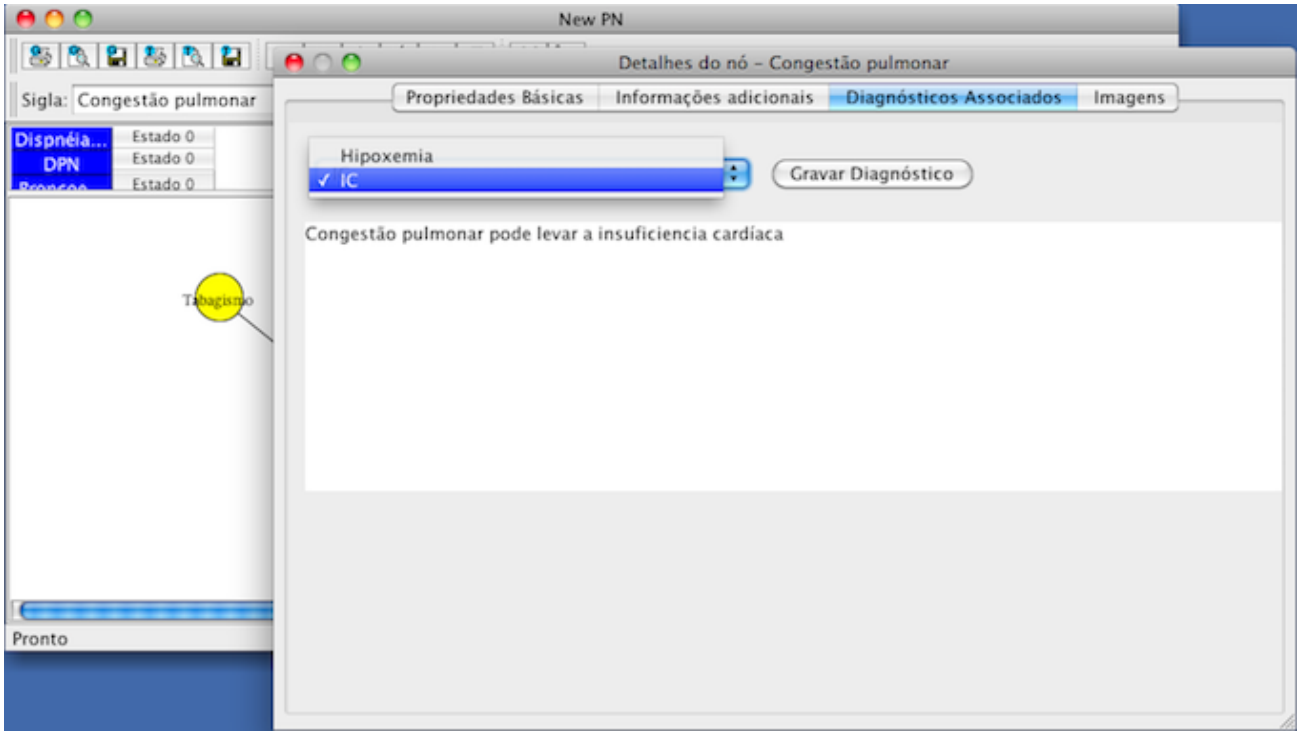


Figura 28: Captura de tela dos diagnósticos associados

```
        textCaminhoImagem.setText(img.getUrl());
        textObsImagem.setText(img.getComentario());
        labelImagem.setIcon(new ImageIcon(img.getUrl())); // TODO: resolver problema imagens na int
    }

    private void setComboDiagnosticos() {
        Node tmpNode;
        Vector nodeList = new Vector(2);

        for (int i = 0; i < nodes.size(); i++) {
           tmpNode = nodes.get(i);
           if (tmpNode.isNodoDiagnostico()) {
               nodeList.add(tmpNode.getName());
               comoNodoDiagnostico.addItem(tmpNode.getName().toString());
           }
        }
    }

    private void adicionaDiagnostico( ) {
        this.nodo.addDiagnostico(this.nodes.get(comoNodoDiagnostico.getSelectedIndex()), textDiagno
    }

    private String getTextoDiagnostico(int index) {
        Vector tmp = this.nodo.getDiagnosticos();
        if (!tmp.isEmpty()) {
            DiagnosticoAssociado diag = (DiagnosticoAssociado) tmp.get(index);
            return diag.getDiagnostico();
        }
        else {
            return null;
        }
    }
    /**
     * This method initializes botaoExcluir1
     *
     * @return javax.swing.JButton
     */
    private JButton getBotaoExcluir() {
        if (botaoExcluir == null) {
            botaoExcluir = new JButton();
            botaoExcluir.setBounds(new Rectangle(673, 258, 88, 29));
            botaoExcluir.setText("Excluir");
        }
        return botaoExcluir;
```

Figura 29: Implementação da tela de diagnósticos associados

Esta imagem, obviamente, deve ser exibida ao usuário. A imagem é demonstrada através da classe *Label*. A classe *Label* possui um método *setIcon*, que atribui uma imagem ao componente. A figura 31 mostra o código responsável pela exibição de imagens.

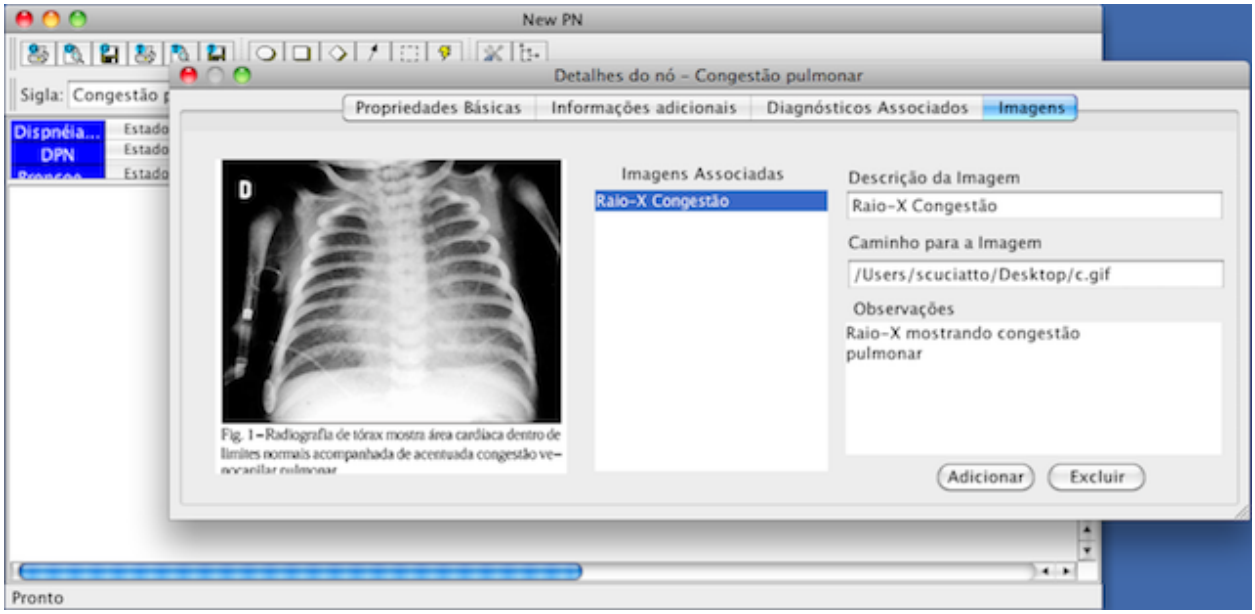


Figura 30: Implementação da tela de imagens

```
private void atualizaListaImagens() {
    Vector imagens = this.nodo.getImagens();
    Imagem img;

    for (int i=0; i<imagens.size(); i++) {
        img = (Imagem) imagens.get(i);

        ((DefaultListModel)(listImagens.getModel())).addElement(img.getTitulo());
    }
    imagens = null;
    img = null;
}

private void mostraDadosImagem(int index) {
    Vector imagens = this.nodo.getImagens();
    Imagem img;

    img = (Imagem) imagens.get(index);

    textTituloImagem.setText(img.getTitulo());
    textCaminhoImagem.setText(img.getUrl());
    textObsImagem.setText(img.getComentario());
    labelImagem.setIcon(new ImageIcon(img.getUrl()));
}

private void setComboDiagnosticos() {
    Node tmpNode;
    Vector nodeList = new Vector(2);

    for (int i = 0; i < nodes.size(); i++) {
       tmpNode = nodes.get(i);
       if (tmpNode.isNodoDiagnostico()) {
           nodeList.add(tmpNode.getName());
           comoNodoDiagnostico.addItem(tmpNode.getName().toString());
       }
    }
}

private void adicionaDiagnostico( ) {
    this.nodo.addDiagnostico(this.nodes.get(comoNodoDiagnostico.getSelectedIndex()), textDiagn
}
```

Figura 31: Implementação da tela da tela de imagens

5.2.3 Integração com a interface do UnBBayes

A integração entre a nova janela e o editor de Redes Bayesianas nativa do UnBBayes, se dá através da classe *GraphPane*. A classe *GraphPane* do pacote *unbbayes.gui*, é responsável pelo desenho da Rede Bayesiana na tela. Esta classe implementa as interfaces *MouseListener* e *MouseMotionListener* para tratar eventos de Mouse. A tela de edição

de nós é acionada toda vez que um duplo clique é dado sobre o mesmo, desta forma, o evento é capturado e a tela exibida. Parte do código de integração entre janela de edição de nós e interface da aplicação é demonstrado na figura 32.

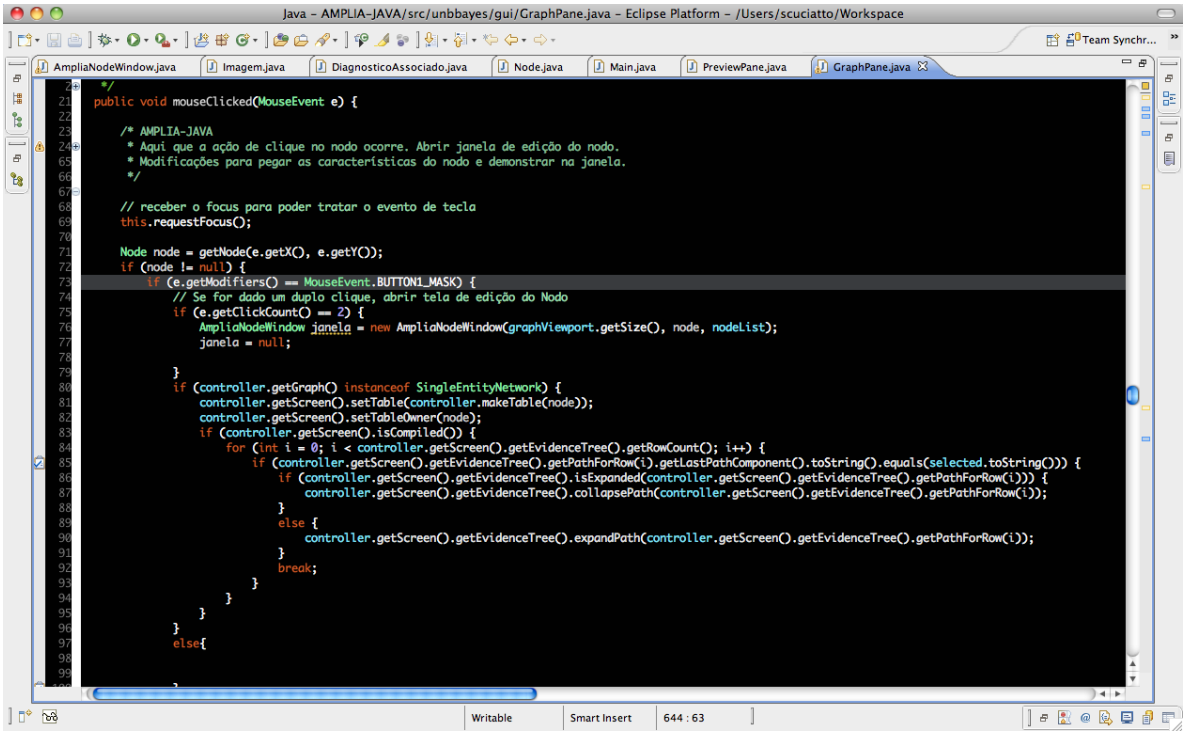


Figura 32: Captura de tela mostrando o código de abertura da janela

A edição da tabela de probabilidades e descrição das variáveis não foi modificada, sendo mantido o padrão do UnBBayes. Uma captura de tela mostrando a tabela de probabilidades, bem como uma rede completa pode ser vista na figura 33.

5.3 Propagação de evidências na rede montada

A propagação de evidências (também conhecida como compilação) da Rede Bayesiana montada, se dá através do algoritmo de árvores de junção. O algoritmo de árvores de junção foi amplamente tratado no capítulo dois deste trabalho. Basicamente, o algoritmo trabalha dividindo em estruturas auxiliares (árvores de junção), e realizando inferência sobre elas. Parte do código de implementação do algoritmo de árvores de junção pode ser visualizado na figura 34.

As implementações apresentadas neste capítulo, tornaram possível o uso do UnBBayes como ferramenta de edição de Redes Bayesianas dentro do AMPLIA. Esta capítulo demonstrou as alteração feitas no editor de Redes Bayesianas do UnBBayes, de forma que

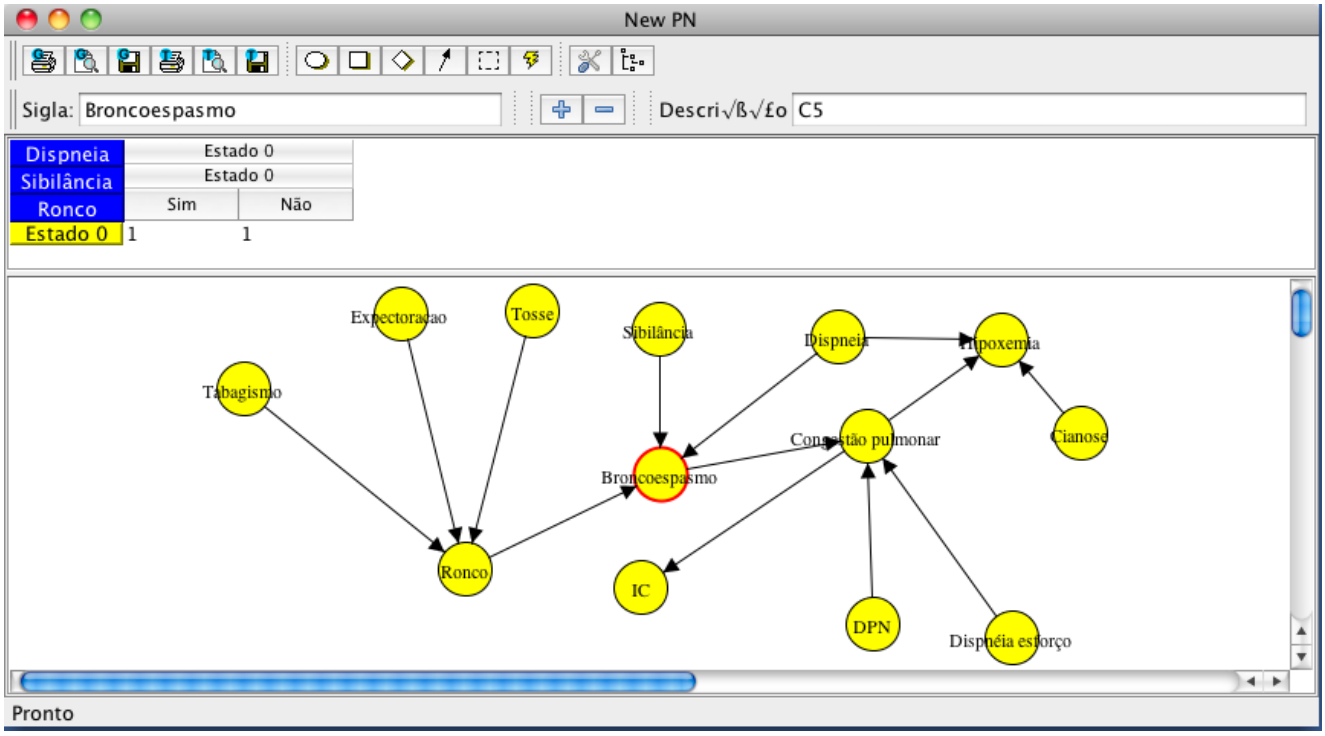
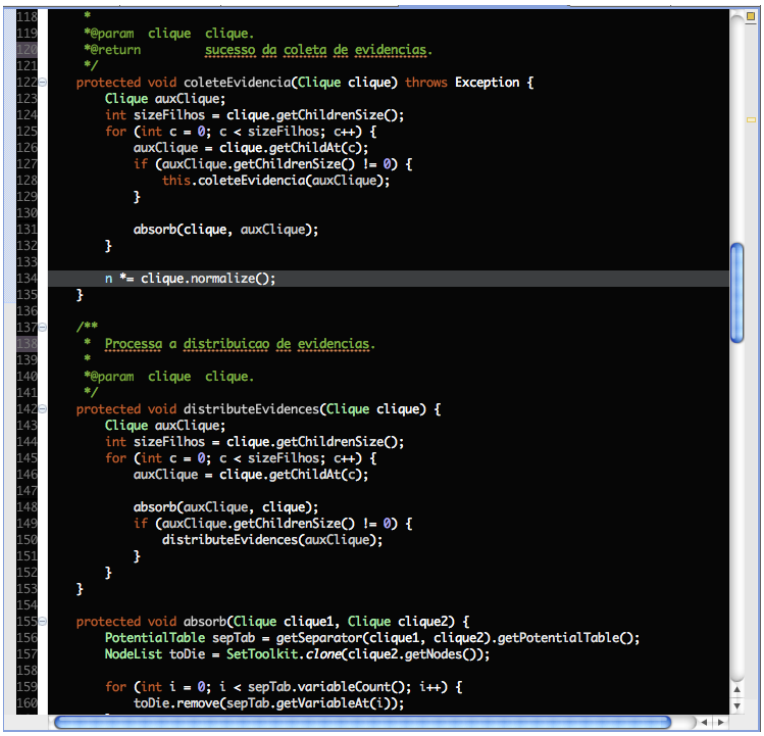


Figura 33: Captura de tela mostrando a tabela de probabilidades



```
     *
     *@param  clique  clique.
     *@return         sucesso da coleta de evidencias.
     */
    protected void coleteEvidencia(Clique clique) throws Exception {
        Clique auxClique;
        int sizeFilhos = clique.getChildrenSize();
        for (int c = 0; c < sizeFilhos; c++) {
            auxClique = clique.getChildAt(c);
            if (auxClique.getChildrenSize() != 0) {
                this.coleteEvidencia(auxClique);
            }

            absorb(clique, auxClique);
        }

        n *= clique.normalize();
    }

    /**
     *  Processa a distribuicao de evidencias.
     *
     *@param  clique  clique.
     */
    protected void distributeEvidences(Clique clique) {
        Clique auxClique;
        int sizeFilhos = clique.getChildrenSize();
        for (int c = 0; c < sizeFilhos; c++) {
            auxClique = clique.getChildAt(c);

            absorb(auxClique, clique);
            if (auxClique.getChildrenSize() != 0) {
                distributeEvidences(auxClique);
            }
        }
    }

    protected void absorb(Clique clique1, Clique clique2) {
        PotentialTable sepTab = getSeparator(clique1, clique2).getPotentialTable();
        NodeList toDie = SetToolkit.clone(clique2.getNodes());

        for (int i = 0; i < sepTab.variableCount(); i++) {
            toDie.remove(sepTab.getVariableAt(i));
```

Figura 34: Captura de tela mostrando parte do algoritmo de árvores de junção

ele possa de adaptar ao projeto AMPLIA. Desta forma, pode-se utilizar a facilidade de um editor gráfico unido a poderosos algoritmos de inferência, responsáveis pela solução da rede.

CONCLUSÃO

O início deste trabalho se deu com uma ampla pesquisa sobre Redes Bayesianas, pesquisa esta que proporcionou um grande referencial teórico sobre o assunto. O capítulo um abordou a teoria da probabilidade, apresentando o axioma base da teoria e como trabalhar com probabilidades incondicionais.

Os conceitos da teoria da probabilidade foram expandidos de forma a apresentar o cálculo de probabilidades condicionais, aplicando o Teorema de Bayes para calculo de probabilidades, levando em conta o surgimento de novas evidências. O Teorema de Bayes dá a base para a inferência Bayesiana, que prega a tomada de decisões baseadas nos métodos estatísticos citados anteriormente.

Ainda no capítulo um, trata-se o problema dos grandes escopos: como analisar problemas complexos usando o Teorema de Bayes. Como solução, as Redes Bayesianas são apresentadas. As Redes Bayesianas tomam proveito de sua estrutura e exploram a relação entre diferentes variáveis, propagando crenças entre elas e atualizando probabilidades. Desta forma, as Redes Bayesianas conseguem calcular as probabilidades conjuntas de um grande número de variáveis, e efetuando inferência sobre as mesmas.

Após apresentação da teoria de Redes Bayesianas no primeiro capítulo, o trabalho trata sobre a realização de inferência nas redes. O capítulo dois discorre sobre a resolução de uma Rede Bayesiana, apresentando os principais algoritmos utilizado na realização desta tarefa. Neste capítulo uma análise do algoritmos é feita, apresentando o principal problema da inferência exata de uma rede, a complexidade NP-Difícil (ℓNP-Hard)) dos mesmos. O segundo capítulo ainda demonstra os principais algoritmos de inferência, tanto exata como aproximada, relatando as características de cada um.

Este trabalho relata a pesquisa e desenvolvimento de um módulo do software AMPLIA. O capítulo três apresenta o projeto, mostrando todas as características desejadas ao software, bem como a arquitetura seguida por versões anteriores. O segundo capítulo ainda cita as características desejadas à nova versão do software AMPLIA.

O editor de Redes Bayesianas desenvolvido aqui, é parte essencial do Agente de Domínio do projeto AMPLIA. O Agente de Domínio está sendo desenvolvido, como trabalho de conclusão de curso nesta Instituição, pelo aluno Paulo Ricardo Muniz Barros. Como este

projeto é amplo, sua implementação não se resume apenas à este trabalho. Os próximos passos são a união do editor Bayesiano ao Agente de Domínio, citado anteriormente.

O Agente de domínio, compara a rede construída pelo aluno com a rede construída pelo especialista, que deve identificar os prováveis conflitos. O resultado desta comparação é encaminhado para o Agente Mediador, que será responsável pela seleção das estratégias pedagógicas utilizadas.

Por fim, conclui-se este trabalho tendo adquirido conhecimento na implementação e análise de Redes Bayesianas. Este trabalho, bem como os demais que fazem parte do projeto AMPLIA, tem como foco a re-engenharia do software, voltado para a WEB. Trabalhos futuros ocorrerão a fim de implementar melhorias no AMPLIA, principalmente no que diz respeito à interação dos usuários com as Redes probabilísticas.

Referências

ALDRICH, J. R. a. fisher on bayes and bayes theorem. *Joural of the International Society for Bayesian Analysis*, v. 1, n. 1, p. 161–170, 2008.

BORSUK, M. E.; BURKHARDT-HOLM, P.; REICHERT, P. *A Bayesian network for investigating the decline in fish catch in Switzerland.* [S.l.], 2002.

BUNTINE, W. Theory refinement on bayesian networks. *Conference on Uncertainty in Artificial Intelligence, 7*, Morgan Kaufmann, p. 52–60, 1991.

CASTILLO, E.; GUTIÉRREZ, J. M.; HADI, A. S. *Sistemas Expertos e Modelos de Redes Probabilisticas.* [S.l.]: Academia Espanola de Ingenieria, 1998.

CHARNIAK, E. Bayesian networks without tears. *AI Magazine*, 1991.

COOPER, G. F.; HERSKOVITZ, E. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, n. 9, p. 309–347, 1992.

DECHTER, R. Bucket elimination: A unifying framework for probabilistic inference. 1996.

ECLIPSE. Novembro 2009. Disponível em: <www.eclipse.org>.

FLORES, C. et al. Projeto amplia - uso de informática na educação médica. 2003.

FLORES, C. D. *Negociação Pedagógica Aplicada a um Ambiente Multiagente de Aprendizagem Colaborativa.* Tese (Doutorado em Ciência da Computação) — PPGC / UFRGS, 2005.

FLORES, C. D. et al. Uma experiência do uso de redes probabilísticas no diagnóstico médico. *Informática Médica*, v. 8, p. 25–29, 2001.

FRIEDMAN, N. et al. Using bayesian networks to analyz expression data. In: *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology.* [S.l.: s.n.], 2000.

GALáN, S. F. et al. Nasonet, joining bayesian networks and time to model nasopharyngeal cancer spread. In: *AIME '01: Proceedings of the 8th Conference on AI in Medicine in Europe.* London, UK: Springer-Verlag, 2001. p. 207–216.

GAREY, M. R.; JOHNSON, D. S. *Computers and Intractability. A guide to the Theory of NP-Completeness.* [S.l.]: W. H. Freeman, 1979.

G.F.COOPER. *Probabilistic Inference Using Belief Networks is NP-Hard.* [S.l.], 1991.

GLUZ, J. a C.; VICCARI, R. M. Linguagens de comunicação entre agentes: Fundamentos, padrões e perspectivas. In: *JORNADA DE MINI-CURSOS DE INTELIGÊNCIA ARTIFICIAL.* [S.l.: s.n.], 2003. p. 53–102.

GOOGLE Code. Novembro 2009. Disponível em: <code.google.com>.

GRAHAM, P. A plan for spam. *http://www.paulgraham.com/spam.html*, 2002.

HORVITZ, E. et al. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. *Microsoft Research*, 1996.

JADE - Java Agent Development Framework. Outubro 2009. Disponível em: <jade.tilab.com>.

JAVA. Outubro 2009. Disponível em: <java.sun.com>.

JENSEN, F. V. *An Introduction to Bayesian Networks*. [S.l.]: UCL Press, 1996.

JENSEN, F. V.; JENSEN, F. Optimal junction trees. In: *In UAI*. [S.l.]: Morgan Kaufmann, 1994. p. 360–366.

KJAERULFF, U. *Triangulation of Graphs - Algorithms Giving Small Total State Space*. [S.l.], 1990.

KJAERULFF, U. B.; MADSEN, A. L. *Probabilistic Networks - An introduction to Bayesian Networks and Influence Diagrams*. [S.l.: s.n.], 2005.

KOLLER, D. et al. Graphical models in a nutshell. In: GETOOR, L.; TASKAR, B. (Ed.). *Introduction to Statistical Relational Learning*. [S.l.]: MIT Press, 2007.

LADEIRA, M. et al. In encontro nacional de inteligÊncia artificial - enia. In: *Ferramenta Aberta e Independente de Plataforma para Redes Probabilística*. [S.l.: s.n.], 2003.

LADEIRA, M.; VICARI, R. M. *Algorítmos de Inferência em Redes Probabilísticas Baseados em Árvore de Junções*. [S.l.], 1999.

LASKEY, K. B. *First-Order Bayesian Logic*. Tese (PhD thesis) — George Mason University, 2006.

MACEDO, A. R. de. Resolução cne/ces 4/2001. In: *Diário Oficial da União*. [S.l.: s.n.], 2001. p. 38.

MARTINS1, V. et al. Uma abordagem de fusão de sinais vitais baseada em redes bayesianas. In: *IV Latin American Congress on Biomedical Engineering 2007, Bioengineering Solutions for Latin America Health*. [S.l.: s.n.], 2007. p. 178–182.

MOORE, D. S. *A Estatística Básica e Sua Prática*. [S.l.]: LTC, 2005.

NEAPOLITAN, R. E. *Learning Bayesian Networks*. 1. ed. [S.l.]: Prentice Hall, 2003.

ONISH, M. S.; CARVALHO, R. N. de. *Framework e API para Construção de Sistemas Inteligentes Baseados em Diagrama de Influências e Rede Bayesiana Múltipla Secionada*. Tese (Bacharelado em Ciência da Computação) — Universidade de Brasília, 2003.

PEARL, J. *Evidential Reasoning Using Stochastic Simulation of Causal Models*. [S.l.], 1987.

PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. [S.l.]: Morgan Kaufmann, 1988.

RUSSEL, S.; NORVIG, P. *Inteligencia Artificial*. 2nd. ed. [S.l.]: Editora Campus, 2004.

SAHAMI, M. et al. A bayesian aproach to filering junk e-mail. *AAAI'98 Workshop on Learning for Text Categorization*, 1998.

SILVA, A. L. B.; LADEIRA, M. B.; OLIVEIRA, M. P. V. de. Aplicação do modelo de redes bayesianas para o gerenciamento de risco de ruptura em cadeias de suprimento. *XI Simpósio de Administração da Produção,Logística e Operações Internacionais - SIMPOI 2008*, 2008.

SUBCLIPSE. Novembro 2009. Disponível em: <subclipse.tigris.org>.

SWING (Java Foundation Classes). Novembro 2009. Disponível em: <java.sun.com/javase/6/docs/technotes/guides/swing>.

UNBBAYES. Junho 2009. Disponível em: <unbbayes.sourceforge.net>. Acesso em: 15/08/2009.

VICCARI, R. M. et al. Multi-agent intelligent environment for medical knowledge. In: *Artificial Intelligence in Medicine*. [S.l.: s.n.], 2003. p. 335–366.

VISUAL Editor Project. Novembro 2009. Disponível em: <www.eclipse.org/vep>.

XIANG, Y.; POOLE, D.; BEDDOES, M. P. Multiply sectioned bayesian networks and junction forests for large knowledge based systems. *Computational Intelligence*, Cambridge, v. 9, n. 2, p. 171–220, 1993.

ZHANG, N.; POOLE, D. A simple approach to bayesian network computations. In: *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*. [S.l.: s.n.], 1994. p. 171–178.

ZHANG, N. L.; POOLE, D. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, v. 5, p. 301–328, 1996.

ANEXO A

Algoritmo de triangularização baseado na Heurística do Peso Mínimo.

```
peso mínimo (N,Adj,S,n,GT) {
for (i=1; i<=n; i++) {
no=N(i); Adjt(no)=Adj(no);
N1(no)=N(no); Adj1(no)=Adj(no);
    }
    n1 = n; ordem=0;
    triangular (N1,Adj1,Adjt,S,Eo,n1,ordem);
    return (N,Adjt,Eo,n);
}

triangular (N1,Adj1,Adjt,S,Eo,n1,ordem) {
algum = true;
while (algum) do {
algum = false;
for (i=1; i<=n1; i++) {
no = N1(i); na = |Adj1(no)|;
if (na <= 1) c = 0;
else c = cordas(N1,Adj1,no,na);
if (c > 0) continue;
for (j=1; j<=na; j++) {
v = Adj1(no)(j);
Adj1(v) = Adj1(v) - no;
}
N1 = N1 - no; algum = true; ordem++;
Eo(no) = ordem; n1 -= 1;
}
}
if (n1==0) return(N,Adjt,Eo);
no = peso (N1,Adj1,S,n1);
```

```
elimine (N1,Adj1,Adjt,no);
ordem++; Eo(no)=ordem; n1 -= 1;
triangularize (N1,Adj1,Adjt,S,Eo,n1,ordem);
}

cordas (N1,Adj1,no,na) {
// Retorna 1 se requer inserir cordas para poder
// eliminar o nó no. Caso contrário retorna 0.
// na: # de adjacentes do nó no.

    for (i=1; i<=na; i++) {
     a1 = Adj1(no)(i);
     for (j=i+1; j<=na; j++) {
     a2 = Adj1(no)(j);
     na2 = |Adj1(a2)|;
     achou = false;
     for (k=1; k<=na2; k++) {
     if (a1 != Adj1(a2)(k)) continue;
     achou = true;
     break; // Finalize a busca pelo for k.
            }
            if (!achou) return 1;
     }
     }
     return 0;
}

peso (N1,Adj1,S,n1) {
 nomin = N1(1);
 na = |Adj1(nomin)|;
 pmin = log2(S(nomin));
 for (i=1; i<=na; i++) {
  v = Adj1(nomin)(i);
  pmin = pmin + log2(S(v));
     }
```

```
      for (i=2; i<=n1; i++) {
       no = N1(i);
       p = log2(S(no));
       na = |Adj1(no)|;
       for (j=1; j<=na; j++) {
       v = Adj1(no)(j);
       p = p + log2(S(v));
         }
         if (p < pmin) {
            pmin = p;
            nomin = no;
         }
      }
      return nomin; mínimo.
}

elimine (N1,Adj1,Adjt,no) {
na = |Adj1(no)|;
for (i=1; i<=na; i++) {
        a1 = Adj1(no)(i);
        for (j=i+1; j<=na; j++) {
            a2 = Adj1(no)(j);
            na2 = |Adj1(a2)|;
            achou = false;
            for (k=1; k<=na2; k++) {
             if (a1 != Adj1(a2)(k)) continue;
             achou = true;
             break;
            }
            if (!achou) {
               Adj1(a1)=Adj1(a1) U a2;
               Adj1(a2)=Adj1(a2) U a1;
               Adjt(a1)=Adjt(a1) U a2;
               Adjt(a2)=Adjt(a2) U a1;
            }
```

```
        }
        for (i=1; i<=na; i++) {
         a1 = Adj1(no)(i);
         Adj1(a1) = Adj1(a1) - no;
        }
        N1 -= no;
        return (N1,Adj1,Adjt);
    }
}
```