UNIVERSIDADE FEEVALE

FERNANDO ALEX HELWANGER

UM EDITOR DE REDES BAYESIANAS COM FOCO EM USABILIDADE

Novo Hamburgo
2016

FERNANDO ALEX HELWANGER

UM EDITOR DE REDES BAYESIANAS COM FOCO EM USABILIDADE

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel em
Ciência da Computação pela
Universidade Feevale

Orientador: Paulo Ricardo Muniz Barros
Coorientador: Alessandro Peixoto de Lima

Novo Hamburgo
2016

AGRADECIMENTOS

Gostaria de agradecer a todos que, de alguma forma, contribuíram para a realização deste trabalho, em especial:

Aos professores da Universidade Feevale que, ao longo dos últimos anos, me propuseram ensinamentos valiosos e me fizeram crescer a ponto de ser capaz de realizar este trabalho.

Ao meu orientador, professor Paulo, que a todo momento soube me direcionar para o caminho correto durante a realização do trabalho.

À minha esposa, Priscila, que também realizou o seu trabalho de conclusão ao mesmo tempo que eu, mas mesmo assim ficou ao meu lado e me apoiou o tempo todo.

À minha família, que compreendeu os momentos de ausência e nunca deixou de me dar carinho e apoio durante este período.

Aos meus amigos, por me apoiarem e darem sugestões durante o desenvolvimento do trabalho.

Aos participantes da etapa de validação, por terem participado e contribuído bastante ao trabalho.

RESUMO

Existem problemas que por natureza possuem resultados incertos, nos quais as variáveis envolvidas definem a probabilidade de determinado evento acontecer. Um exemplo é o processo de diagnóstico médico, no qual os sintomas do paciente definem a probabilidade da existência de determinada enfermidade. Para este tipo de problema, redes bayesianas podem ser empregadas como uma maneira simples de representação. Com o auxílio desta rede é possível simular diferentes resultados alterando as variáveis de entrada, o que se mostra bastante útil para simuladores no ensino. Porém, os softwares para edição de redes bayesianas existentes mostram-se com uma usabilidade aquém do necessário para boa parte de seus usuários. Como redes bayesianas podem ser empregadas em diversos domínios, os usuários dos softwares para edição nem sempre possuem uma grande experiência com computação e acabam encontrando dificuldades na criação destas redes. O objetivo deste trabalho é fazer uma avaliação da usabilidade dos softwares para edição de redes bayesianas existentes e, a partir dos resultados dessa avaliação, desenvolver um novo software que supra a necessidade de seus usuários sem que eles sejam afetados por uma baixa usabilidade. Para que o desenvolvimento desse software fosse possível, também foi criada uma biblioteca de inferência em redes bayesianas que pode ser reutilizada em outros projetos. Ao final, foi realizada uma validação, utilizando testes de usabilidade com usuários, na qual os resultados indicam uma melhora na usabilidade do editor desenvolvido em relação a outros estudados neste trabalho.

Palavras-chave: Redes bayesianas. Usabilidade. Editor.

ABSTRACT

There are problems that by nature have uncertain results, in which the variables involved define the probability of a given event happen. An example is the medical diagnostic procedure, in which the patient's symptoms define the likelihood of a certain condition. For this type of problem, Bayesian networks can be used as a simple way of representation. With the help of such network, the user can simulate different results by changing the input variables, which proves to be quite useful for simulators in teaching. However, software for existing Bayesian networks edition show up with usability below the necessary for most of its users. Because Bayesian networks can be employed in various fields, users of the software for editing don’t always have great experience with computing and end up finding difficulties in establishing such networks. The objective of this study is to evaluate the usability of existing software for editing Bayesian networks, and from the results of this study, develop a new software that meets the needs of its users without them being affected by a low usability. In order to make this software development possible, a Bayesian network inference library that can be reused in other projects was created. In the end, a validation was performed, using usability tests with users, in which the results indicate an improvement in the usability of the developed editor in relation to other softwares studied in this work.

Key words: Bayesian networks. Usability. Editor.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Program Interface*
CPT	*Conditional Probability Table*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
NBR	Normais Brasileiras

SUMÁRIO

INTRODUÇÃO

Existem diversos problemas que devem considerar um nível de incerteza em sua resolução, por possuírem amostras imprecisas ou teorias incompletas. Para este tipo de problema, podem ser usados cálculos probabilísticos, que trazem resultados com um determinado grau de confiança. Uma maneira de realizar estes cálculos probabilísticos é através da utilização de redes bayesianas (RUSSEL; NORVIG, 1995).

Redes bayesianas envolvem uma parte qualitativa e outra quantitativa. A parte qualitativa é representada por grafos acíclicos dirigidos, nos quais os nodos representam variáveis aleatórias do problema em questão e as arestas representam as relações entre essas variáveis. A parte quantitativa é definida pelas probabilidades atribuídas a cada estado das variáveis aleatórias (WIEGERINCK; KAPPEN; BURGERS, 2010).

As redes bayesianas podem ser empregadas na solução de qualquer problema que envolva incerteza. Um exemplo deste tipo de problema é o processo de diagnóstico, no qual um especialista em determinada área precisa encontrar a causa de um problema a partir dos efeitos observados (RUSSEL; NORVIG, 1995).

No ensino da saúde, o aluno precisa praticar a construção de modelos hipotéticos que relacionem doenças com suas causas e sintomas. É preciso que ele possa avaliar a solução e tomar ações no decorrer deste processo (SEIXAS et al., 2002). Neste contexto, o uso de um software no qual o aluno possa simular situações reais e testar suas ações em tempo real mostra-se bastante útil. Para realizar estas simulações, o uso de redes bayesianas pode ser empregado.

Atualmente, existem softwares que permitem a construção e simulação de redes bayesianas. Porém, em um dos projetos em que estas foram empregadas no ensino da saúde, existem relatos de que os alunos encontraram dificuldades na utilização do editor de redes (MARONI, 2013), o que é um indício de problemas de usabilidade.

Este trabalho propõe uma análise nos editores de redes bayesianas existentes, para identificar boas práticas de usabilidade presentes em cada um deles, assim como oportunidades de melhorias. A partir dos dados observados nessa análise, um novo editor será desenvolvido aproveitando as oportunidades de melhorias encontradas.

No primeiro capítulo, será apresentada toda a teoria necessária para a definição e cálculos de redes bayesianas, o que será fundamental no desenvolvimento do novo editor. No segundo capítulo, serão apresentados conceitos de usabilidade e uma metodologia para o

desenvolvimento do um software com foco em usabilidade. No terceiro capítulo, será feita a análise dos editores existentes no mercado, na qual serão levantados itens que servirão como base para melhorias no novo editor. No quarto capítulo, será apresentado o desenvolvimento do novo editor de redes bayesianas. Para que fosse possível realizar o desenvolvimento deste editor, foi criada uma biblioteca de inferência que pode ser reutilizável em outros projetos no futuro. Essa biblioteca também está descrita no quarto capítulo. Por fim, no quinto capítulo serão apresentados os resultados da validação do editor desenvolvido.

1 TEORIA DA PROBABILIDADE E REDES BAYESIANAS

Alguns problemas necessitam de raciocínio probabilístico por envolverem incertezas. Por exemplo, Russel e Norvig (1995) acreditam que o processo de diagnóstico (independente se for médico, de reparo automobilístico ou qualquer outra área) quase sempre envolve incerteza. Nesses casos, o especialista precisa identificar a causa do problema a partir de sinais e sintomas, ou seja, dos efeitos da causa observada.

Porém, a presença de um sintoma não necessariamente indica com 100% de certeza a causa do problema. Por exemplo, se um paciente possui dor de dente, há uma probabilidade de ele possuir uma cavidade em algum dente, mas não há certeza disto. Nestes casos, uma representação utilizando lógica de primeira ordem não é o suficiente, é preciso usar teorias probabilísticas para modelar o problema, pelas seguintes motivos (RUSSEL; NORVIG, 1995):

- **Preguiça:** é muito trabalhoso listar todos os conjuntos de elementos para construir uma regra sem exceções e, mesmo que construída, é muito difícil de trabalhar com regras assim;
- **Ignorância teórica:** não existe conhecimento teórico suficiente sobre o assunto para criar as regras necessárias;
- **Ignorância prática:** mesmo possuindo as regras necessárias, não é possível testá-las na prática em todos os casos.

Portanto, "a principal vantagem de raciocínio probabilístico sobre o raciocínio lógico é o fato de que agentes podem tomar decisões racionais mesmo quando não existe informação suficiente para se provar que uma ação funcionará" (CHARNIAK apud MAQUES; DUTRA, 2002, p. 2).

Redes bayesianas podem ser utilizadas como uma maneira eficiente para representar o raciocínio probabilístico. Elas são estruturadas na forma de um grafo acíclico dirigido (em inglês, *directed acyclic graph*), no qual os nodos representam variáveis aleatórias e as arestas podem ser interpretadas como a indicação de uma relação de causa e efeito entre as variáveis. Utilizando esta rede, inferências podem ser feitas para verificar qual a probabilidade de determinado evento acontecer (NILSSON, 1998).

Este capítulo tem como objetivo apresentar a teoria da probabilidade necessária para resolver problemas que envolvem incerteza e de que maneira redes bayesianas podem ser empregadas para resolver estes problemas.

1.1 Teoria da probabilidade

1.1.1 Conceitos fundamentais

Quando um problema que envolve incerteza é observado, são envolvidas variáveis aleatórias representadas por V_1, V_2, ..., V_k. Os possíveis valores dessas variáveis são representados por v_1, v_2, ..., v_k. Estas variáveis podem ser de diferentes tipos, de acordo com o problema em questão (NILSSON, 1998):

- **Booleanas:** caso representem proposições, de domínio "verdadeiro" ou "falso";
- **Numéricas:** caso representem medidas físicas, como altura, largura, velocidade, entre outros;
- **Categóricas:** caso representem categorias, como cores, letras, etc.

Uma função chamada de *joint probability function*, que mapeia um conjunto de variáveis em um valor entre 0 e 1, é representada por $p(V_1 = v_1, V_2 = v_2, ..., V_k = v_k)$ (NILSSON, 1998). Por exemplo, ao considerar três lançamentos de uma moeda, representados pelas variáveis M_1, M_2 e M_3, que podem possuir os valores "cara" ou "coroa", $p(M_1 = \text{cara}) = 0{,}5$, representando 50% de chance de ser "cara" no primeiro lançamento. A probabilidade dos três lançamentos resultarem em "cara", representada por $p(M_1 = \text{cara}, M_2 = \text{cara}, M_3 = \text{cara})$, é de 0,125.

A definição dessas probabilidades pode ser dada a partir do processamento de dados ou do julgamento de especialistas no domínio, desde que satisfaçam os seguintes axiomas básicos da probabilidade (NILSSON, 1998):

i. $0 \leq p(V_1 = v_1, V_2 = v_2, \dots, V_k = v_k) \leq 1$

ii. $\sum p(V_1, V_2, \dots, V_k) = 1$

onde o somatório é sobre todos os valores de todas as variáveis.

A partir dessas definições, é possível montar uma tabela chamada de *full joint distribution*, que considera todas as combinações de valores para cada variável. Russel e Norvig (1995) mostram um exemplo considerando três variáveis aleatórias de domínio booleano: "toothache" (paciente possui dor de dente), "catch" (sonda de aço do dentista prende no dente) e "cavity" (paciente possui cavidade no dente). A tabela referente a este exemplo pode ser observada no Quadro 1.1.

Quadro 1.1 – Exemplo de *full joint distribution* para um caso de dor de dente.

	toothache		¬toothache	
	catch	¬catch	catch	¬catch
Cavity	0,108	0,012	0,072	0,008
¬cavity	0,016	0,064	0,144	0,576

Fonte: Russel e Norvig (1995).

A partir dessa tabela, é possível responder a questões como "qual a probabilidade de um paciente possuir cavidade?" somando todos os valores das células nas quais a variável "cavity" é verdadeira. Este processo é chamado de *marginalization* (RUSSEL; NORVIG, 1995).

$$p(cavity) = 0{,}108 + 0{,}012 + 0{,}072 + 0{,}008 = 0{,}2 \quad (1.1)$$

Também é possível obter a probabilidade de que variáveis diferentes estejam um determinado estado ao mesmo tempo. Por exemplo, para obter a probabilidade de um paciente possuir cavidade e não possuir dor de dente, o seguinte cálculo pode ser feito:

$$p(cavity, \neg toothache) = 0{,}072 + 0{,}008 = 0{,}08 \quad (1.2)$$

1.1.2 Probabilidade condicional

Caso o valor de alguma variável seja conhecido, ele pode ser utilizado para obter a probabilidade de que outras variáveis aconteçam. Para isso, utiliza-se a probabilidade condicional. A probabilidade de V_i dado V_j é representada por $p(V_i \mid V_j)$ e seu valor é definido ela seguinte equação (NILSSON, 1998):

$$p(V_i|V_j) = \frac{p(V_i, V_j)}{p(V_j)} \quad (1.3)$$

Por exemplo, caso seja conhecido que um paciente possui dor de dente, essa informação pode ser utilizada para calcular a probabilidade de que ele possua uma cavidade.

$$p(cavity|toothache) = \frac{p(cavity, toothache)}{p(toothache)} = \frac{0{,}12}{0{,}2} = 0{,}6 \quad (1.4)$$

Ou seja, a probabilidade de um paciente possuir cavidade é de 0,2, mas, caso seja sabido que ele possui dor de dente, essa probabilidade sobe para 0,6. Este tipo de cálculo envolvendo probabilidade condicional é chamado de inferência probabilística (NILSSON, 1998).

A equação da probabilidade condicional também pode ser escrita na forma da regra do produto (RUSSEL; NORVIG, 1995)

$$p(V_i, V_j) = p(V_i|V_j)p(V_j) \quad (1.5)$$

e aplicada de forma geral como (NILSSON, 1998)

$$p(V_1, V_2, \dots, V_k) = \prod_{i=1}^{k} p(V_i|V_{i-1}, \dots, V_1) \quad (1.6)$$

Aplicando essa regra no problema da consulta ao dentista, tem-se que:

$$\begin{aligned} &p(cavity, catch, toothache) \\ &= p(cavity|catch, toothache)p(catch|toothache)p(toothache) \end{aligned} \quad (1.7)$$

Utilizando a regra do produto e considerando que p(A, B) = p(B, A), a seguinte equação pode ser escrita:

$$p(V_i, V_j) = p(V_i|V_j)p(V_j) = p(V_j|V_i)p(V_i) = p(V_j, V_i) \quad (1.8)$$

A partir dessa equação, é obtida a chamada de regra de Bayes (NILSSON, 1998):

$$p(V_i|V_j) = \frac{p(V_j|V_i)p(V_i)}{p(V_j)} \quad (1.9)$$

A regra de Bayes pode ser útil na prática para o processo de diagnóstico. Ela pode ser reescrita como:

$$p(causa|efeito) = \frac{p(efeito|causa)p(causa)}{p(efeito)} \quad (1.10)$$

Por exemplo, caso o médico saiba que a meningite causa torcicolo em 70% dos casos, que a probabilidade de um paciente possuir meningite é de 1/50000 e que a probabilidade de um paciente possuir torcicolo é de 1%, pode-se calcular (RUSSEL; NORVIG, 1995):

$$p(meningite|torcicolo) = \frac{0{,}7 \times 0{,}00002}{0{,}01} = 0{,}0014 \quad (1.11)$$

Ou seja, a probabilidade de um paciente possuir meningite caso possua torcicolo é de 0,14%.

1.1.3 Independência

Algumas variáveis podem não ter relação com as outras, dependendo do problema modelado. Seguindo o exemplo da consulta ao dentista, caso fosse adicionada uma variável categórica "clima" com 4 estados possíveis, a tabela de probabilidades teria que ser expandida para 32 células. Usando a regra do produto, obtém-se a seguinte equação:

$$\begin{aligned} &p(cavity, catch, toothache, cloudy) \\ &= p(cloudy|cavity, catch, toothache)p(cavity, catch, toothache) \end{aligned} \quad (1.12)$$

Além disso, as outras variáveis são independentes do clima, pois não tem nenhuma influência nele, portanto:

$$p(cloudy|cavity, catch, toothache) = p(cloudy) \quad (1.13)$$

Essa propriedade é chamada de independência absoluta (RUSSEL; NORVIG, 1995). Isso permite que seja feita a seguinte separação:

$$\begin{aligned} &p(cavity, catch, toothache, cloudy) \\ &\qquad = p(cavity, catch, toothache)p(cloudy) \end{aligned} \quad (1.14)$$

Desta forma, ao invés de construir uma tabela com 32 valores, pode-se manter a tabela de 8 valores e criar uma outra de 4 valores para o clima. Para calcular as probabilidades, basta usar as tabelas separadamente e multiplicar o valor de cada uma. Porém, este tipo de independência não é tão comum na prática (RUSSEL; NORVIG, 1995). A independência condicional ocorre mais frequentemente.

Pode-se dizer que uma variável V é condicionalmente independente de um conjunto de variáveis V_i, dado um conjunto de variáveis V_j, se $p(V|V_i, V_j) = p(V|V_j)$. Para representar isto, é usada a notação $I(V, V_i|V_j)$. Isso significa que se o valor de V_j for conhecido, V_i não influencia em nada a probabilidade de V (NILSSON, 1998).

Por exemplo, para diagnosticar o problema de um carro não ligar, podem ser consideradas as seguintes variáveis:

- **vazamento:** probabilidade de haver um vazamento no tanque de gasolina do carro;
- **gasolina:** probabilidade de haver gasolina no carro;
- **liga:** probabilidade do carro ligar ao girar a chave.

Para a análise completa do problema, muitas outras variáveis deveriam ser consideradas (probabilidade das mais diversas falhas no motor, probabilidade de o motorista ter esquecido de abastecer, etc.), mas para ilustrar a independência condicional este conjunto basta. Neste cenário, a probabilidade de haver vazamento influencia na probabilidade de não haver gasolina, que por sua vez influencia na probabilidade do carro não ligar. Desta forma, pode-se dizer que, indiretamente, a variável "vazamento" influencia a variável "liga". Porém, caso seja sabido se o carro possui ou não gasolina, a variável "vazamento" não influencia em nada na probabilidade da variável "liga". Portanto, a variável "liga" é condicionalmente independente da variável "vazamento" dada a variável "gasolina", o que é representado por:

$$I(liga, vazamento|gasolina) \quad (1.15)$$

Ao considerar duas variáveis condicionalmente independentes, V_i e V_j, dado um conjunto de variáveis V, a seguinte equação pode ser feita:

$$p(V_i, V_j|V) = p(V_i|V)p(V_j|V) \quad (1.16)$$

Essa equação pode ser aplicada de forma geral, dado um conjunto de variáveis V_1, ..., V_k, no qual todas as variáveis são condicionalmente independentes das outras dado um conjunto de variáveis V, da seguinte maneira (NILSSON, 1998):

$$p(V_1, V_2, \dots, V_k|V) = \prod_{i=1}^{k} p(V_i|V) \quad (1.17)$$

A independência condicional é aproveitada para uma representação mais simples de problemas probabilísticos em estruturas chamadas de redes bayesianas.

1.2 Redes bayesianas

Para resolver problemas probabilísticos, a tabela representando a *full joint distribution* pode ser utilizada. Porém, quanto mais variáveis são adicionadas ao problema, o número de entradas nessa tabela cresce exponencialmente. Problemas reais tendem a ter um número

relativamente grande de variáveis, o que torna essa abordagem ineficiente e muitas vezes inviável. Para uma representação mais simples deste tipo de problema, aproveitando as propriedades de independência entre as variáveis, podem ser utilizadas redes bayesianas (RUSSEL; NORVIG, 1995).

1.2.1 Representação e cálculo

Redes bayesianas são representadas por grafos acíclicos dirigidos, com a seguinte especificação (RUSSEL; NORVIG, 1995):

1. Cada nodo representa uma variável aleatória;
2. Se houver uma aresta do nodo X ao nodo Y, X é considerado pai de Y;
3. Cada nodo contém uma distribuição de probabilidade condicional que quantifica o efeito de seus pais em suas probabilidades.

A topologia da rede especifica quais são as relações de independência condicional entre as variáveis observadas. Cada variável representada por um nodo, dados os valores das variáveis representadas pelos seus nodos pais, é condicionalmente independente de todas as outras variáveis que não sejam descendentes dela.

As arestas que conectam os nodos da rede podem ser interpretadas como uma relação de causa e efeito, na qual o nodo pai é a causa do efeito representado pelo nodo filho. Este tipo de interpretação é muito próximo do raciocínio que seres humanos especialistas em alguma área utilizam, portanto, a representação é feita de forma natural (NILSSON, 1998).

A construção de uma rede bayesiana envolve duas partes, sendo uma qualitativa e a outra quantitativa. A parte qualitativa é a estrutura do grafo que representa a rede. A parte quantitativa é representada pelas tabelas de probabilidade associadas a cada variável. Essa especificação pode ser inferida de dados, construída manualmente por um especialista ou uma mistura das duas alternativas (WIEGERINCK; KAPPEN; BURGERS, 2010).

A problema da consulta ao dentista, apresentado anteriormente, pode ser representado por uma rede bayesiana com a estrutura apresentada na Figura 1.1.

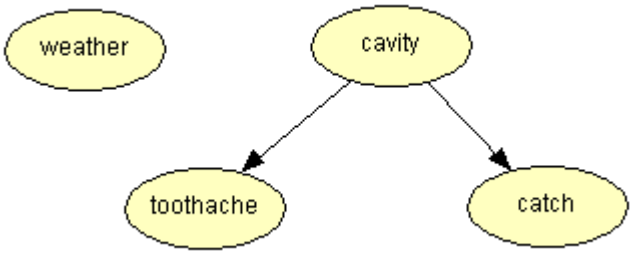


Figura 1.1 – Estrutura de uma rede bayesiana
Fonte: Russel e Norvig (1995)

Nesta rede, pode-se dizer que as variáveis “toothache” e “catch” são condicionalmente independentes dada a variável “cavity”, pois não há nenhuma ligação direta entre elas, apenas indiretamente por meio de “cavity”. Seguindo o raciocínio de causa e efeito, pode-se interpretar essa rede dizendo que a variável “cavity” é a causa dos efeitos “toothache” e “catch”. Além disso, como não há nenhuma ligação da variável “weather” com as outras, fica caracterizado que existe uma independência incondicional (ou absoluta) entre elas.

Além dessa representação das relações de independência entre as variáveis, é preciso associar uma tabela de probabilidade condicional (do inglês *conditional probability table*, ou simplesmente CPT) a cada uma delas.

Russel e Norvig (1995) apresentam um exemplo que ilustra essa definição. Este exemplo representa um problema relacionado ao acionamento de um alarme em uma residência e envolve cinco variáveis aleatórias:

- **burglary:** probabilidade de haver um roubo na residência;
- **earthquake:** probabilidade de haver um terremoto;
- **alarm:** probabilidade de o alarme ser acionado;
- **john_calls:** probabilidade de John ligar avisando que o alarme foi acionado;
- **mary_calls:** probabilidade de Mary ligar avisando que o alarme foi acionado.

A Figura 1.2 mostra a rede bayesiana criada e as CPTs associadas a cada um dos nodos.

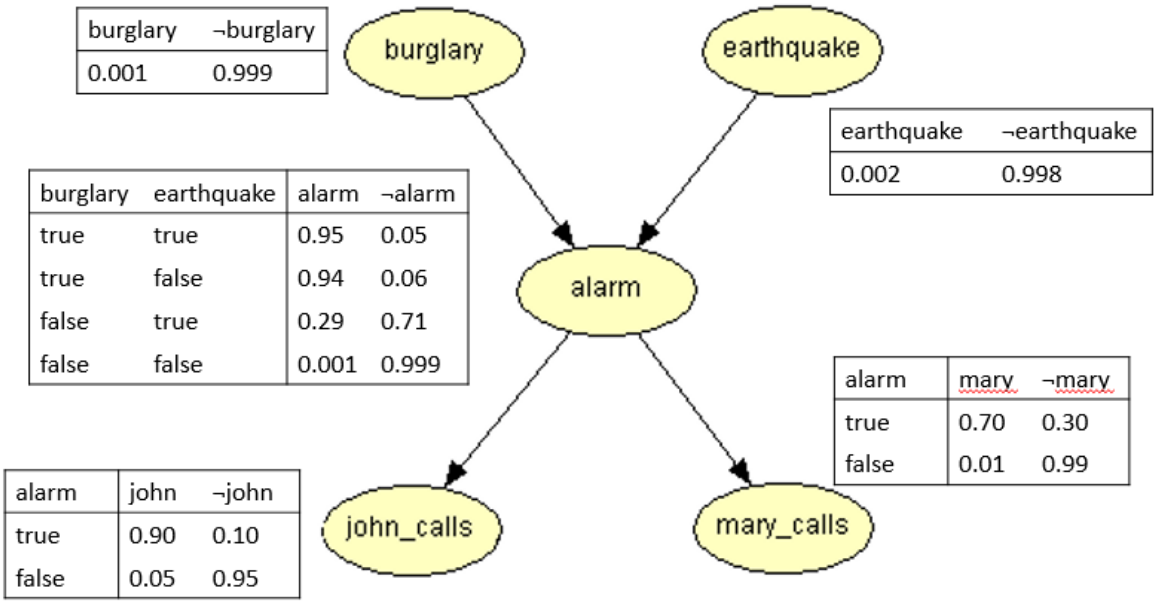


Figura 1.2 – Estrutura de uma rede bayesiana com CPTs
Fonte: Russel e Norvig (1995)

Como esta rede possui 5 variáveis de domínio booleano, caso o mesmo problema fosse representado utilizando a *full joint distribution*, teriam que ser considerados 32 (2^5) possibilidades. Por aproveitar as relações de independência condicional, o mesmo problema pode ser representado pela rede bayesiana, somando as possibilidades de cada uma das CPTs, considerando 20 possibilidades, o que mostra uma redução significativa no tamanho do problema. Como o crescimento da *full joint distribution* em relação ao número de variáveis é exponencial, quanto maior o número, a utilização de redes bayesianas tende a trazer uma redução ainda maior no tamanho. Além disso, por trabalhar com tabelas menores ao invés de uma única tabela, o gerenciamento delas torna-se mais simples.

Para que sejam calculadas as probabilidades de todos os nodos da rede bayesiana, utiliza-se a seguinte fórmula (NILSSON, 1998):

$$p(V_1, V_2, \dots, V_k) = \prod_{i=1}^{k} p(V_i | Pais(V_i)) \qquad (1.18)$$

Utilizando essa fórmula é possível montar uma *full joint distribution* para qualquer rede bayesiana e utilizar os mesmos cálculos da teoria da probabilidade para realizar inferências na rede. Por exemplo, no caso do alarme, caso seja sabido que John e Mary ligaram avisando que o alarme tocou, pode-se calcular a probabilidade de que tenha ocorrido um roubo da

seguinte forma, utilizando a propriedade da probabilidade condicional (para simplificar, as variáveis foram abreviadas em suas letras iniciais):

$$p(B|J,M) = \frac{p(B,J,M)}{p(J,M)} \quad (1.19)$$

Para calcular p(B, J, M), deve-se considerar todas as combinações de estados possíveis em que B, J e M sejam verdadeiros:

$$\begin{aligned} &p(B) \times p(E) \times p(A|B,E) \times p(J|A) \times p(M|A) \\ &p(B) \times p(\neg E) \times p(A|B,\neg E) \times p(J|A) \times p(M|A) \\ &p(B) \times p(E) \times p(\neg A|B,E) \times p(J|\neg A) \times p(M|\neg A) \\ &p(B) \times p(\neg E)p \times (\neg A|B,\neg E) \times p(J|\neg A) \times p(M|\neg A) \end{aligned} \quad (1.20)$$

Feito isso, os valores para cada probabilidade, obtidos por meio das CPTs atribuídas a cada variável, devem ser multiplicados, o que retorna os seguintes valores:

$$\begin{aligned} &0{,}001 \times 0{,}002 \times 0{,}950 \times 0{,}900 \times 0{,}070 = 0{,}000001197 \\ &0{,}001 \times 0{,}998 \times 0{,}940 \times 0{,}900 \times 0{,}070 = 0{,}0005910156 \\ &0{,}001 \times 0{,}002 \times 0{,}050 \times 0{,}050 \times 0{,}010 = 0{,}00000000005 \\ &0{,}001 \times 0{,}998 \times 0{,}060 \times 0{,}050 \times 0{,}010 = 0{,}00000002994 \end{aligned} \quad (1.21)$$

Somando estes valores, obtém-se 0,00059224259. Seguindo o mesmo processo, tem-se que p(J, M) = 0,002084100239. Portanto:

$$p(B|J,M) = \frac{p(B,J,M)}{p(J,M)} = \frac{0{,}00059224259}{0{,}002084100239} = 0{,}2842 \quad (1.22)$$

Isso significa que, sabendo que John e Mary ligaram avisando que o alarme foi acionado, a chance de um roubo ter ocorrido é de aproximadamente 28%.

1.2.2 Utilização

Redes bayesianas podem ser utilizadas nas mais diversas áreas que necessitem de raciocínio probabilístico. Uma dessas áreas é no processo de diagnóstico médico, no qual o paciente possui uma série de sinais e sintomas que indicam as probabilidades de ele possuir doenças presentes em um determinado conjunto (MARQUES; DUTRA, 2002). Para auxiliar este processo de diagnóstico, tanto na prática quanto no ensino, pode ser utilizado algum

software que interprete os sinais e sintomas e realize inferências para calcular as probabilidades (SEIXAS et al., 2002).

No contexto do ensino, o aluno de medicina precisa praticar a construção de modelos hipotéticos que relacionem doenças com as suas causas e sintomas, além de avaliar a situação e tomar decisões ao longo deste processo. Considerando uma abordagem construtivista de ensino, na qual o conhecimento deve ser construído por cada indivíduo e não passado de uma pessoa para a outra, o uso de simuladores virtuais mostra-se bastante útil. As redes bayesianas são uma representação natural para o conhecimento probabilístico, portanto, elas podem ser aplicadas nesses simuladores (SEIXAS et al., 2002).

Atualmente, existem projetos que utilizam redes bayesianas com este intuito. Alguns exemplos são o AMPLIA (SEIXAS, 2005), o SimDeCS (BARROS et al., 2012) e o Health Simulator (LIMA et al., 2015). A seguir, eles serão brevemente apresentados, bem como a tecnologia utilizada para empregar o uso de redes bayesianas.

O AMPLIA é um projeto no qual o aluno representa seu raciocínio de diagnóstico por meio de uma rede bayesiana e este raciocínio é avaliado por agentes inteligentes (SEIXAS, 2005). O editor de redes bayesianas do AMPLIA é baseado no editor gráfico para redes bayesianas SEAMED, aplicação desktop desenvolvida em Delphi 6.0 (FLORES, 2005).

O SimDeCS é um simulador no qual um especialista constrói uma rede bayesiana baseada em uma diretriz clínica. Após isso, o professor monta um caso clínico que utiliza o conhecimento modelado nesta rede bayesiana. Posteriormente, na forma de um jogo sério, o aluno de medicina realiza um atendimento a um paciente virtual conforme descrito neste caso clínico. Este processo é avaliado por agentes inteligentes na forma de uma negociação pedagógica (FONSECA, 2013). Neste projeto, a inferência de redes bayesianas é feita pelo UnBBayes e a edição delas é feita pelo software Hugin. Ambos serão apresentados no terceiro capítulo deste trabalho.

O Health Simulator é um software do tipo paciente virtual, com o intuito de auxiliar no ensino na área da saúde. A sua estruturação quanto a modelagem do problema é semelhante ao SimDeCS, no qual um especialista cria uma rede bayesiana que é utilizada pelo professor na montagem de um caso clínico para a simulação. Porém, segundo Lima et al. (2015, p. 286),

> a experiência adquirida no desenvolvimento do SIACC e SimDeCS mostrou que era necessário desenvolver um simulador mais realístico, que forneça mais liberdade ao aluno para testar suas hipóteses diagnósticas e experimentar na prática o que este encontraria na área da saúde.

Conforme relatado por Maroni (2013), o fato do aluno usuário do AMPLIA precisar utilizar um editor de redes bayesianas para modelar suas hipóteses muitas vezes atrapalha o objetivo da aprendizagem, pois ele encontra dificuldades de utilização. Isto é um indicativo de que o editor utilizado no AMPLIA possui problemas de usabilidade.

Apesar de o SimDeCS ter removido do aluno a tarefa de edição de redes, o especialista na área ainda precisa criá-las e editá-las. Como a montagem do caso clínico e a simulação do atendimento médico dependem dessa edição, é essencial que ela esteja correta.

Como o Health Simulator mantém o conceito de modelagem do conhecimento do especialista utilizando redes bayesianas, surge a necessidade de um editor com foco em usabilidade, para que possa ser utilizado por especialistas que não possuam necessariamente um conhecimento aprofundado em computação.

No terceiro capítulo, serão apresentados alguns dos editores de redes bayesianas existentes no mercado. No próximo capítulo, serão apresentados conceitos de usabilidade que serão utilizadas para a avaliação destes trabalhos relacionados.

2 USABILIDADE

Este capítulo tem como objetivo apresentar os conceitos de usabilidade relevantes para a avaliação dos softwares para criação e edição de redes bayesianas existentes no mercado. Além desses conceitos, será apresentada uma metodologia para construção de projetos com foco em usabilidade. Esta metodologia será utilizada no desenvolvimento do novo editor de redes bayesianas que, com base na avaliação das alternativas existentes, apresente melhorias em relação a elas.

2.1 Conceitos de usabilidade

A norma NBR ISO/IEC 9126-1 (2003) define um modelo para a qualidade do software em que seus atributos são classificados em seis categorias: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Nesta norma, a usabilidade é definida como a "capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições específicas" (NBR ISO/IEC 9126-1, 2003, p. 9).

Para Reiss (2015, p. xvii, tradução nossa), "Usabilidade lida com a habilidade de um indivíduo realizar uma tarefa específica ou meta mais ampla enquanto "usa" o que você estiver investigando, melhorando ou desenvolvendo". Nesta definição há um foco maior no fato de que o usuário está tentando realizar uma tarefa enquanto usa o software, e este objetivo deve ser atingido.

A norma ISO/IEC 9241-210 (2010) define usabilidade como uma medida para como um sistema, produto ou serviço pode ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um determinado contexto de uso. Essa norma ainda detalha o significado dos seguintes termos:

- **Eficácia:** precisão e plenitude com que os usuários atingem objetivos específicos;
- **Eficiência:** recursos gastos em relação à eficácia, quanto menos recursos gastos, maior a eficiência;
- **Satisfação:** ausência de desconforto e atitudes positivas em relação ao uso do produto.

Desta forma, a usabilidade, além de medir a capacidade de um usuário atingir seus objetivos enquanto usa determinado software, também considera a que ele deve atingir estes

objetivos da maneira mais eficiente possível, ou seja, gastando o mínimo de recursos. Estes recursos envolvem tempo, dinheiro, esforço mental, entre outros. Além disso, um produto com boa usabilidade considera que o usuário ficará satisfeito durante o uso.

Preece, Rogers e Sharp (2005) consideram que, para obter uma boa usabilidade, o software deve atingir seis metas, sendo elas:

- **Eficácia:** meta bastante geral que mede o quão bom o software é em fazer o que se espera dele. Um software que possua uma boa eficácia permite que o usuário atinja seus objetivos com resultados satisfatórios;
- **Eficiência:** refere-se a medir se a maneira como o software auxilia seus usuários a realizarem as tarefas é a mais produtiva. Por exemplo, se uma funcionalidade usada frequentemente exige um número excessivo de passos para ser concluída, ela poderia ser mais eficiente reduzindo este número de passos;
- **Segurança:** protege o usuário de condições perigosas e situações indesejáveis. Um exemplo seria quando um usuário pressiona acidentalmente a tecla para apagar um arquivo, uma mensagem de confirmação pode ser adotada como uma medida de segurança para que isso somente aconteça após o usuário confirmar que é isso mesmo o que ele deseja;
- **Utilidade:** considera se o software provê as funcionalidades necessárias para que o usuário realize a tarefa da maneira que deseja. Por exemplo, um software de desenho que não proporcione ao usuário a capacidade de utilizar as mãos livres, obrigando o uso do mouse, para determinado público, mostra-se com uma utilidade baixa;
- **Capacidade de aprendizagem:** refere-se a quão fácil é aprender a usar o software em questão. O tempo que os usuários estão dispostos a investir no aprendizado de um software pode variar. Quanto mais complexo é o software e mais valor ele agrega ao usuário, um tempo maior para o aprendizado o usuário tende a dispor. Porém, é importante tentar diminuir esse tempo, deixando o uso das funcionalidades principais da maneira mais clara possível;

- **Capacidade de memorização:** esta meta é importante quando o usuário não usa o sistema de maneira frequente, pois quando retornar, não terá que relembrar, possivelmente com a ajuda de alguém, onde fica cada recurso que ele deseja. Quando as ações que o usuário precisa executar são obscuras ou ilógicas, a sua capacidade de memorização é reduzida.

Preece, Rogers e Sharp (2005) também consideram que, além das metas decorrentes da usabilidade, é preciso considerar as metas decorrentes da experiência do usuário. A diversidade de tecnologias (como realidade virtual, internet e computação móvel), aplicadas a diferentes áreas, trouxeram um novo conjunto de interesses para os usuários. Os usuários esperam sistemas que sejam satisfatórios, agradáveis, divertidos, interessantes, úteis, motivadores, esteticamente apreciáveis, incentivadores de criatividade, compensadores e emocionalmente adequados.

Muitas vezes essas metas são conflitantes com as metas de usabilidade. Por exemplo, uma funcionalidade pode ser apresentada de maneira mais divertida para o usuário, porém menos eficiente. Da mesma forma, talvez possa ser impossível que o sistema seja seguro e divertido ao mesmo tempo. É necessário avaliar as opções dentro do contexto de uso, tarefas a serem realizadas e perfil de usuário, para então escolher a mais adequada.

Jha e Duffy (2002) realizaram um estudo no qual identificaram as chamadas dez regras de ouro para o desenvolvimento de software para a educação médica. O Quadro 2.1 mostra essas regras. Conforme descrito no subcapítulo 1.2.2, é comum o uso de redes bayesianas na área do ensino da saúde, portanto, as pessoas envolvidas no ensino da saúde mostram-se como potenciais usuários para um editor de redes bayesianas. Devido a isso, é importante que essas regras sejam observadas durante o desenvolvimento do editor.

A usabilidade pode ser um fator decisivo no sucesso de um produto. Se o usuário tiver uma má experiência de uso com determinado produto, ele geralmente deixará de usá-lo. Caso a experiência de uso for mediana e um competidor possuir um produto que proporcione uma experiência melhor, o usuário tenderá a trocar para o produto do competidor (GARRET, 2011).

Quadro 2.1 – Dez regras de ouro para o desenvolvimento de software para a educação médica.

Regra 1	Conteúdo deve ser adequado para o propósito educacional, de bons padrões e relevante para a prática clínica.
Regra 2	Conteúdo deve ser baseado em evidências, e não em opiniões.
Regra 3	Use *hypermedia* e *hypertext* para promover o conhecimento.
Regra 4	Garanta que a apresentação seja interessante, agradável e desafiadora.
Regra 5	Use multimídia apropriada.
Regra 6	Use uma configuração baseada em problemas.
Regra 7	O conteúdo e tarefas devem simular habilidades de análise e solução de problemas.
Regra 8	O produto deve ser amigável e de fácil navegação.
Regra 9	Forneça o impulso adequado para o uso.
Regra 10	Mantenha os custos baixos e cronograma de produção rigoroso.

Fonte: Jha e Duffy (2002, tradução nossa).

2.2 Princípios de design

A usabilidade pode ser avaliada também pelos princípios de design. Os princípios de design são derivados de uma mistura de conhecimento teórico, experiência e senso comum. Eles tendem a ser escritos de maneira prescritiva, sugerindo o que deve ou não ser utilizado. Não especificam como projetar uma interface real, mas servem como itens que devem ser considerados ao projetar uma (PREECE; ROGERS; SHARP, 2005). Os mais comuns, descritos por Norman (1988), serão demonstrados a seguir.

- **Visibilidade:** quanto mais visíveis forem as funções, mais rapidamente os usuários saberão como proceder. Se elas estiverem escondidas, é mais difícil de encontrá-las e saber como utilizá-las (PREECE; ROGERS; SHARP, 2005);
- ***Feedback:*** se refere ao retorno de informações decorrentes das ações realizadas. O *feedback* pode ser de vários tipos, como áudio, verbal, visual e combinações entre eles. Utilizar o *feedback* de maneira certa pode proporcionar o nível de visibilidade adequado para a interação do usuário (PREECE; ROGERS; SHARP, 2005).
- **Restrições:** refere-se a delimitar quais ações o usuário pode tomar em determinado momento. Um exemplo desse princípio acontece quando opções de um sistema ficam desativadas por determinado motivo. Com a redução de opções incorretas que o usuário pode tomar, a chance de erros é reduzida (PREECE; ROGERS; SHARP, 2005).

- **Mapeamento:** refere-se à relação entre os controles e seus efeitos no mundo. Um exemplo de bom mapeamento são as setas do teclado (controles) e seus efeitos (mover o cursor para uma direção). A tecla que move o cursor para cima fica em cima das outras, a que move para esquerda fica na esquerda, etc. Desta forma, há um mapeamento natural na maneira em que elas estão dispostas com os resultados que elas produzem. Caso as teclas estivessem dispostas de outra maneira, o mapeamento com os resultados poderia não ser tão natural (PREECE; ROGERS; SHARP, 2005).
- **Consistência:** diz respeito a projetar interfaces que sejam semelhantes para a realização de tarefas semelhantes. Um dos benefícios de interfaces consistentes é que são mais fáceis de aprender e usar. (PREECE; ROGERS; SHARP, 2005).

2.3 Testes de usabilidade

Para Krug (2006), um teste de usabilidade caracteriza-se quando é mostrado algo (independente se for um site, um protótipo ou rascunhos) a um usuário por vez, e solicitado para que este usuário, ou descobrir o que foi mostrado é, ou tentar usá-lo para realizar determinada tarefa.

Além disso, Krug (2006) acredita que o número ideal de usuários por rodada de testes de usabilidade é três, no máximo quatro. Isto é justificado por dois motivos. Um deles é que os primeiros três usuários normalmente encontrarão quase todos os problemas realmente relevantes. O outro motivo é que é mais importante realizar várias rodadas de testes do que poucas rodadas, e um número pequeno de usuários favorece isto. Desta forma, tem-se um processo mais iterativo, onde os problemas encontrados pelos usuários são corrigidos entre uma rodada e outra.

Reiss (2012) considera que o método preferido para testes de usabilidade são os testes *think aloud*. Neste tipo de teste, é solicitado ao usuário realizar determinadas tarefas, como preencher um formulário, procurar uma informação, etc. Estas tarefas definem o protocolo do teste. Feito isso, é solicitado que o usuário diga em voz alta o que está pensando enquanto realiza as tarefas definidas. A partir das informações ditas pelo usuário, é possível identificar em quais pontos ele teve dificuldade de entendimento, para que esses pontos possam ser melhorados. Caso o usuário fique em silêncio, o observador deve fazer perguntas do tipo:

- O que você está pensando agora?
- Para onde você está olhando?
- O que você quer fazer no momento?

Para a escolha dos usuários, o ideal é que sejam do público alvo do produto que está sendo produzido. Porém, tanto Reiss (2012) quanto Krug (2006) concordam que, se não for viável a presença de um usuário do público alvo específico em todas as rodadas de testes, é preferível que sejam escolhidos usuários de outras áreas mesmo assim. O observador deve estar ciente disso e interpretar os resultados dos testes de acordo.

2.4 Metodologia para construção de projetos com foco em usabilidade

O Projeto E é uma metodologia para guiar o desenvolvimento de projetos com interfaces gráficas amigáveis (MEURER; SZABLUK, 2010). A estrutura da metodologia tem como base o modelo definido por James Jesse Garret (2003), e divide o projeto em seis etapas: estratégia, escopo, estrutura, esqueleto, estética e execução.

O Projeto E não é necessariamente um processo sequencial, ou seja, é possível alterar os itens pertinentes a cada uma das etapas em qualquer momento do projeto (MEURER; SZABLUK, 2010). A Figura 2.1 mostra as etapas que constituem o Projeto E.

Figura 2.1 – Etapas do Projeto E original
Fonte: Meurer e Szabluk (2010)

Na versão atual do Projeto E, foi feita uma reestruturação das etapas, com a finalidade de ser melhor adaptada a um contexto de ensino e aprendizagem (MEURER, 2014). Nessa reestruturação, as etapas que constituem o projeto são: contextualização, desconstrução,

verificação, reconstrução, diferenciação, identidade, desenvolvimento e validação. A Figura 2.2 apresenta as etapas após a reestruturação.

Projeto E Meurer e Szabluk (2012)	Projeto E como modelo de ABP	Ações de cada etapa	Habilidades requeridas
Estratégia	Contextualização Desconstrução Verificação	Indentificar, compreender e motivar Investigar, observar e analisar em detalhes Entender restrições, requisitos e possibilidades	Comunicação Investigação Argumentação Autonomia Discussão Expressão Gráfica Senso Estético Reflexão Colaboração Pensamento Crítico Planejamento Projetação Autogestão Autoavaliação Apresentação
Escopo Estrutura Esqueleto	Reconstrução Diferenciação	Definir, conceituar, organizar, esquematizar, desenhar Posicionar e diferenciar	
Estética	Identidade	Promover e destacar através da originalidade	
Execução	Desenvolvimento Validação	Modelar e projetar Testar e redifinir	

Figura 2.2 – Etapas do Projeto E após reestruturação
Fonte: Meurer (2014)

Para o desenvolvimento do novo editor de redes bayesianas com foco em usabilidade, será utilizado o Projeto E com algumas modificações, pois alguns itens definidos em suas etapas não se aplicam ao escopo deste trabalho. Por exemplo, na etapa de contextualização, o Projeto E define a tarefa de definição de papéis da equipe (MEURER, 2014). Essa etapa não será realizada pois o trabalho será desenvolvido por um único aluno e não é preciso uma documentação que permita a integração de equipes. Ainda assim, o método possui uma série de tarefas relevantes que auxiliarão no desenvolvimento de um editor de redes bayesianas com foco em usabilidade. A seguir, as etapas e tarefas que constituem o Projeto E serão apresentadas.

2.4.1 Contextualização

O objetivo da etapa de contextualização é identificar, definir e delimitar o problema (MEURER, 2014). Dentre as tarefas definidas nessa etapa, serão executadas as seguintes:

- **Situação inicial bem definida (SIBD) e situação final bem definida (SFBD):** para ter uma SIBD, é preciso identificar qual o produto a ser desenvolvido e por quem e em quais situações ele será utilizado. Para a SFDB, é preciso identificar quais tecnologias, processos de construção e incentivos de uso serão aplicados;

- **Taxonomia:** é uma forma de classificar os requisitos envolvidos no projeto. Um exemplo de taxonomia pode ser observado na Figura 2.3;
- **Equalização dos fatores projetuais:** definir a importância de um ou mais fatores relevantes ao design do projeto, sendo eles: antropológicos, ecológicos, ergonômicos, econômicos, mercadológicos, tecnológicos, filosóficos, geométricos e psicológicos.

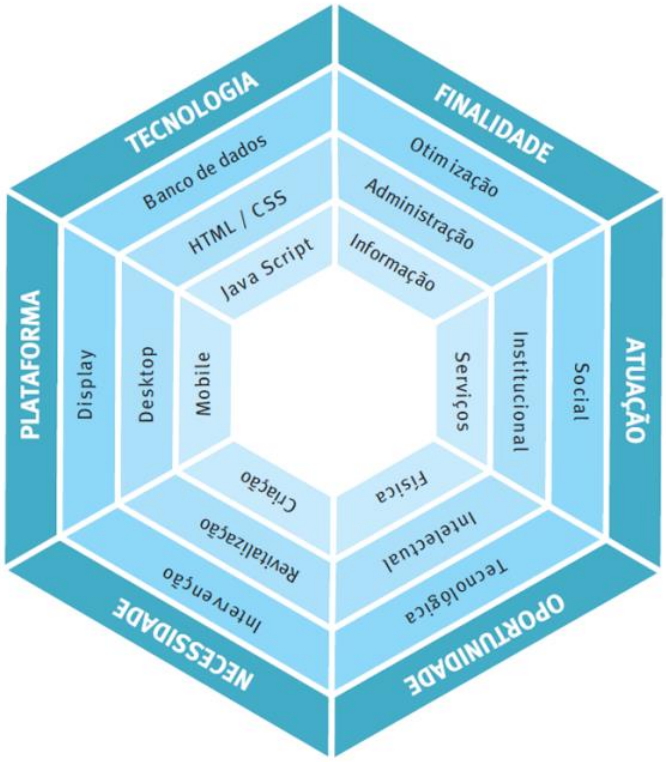


Figura 2.3 – Exemplo de taxonomia
Fonte: Meurer (2014)

2.4.2 Desconstrução

Nesta etapa, são feitas profundas investigações, análises e avaliações de conteúdo, conceitos e contextos que possam servir de referência para influenciar o desenvolvimento do projeto. Nesta etapa, produtos similares são analisados. O resultado da desconstrução serve como base para uma argumentação coerente nas tomadas de decisão futuras do projeto (MEURER, 2014). Algumas das atividades da desconstrução são:

- **Similares e referências:** envolve o estudo de projetos similares, que permite situar o produto em relação a outros existentes no mercado. É importante para evitar reinvenções. É recomendado que a análise não seja apenas por produtos

similares do mesmo nicho em questão, mas que sejam incluídos outros produtos que sirvam de referências relevantes ao projeto. Para cada produto, é preciso identificar e descrever pontos positivos e negativos. A partir disso, os produtos podem ser classificados em relação a sua usabilidade e funcionalidades, para que sejam identificadas quais são boas referências;

- **Linha do tempo:** é preciso avaliar a linha do tempo de produtos similares para que tendências evolutivas sejam observadas. Os aspectos observados podem ser tecnológicos, estruturais, funcionais, estéticos, entre outros. A partir dessa observação, é possível prever situações futuras, tanto oportunas quanto indesejáveis;
- **Comparativa de funcionalidades:** a partir das referências selecionadas, é preciso criar um comparativo de ferramentas e funcionalidades presentes em cada uma delas, identificando as com maior relevância. Os itens encontrados podem servir de referência para o desenvolvimento de produtos futuros;
- **Da identidade gráfico-visual:** constitui uma análise da identidade gráfico-visual das referências selecionadas. A identidade gráfico-visual é constituída por quatro elementos institucionais: logotipo, símbolo, cores padrão e tipografia.

2.4.3 Verificação

Na etapa de verificação, com base nos resultados obtidos na etapa de desconstrução, deve ser definida uma lista de restrições, requisitos e possibilidades a serem consideradas no novo produto (MEURER, 2014).

2.4.4 Reconstrução

Esta fase caracteriza o início do desenvolvimento do novo produto. Segundo Meurer (2014), o objetivo desta etapa é definir e conceituar a arquitetura informacional do produto, organizar e hierarquizar os conteúdos, planejar e esquematizar o modelo interacional, desenhar o fluxo de tarefas que constituem as ferramentas e funcionalidades e estabelecer a estrutura das telas. Estes objetivos podem ser atingidos por meio das técnicas apresentadas a seguir.

- **Definição das ferramentas, funcionalidades e conteúdos:** é preciso estabelecer a organização, hierarquia e inter-relação entre conteúdos, funcionalidades e ferramentas, dividindo-os em categorias ou módulos. A partir dessa definição, é possível ter uma ideia correta do escopo;
- ***Wireframes, wireflows* e casos de uso:** *wireframes* são esboços preliminares das telas do produto que definem a organização dos elementos. *Wireframes* precedem o design visual, portanto, não devem ser usados como parâmetro para aspectos estéticos. Para apresentar passos de uma tarefa, cria-se uma sequência de *wireframes*, chamada *wireflow*.

2.4.5 Identidade

Nesta etapa é criada a identidade gráfico-visual do novo produto. A partir dos *wireframes* definidos na reconstrução, são criados os elementos visuais necessários para que o produto transmita empatia, credibilidade e confiabilidade (MEURER, 2014). De acordo com Norman (2005), uma pesquisa feita apontou que usuários consideram produtos com estética superior 25% mais fáceis de usar, o que indica que estes fatores agreguem usabilidade.

2.4.6 Diferenciação

Esta fase procura avaliar a identidade visual do produto. Uma maneira de avaliar é através do método de mapeamento de expressões (MEMÓRIA, 2005), no qual os usuários avaliam o produto posicionando-o num eixo com palavras de diferentes valores em cada extremidade. A partir dessa análise, é possível ter uma visão se a identidade pretendida é a mesma percebida pelo usuário. Caso não seja, é preciso revê-la (MEURER, 2014).

2.4.7 Desenvolvimento

Normalmente é iniciada após as restrições, requisitos e possiblidades forem definidas. Na medida que novas especificações forem definidas, elas são passadas ao desenvolvimento. Em geral, o desenvolvimento da interface gráfica só é iniciado após a definição dos leiautes das principais telas terem sido definidos e aprovados (MEURER, 2014).

2.4.8 Validação

Esta etapa diz respeito a execução de testes técnico-funcionais e com usuários. Normalmente é iniciada logo após os primeiros resultados da etapa de desconstrução e segue ao decorrer do desenvolvimento do produto. Ela visa encontrar e corrigir problemas de usabilidade encontrados no produto (MEURER, 2014).

3 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados alguns dos editores de redes bayesianas existentes no mercado. Além disso, será realizada a etapa de desconstrução do Projeto E, apresentada no capítulo 2. Essa etapa envolve realizar análises em produtos similares que possam servir de referência para o desenvolvimento do produto novo.

Para cada um dos produtos analisados, será feita uma apresentação breve do fluxo de tarefas que o usuário deve fazer para montar uma rede bayesiana e realizar inferências nela. Também serão levantadas características relevantes de cada um deles, observadas pelo autor deste trabalho. Ao final, será feita uma análise comparando as funcionalidades presentes em cada um dos editores, para que fique mais claro o escopo de cada um deles.

3.1 Hugin

O Hugin é uma ferramenta de propósito geral para modelos probabilísticos gráficos (MADSEN et al., 2003). Ele é desenvolvido pela Hugin Expert A/S, empresa dinamarquesa fundada em 1989. A empresa surgiu a partir de um projeto de pesquisa para ajudar a diagnosticar doenças musculares e nervosas (HUGIN, 2016). Uma imagem do software pode ser visualizada na Figura 3.1.

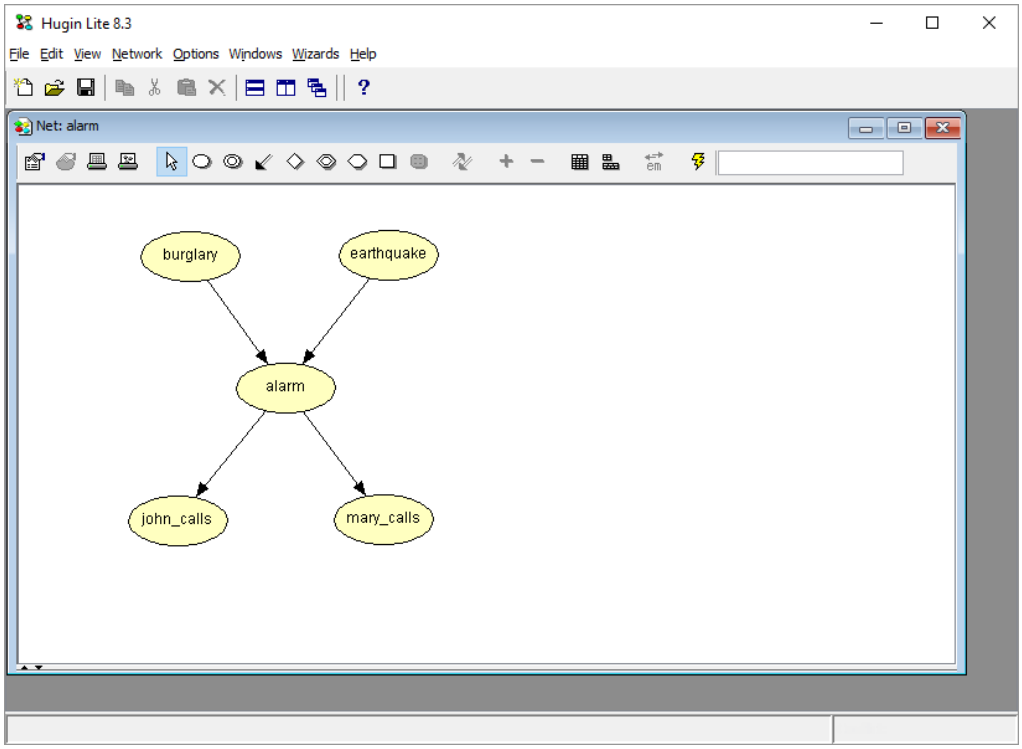


Figura 3.1 – Captura de tela do Hugin: visão geral
Fonte: Capturado pelo autor

No Hugin, cada rede bayesiana é aberta em uma janela para edição e simulação. Dentro dessa janela, há uma barra de ferramentas na parte superior. Nessa barra, ficam disponibilizadas uma série de ferramentas para inserir elementos na rede, sendo elas:

- **Discrete chance tool:** utilizada para inserir nodos representando variáveis aleatórias discretas com um número finito de estados;
- **Continuous chance tool:** utilizada para inserir nodos representando variáveis aleatórias com uma função de distribuições contínua;
- **Link tool:** utilizada para inserir uma ligação causal entre um nodo pai e um nodo filho;
- **Utility tool:** utilizada para inserir nodos representando uma função de utilidade em diagramas de influência. Uma função de utilidade define um valor para cada combinação dos pais do nodo de utilidade;
- **Function tool:** utilizada para inserir nodos representado uma função. Nodos de função não são usados no processo de inferência, mas seu valor é definido em função dos resultados obtidos da inferência dos seus nodos pais;
- **Discrete function tool:** utilizada para definir um nodo representando uma distribuição discreta em função dos valores dos nodos pais;
- **Discrete decision tool:** utilizada para inserir um nodo que representa uma decisão a ser tomada pelo usuário com base nos valores de seus nodos pais em um diagrama de influência;
- **Instance node tool:** utilizada para inserir nodos de instância. Nodos de instância representam a instância de outra rede bayesiana, ou seja, uma sub rede.

De acordo com a documentação do Hugin, há restrições na utilização de nodos de chance contínua, parcialmente porque a teoria ainda está em desenvolvimento. As restrições são as seguintes:

- Atualmente somente são suportadas distribuições de Gauss (normais);
- Um nodo contínuo não pode ser pai de um nodo discreto;
- Nodos contínuos não podem ser utilizados em diagramas de influência, ou seja, não podem existir em uma rede contendo nodos de utilidade ou decisão.

Ao clicar com o botão direito em um dos nodos da rede, o usuário pode escolher a opção “Open Tables”, que abre a tabela de probabilidade condicional associada ao nodo. O usuário pode editar essa tabela definindo a probabilidade de cada uma das combinações de estados possíveis, conforme mostrado na Figura 3.2.

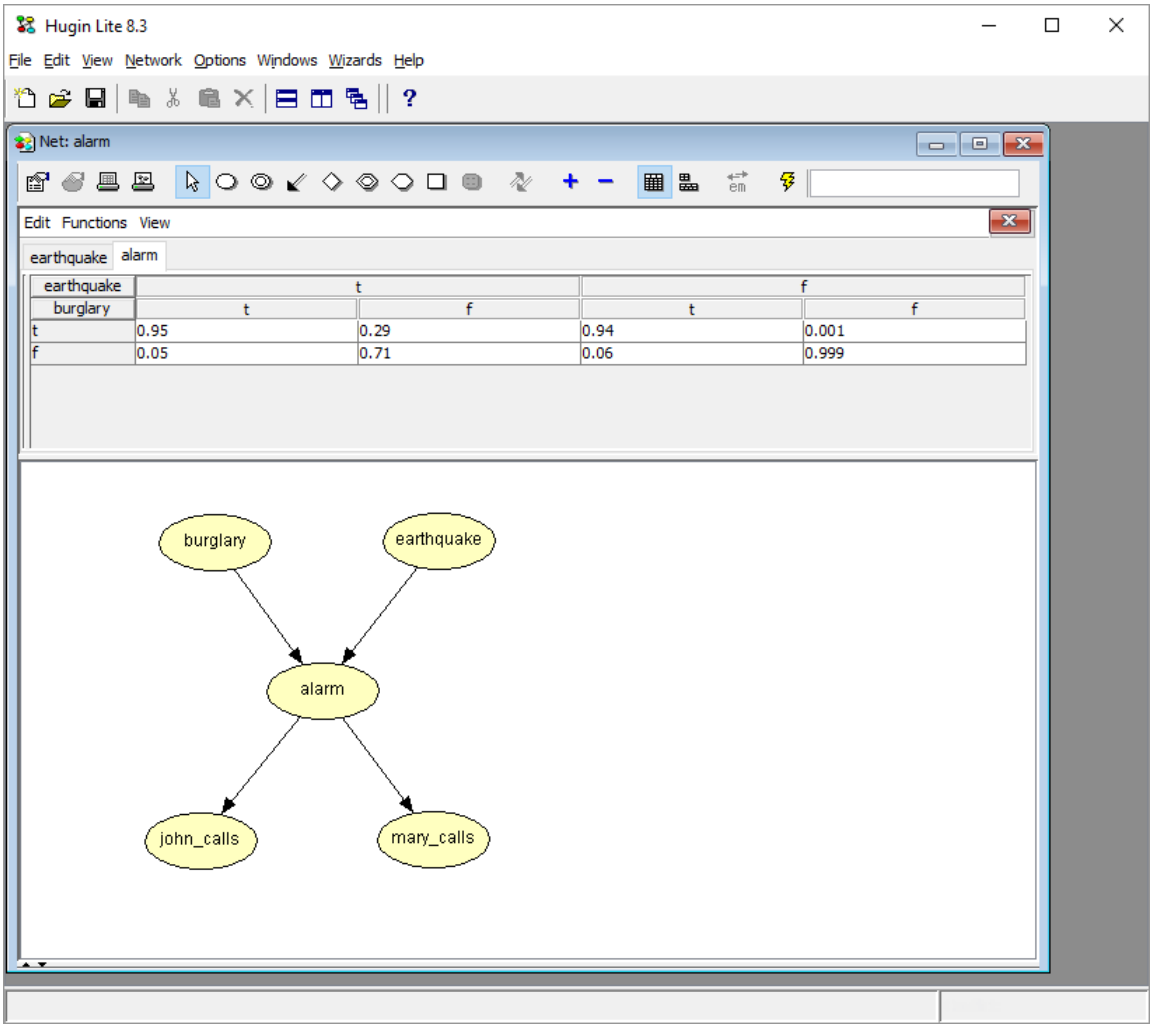


Figura 3.2 – Captura de tela do Hugin: edição das tabelas de probabilidade condicional
Fonte: Capturado pelo autor

Na barra de ferramentas superior o usuário pode clicar em um botão que altera a rede do modo de edição para o modo de execução. Neste modo, é possível visualizar as probabilidades de cada estado de cada nodo após a inferência. Também é possível informar os valores conhecidos de cada nodo clicando duas vezes em um dos seus estados, o que fica sinalizado em vermelho. Quando isso é feito, as probabilidades de todos os nodos da rede são recalculadas. O modo de execução pode ser visualizado na Figura 3.3.

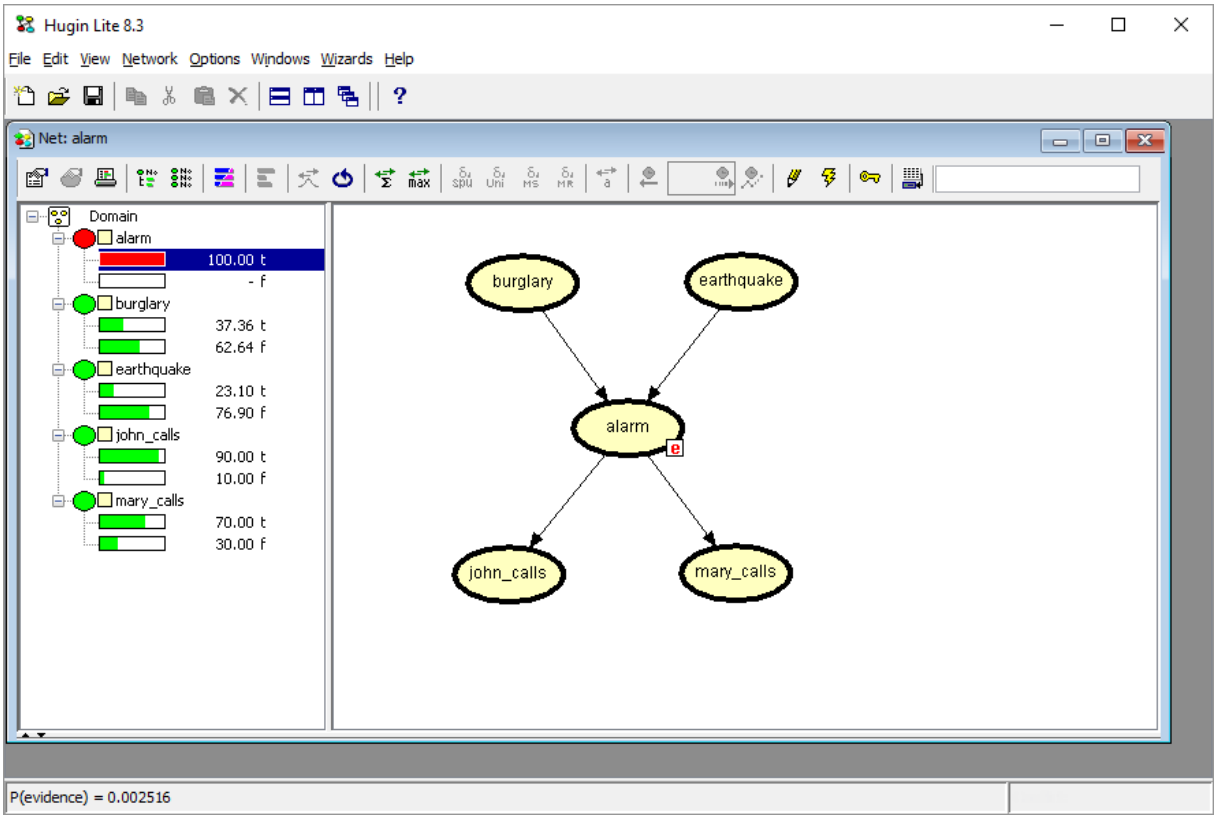


Figura 3.3 – Captura de tela do Hugin: modo de execução
Fonte: Capturado pelo autor

Além do editor gráfico, o Hugin também disponibiliza um conjunto de APIs para que programadores possam utilizar o motor de inferência do Hugin para criar suas próprias aplicações. Atualmente, as APIs do Hugin estão disponíveis como bibliotecas C, C++, .NET, Java e ActiveX.

3.1.1 Características observadas

O fato de o Hugin disponibilizar bibliotecas com APIs para diversas linguagens dá uma liberdade grande para que programadores utilizem redes bayesianas criadas no Hugin em suas próprias aplicações. Outro aspecto interessante do Hugin é que ele possui um conjunto diversificado de tipos de nodos, o que torna a ferramenta útil, ou mais eficiente, para um número maior de problemas.

O fato de o Hugin possuir o código fonte fechado, dependendo do cenário pretendido de aplicação das redes bayesianas, pode não ser o ideal, já que impede que modificações ou consultas sejam feitas no código do programa.

O fato de o modo de edição e execução serem separados pode diminuir a velocidade com que o usuário pode fazer simulações. Por exemplo, se ele, durante o modo de execução,

desejar alterar a probabilidade de algum estado de um nodo da rede, será necessário ir para o modo de edição, editar o estado desejado e então voltar para o modo de execução.

Um caso de uso que ficou confuso no Hugin foi a edição das CPTs quando o nodo possuía muitas combinações possíveis de estados. A Figura 3.2 mostra um exemplo, onde não fica imediatamente claro quais são os estados dos nodos pais e quais são os estados do nodo que está em edição no momento. Além disso, deve ser considerado que, no exemplo, todos os nodos são de domínio booleano e o nodo em questão possui apenas dois pais. Em aplicações reais, esse número tende a ser maior, o que dificultaria ainda mais a edição.

3.2 Netica

De acordo com a definição presente em seu site, o Netica é “Um pacote de software completo para resolver problemas utilizando redes bayesianas e diagramas de influência” (NETICA, 2016, tradução nossa). Da mesma forma que o Hugin, ele possui um editor gráfico e um conjunto de APIs que programadores podem utilizar em suas aplicações. As APIs estão disponíveis nas linguagens Java, C, C#, Visual Basic/COM, C++, Matlab e CLisp (NETICA, 2016).

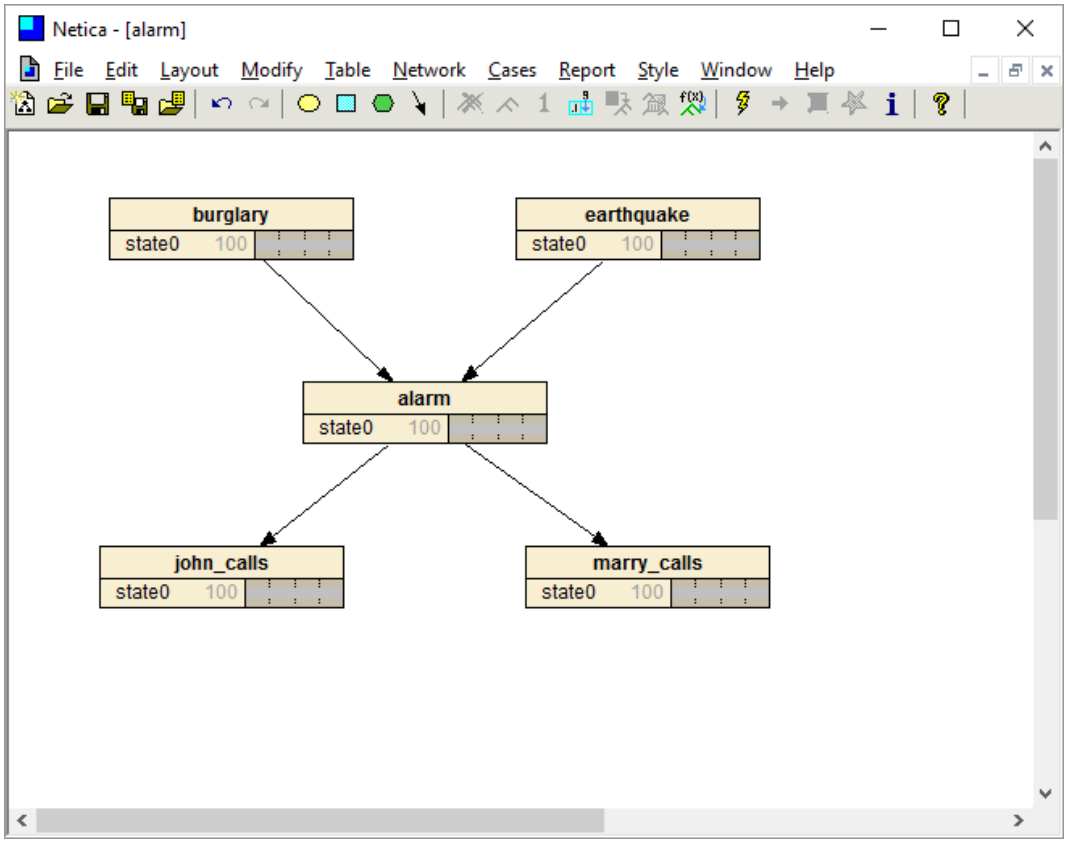


Figura 3.4 – Captura de tela do Netica: estrutura da rede
Fonte: Capturado pelo autor

Na Figura 3.4, é possível visualizar a tela principal de edição de redes bayesianas do Netica. Nela, há uma barra de ferramentas na parte superior onde é possível selecionar recursos para serem incluídos na rede bayesiana, sendo eles:

- **Nature node:** um nodo que representa uma variável aleatória na rede bayesiana. Pode ser tanto discreta quanto contínua;
- **Decision node:** um nodo que representa uma decisão a ser tomada por um usuário da rede, utilizada em diagramas de influência;
- **Utility node:** um nodo que representa um valor utilidade que deve ser maximizado em um diagrama de influência;
- **Link:** utilizada para adicionar ligações entre os nodos da rede.

Após montar a estrutura da rede, é preciso informar os estados para cada um dos nodos. Para isso, é necessário clicar com o botão direito do mouse no nodo desejado e escolher a opção "Modify / Set States...". Feito isso, será exibida a janela mostrada na Figura 3.5, onde os nomes dos estados devem ser inseridos, um em cada linha.

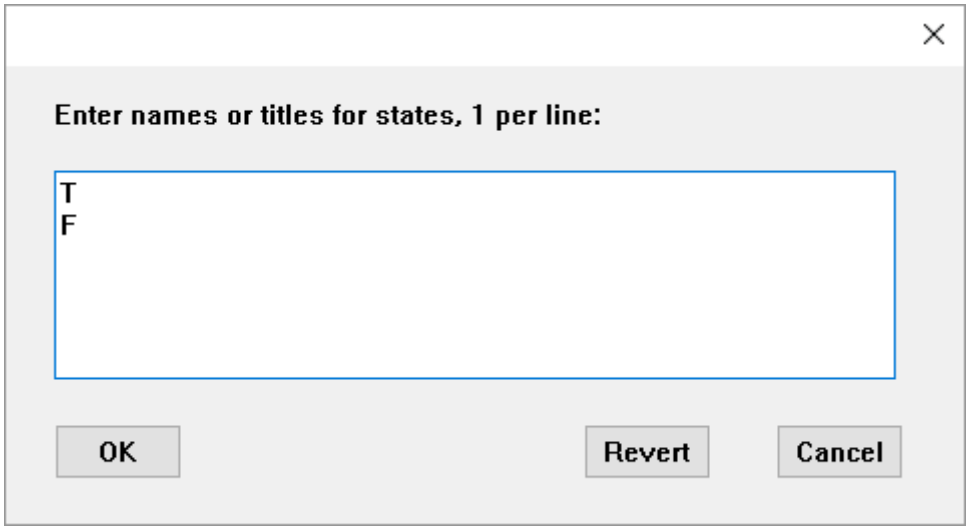


Figura 3.5 – Captura de tela do Netica: tela de edição de estados
Fonte: Capturado pelo autor

Após definir os estados de cada nodo, é preciso definir e probabilidade de cada um deles. Para isso, é preciso clicar com o botão direito do mouse no nodo desejado e escolher a opção "Table...' no menu de contexto. Feito isso, será exibida a tela mostrada na Figura 3.6, onde é preciso informar as probabilidades para cada combinação de estado possível.

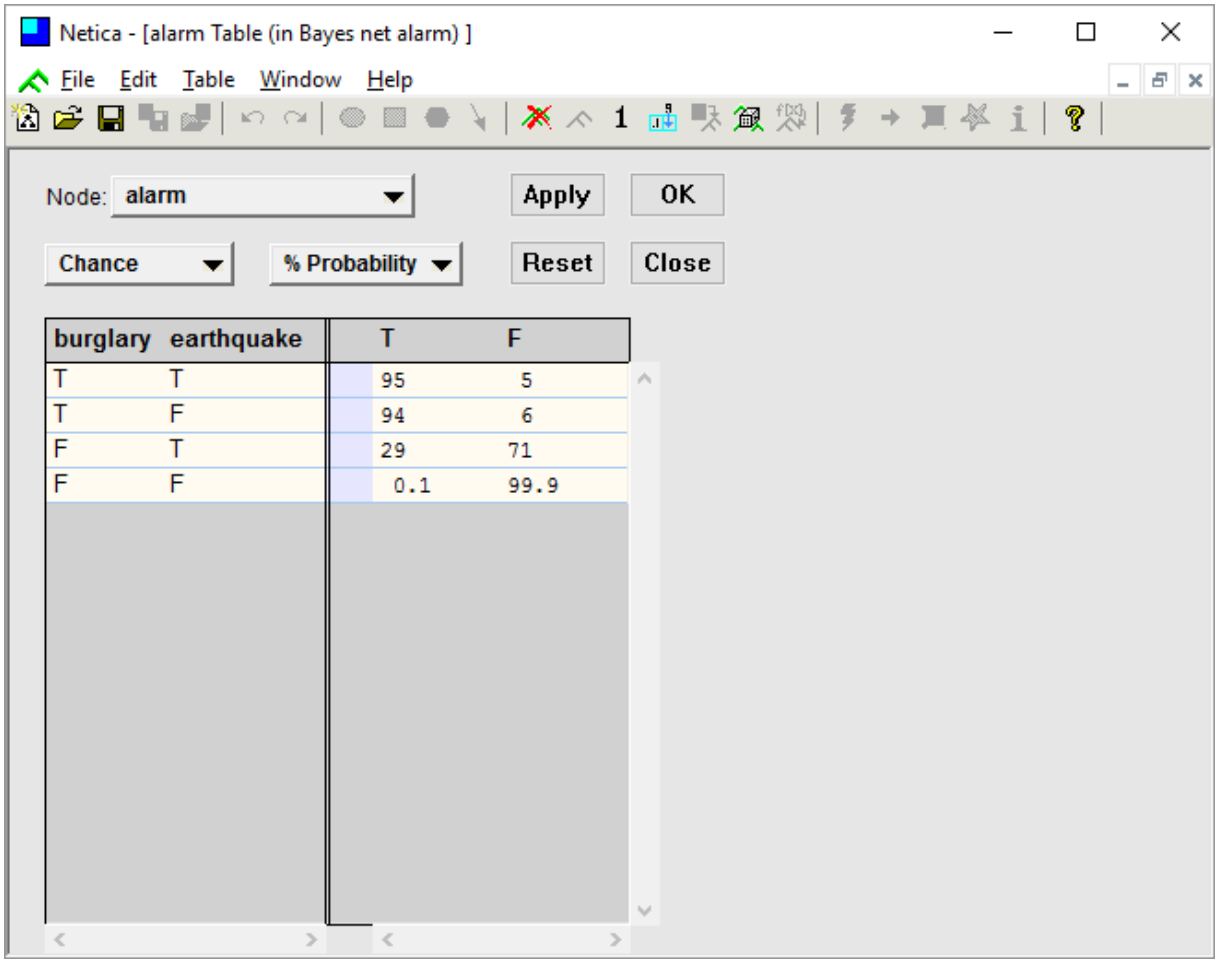


Figura 3.6 – Captura de tela do Netica: tela de edição de probabilidades
Fonte: Capturado pelo autor

No Netica, após definir as probabilidades de cada estado, elas não são propagadas automaticamente para todos os nodos da rede. Por isso, após todas as probabilidades serem definidas, é preciso clicar no botão “Compile” presente na barra de ferramentas superior.

Após a rede estar compilada, é possível realizar inferências nos nodos criados. Para informar um estado já conhecido para um dos nodos, basta clicar em cima no estado desejado. A Figura 3.7 mostra os resultados após a inferência na rede informando o estado do nodo “alarm” como T.

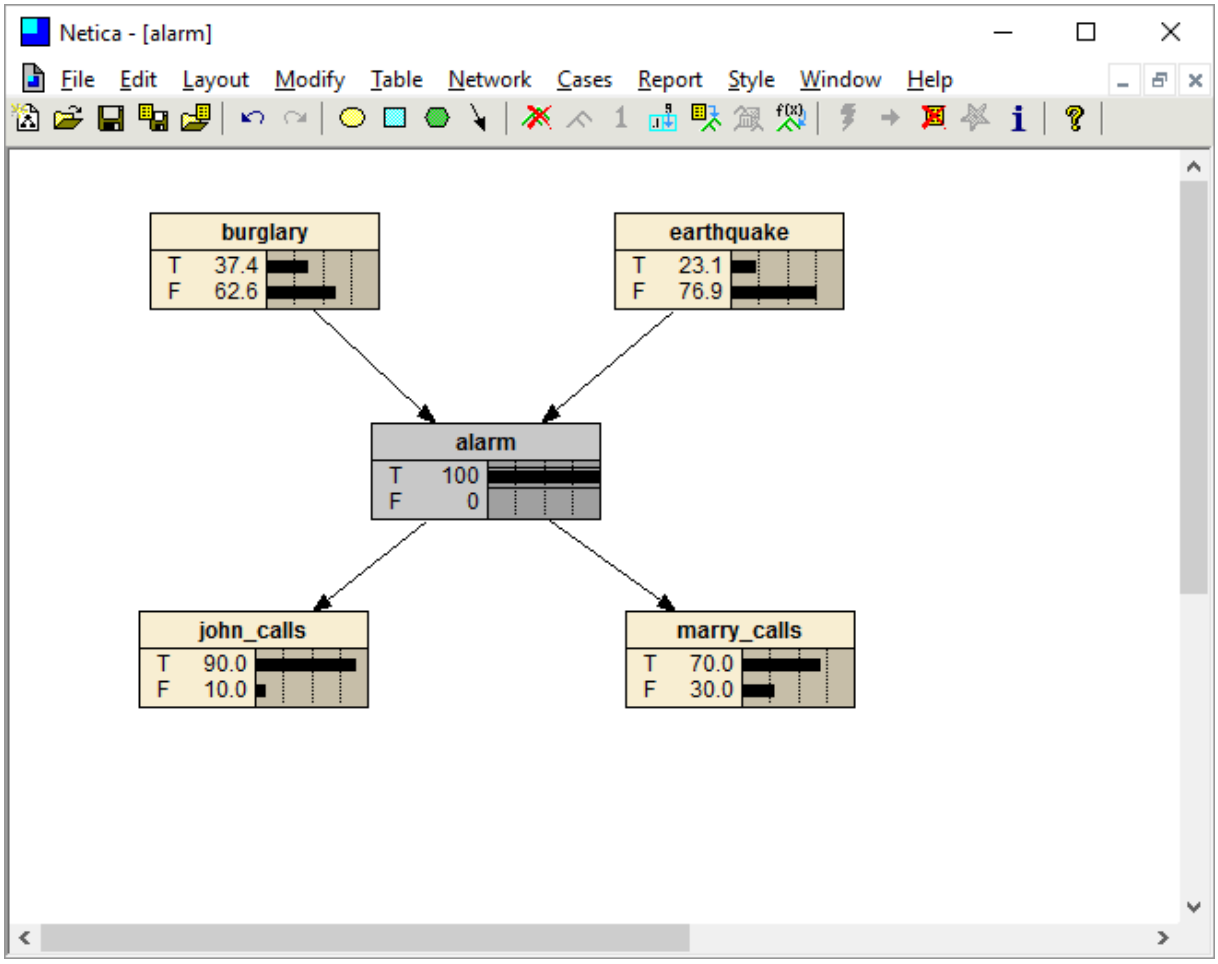


Figura 3.7 – Captura de tela do Netica: tela de inferências
Fonte: Capturado pelo autor

3.2.1 Características observadas

Assim como o Hugin, o Netica possui uma série de APIs para uso de programadores em suas próprias aplicações, o que pode ser bastante útil dependendo do contexto de aplicação. Ele também possui uma grande diversidade de tipos de nodos, o que faz com que a ferramenta seja útil em diferentes aplicações.

No Netica, a edição da tabela de probabilidades mostrou-se bastante natural, já que ficou evidente quais estados representam os estados dos nodos pai e quais são referentes ao nodo que é editado, conforme mostrado pela Figura 3.6. Também é possível selecionar se as probabilidades serão informadas em valores percentuais, de 0 a 100, ou em valores numéricos representando a probabilidade, de 0 a 1.

Além disso, como não existe a separação de um modo de edição ou execução, o Netica mostra-se como uma ferramenta com um fluxo ágil para simulações, já que o usuário fica a poucos cliques da edição das CPTs enquanto pode visualizar os resultados da inferência.

O Netica possui mais de uma maneira para informar os estados dos nodos, mas, apesar de elas tornarem a maneira de informar os valores mais flexível, ficou confuso para o usuário entender qual delas utilizar. Além disso, foram encontrados vários *bugs* ao tentar realizar este processo, o que dificultou bastante a execução da tarefa.

Outro aspecto observado é que a compilação da rede não é automática. Sendo assim, após o usuário informar as probabilidades da rede, ele precisa manualmente clicar no botão para recompilar. Caso o usuário não perceba isso, ele pode assumir valores incorretos na inferência da rede.

Além destes pontos, assim como o Hugin, o Netica também possui o código fonte fechado, o que pode não ser o ideal dependendo da aplicação pretendida.

3.3 UnBBayes

De acordo com a definição presente em seu site, o UnBBayes é um software de código aberto para modelagem, aprendizado e raciocínio baseado em redes bayesianas (UNBBAYES, 2016, tradução nossa). Ele é um *framework* para montagem e inferência de redes bayesianas desenvolvido pela Universidade de Brasília (SANTOS, 2009). Ao abrir o UnBBayes e criar uma nova rede, é exibida a tela mostrada na Figura 3.8.

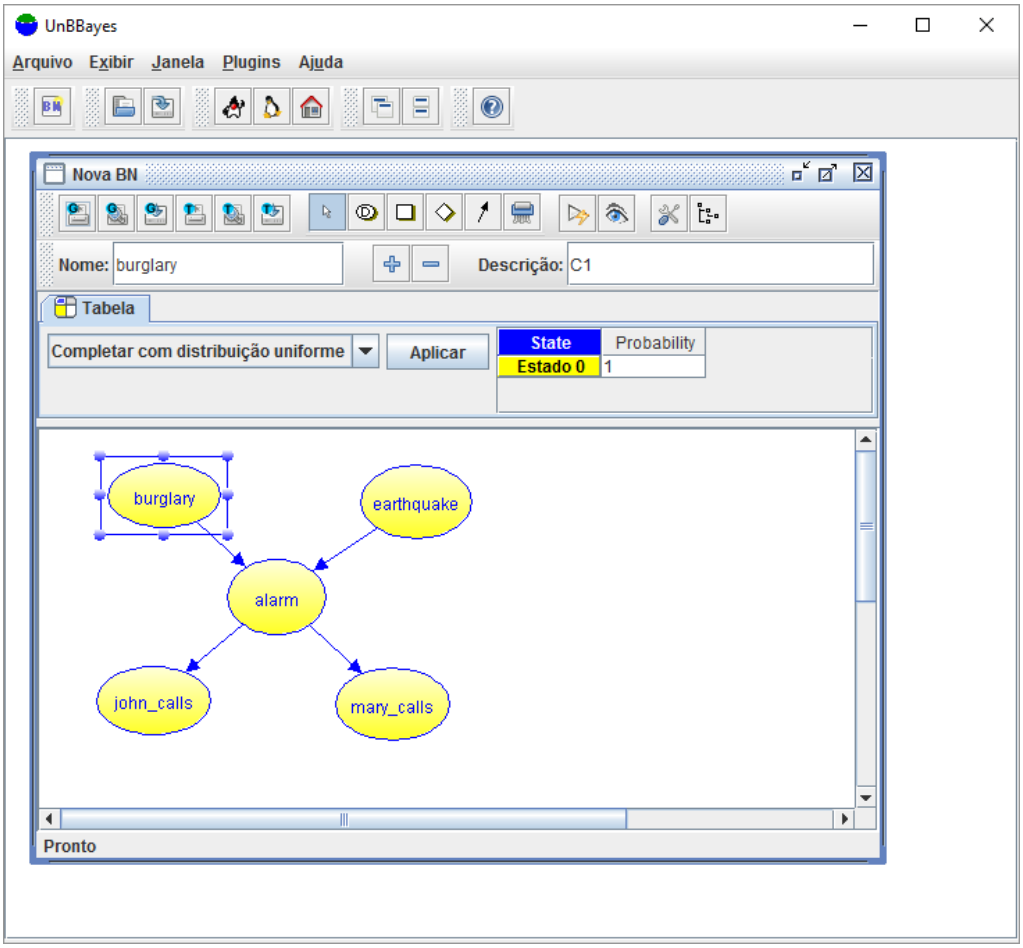


Figura 3.8 – Captura de tela do UnBBayes: tela principal
Fonte: Capturado pelo autor

Assim como nas outras ferramentas avaliadas, na parte superior existe uma barra de ferramentas que pode ser utilizada para inserir nodos de probabilidade, decisão e utilidade, além de arcos para as ligações entre eles. No UnBBayes, não é possível adicionar nodos representando variáveis contínuas, apenas discretas. Após definir a estrutura da rede, para adicionar os estados nos nodos presentes, é preciso selecionar um deles e clicar no botão “+” que fica logo abaixo da barra de ferramentas. Feito isso, um novo estado será adicionado na CPT na aba “Tabela”, que pode ser editada imediatamente. Uma figura da rede com os estados e probabilidades preenchidos pode ser observada na Figura 3.9.

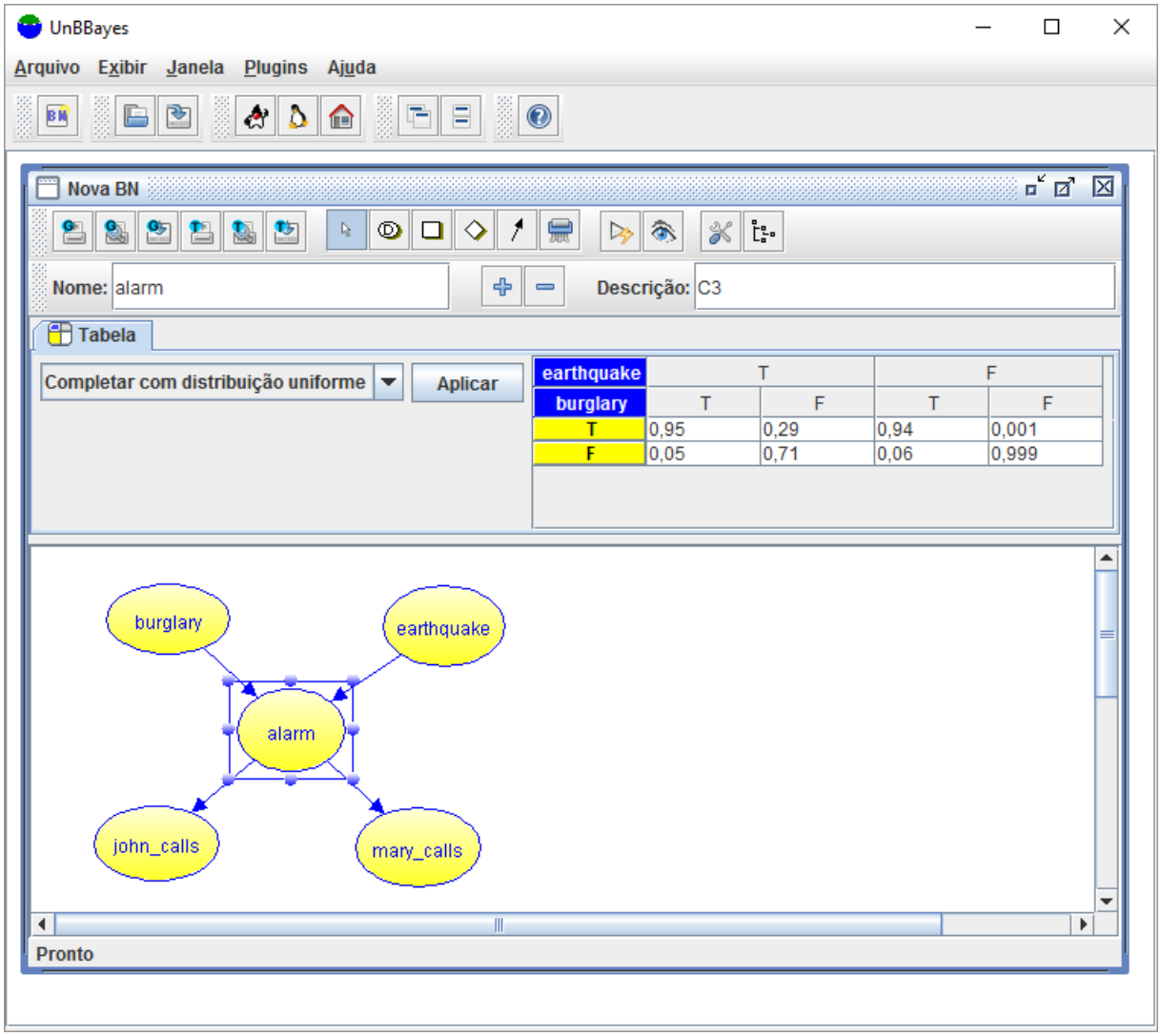


Figura 3.9 – Captura de tela do UnBBayes: rede com probabilidades preenchidas
Fonte: Capturado pelo autor

Quando a rede estiver finalizada, é possível entrar no modo de execução clicando no botão “Compilar árvore de junção”. Nesta tela, na parte esquerda, há uma lista com os nodos e seus estados, onde são exibidos os resultados da inferência. Além disso, há uma exibição da rede com os resultados em cada nodo, parecida com a exibição do Netica. Ao clicar em um dos estados de um nodo, tanto na listagem da esquerda quando diretamente no nodo, a crença de que aquela variável está naquele estado é inserida. Porém, a propagação das crenças não é automática, é preciso clicar no botão “Propagar evidências” para que os resultados da inferência sejam exibidos. Uma visualização desta tela pode ser observada na Figura 3.10.

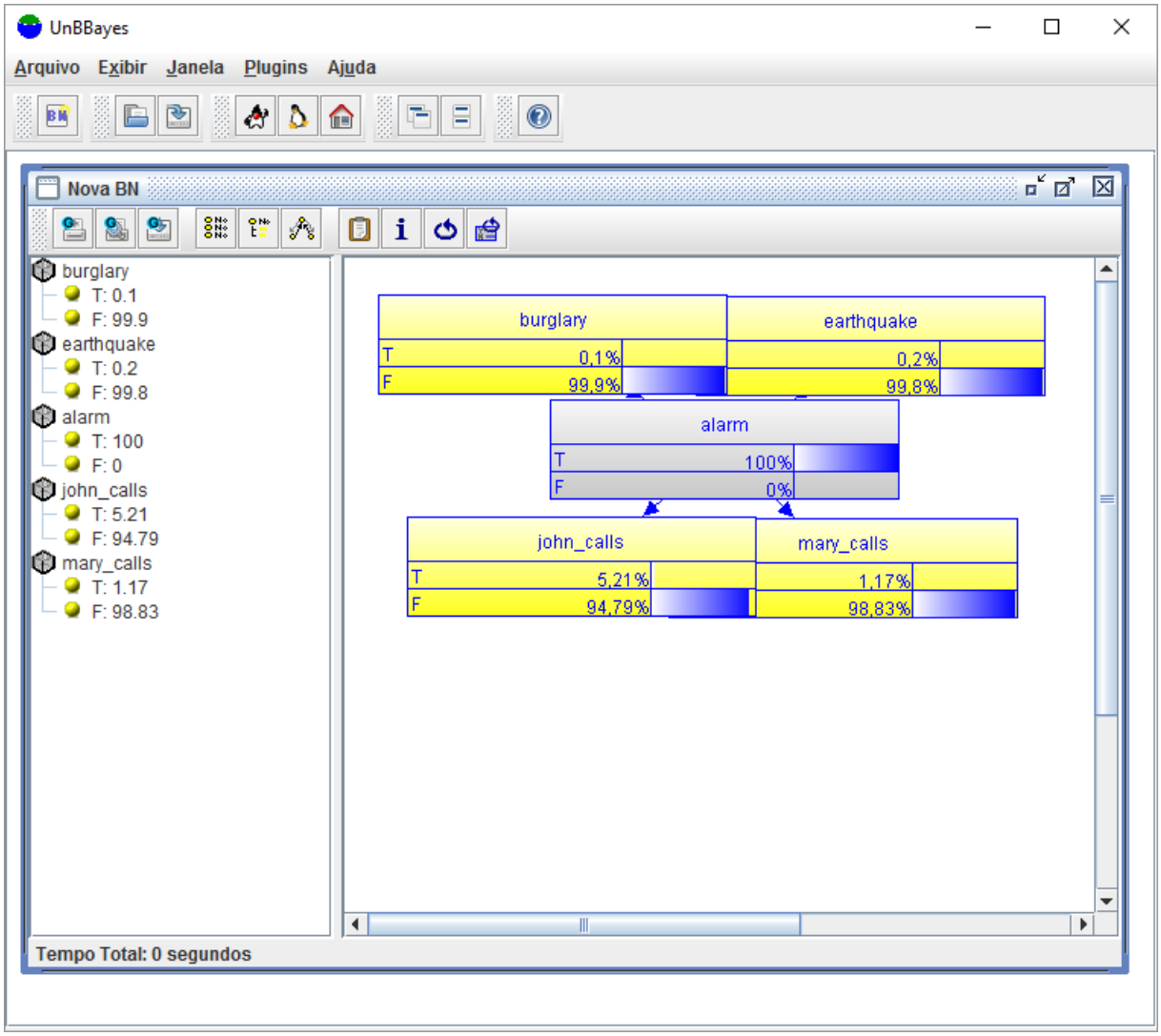


Figura 3.10 – Captura de tela do UnBBayes: modo de execução
Fonte: Capturado pelo autor

O UnBBayes também disponibiliza uma API para utilização por programadores, mas apenas para a linguagem Java.

3.3.1 Características observadas

O UnBBayes é de código aberto, o que permite que modificações e consultas ao seu código sejam feitas. Isto o torna uma ferramenta bastante flexível, pois, dependendo da aplicação, isto pode ser necessário.

Outro aspecto interessante é que a CPT é de fácil acesso para edição. Basta selecionar o nodo desejado e ela ficará disponível na parte superior da rede para edição, conforme mostrado na Figura 3.9.

Apesar de o UnBBayes disponibilizar uma API para programadores, ela só está disponível na linguagem Java. Nos demais softwares analisados, o conjunto de linguagens em que a API está disponível é maior. O UnBBayes também não possui nodos contínuos. Dependendo da aplicação da ferramenta, este fator pode ser importante.

Apesar de a tabela para edição de estados ser de fácil acesso, não é óbvio para o usuário de que nesta tabela ele pode alterar o nome dos estados, o que pode fazê-lo pensar que não é possível editar. Como a tabela de edição das probabilidades é bem semelhante à do Hugin, ela possui os mesmos problemas em relação a complexidade, que cresce de forma proporcional ao número de estados.

No modo de edição, é possível informar os valores dos estados conhecidos. Porém, nas outras ferramentas, para remover uma crença informada, basta clicar novamente no estado que foi informado. No UnBBayes, é preciso ir até o botão “Reiniciar crenças”, o que, além de não ser tão ágil, pode não ser óbvio ao usuário.

Além disso, apesar de ele possuir os modos de edição e execução separados, a propagação das probabilidades não é automática, é preciso realizar a ação manualmente. Desta forma, além de o usuário precisar ficar alternando entre os dois modos, ele precisa realizar a propagação manualmente, o que, além de não facilitar a tarefa de simulações, pode fazer com que ele visualize probabilidades inconsistentes com os estados caso não perceba que a propagação precise ser feita.

3.4 AMPLIA

O AMPLIA é um projeto no qual o aluno representa seu raciocínio de diagnóstico por meio de uma rede bayesiana e este raciocínio é avaliado por agentes inteligentes (SEIXAS, 2005). O editor de redes bayesianas do AMPLIA é baseado no editor gráfico para redes bayesianas SEAMED, aplicação desktop desenvolvida em Delphi 6.0 (FLORES, 2005).

O objetivo do AMPLIA é muito mais específico do que os outros softwares avaliados até o momento. Enquanto os outros editores são de propósito geral, o AMPLIA é um software direcionado para o ensino na área da saúde. Por este motivo, este trabalho avaliará apenas o editor de redes bayesianas contido no AMPLIA, e não o software como um todo. A tela de edição de redes bayesianas do AMPLIA pode ser visualizada na Figura 3.11.

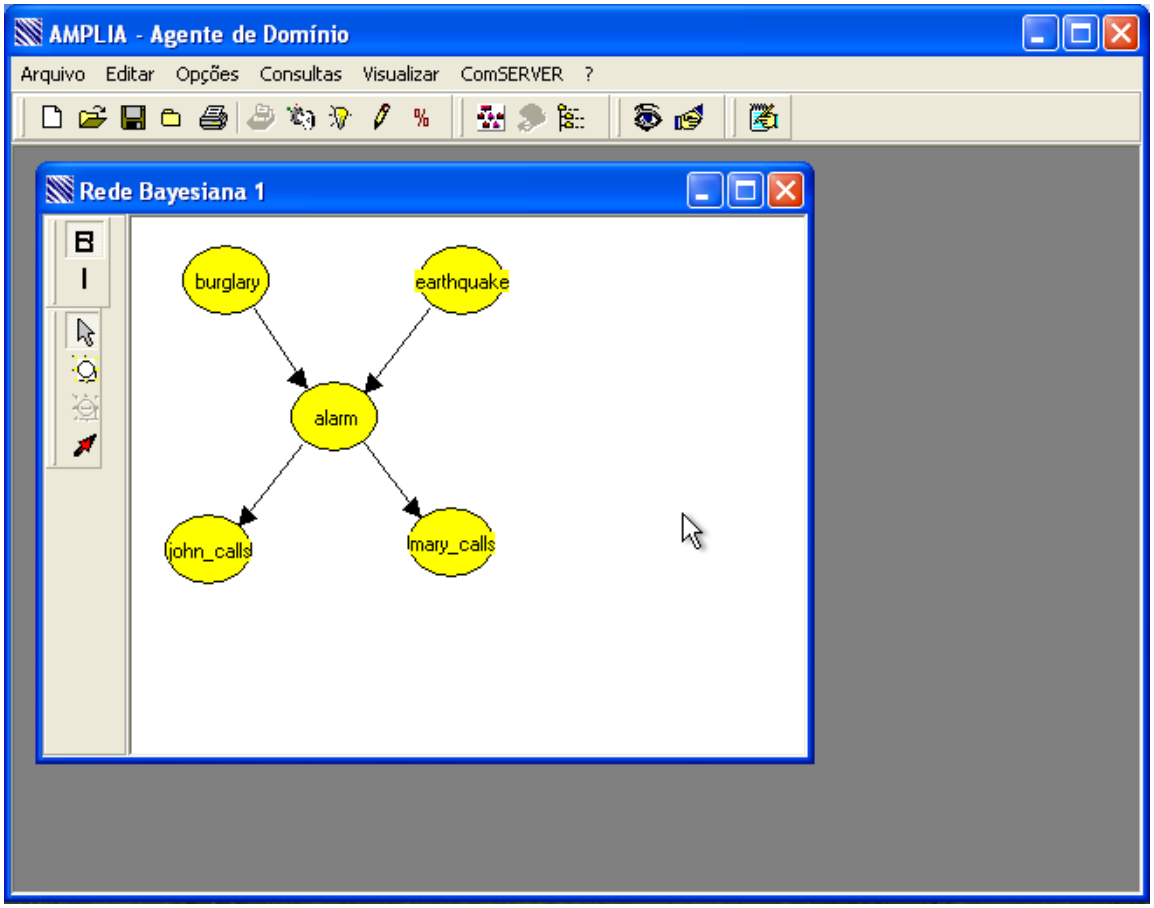


Figura 3.11 – Captura de tela do AMPLIA: edição da rede bayesiana
Fonte: Capturado pelo autor

No AMPLIA, a barra de ferramentas para incluir itens na rede fica na parte esquerda. Nesta barra, é possível alternar entre os modos “Rede bayesiana” e “Diagrama de influência”. No modo “Rede bayesiana”, é possível incluir nodos representando variáveis aleatórias discretas e as ligações entre eles. No modo “Diagrama de influência”, além dos mesmos itens disponíveis nas redes bayesianas, é possível incluir nodos do tipo decisão e utilidade. Para editar as propriedades de um nodo, como seu nome, tipo, estados e probabilidades, é preciso clicar com o botão direito do mouse nele e escolher a opção “Propriedades”.

A tela de “Propriedades Básicas” pode ser visualizada na Figura 3.12. Nela, é possível definir quais estados o nodo possui, seu nome e seu tipo, entre outras propriedades pertinentes ao contexto do AMPLIA.

Ao trocar para a aba “Probabilidades Condicionais”, é exibida a tela mostrada na Figura 3.13. Nesta tela, os valores das probabilidades de cada estado podem ser preenchidos diretamente nas células da tabela ou ajustados utilizando os *sliders* verticais presentes na direita. Ao preencher a probabilidade de um estado, a probabilidade dos outros é ajustada automaticamente para totalizar 1.

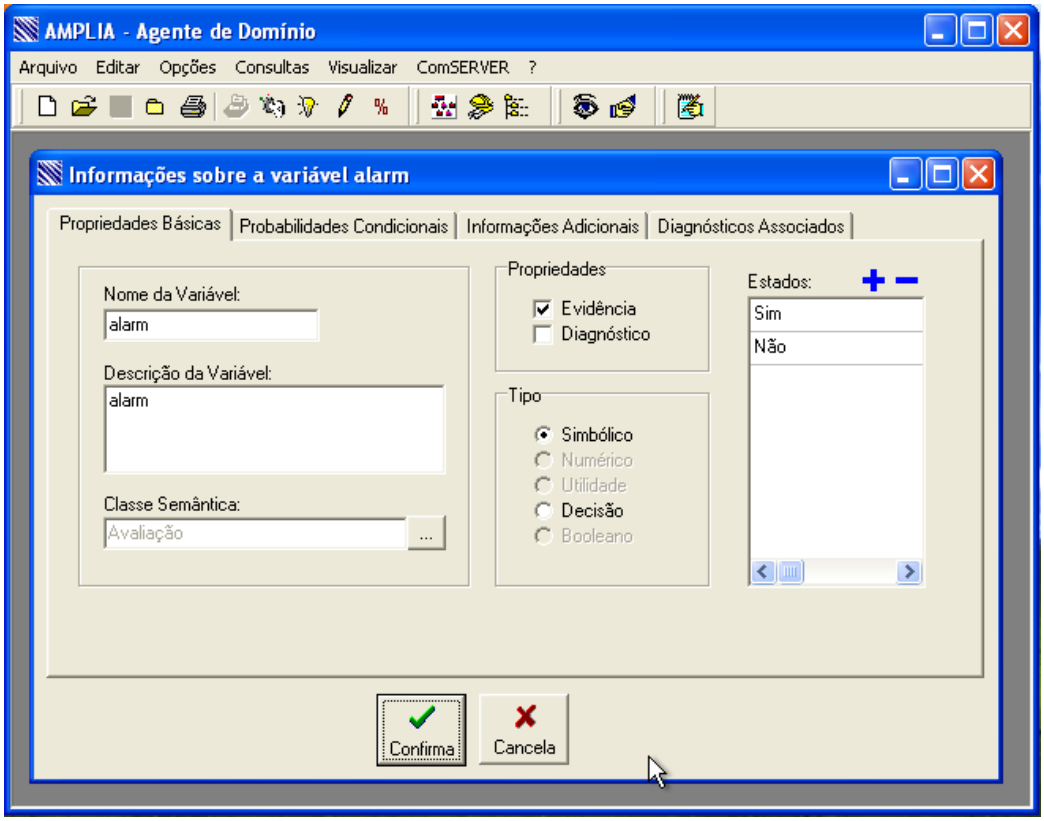


Figura 3.12 – Captura de tela do AMPLIA: edição das propriedades básicas do nodo
Fonte: Capturado pelo autor

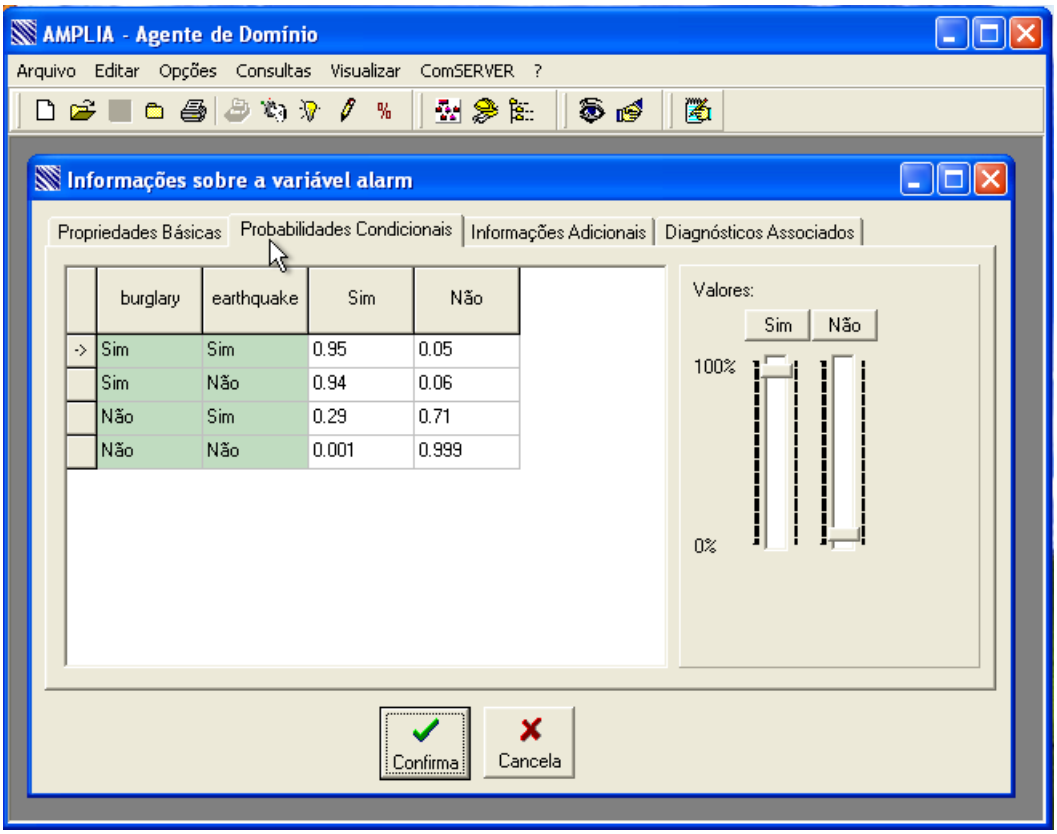


Figura 3.13 – Captura de tela do AMPLIA: edição das probabilidades condicionais do nodo
Fonte: Capturado pelo autor

Feita a montagem da rede, é possível realizar a inferência ao clicar no botão “Consulta”. Quando isso é feito, é exibida uma tela onde o usuário pode informar os estados conhecidos de alguns nodos da rede, conforme mostrado pela Figura 3.14. Após a definição dos estados conhecidos, o usuário pode clicar em “Processar” para que a inferência seja feita e os resultados exibidos, conforme mostrado na Figura 3.15.

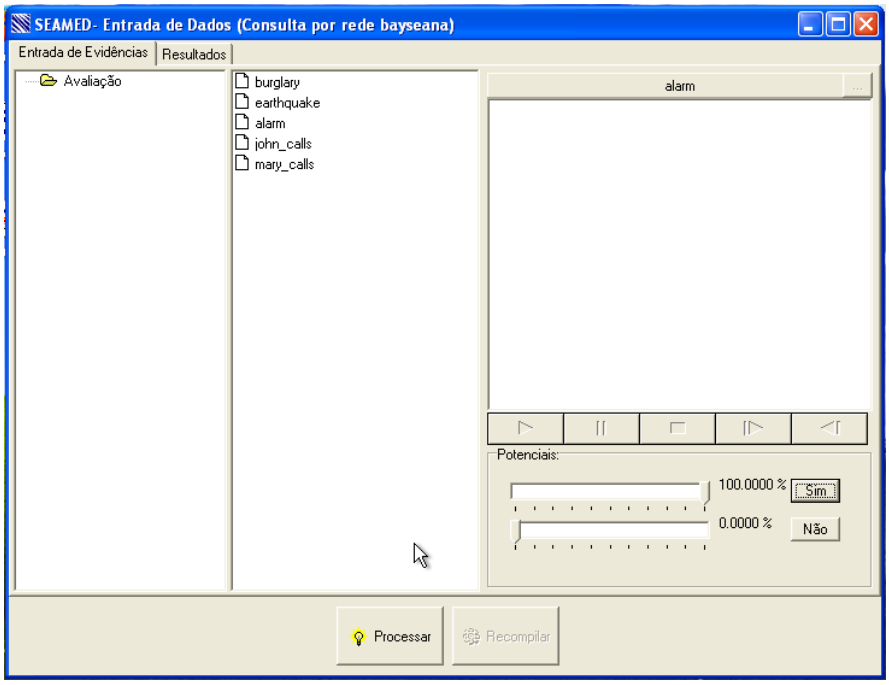


Figura 3.14 – Captura de tela do AMPLIA: tela de entrada de evidências
Fonte: Capturado pelo autor

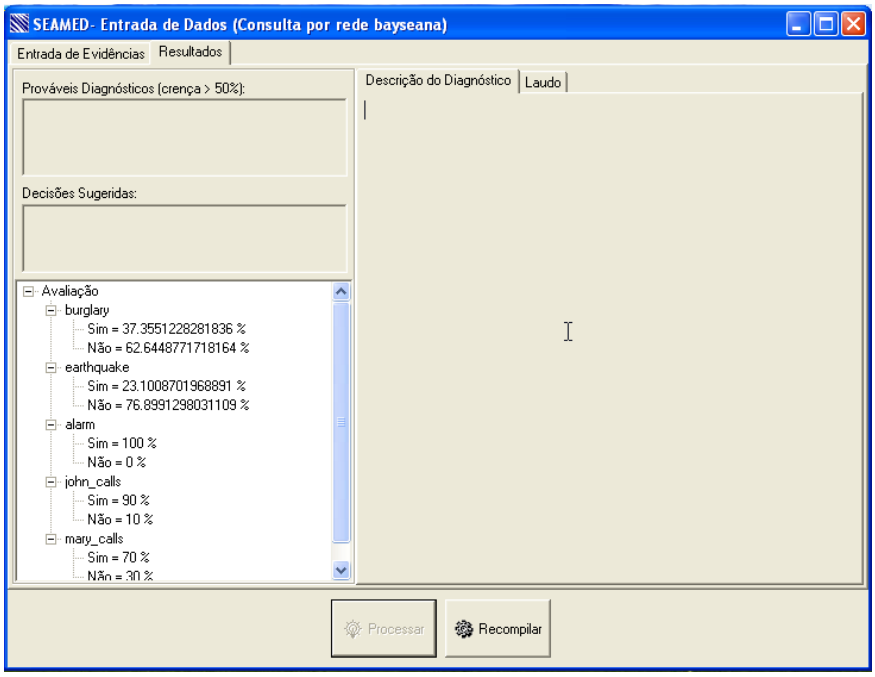


Figura 3.15 – Captura de tela do AMPLIA: tela de resultados
Fonte: Capturado pelo autor

3.4.1 Características observadas

Ao observar as características do AMPLIA, vale ressaltar que ele é um software destinado ao ensino da saúde, e não um editor de redes bayesianas de propósito geral. Por este motivo, algumas das funcionalidades desejadas nos outros editores, como, por exemplo, uma diversidade de tipos de nodos, APIs disponíveis e código aberto, não estão presentes no AMPLIA, pois não se enquadram em seu escopo.

Ao editar as probabilidades de um nodo, o valor dos outros estados é ajustado automaticamente para totalizar 1, o que se mostrou bastante conveniente quando o nodo possuía apenas dois estados. Porém, quando existe um número maior de estados, isso torna a edição um pouco confusa. Por exemplo, para três estados, quando o usuário informa o valor do primeiro, os outros dois são ajustados proporcionalmente para que a soma de todos totalize 1. Feito isso, quando ele for editar a probabilidade do segundo estado, a probabilidade do primeiro, que já havia sido informada, será ajustada, junto com a probabilidade do terceiro, para totalizar 1.

Para mitigar essa situação, quando o nodo possui mais de três estados é exibida uma caixa de seleção com o texto “Fix” em baixo de cada um dos estados para que o valor dele não seja atualizado automaticamente. Embora isto resolva a problema, não é óbvio para o usuário. De acordo com Krug (2006), o mais importante para que um produto seja fácil de usar é tentar torná-lo o mais óbvio possível, fazendo com que o usuário, nas palavras do autor, “não precise pensar” para realizar as tarefas.

Ao executar o editor de redes bayesianas do AMPLIA, não foi possível redimensionar o tamanho de um dos nodos, fazendo com que o seu nome ficasse maior do que o nodo em si, como pode ser observado no nodo “earthquake” da Figura 3.11.

Para realizar a inferência no AMPLIA, se comparado aos outros editores, uma quantidade grande de passos e telas é preciso para chegar nos resultados. Isso torna o fluxo da tarefa de realizar simulações não ser tão ágil.

3.5 Comparativa de funcionalidades

A partir das características descritas neste capítulo em relação aos editores de redes bayesianas existentes, é possível montar um quadro para que seja feita uma comparação entre as funcionalidades presentes neles. O Quadro 3.1 exibe a comparativa para os editores analisados. Desta forma, o escopo de cada um pode ser observado com mais facilidade.

Quadro 3.1 – Comparativa de funcionalidades entre os editores analisados.

	Hugin	**Netica**	**UnBBayes**	**AMPLIA**
Possui nodos de estados discretos	Sim	Sim	Sim	Sim
Possui nodos de estados contínuos	Sim	Sim	Não	Não
Suporte a diagramas de influência	Sim	Sim	Sim	Sim
Impressão da estrutura da rede bayesiana	Sim	Sim	Não	Não
Impressão das CPTs da rede bayesiana	Sim	Não	Não	Não
Código fonte	Fechado	Fechado	Aberto	Fechado
APIs	Disponíveis em diversas linguagens	Disponíveis em diversas linguagens	Disponível apenas na linguagem Java	Não disponível
Facilidade de edição das CPTs	Dificuldade de visualizar quais estados são referentes ao nodo em edição e quais são referentes aos estados dos nodos pai	Quais estados se referem aos nodos pais e quais se referem ao nodo atual de fácil visualização	Dificuldade de visualizar quais estados são referentes ao nodo em edição e quais são referentes aos estados dos nodos pai	Quais estados se referem aos nodos pais e quais se referem ao nodo atual de fácil visualização
Agilidade na edição das probabilidades durante a inferência	Modo de edição e execução separados, o que causa uma troca de contexto pelo usuário	Não existe separação entre o modo de edição e execução, o que torna a ferramenta bastante ágil para esta tarefa	Modo de edição e execução separados, o que causa uma troca de contexto pelo usuário	Modo de edição e execução separados, o que causa uma troca de contexto pelo usuário. Além disso, vários passos são necessários para realizar a inferência.

Fonte: Itens levantados pelo autor.

4 DESENVOLVIMENTO DO EDITOR

Com base nos resultados obtidos no estudo dos trabalhos relacionados, um novo editor foi desenvolvido. Este capítulo relatará as etapas deste processo de desenvolvimento. Primeiramente serão apresentados os *wireframes* das telas, que foram criados levando em conta os resultados da análise dos trabalhos relacionados, buscando uma boa usabilidade. Em seguida, serão apresentadas as tecnologias escolhidas para o desenvolvimento do editor. Após isso, será apresentado o algoritmo utilizado para realizar as inferências nas redes bayesianas. Por fim, será apresentado o editor desenvolvido.

4.1 Wireframes

Para a criação dos *wireframes*, foi utilizada a ferramenta *Pencil*. Os *wireframes* buscam representar a estrutura da aplicação, com foco em funcionalidades e comportamentos. Desta forma, os *wireframes* não representam questões estéticas, como, por exemplo, tipografia, cores, etc.

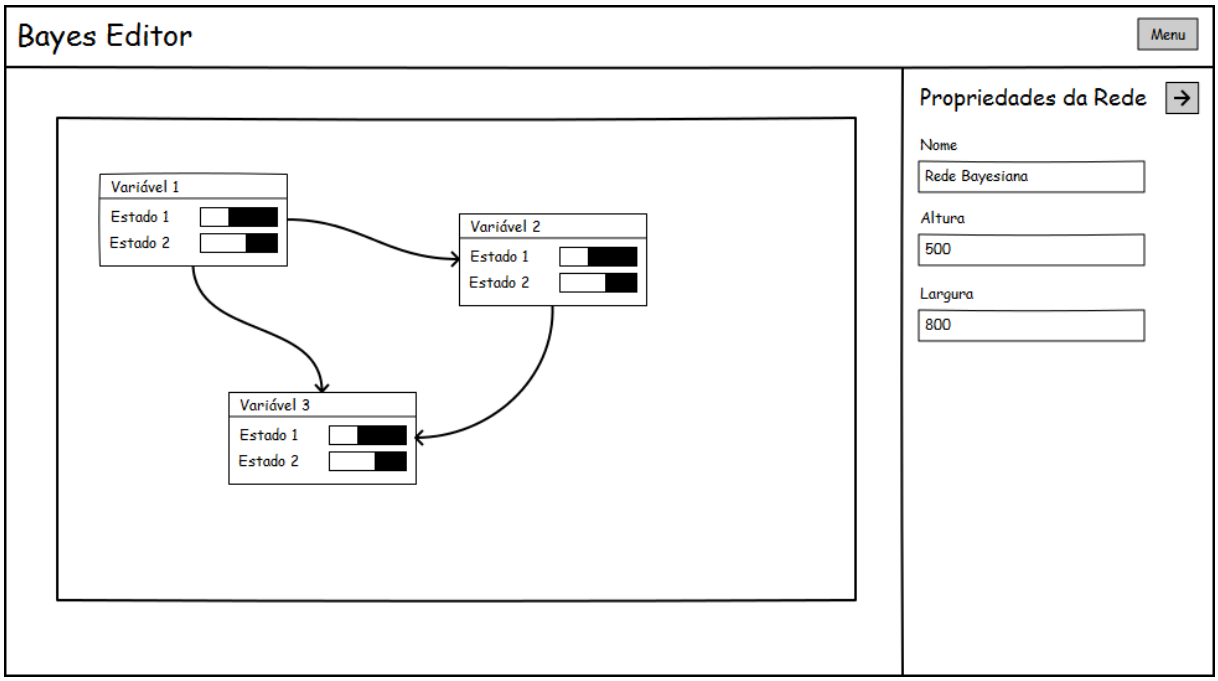


Figura 4.1 – *Wireframe* da tela principal
Fonte: Construída pelo autor

A Figura 4.1 mostra o *wireframe* da tela principal do software. Nesta tela, o usuário pode acompanhar em tempo real o resultado da inferência na rede bayesiana, pois após qualquer modificação na rede, o resultado é atualizado e exibido pelas barras ao lado de cada estado das variáveis. A barra vazia representa a probabilidade de 0% enquanto a barra cheia representa a

probabilidade de 100%. Ao passar o mouse em cima de uma dessas barras, o valor numérico é exibido ao usuário.

Também é possível dar um clique duplo em cima de um dos estados para que uma evidência seja informada para a inferência na rede bayesiana. Quando isso acontece, a cor da barra é alterada e sua probabilidade vai para 100%, enquanto a probabilidade dos outros estados da mesma variável é informada como 0%. O resultado da inferência das outras variáveis também é atualizado, levando em consideração a evidência informada. Para remover a evidência, basta dar um clique duplo sobre ela novamente.

Na tela principal há um cabeçalho com um botão para acesso ao menu. Ao clicar nele, o usuário pode acessar as seguintes opções:

- **Nova Rede:** carrega uma rede bayesiana vazia para que o usuário possa construí-la. Caso uma rede bayesiana esteja em edição, é dado um aviso ao usuário de que os dados dela serão perdidos caso ele não os salve, com opção de cancelar a criação de uma nova rede;
- **Abrir Rede:** abre uma rede bayesiana salva previamente;
- **Salvar Rede:** salva a rede bayesiana em um arquivo.

Além disso, na direita há um painel de propriedades no qual o usuário pode configurar as propriedades da rede. O nome da rede serve apenas para identificá-la. Ao alterar o tamanho da rede nos campos altura e largura, a área para a edição da rede é ajustada. Desta forma, o usuário pode configurar o tamanho que for mais conveniente de acordo com o tamanho de sua rede bayesiana e da resolução de seu monitor.

Ao clicar no botão representado por uma seta que fica dentro do painel de propriedades, o painel é escondido até que o botão seja clicado novamente, conforme mostrado pela Figura 4.2. Desta forma, o espaço do monitor pode ser melhor aproveitado durante a montagem da rede bayesiana, pois mais espaço fica disponível para a visualização da estrutura da rede.

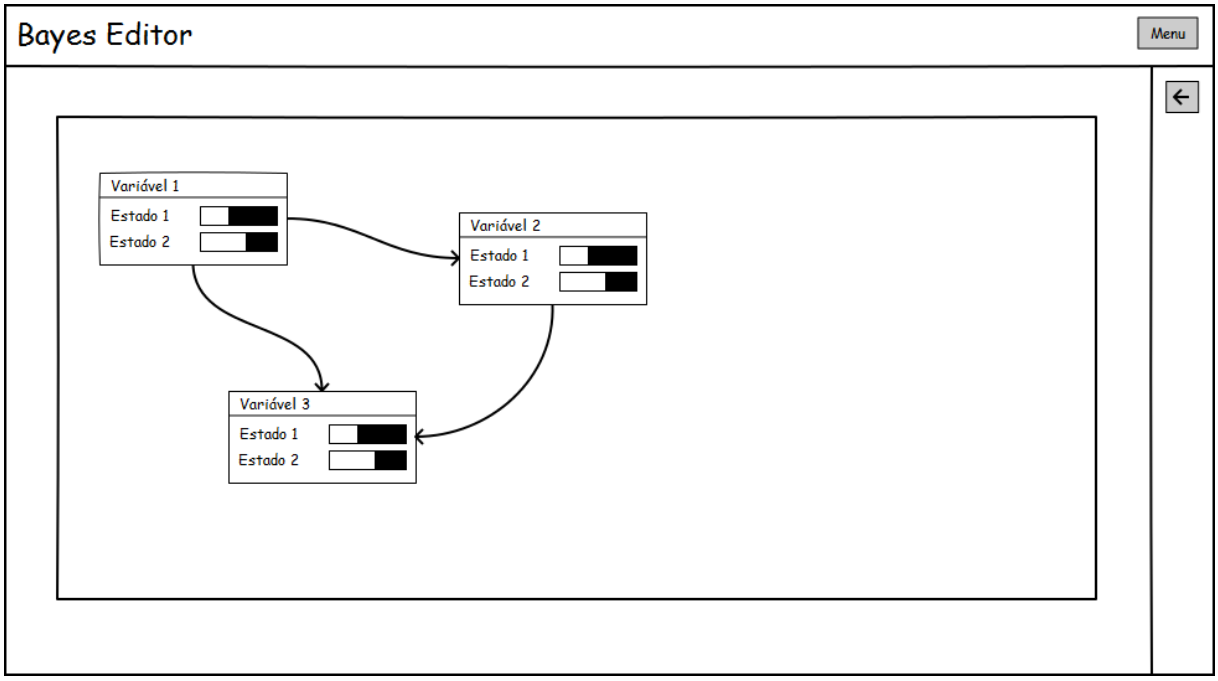


Figura 4.2 – *Wireframe* da tela principal com painel de propriedades escondido
Fonte: Construída pelo autor

No retângulo maior, o usuário pode montar a estrutura da rede bayesiana. Ao clicar com o botão direito do mouse em qualquer parte desta área, há uma opção para adicionar uma variável na rede. Ao clicar nesta opção, a tela mostrada na Figura 4.3 é exibida.

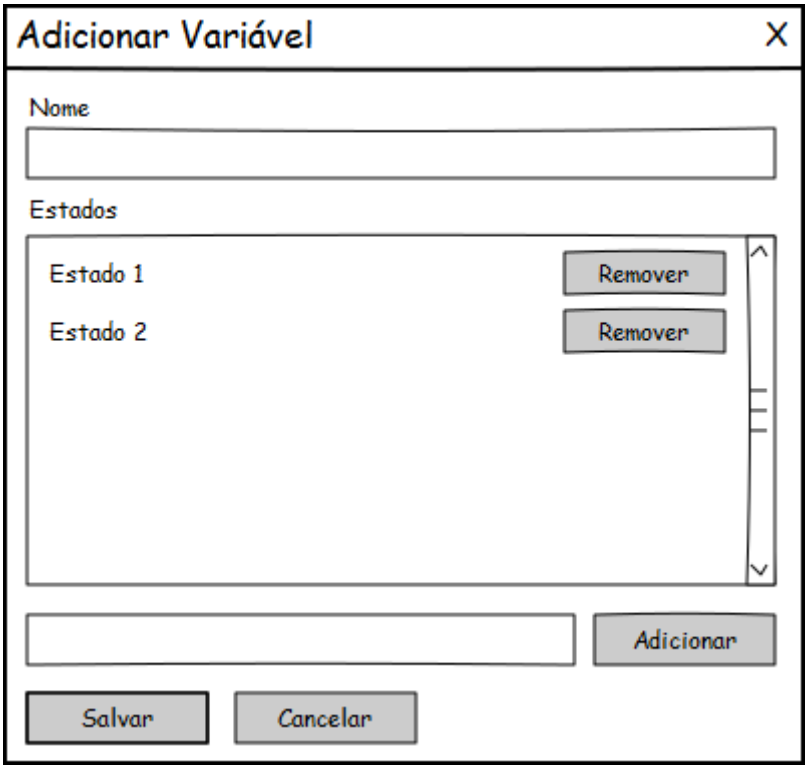


Figura 4.3 – *Wireframe* da tela para adicionar variável
Fonte: Construída pelo autor

Nesta tela o usuário pode preencher o nome da nova variável e montar a lista de estados que ela possuirá. Ao adicionar um novo estado é validado que ele tenha sido preenchido e que já não exista outro estado com o mesmo nome na lista. Ao clicar em salvar, a nova variável será adicionada e exibida na tela principal. Antes de salvar, as seguintes validações são realizadas e, caso alguma delas não passe, o usuário é notificado e precisa corrigir os dados antes de tentar salvar novamente:

- O nome da variável não pode estar vazio;
- Não pode existir outra variável com o mesmo nome;
- A variável precisa possuir pelo menos um estado.

Quando o usuário clicar em alguma das variáveis da rede, ela será selecionada. A Figura 4.4 mostra a tela principal do software quando a variável com nome *Variável 2* está selecionada. Quando isto acontece, o painel de propriedades é alterado, permitindo que o usuário altere o nome, estados e tabela de probabilidades da variável. Ao alterar o nome, também é feita a validação de nome vazio ou duplicado, assim como na inclusão na variável.

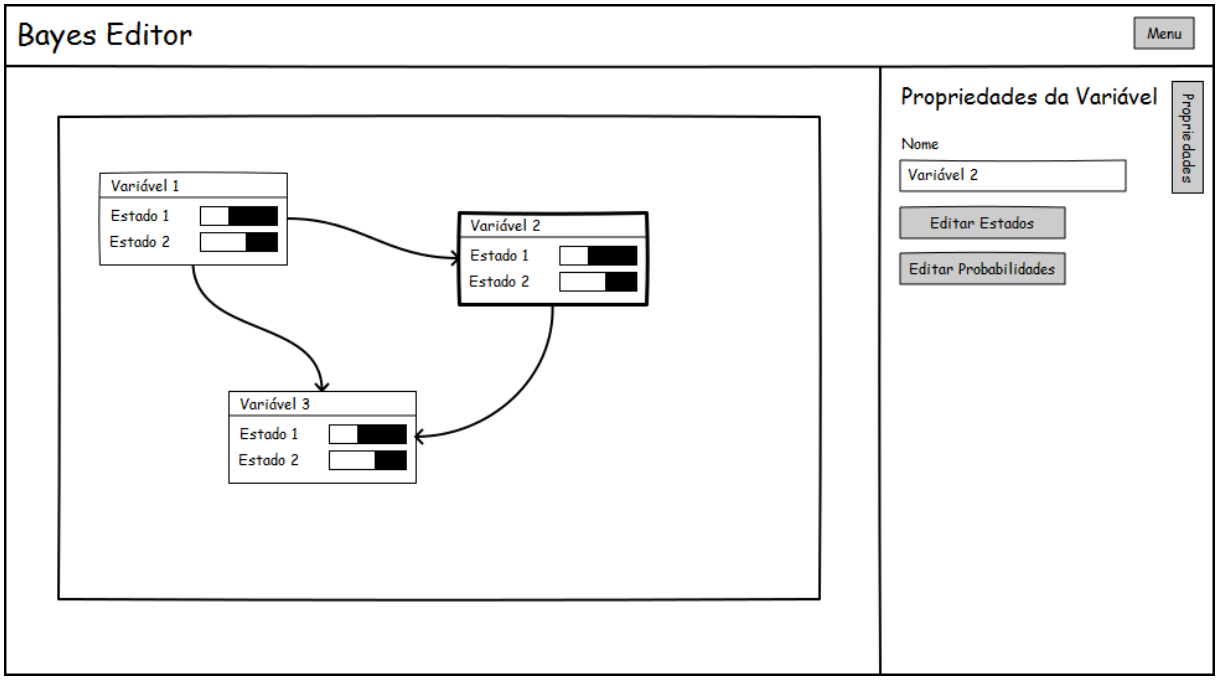


Figura 4.4 – *Wireframe* da tela principal com variável selecionada
Fonte: Construída pelo autor

Além de usar as opções do painel de propriedades, o usuário pode clicar com o botão direito do mouse em cima de uma variável. Quando isso for feito, será exibido um menu de contexto com as opções adicionar ligação, editar estados, editar probabilidades e remover variável. As opções de editar estados e editar probabilidades são as mesmas disponíveis no

painel de propriedades. No painel de propriedades essas opções tem uma visibilidade melhor, porém no menu de contexto elas permitem que o usuário faça as operações com maior agilidade.

Ao selecionar a opção adicionar ligação, o usuário deverá ligar a variável selecionada à outra. Com isso, será estabelecida uma ligação na rede bayesiana entre elas. Essa ligação será exibida na tela pelas setas. Antes de estabelecer a ligação, o editor valida que com a nova ligação não sejam introduzidos ciclos no grafo.

Ao selecionar a opção para editar estados, a tela mostrada na Figura 4.5 será exibida. Nesta tela, o usuário poderá alterar os estados da variável da mesma forma em que eles foram definidos na tela para adição de variáveis, com as mesmas validações.

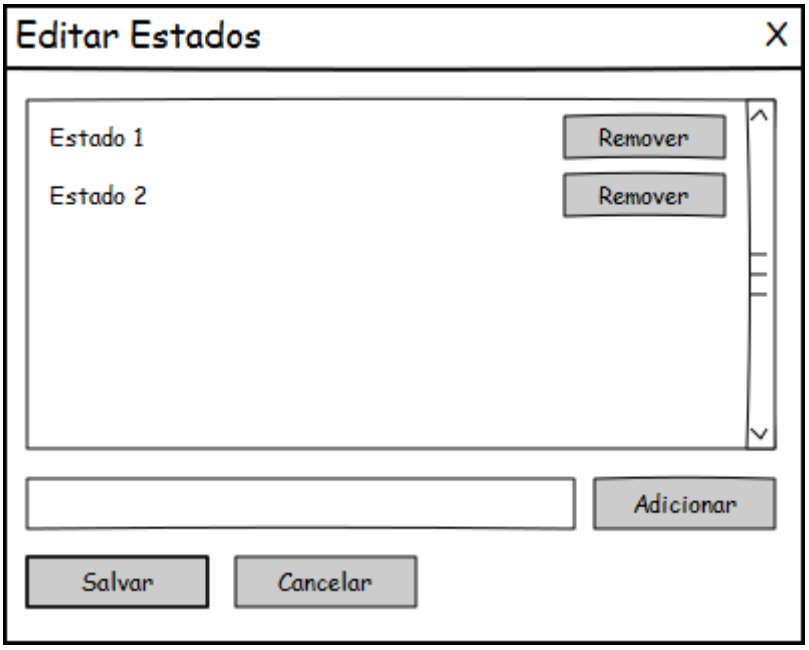


Figura 4.5 – *Wireframe* da tela de edição de estados
Fonte: Construída pelo autor

Ao selecionar a opção de editar probabilidades, a tela representada pela Figura 4.6 é exibida. No lado esquerdo da tabela, são exibidas as combinações de estados de das variáveis que são pais da que está em edição. No lado direito, os valores das probabilidades de cada um dos estados podem ser informados. Se a variável não possuir pais, a parte da esquerda é omitida, e apenas os valores para os estados são informados.

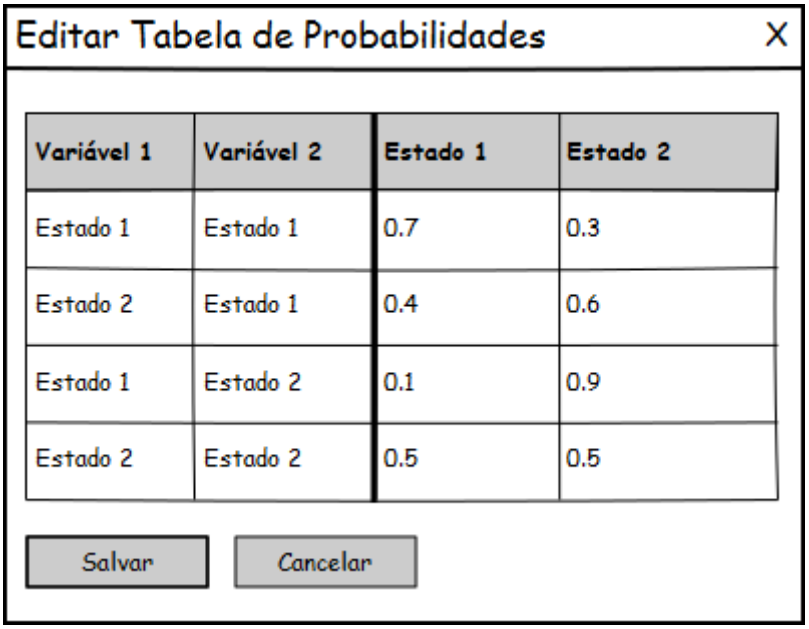


Figura 4.6 – *Wireframe* da tela de edição de probabilidades
Fonte: Construída pelo autor

Ao clicar em salvar, é feita a validação de que a probabilidade definida por cada linha deve somar 1. Caso alguma não passe nessa validação, o usuário é notificado e precisa corrigir antes de salvar. Ao salvar com sucesso, os valores da inferência são atualizados na tela principal.

No menu de contexto, se a opção de remover variável for escolhida, a variável é removida da estrutura da rede. Com estes *wireframes*, a estrutura e *layout* do editor de redes bayesianas foi especificado. Além disso, os fluxos de trabalho necessários ao usuário do editor foram descritos neste subcapítulo do trabalho.

4.2 Escolhas tecnológicas

O software foi desenvolvido utilizando tecnologias da plataforma web, como JavaScript, HTML e CSS. Esta decisão foi tomada por, principalmente, dois fatores. O primeiro é o fato de que desta forma o editor funcionará em múltiplas plataformas. O segundo é que desta forma o usuário não precisa instalar nada em seu computador local, o que facilita o acesso de qualquer dispositivo.

O editor será desenvolvido em dois projetos separados: uma biblioteca de inferência e a interface gráfica do editor. Desta forma, a biblioteca de inferência pode ser utilizada nos mais diversos projetos no futuro. Alguns exemplos de projetos seriam *web services*, outras interfaces gráficas para criação de redes, simuladores aplicados ao ensino que utilizem redes bayesianas, entre outros.

Apesar da utilização da plataforma web para o desenvolvimento do editor, todo o processamento da edição e inferência da rede bayesiana será processado no cliente via JavaScript. O único tráfego que ocorrerá na rede será o *download* inicial dos arquivos. Após isso, todo o processamento será feito no cliente para que se obtenha um tempo de resposta satisfatório.

O desenvolvimento do trabalho utiliza a especificação ECMAScript 2016 da linguagem JavaScript. Para que mesmo alguns navegadores não atualizados não tenham problemas em interpretar as funcionalidades da linguagem, foi utilizado o Babel, que compila o código ECMAScript 2016 para a versão ECMAScript 5, que está disponível em grande parte dos navegadores. Além disso, foi utilizado o Flow, que é uma ferramenta que pode ser utilizada para fazer verificação estática de tipos em um projeto JavaScript.

4.3 Biblioteca de inferência

Na biblioteca de inferência, a rede bayesiana é representada por um objeto que pode ser descrito em formato JSON (*JavaScript Object Notation*). Uma rede de exemplo pode ser observada na Figura 4.7 e sua representação em formato JSON na Figura 4.8. O objeto da rede possui uma propriedade para cada variável, e o nome da propriedade é o identificador da respectiva variável. Cada variável possui as seguintes propriedades:

- **id:** identificador da variável;
- **states:** lista de estados possíveis para a variável;
- **parents:** lista de identificadores das variáveis que são pais da variável em questão;
- **cpt:** tabela de probabilidades condicionais associada à variável.

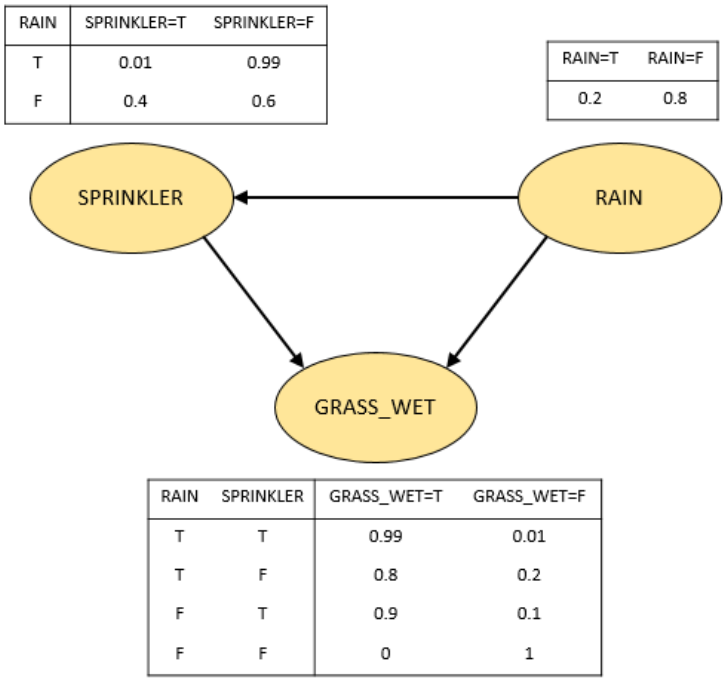


Figura 4.7 – Rede bayesiana de exemplo
Fonte: Construída pelo autor

```
{
  "RAIN": {
    "id": "RAIN",
    "states": [ "T", "F" ],
    "parents": [],
    "cpt": { "T": 0.2, "F": 0.8 }
  },
  "SPRINKLER": {
    "id": "SPRINKLER",
    "states": [ "T", "F" ],
    "parents": [ "RAIN" ],
    "cpt": [
      { "when": { "RAIN": "T" }, "then": { "T": 0.01, "F": 0.99 } },
      { "when": { "RAIN": "F" }, "then": { "T": 0.4, "F": 0.6 } }
    ]
  },
  "GRASS_WET": {
    "id": "GRASS_WET",
    "states": [ "T", "F" ],
    "parents": [ "RAIN", "SPRINKLER" ],
    "cpt": [
      { "when": { "RAIN": "T", "SPRINKLER": "T" }, "then": { "T": 0.99, "F": 0.01 } },
      { "when": { "RAIN": "T", "SPRINKLER": "F" }, "then": { "T": 0.8, "F": 0.2 } },
      { "when": { "RAIN": "F", "SPRINKLER": "T" }, "then": { "T": 0.9, "F": 0.1 } },
      { "when": { "RAIN": "F", "SPRINKLER": "F" }, "then": { "T": 0, "F": 1 } }
    ]
  }
}
```

Figura 4.8 – Rede bayesiana representada em JSON
Fonte: Construída pelo autor

Para as variáveis que não possuem nenhum nodo pai, a CPT é representada por um objeto no qual cada propriedade representa a probabilidade de um de seus estados, como pode ser observado na variável *RAIN* no exemplo da Figura 4.8. Quando a variável possuir pais, a

CPT é representada por uma lista de objetos com as propriedades *when* e *then*. A propriedade *when* é representada por um objeto descrevendo uma das combinações possíveis dos estados de cada um dos nodos pais da variável. A propriedade *then* representa as probabilidades para cada um dos estados da variável, no mesmo formato da CPT quando a variável não possui pais. Desta forma, essa CPT pode ser entendida com uma lista de objetos que descrevem as probabilidades (objeto *then*) para uma condição (objeto *when*). Essa lista de objetos deve possuir todas as combinações possíveis de estados dos pais da variável, como pode ser observado nas variáveis *SPRINKLER* e *GRASS_WET* no exemplo da Figura 4.8.

Durante o trabalho, foram realizadas duas implementações de algoritmos diferentes para fazer a inferência na rede. Ambos utilizam a mesma estrutura para a representação da rede bayesiana apresentada anteriormente. A seguir, cada um deles será descrito.

4.3.1 Inferência por enumeração

A inferência por enumeração é o método mais simples. Porém, considerando apenas variáveis booleanas, o algoritmo tem complexidade $O(2^n)$ (RUSSEL; NORVIG, 1995). Para realizar a inferência na rede utilizando este algoritmo, o primeiro passo é gerar uma lista com todas as combinações de estados possíveis, como pode ser visto na Figura 4.9.

```
[
  { "RAIN": "T", "SPRINKLER": "T", "GRASS_WET": "T" },
  { "RAIN": "T", "SPRINKLER": "T", "GRASS_WET": "F" },
  { "RAIN": "T", "SPRINKLER": "F", "GRASS_WET": "T" },
  { "RAIN": "T", "SPRINKLER": "F", "GRASS_WET": "F" },
  { "RAIN": "F", "SPRINKLER": "T", "GRASS_WET": "T" },
  { "RAIN": "F", "SPRINKLER": "T", "GRASS_WET": "F" },
  { "RAIN": "F", "SPRINKLER": "F", "GRASS_WET": "T" },
  { "RAIN": "F", "SPRINKLER": "F", "GRASS_WET": "F" }
]
```

Figura 4.9 – Combinações de estados da rede bayesiana
Fonte: Construída pelo autor

A função que gera esta lista de estados é mostrada na Figura 4.10. Conforme mostrado na linha 1, ela recebe uma rede bayesiana e retorna uma lista de combinações. Toda a geração de combinações é feita pela função *makeCombinations*, chamada na linha 4. Após elas terem sido geradas, são retornadas na linha 6.

```
1   function buildCombinations(network: Network): Combinations[] {
2     const combinations: Combinations[] = [];
3
4     makeCombinations(Object.keys(network));
5
6     return combinations;
7
8     function makeCombinations(nodes: string[], acc: Combinations = {}): void {
9       if (nodes.length === 0) {
10        combinations.push(acc);
11        return;
12      }
13
14      const [ node: string, ...rest: string[] ] = nodes;
15      const states: string[] = network[node].states;
16
17      for (let i = 0; i < states.length; i++) {
18        const state: string = states[i];
19
20        makeCombinations(rest, {
21          ...acc,
22          [node]: state
23        });
24      }
25    }
26  }
```

Figura 4.10 – Algoritmo para gerar combinações de estados da rede bayesiana
Fonte: Construída pelo autor

A função *makeCombinations* recebe uma lista com os identificadores de cada variável da rede e um objetivo onde uma das possíveis combinações vai sendo montada, inicialmente vazio. Na linha 9, é verificado se não há mais variáveis a serem iteradas e, se este for o caso, quer dizer que chegou ao fim de uma das possíveis combinações. Neste caso, ela é adicionada na lista de combinações e a função encerra.

Caso haja mais variáveis para combinar, a primeira da lista é extraída na variável *node*, e o restante das variáveis é armazenada na variável *rest*, conforme mostrado na linha 14. Após isso, todos os estados da variável *node* são iterados, e a função é chamada recursivamente para cada um, passando apenas a lista de variáveis restante como primeiro parâmetro e adicionando na variável *acc* uma das combinações de estado feitas com a variável atual. Com isso, todas as combinações serão geradas e um objeto como o da Figura 4.9 será obtido.

Feito isso, o próximo passo é filtrar apenas as combinações que interessam para a inferência em questão. Por exemplo, para descobrir a probabilidade de que a variável *GRASS_WET* esteja com o estado *T*, é preciso listar apenas as combinações em que o valor da variável *GRASS_WET* seja *T*, o que resulta na lista mostrada na Figura 4.11.

```
[
  { "RAIN": "T", "SPRINKLER": "T", "GRASS_WET": "T" },
  { "RAIN": "T", "SPRINKLER": "F", "GRASS_WET": "T" },
  { "RAIN": "F", "SPRINKLER": "T", "GRASS_WET": "T" },
  { "RAIN": "F", "SPRINKLER": "F", "GRASS_WET": "T" }
]
```

Figura 4.11 – Combinações de estados da rede bayesiana após filtragem
Fonte: Construída pelo autor

A função que realiza esta tarefa é mostrada na Figura 4.12. Conforme mostrado na linha 1, ela recebe a lista de combinações e uma combinação de todos para filtrar. Ao final, uma nova lista de combinações filtrada é retornada. Seguindo o exemplo de filtrar apenas a variável *GRASS_WET* como *T*, a variável *nodesToFilter* teria apenas uma propriedade *GRASS_WET* com o valor *T*. Caso a inferência fosse para encontrar a probabilidade de que várias variáveis aconteçam simultaneamente, todas elas seriam adicionadas na variável *nodesToFilter*.

```
1  function filterCombinations(combinations: Combinations[], nodesToFilter: Combinations): Combinations[] {
2    const idsToFilter = Object.keys(nodesToFilter);
3
4    return combinations.filter(row => {
5      for (let i = 0; i < idsToFilter.length; i++) {
6        const idToFilter = idsToFilter[i];
7
8        if (row[idToFilter] !== nodesToFilter[idToFilter]) {
9          return false;
10       }
11     }
12
13     return true;
14   });
15 }
```

Figura 4.12 – Algoritmo para filtrar estados nas combinações de estados da rede
Fonte: Construída pelo autor

Para realizar a filtragem, para cada linha de combinações (linha 4), todas as variáveis adicionadas na variável *nodesToFilter* são percorridas (linha 5) e, caso a combinação corrente possua uma delas com estado diferente do que o informado em *nodesToFilter*, ela não é adicionada no resultado, conforme mostrado na condição das linhas 8 a 10. Caso contrário, ela é mantida nas combinações. Com isso, um objeto semelhante ao da Figura 4.11 é obtido.

Após filtrar a lista de combinações, é preciso calcular o resultado da inferência. Para isso, é preciso calcular, para cada linha, o resultado da multiplicação das probabilidades de cada uma das variáveis nos estados presentes da linha. O valor das probabilidades é obtido na CPT de cada variável. Depois de realizar este cálculo, é preciso somar os resultados de cada linha e o valor resultante é o resultado da inferência. Por exemplo, o cálculo da terceira linha da Figura 4.3 ficaria da seguinte forma:

$$p(\neg rain) * p(sprinkler|\neg rain) * p(grass\_wet|\neg rain, sprinkler) \quad (4.1)$$

Substituindo cada probabilidade pelo valor obtido na CPT da variável, obtêm-se:

$$0{,}8 * 0{,}4 * 0{,}9 = 0{,}288 \quad (4.2)$$

Seguindo o mesmo processo para todas as linhas, obtêm-se o seguinte somatório:

$$0{,}00198 + 0{,}1584 + 0{,}288 + 0 = 0{,}44838 \quad (4.3)$$

Ou seja, a probabilidade de que a variável *GRASS_WET* seja *T* é de 44,838%. Para realizar este cálculo, foi utilizado o código mostrado na Figura 4.13. Essa função recebe o objeto da rede bayesiana utilizada no cálculo e a lista de combinações já filtrada, e retorna a probabilidade de que elas aconteçam, conforme mostrado nas linhas 1 a 4.

Cada uma das combinações é percorrida (linha 7), e para cada uma delas a lista de variáveis contidas nela é percorrida (linha 13). Caso a variável não possua pais, o valor da probabilidade do estado da combinação corrente é obtido acessando a propriedade *cpt* diretamente (linhas 18 e 19). Caso ela possua pais, a linha da *cpt* em que a variável *when* respeite os estados da combinação atual é encontrada e o valor da probabilidade é obtido da propriedade *then* da mesma linha (linhas 21 a 34). Todos os valores obtidos para uma linha de combinação são multiplicados e o resultado armazenado na variável *rowProduct* e, ao final de cada combinação, o valor é armazenado na lista dos produtos (linha 38). Ao final, todos os produtos armazenados na variável *rowsProducts* são somados (linhas 41 a 45) e o valor resultante é retornado. Este valor é o resultado da inferência na rede.

```
function calculateProbabilities(
  network: Network,
  combinations: Combinations[]
): number {
  const rowsProducts: number[] = [];

  for (let i = 0; i < combinations.length; i++) {
    let rowProduct = 1;

    const row = combinations[i];
    const ids = Object.keys(row);

    for (let j = 0; j < ids.length; j++) {
      const nodeId = ids[j];
      const node = network[nodeId];
      const cpt = (node.cpt : any);

      if (node.parents.length === 0) {
        rowProduct *= cpt[row[nodeId]];
      } else {
        const when = {};

        for (let k = 0; k < node.parents.length; k++) {
          const parent = node.parents[k];
          when[parent] = row[parent];
        }

        for (let k = 0; k < cpt.length; k++) {
          const cptRow = cpt[k];
          if (equal(cptRow.when, when)) {
            rowProduct *= cptRow.then[row[nodeId]];
            break;
          }
        }
      }
    }

    rowsProducts.push(rowProduct);
  }

  let probability = 0;

  for (let i = 0; i < rowsProducts.length; i++) {
    probability += rowsProducts[i];
  }

  return probability;
}
```

Figura 4.13 – Algoritmo para calcular a probabilidade de uma lista de combinações
Fonte: Construída pelo autor

O algoritmo apresentado mostra como realizar inferência sem evidências informadas. Para inferências que possuam evidências, a propriedade da probabilidade condicional apresentada no Capítulo 1 é utilizada:

$$p(V_i|V_j) = \frac{p(V_i, V_j)}{p(V_j)} \tag{4.4}$$

Desta forma, o algoritmo é executado duas vezes, uma para $p(V_i, V_j)$ e outra para $p(V_j)$. Ao final, é feita a divisão definida pela propriedade para obter o resultado da inferência com evidências.

4.3.2 Inferência por eliminação de variáveis

No algoritmo de inferência por enumeração, ao calcular a probabilidade das combinações, vários cálculos são repetidos. Para evitar este problema, técnicas de programação dinâmica podem ser utilizadas. Uma das formas de resolver este problema é utilizando o algoritmo de eliminação de variáveis (RUSSEL; NORVIG, 1995).

O primeiro passo desse algoritmo é construir fatores com base na tabela de probabilidades de cada variável. Inicialmente, é definido um fator para cada variável. Cada fator representa as probabilidades para determinadas combinações de variáveis. Por exemplo, utilizando a rede de exemplo da Figura 4.7, o fator construído a partir da tabela de probabilidades da variável SPRINKLER teria o formato apresentado na Figura 4.14, e sua representação em JSON é mostrada na Figura 4.15.

RAIN	SPRINKLER	Probabilidade
F	F	0.6
F	T	0.4
T	F	0.99
T	T	0.01

Figura 4.14 – Fator de exemplo para eliminação de variáveis
Fonte: Construída pelo autor

```
[
  { "states": { "RAIN": "F", "SPRINKLER": "F" }, "value": 0.6 },
  { "states": { "RAIN": "F", "SPRINKLER": "T" }, "value": 0.4 },
  { "states": { "RAIN": "T", "SPRINKLER": "F" }, "value": 0.99 },
  { "states": { "RAIN": "T", "SPRINKLER": "T" }, "value": 0.01 }
]
```

Figura 4.15 – Fator de exemplo para eliminação de variáveis em JSON
Fonte: Construída pelo autor

O código utilizado para a construção destes fatores é mostrado na Figura 4.16. Na linha 1, é possível observar que a função recebe como parâmetro um nodo da rede e uma lista de evidências (que é opcional), e tem como retorno o fator construído.

Na linha 5, é verificado se o nodo informado possui pais. Caso ele não possua, apenas os seus próprios estados devem ser levados em conta na hora de construir o fator. Nesse caso, o código das linhas 6 a 13 adiciona uma linha no fator final para cada estado do nodo, buscando o valor da sua tabela de probabilidades.

Caso a tabela possua pais, é preciso percorrer todas as linhas da sua tabela de probabilidades. Cada linha dessa tabela possui uma possível combinação de pais, portanto, para cada uma dessas linhas, os estados do nodo também são percorridos e adicionados à combinação dos estados dos pais, formando uma nova combinação. Essa combinação é adicionada no fator final, com o valor obtido da linha corrente da tabela de probabilidades. O código que faz isto pode ser observado nas linhas 15 a 24.

Caso tenham sido passadas evidências para a função, é preciso testar, para cada linha dos fatores finais, se ela possui alguma variável com estado diferente do que foi informado nas evidências. Se este for o caso, é preciso que a linha seja removida dos fatores finais. O código responsável por essa tarefa pode ser observado nas linhas 27 a 41.

```
function buildFactor(node: Node, giving: ?Combinations): Factor {
  const factor = [];
  const cpt = (node.cpt : any);

  if (node.parents.length === 0) {
    for (let i = 0; i < node.states.length; i++) {
      const state = node.states[i];

      factor.push({
        states: { [node.id]: state },
        value: cpt[state]
      });
    }
  } else {
    for (let i = 0; i < cpt.length; i++) {
      for (let j = 0; j < node.states.length; j++) {
        const state = node.states[j];

        factor.push({
          states: { ...cpt[i].when, [node.id]: state },
          value: cpt[i].then[state]
        });
      }
    }
  }

  if (giving) {
    const givingIds = Object.keys(giving);

    for (let i = factor.length - 1; i >= 0; i--) {
      for (let j = 0; j < givingIds.length; j++) {
        const givingId = givingIds[j];

        if (factor[i].states[givingId] &&
            factor[i].states[givingId] !== giving[givingId]) {
          factor.splice(i, 1);
          break;
        }
      }
    }
  }

  return factor;
}
```

Figura 4.16 – Código para a construção de fatores iniciais
Fonte: Construída pelo autor

Feita essa construção inicial de fatores, o algoritmo precisa eliminar todas as variáveis que não estão envolvidas na inferência em questão. Para que isso seja feito, é necessário definir duas operações que serão realizadas sobre os fatores. Uma delas é a operação de juntar dois fatores e a outra é a operação de eliminar uma variável de um fator.

A Figura 4.17 mostra um exemplo da operação de juntar fatores. Neste exemplo, a operação tem como entrada os fatores iniciais definidos a partir da tabela de probabilidades das variáveis *RAIN* e *SPRINKLER*. A saída é um novo fator com todas as combinações de estados das variáveis presentes nos fatores de entrada, multiplicando a probabilidade encontrada em cada um deles para obter a probabilidade do novo fator.

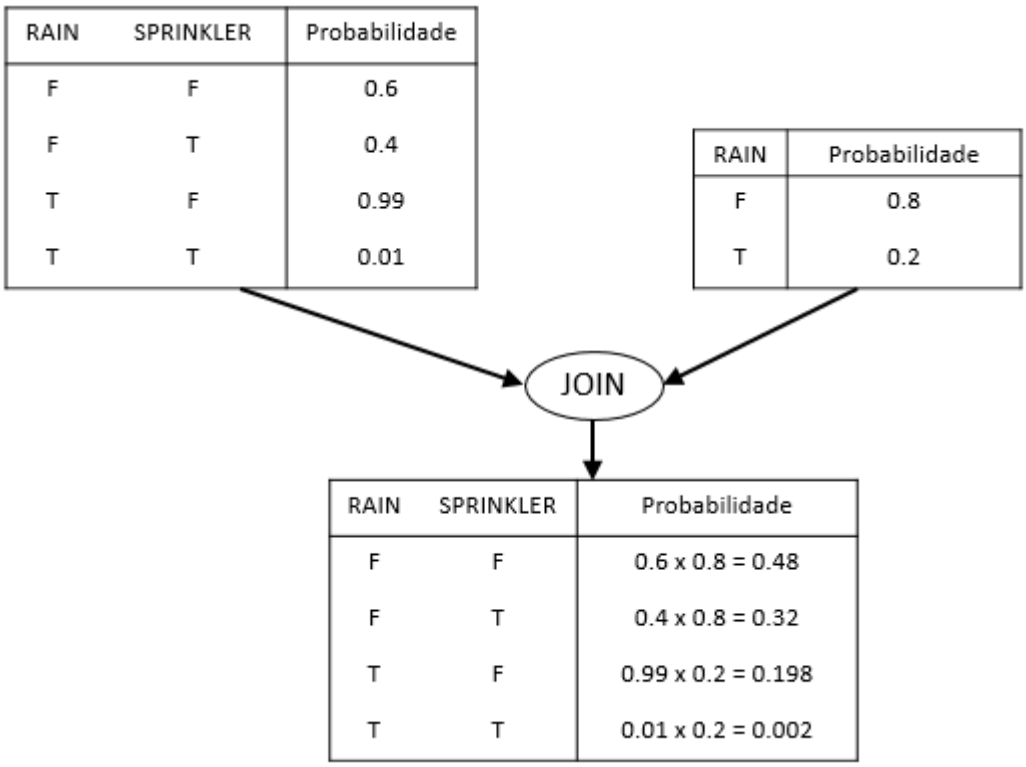


Figura 4.17 – Exemplo de operação para juntar fatores
Fonte: Construída pelo autor

O código para executar esta operação pode ser observado na Figura 4.18. A função recebe como parâmetro dois fatores e retorna um novo fator (linha 1). A primeira etapa desta função consiste em gerar um novo fator com todas as combinações de variáveis contidas nos dois fatores de entrada, sem calcular a probabilidade dele (linhas 4 a 19).

A segunda etapa consiste em procurar, para cada linha gerada no novo fator, as respectivas probabilidades nos fatores de entradas. Com as funcionalidades encontradas, basta multiplicá-las e atualizar a probabilidade da linha no novo fator (linhas 21 a 40).

```
function joinFactors(f1: Factor, f2: Factor): Factor {
  const newFactor = [];

  for (let i = 0; i < f1.length; i++) {
    for (let j = 0; j < f2.length; j++) {
      const states = { ...f1[i].states, ...f2[j].states };
      const nodeIds = Object.keys(states);

      const alreadyExists = newFactor.some(x => {
        return nodeIds.every(
          nodeId => x.states[nodeId] === states[nodeId]
        );
      });

      if (!alreadyExists) {
        newFactor.push({ states, value: 0 });
      }
    }
  }

  const nodeIdsF1 = Object.keys(f1[0].states);
  const nodeIdsF2 = Object.keys(f2[0].states);

  for (let i = 0; i < newFactor.length; i++) {
    const rowNewFactor = newFactor[i];

    const rowF1 = f1.find(x => {
      return nodeIdsF1.every(nodeId =>
        x.states[nodeId] === rowNewFactor.states[nodeId]
      );
    });

    const rowF2 = f2.find(x => {
      return nodeIdsF2.every(nodeId =>
        x.states[nodeId] === rowNewFactor.states[nodeId]
      );
    });

    rowNewFactor.value = rowF1.value * rowF2.value;
  }

  return newFactor;
}
```

Figura 4.18 – Código para a operação de juntar fatores
Fonte: Construída pelo autor

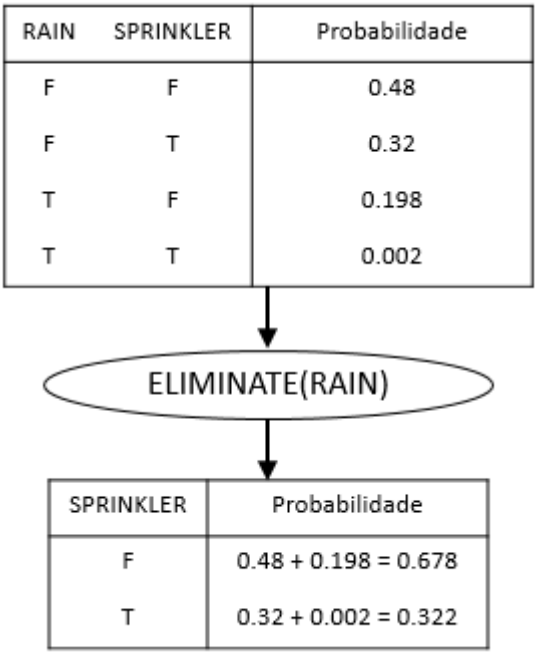


Figura 4.19 – Exemplo de operação para eliminar variável
Fonte: Construída pelo autor

Um exemplo para a operação de eliminar uma variável de um fator pode ser observado na Figura 4.19. Ela recebe como parâmetro um fator existente e uma variável a ser eliminada. No exemplo, a variável *RAIN* foi eliminada. A operação consiste em remover a coluna da variável a ser eliminada e, para as linhas que possuírem a mesma combinação de estados, somar as probabilidades.

O código responsável por realizar esta operação pode ser observado na Figura 4.20. Na assinatura da função (linha 1) é possível observar que ela recebe como parâmetro um fator e uma variável a ser eliminada e retorna um novo fator. Cada uma das linhas do fator é percorrida (linha 4) e, para cada uma, uma cópia da combinação de estados é criada, removendo a variável a ser eliminada (linhas 5 a 7). Após isso, é procurado no novo fator construído se já existe uma linha para a nova combinação de variáveis (linhas 9 a 13). Se não existir, uma nova linha é incluída no novo fator, atribuindo à probabilidade o valor da linha do fator original (linhas 16 a 19). Caso já exista uma linha, o valor da sua probabilidade é somado com o da linha do fator original (linha 21).

```
function eliminateVariable(factor: Factor, variable: string): Factor {
  const newFactor = [];

  for (let i = 0; i < factor.length; i++) {
    const states = { ...factor[i].states };

    delete states[variable];

    const nodeIds = Object.keys(states);

    const existingRow = newFactor.find(x => {
      return nodeIds.every(nodeId => x.states[nodeId] === states[nodeId]);
    });

    if (existingRow === undefined) {
      newFactor.push({
        states,
        value: factor[i].value
      });
    } else {
      existingRow.value += factor[i].value;
    }
  }

  return newFactor;
}
```

Figura 4.20 – Código para a operação de eliminar variável
Fonte: Construída pelo autor

Definidas as operações de juntar fatores e eliminar variável, é possível utilizá-las nos fatores construídos a partir das tabelas de probabilidades de uma rede bayesiana para realizar a inferência. Os fatores iniciais para a rede bayesiana mostrada na Figura 4.7 são apresentados na Figura 4.21. Para realizar a inferência $p(GRASS\_WET = T)$, é preciso eliminar todas as variáveis da rede exceto *GRASS_WET*.

Para eliminar a variável *SPRINKLER*, primeiro é preciso juntar todos os fatores que possuem ela e depois eliminá-la do fator resultante, conforme mostrado na Figura 4.22. Feita essa operação, sobram dois fatores para continuar a inferência, conforme mostrados na Figura 4.23. A outra variável que resta eliminar é a variável *RAIN*. Para isso, o mesmo processo é feito com os dois fatores restantes, conforme mostrado pela Figura 4.24.

Feito isso, como não restam mais variáveis para eliminar, basta procurar no fator restante a linha em que a variável *GRASS_WET* seja *T*. Desta forma, o mesmo valor de 44,838% que foi encontrado no algoritmo por enumeração é obtido. Os algoritmos completos para a inferência tanto por enumeração quanto por eliminação de variáveis podem ser encontrados, respectivamente, nos apêndices A e B deste trabalho.

RAIN	Probabilidade
F	0.8
T	0.2

RAIN	SPRINKLER	Probabilidade
F	F	0.6
F	T	0.4
T	F	0.99
T	T	0.01

RAIN	SPRINKLER	GRASS_WET	Probabilidade
F	F	F	1
F	F	T	0
F	T	F	0.1
F	T	T	0.9
T	F	F	0.2
T	F	T	0.8
T	T	F	0.01
T	T	T	0.99

Figura 4.21 – Fatores iniciais para exemplo de eliminação de variáveis
Fonte: Construída pelo autor

RAIN	SPRINKLER	GRASS_WET	Probabilidade
F	F	F	1 x 0.6 = 0.6
F	F	T	0 x 0.6 = 0
F	T	F	0.1 x 0.4 = 0.04
F	T	T	0.9 x 0.4 = 0.36
T	F	F	0.2 x 0.99 = 0.198
T	F	T	0.8 x 0.99 = 0.792
T	T	F	0.01 x 0.01 = 0.0001
T	T	T	0.99 x 0.01 = 0.0099

RAIN	GRASS_WET	Probabilidade
F	F	0.6 + 0.04 = 0.64
F	T	0 + 0.36 = 0.36
T	F	0.198 + 0.0001 = 0.1981
T	T	0.792 + 0.0099 = 0.8019

Figura 4.22 – Passos para eliminar a variável SPRINKLER
Fonte: Construída pelo autor

RAIN	Probabilidade
F	0.8
T	0.2

RAIN	GRASS_WET	Probabilidade
F	F	0.64
F	T	0.36
T	F	0.1981
T	T	0.8019

Figura 4.23 – Fatores resultantes após eliminar variável SPRINKLER
Fonte: Construída pelo autor

RAIN	GRASS_WET	Probabilidade
F	F	0.64 x 0.8 = 0.512
F	T	0.36 x 0.8 = 0.288
T	F	0.1981 x 0.2 = 0.03962
T	T	0.8019 x 0.2 = 0.16038

GRASS_WET	Probabilidade
F	0.512 + 0.03962 = 0.55162
T	0.288 + 0.16038 = 0.44838

Figura 4.24 – Fator resultante após eliminar variável RAIN
Fonte: Construída pelo autor

4.3.3 Comparação de performance entre os algoritmos

Para realizar inferências únicas em redes extremamente pequenas, ambos os algoritmos se mostraram satisfatórios. Porém, para redes relativamente grandes, apenas o algoritmo de eliminação de variáveis mostrou-se viável. Para visualizar a diferença entre eles, foram feitos testes com redes de n variáveis, todas de domínio booleano e ligadas em série, ou seja, a variável i tem como pai a variável $i - 1$. A Figura 4.25 apresenta a estrutura dessas redes.

Figura 4.25 – Redes bayesianas usadas em testes de performance
Fonte: Construída pelo autor

Os testes foram executados em um computador com processador Intel Core i5-3230M 2.60GHz, memória RAM de 8GB e 1600 MHz, SSD Samsung EVO 850 de 250 GB e sistema operacional Windows 10 de 64 bits. Para cada cenário, foram feitas 3 inferências. A primeira é apenas para descobrir a probabilidade de que a variável $v3$ seja T. A segunda é para descobrir a probabilidade de que as variáveis $v1$, $v2$ e $v3$ sejam T. A terceira é para descobrir a probabilidade de que $v3$ seja T dado como evidência que $p(v1 = T)$. Desta forma, é feita uma inferência à uma única variável, uma em múltiplas variáveis e uma com evidências.

Tabela 4.1 – Resultados de testes de performance entre os algoritmos

Tipo do algoritmo	**Quantidade de variáveis**	**p(v3=T)**	**p(v1=T, v2=T, v3=T)**	**p(v3=T | v1=T)**
Enumeração	5	5ms	1ms	2ms
	10	109ms	31ms	123ms
	15	4202ms	784ms	4905ms
	18	40213ms	7688ms	47509ms
Eliminação de Variáveis	5	3ms	2ms	1ms
	10	5ms	2ms	2ms
	15	8ms	3ms	5ms
	18	11ms	5ms	6ms
	50	28ms	14ms	11ms
	100	44ms	33ms	27ms
	150	68ms	53ms	52ms
	200	101ms	82ms	76ms

Fonte: Construída pelo autor.

Os resultados dos testes para cada um dos algoritmos podem ser observados na Tabela 4.1. Para cada cenário, o tempo decorrido para chegar ao resultado é mostrado em

milissegundos. Após 15 variáveis, o algoritmo por enumeração mostrou-se completamente inviável para uso no editor. Já o algoritmo de eliminação de variáveis, mesmo para número bem superiores, não teve uma queda grande de performance.

Além disso, existem diversos fatores que contribuem para aumentar o tempo levado até o algoritmo encontrar o resultado. Alguns deles são: quantidade de variáveis presentes na rede; quantidade de estados em cada variável; quantidade de ligações de uma variável até a outra. Portanto, uma rede que possuir menos variáveis, porém estados e ligações mais complexas, pode levar mais tempo para calcular do que uma rede com mais variáveis, porém menos estados e ligações.

4.4 Editor

A Figura 4.26 mostra uma captura de tela do editor de redes bayesianas desenvolvido. O software foi nomeado provisoriamente de Bayes Editor, pois a definição de marca e identidade visual não está no escopo deste trabalho. A rede que está sendo exibida é a mesma utilizada nos exemplos para os algoritmos de inferência, apresentada na Figura 4.7. Todo o fluxo das tarefas que o usuário pode executar já foram descritos no subcapítulo 4.1.

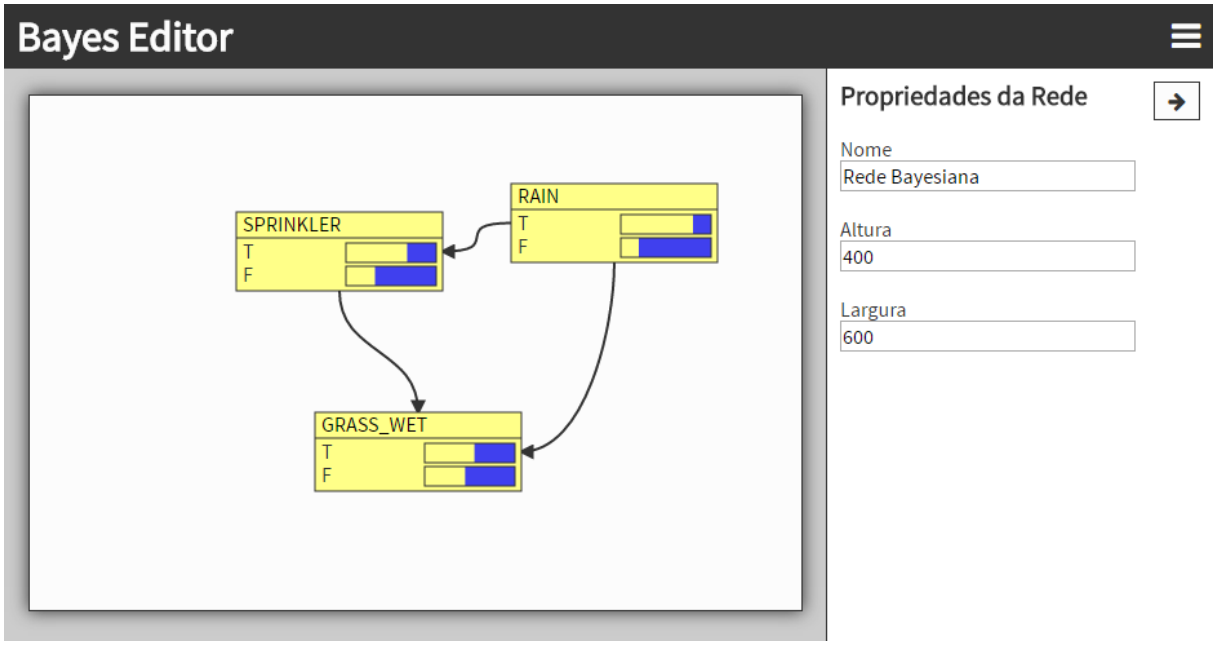


Figura 4.26 – Captura de tela do editor desenvolvido
Fonte: Construída pelo autor

Para a construção de toda a interface gráfica do software, foi utilizada a biblioteca React. Esta biblioteca serve para que a interface gráfica possa ser desenvolvida de forma declarativa, baseada em componentes. A Figura 4.27 mostra o componente definido para a

exibição dos botões do editor. Na linha 5, é definida uma função que recebe propriedades via parâmetro. Essas propriedades incluem o estilo, texto e também a função a ser executada quando o usuário clicar no botão. Com base nessas propriedades, uma *tag* HTML *button* é inserida na tela.

```
1   import React from 'react';
2   import classNames from 'classnames';
3   import styles from './styles.css';
4
5   const Button = props => (
6     <button
7       type="button"
8       className={classNames(styles.button, props.className, {
9         [styles.primary]: props.primary,
10      })}
11      style={props.style}
12      title={props.title}
13      onClick={props.onClick}
14    >
15      {props.children}
16    </button>
17  );
18
19  export default Button;
```

Figura 4.27 – Código do componente para botões do editor
Fonte: Construída pelo autor

Todos os outros componentes são definidos de forma semelhante, porém alguns com complexidade muito maior que os outros. Além disso, alguns componentes utilizam outros, por exemplo, o componente do painel de propriedades utiliza o componente dos botões internamente. Ao final do desenvolvimento, foram criados os seguintes componentes:

- **AddNodeModal:** tela para adicionar nova variável;
- **App:** componente raiz que engloba a aplicação inteira;
- **Button:** componente para todos os botões da aplicação;
- **Canvas:** área branca em que as variáveis da rede são exibidas;
- **ContextMenu:** menu de contexto que é exibido quando o usuário clica com o botão direito no canvas de edição;
- **EditCptModal:** tela para a edição da tabela de probabilidades de uma variável;

- **EditStatesList:** lista de estados, usada tanto na tela de edição de estados quanto na tela de adicionar variável;
- **EditStatesModal:** tela para a edição de estados de uma variável;
- **Header:** cabeçalho da aplicação, com título e acesso ao menu;
- **Node:** representa uma variável da rede;
- **PropertiesPanel:** painel de propriedade para editar a rede e variáveis.

O Quadro 4.1 mostra a comparativa de funcionalidades feita no Capítulo 3, porém agora inserindo o editor desenvolvido. É possível visualizar que o Bayes Editor tem um número de funcionalidades mais limitado em relação aos editores existentes. Porém, a decisão durante o desenvolvimento foi justamente criar um conjunto mínimo de funcionalidades que permitam aos usuários criar redes bayesianas, desde que as funcionalidades disponíveis possuam uma boa usabilidade.

Além disso, apesar de o editor desenvolvido não ter uma funcionalidade específica para impressão da estrutura da rede, como ele é uma aplicação web, pode ser utilizada a funcionalidade de impressão do navegador. Porém, uma funcionalidade específica para impressão poderia deixar a rede em um formato mais adequado para a impressão.

Tanto a biblioteca de inferência como o editor gráfico de redes bayesianas foram desenvolvidos com código aberto e podem ser encontrados, respectivamente, em: https://github.com/fhelwanger/bayesjs e https://github.com/fhelwanger/bayesjs-editor. O próximo capítulo apresentará o método e os resultados da validação do editor.

Quadro 4.1 – Comparativa de funcionalidades entre os editores analisados e o desenvolvido.

	Hugin	**Netica**	**UnBBayes**	**AMPLIA**	**Bayes Editor**
Possui nodos de estados discretos	Sim	Sim	Sim	Sim	Sim
Possui nodos de estados contínuos	Sim	Sim	Não	Não	Não
Suporte a diagramas de influência	Sim	Sim	Sim	Sim	Não
Impressão da estrutura da rede bayesiana	Sim	Sim	Não	Não	Não
Impressão das CPTs da rede bayesiana	Sim	Não	Não	Não	Não
Código fonte	Fechado	Fechado	Aberto	Fechado	Aberto
APIs	Disponíveis em diversas linguagens	Disponíveis em diversas linguagens	Disponível apenas na linguagem Java	Não disponível	Disponível apenas na linguagem JavaScript
Facilidade de edição das CPTs	Dificuldade de visualizar quais estados são referentes ao nodo em edição e quais são referentes aos estados dos nodos pai	Quais estados se referem aos nodos pais e quais se referem ao nodo atual de fácil visualização	Dificuldade de visualizar quais estados são referentes ao nodo em edição e quais são referentes aos estados dos nodos pai	Quais estados se referem aos nodos pais e quais se referem ao nodo atual de fácil visualização	Quais estados se referem aos nodos pais e quais se referem ao nodo atual de fácil visualização
Agilidade na edição das probabilidades durante a inferência	Modo de edição e execução separados, o que causa uma troca de contexto pelo usuário	Não existe separação entre o modo de edição e execução, o que torna a ferramenta bastante ágil para esta tarefa	Modo de edição e execução separados, o que causa uma troca de contexto pelo usuário	Modo de edição e execução separados, o que causa uma troca de contexto pelo usuário. Além disso, vários passos são necessários para realizar a inferência.	Não existe separação entre o modo de edição e execução, o que torna a ferramenta bastante ágil para esta tarefa

Fonte: Itens levantados pelo autor.

5 VALIDAÇÃO E RESULTADOS

Neste capítulo do trabalho, será apresentada a metodologia da validação realizada. Os conceitos de testes de usuários, já apresentados no Capítulo 3, foram os principais recursos utilizados nesta etapa. Após isso, serão relatados os resultados obtidos.

5.1 Metodologia

Conforme já descrito no Capítulo 3, o ideal é que sejam feitas várias rodadas de testes com usuários. Com isso, as oportunidades de melhorias encontradas em uma rodada são implementadas no software e uma nova rodada é realizada com os usuários, para verificar se os objetivos foram atingidos e se outras oportunidades de melhorias são encontradas. Porém, o escopo deste trabalho compreende apenas uma única rodada de testes. Além disso, outros editores existentes poderiam ser utilizados nos testes com usuários. Devido a estes motivos, a validação realizada neste trabalho é uma validação parcial.

Para a validação do trabalho, foram realizados testes com usuários. Para estes testes, foram utilizados ao total 9 usuários, sendo eles, 2 estudantes de biomedicina, 2 estudantes de enfermagem e 5 estudantes de áreas relacionadas à computação. Todos estes participantes são integrantes do grupo de pesquisa de Computação Aplicada da Universidade Feevale. Estes usuários foram convidados para uma oficina que envolvia a capacitação deles em três itens: conceitos de redes bayesianas, utilização do editor Hugin e utilização do editor Bayes Editor. A utilização do editor Hugin foi realizada para que seja possível fazer uma comparação relativa entre os resultados.

Foi utilizado como instrumento de avaliação dos resultados um questionário contendo 9 questões optativas sobre usabilidade, seguindo a escala Likert, e uma questão descritiva, solicitando críticas e sugestões gerais. Nas questões optativas, haviam opções de 1 a 5, sendo 1 o equivalente a discordo totalmente e o 5 o equivalente a concordo totalmente.

Para o convite dos usuários, foi enviado um e-mail a todos eles, solicitando a confirmação de presença em duas noites em uma sala da Universidade Feevale reservada para a oficina. A confirmação foi realizada por todos os participantes. O primeiro dia compreendeu a apresentação de conceitos de redes bayesianas, guiada pelos slides presentes no apêndice C, e a utilização do editor Hugin. No segundo dia foi realizada a utilização do Bayes Editor e aplicação do questionário. Na capacitação sobre conceitos de redes bayesianas, foram apresentados os seguintes itens:

- **Aplicações de redes bayesianas:** foi explicado que redes bayesianas são empregadas em problemas que envolvem incerteza e foram dados exemplos de áreas em que elas podem ser empregadas;
- **Estrutura de uma rede bayesiana:** foram explicados os componentes que constituem uma rede bayesiana e suas restrições, tanto a estrutura lógica da rede quanto as probabilidades associadas a cada variável;
- **Tipos de inferência:** diversos tipos de inferências possíveis em uma rede bayesiana foram apresentados, como, por exemplo, inferência em uma única variável, inferência em múltiplas variáveis e inferências com evidências;
- **Passos para construir uma rede bayesiana:** baseado nos passos para construção de redes bayesianas descritos por Lucas (2004), foi apresentada uma metodologia para a construção de redes bayesianas que compreende, sequencialmente: selecionar as variáveis relevantes; identificar o relacionamento entre as variáveis; definir as probabilidades de cada uma das variáveis e avaliar o resultado da rede.

Após a apresentação destes conceitos, foi montada, pelo autor deste trabalho, juntamente com os usuários, a rede da Figura 4.7 em ambos os editores. Após isso, foi solicitado que eles criassem nos editores a rede da Figura 5.1 sozinhos. Essa rede foi elaborada por alunos do curso de biomedicina, que representa o problema de verificar se uma pessoa é apta para realizar uma doação de sangue. Durante os testes, foi solicitado que os usuários falassem quais dificuldades encontraram durante a utilização dos editores, que foram observadas pelo autor deste trabalho.

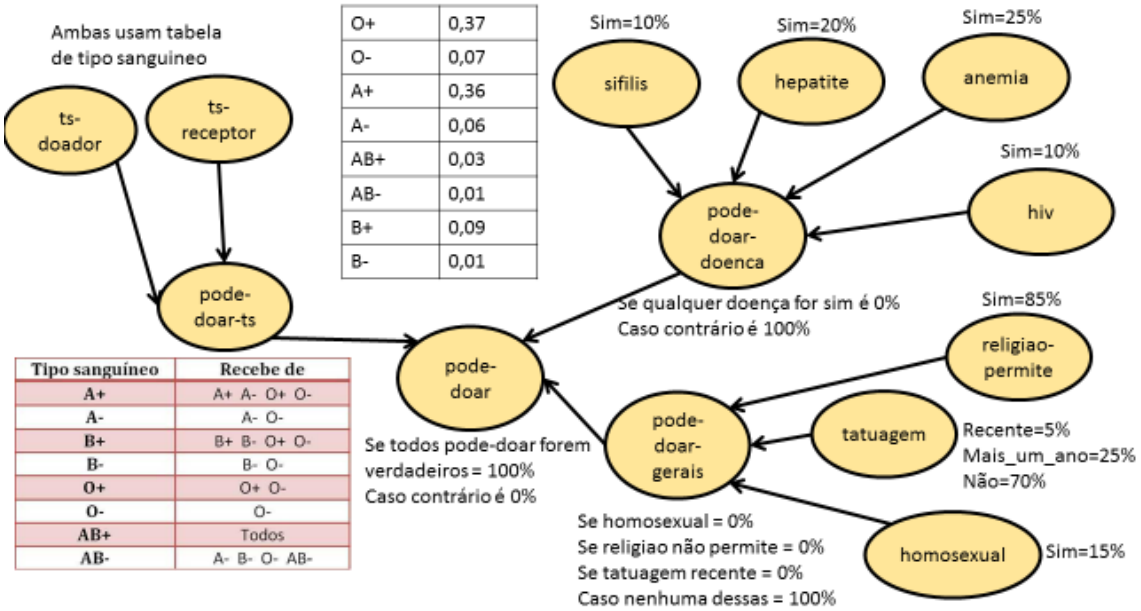


Figura 5.1 – Rede de doação de sangue
Fonte: Construída pelo autor

5.2 Resultados

A tabela 5.1 apresenta as médias de respostas para cada pergunta após os testes de usabilidade para os dois editores. O questionário completo pode ser encontrado no apêndice D deste trabalho.

Com base nas respostas da primeira questão, é possível observar que ambos os editores se mostraram satisfatórios em relação ao fornecimento de funcionalidades para a criação de redes bayesianas. As respostas da segunda questão já mostram que o fluxo de trabalho adotado no Bayes Editor se mostrou mais ágil, provavelmente por se basear na ideia de não possuir telas separadas para a montagem e a inferência na rede.

As respostas para as questões de número 3, 7 e 8 mostram que o fluxo de trabalho do Bayes Editor se mostrou mais intuitivo. Conforme relatado no Capítulo 3, pela análise do autor deste trabalho, a edição da tabela de probabilidades no Hugin era um tanto confusa. Nos testes com usuários esse problema mostrou-se bastante aparente e, em geral, as pessoas conseguiram compreender melhor a forma adotada no Bayes Editor para a disposição das tabelas de probabilidades. Além disso, a redução no número de telas para tornar o editor mais simples também contribui para uma interface gráfica mais simples.

Tabela 5.1 – Respostas do questionário dos testes de usabilidade

Número	**Questão**	**Hugin**	**Bayes Editor**
1	O editor fornece as funcionalidades necessárias para a criação e inferência em uma rede bayesiana	4,33	4,68
2	O editor permite que a criação e inferência em uma rede bayesiana seja realizada com rapidez	2,78	4,44
3	O editor possui um fluxo de trabalho que é claro e não gera confusões	2,33	4,67
4	Os textos e mensagens utilizados no editor são claros	3,22	4,44
5	O editor não apresenta erros durante a execução	4	4,89
6	O editor informa ao usuário quando dados inválidos são informados	3,78	4,78
7	A interface gráfica utilizada pelo editor é intuitiva	2,67	4,44
8	É fácil lembrar de como executar as operações no editor	3,33	4,89
9	O editor fornece os resultados com tempo de resposta satisfatório	4	4,44

Fonte: Construída pelo autor.

Na questão 4, os usuários acreditam que os textos e mensagens do Bayes Editor se mostraram mais claros do que os do Hugin. Por conta de o Bayes Editor possuir um conjunto mais limitado de funcionalidades, foi possível utilizar termos bastante simples para descrever os elementos da interface gráfica, já o Hugin apresenta termos bastante técnicos, como *Discrete change tool* e *Function tool*. Isto pode explicar a escolha dos usuários.

Em geral os usuários concordam que os editores não apresentaram erros durante a execução, conforme mostra a questão 5. Alguns usuários reclamaram que o Hugin não aceitava acentos ou espaços nos nomes das variáveis, isso pode ter sido percebido por alguns como erro

e explicaria a média um pouco inferior em relação ao Bayes Editor. Erros não tratados não foram observados durante a validação.

Conforme pode ser observado nas respostas da questão 9, os editores de maneira geral forneceram resultados em tempo satisfatório para as redes montadas. Porém, um dos usuários gerou algumas ligações mais complexas durante a montagem da rede no Bayes Editor e foi possível observar lentidões durante a execução do editor. Para que os resultados sejam mostrados em tempo real no editor, é preciso que seja feita uma inferência para cada estado de cada variável da rede a cada modificação. Devido a isto, mesmo o algoritmo de eliminação de variáveis levando alguns milissegundos para inferências isoladas, ao somar todas elas, em redes complexas fica perceptível a lentidão. Isso é um indício de que mesmo o algoritmo de eliminação de variáveis pode não ser adequado para a proposta do editor quando usado em redes maiores.

Além das respostas das questões optativas que indicam os índices de usabilidade dos editores, foram obtidas diversas sugestões de melhorias, tanto por meio do campo para sugestões e críticas gerais do questionário, quanto por meio de observações feitas pelo autor deste trabalho durante os testes. A maioria dessas sugestões são itens bastante simples, mas que aumentariam ainda mais a usabilidade do software.

Um dos itens que foi sugerido por boa parte dos usuários foi a questão de que o Bayes Editor poderia usar teclas de atalho para algumas operações ao invés do mouse. Ao adicionar uma variável, editar os estados ou editar as tabelas de probabilidades das redes, o usuário é obrigado a clicar no botão de confirmação da janela para concluir o processo. Foi sugerido que, ao pressionar a tecla ENTER, o processo também seja concluído. Também foi sugerido que, além de possuir um menu de contexto para apagar uma variável da rede, quando a variável estiver selecionada, seja possível excluí-la pressionando a tecla DELETE.

Outra sugestão foi que seja possível ligar as variáveis da rede sem a necessidade de acessar um menu de contexto. Alguns softwares existentes mostram pontos nas bordas dos objetos que podem ser ligados, e basta ligar estes pontos com o mouse para estabelecer a relação entre eles.

Também foram dadas sugestões para que a estrutura lógica da rede possa ter uma representação visual mais organizada. Umas delas é que, ao selecionar uma das variáveis, todas as setas que estão ligadas a ela fiquem realçadas com uma cor diferente. Desta forma, quando houverem ligações complexas, o usuário pode ter uma melhor visualização da estrutura da rede.

Além disso, foi sugerido que a posição das setas possa ser ajustada manualmente. Na versão do software utilizada na validação, o usuário só podia ajustar as posições das variáveis e as setas são ajustadas automaticamente. Porém, o ajuste automático pode nem sempre trazer a disposição das setas para o entendimento do usuário.

As telas de edição de estados e edição de probabilidades possuem como título, respectivamente, os textos “Editar Estados” e “Editar Tabelas de Probabilidade”. Porém, em nenhum lugar dessas telas é exibido o nome da variável em edição. Isso pode gerar confusão caso não lembrassem qual variável estava sendo editada. Foi notado que os usuários sentiram falta da exibição desta informação, que poderia ser incluída no título dessas telas.

Alguns usuários informaram que prefeririam que, para a exibição dos resultados das inferências, fossem utilizados números ao invés das barras gráficas. A versão utilizada na validação exibia um texto com o valor numérico quando o usuário passava o mouse sobre a barra, porém, o acesso à essa informação é mais difícil. Isso seria resolvido com a inclusão de uma opção no software para que seja possível escolher se a exibição dos resultados será na forma de números ou barras.

Outro item levando foi que a versão do software utilizada na validação não permitia a seleção de múltiplas variáveis simultaneamente. Dessa forma, mover várias variáveis de lugar se torna bastante trabalhoso. Com a implementação dessa funcionalidade, as operações de mover e apagar múltiplas variáveis se tornaria mais rápida.

Também foi sugerida uma funcionalidade muito semelhante à do AMPLIA, para que, ao editar a probabilidade de um estado de uma variável com apenas dois estados possíveis, o outro estado seja preenchido automaticamente com o valor necessário para que a soma de ambos totalize 1. Isso agilizaria bastante o preenchimento das tabelas.

Foi observado que, quando a tabela de probabilidades possui uma lista muito grande de combinações, é exibida uma barra de rolagem para que o usuário possa navegar por toda a tabela. Porém, ao mover a barra de rolagem para baixo, o cabeçalho da tabela é escondido, o que pode gerar confusão durante a edição da tabela. O ideal seria que o cabeçalho permanecesse fixo e apenas o corpo da tabela rolasse.

CONCLUSÃO

A utilização de redes bayesianas mostra ser uma maneira eficaz para a solução de problemas que envolvem a incerteza. Por representarem relações de causa e efeito entre variáveis, são semelhantes ao raciocínio de especialistas em determinada área. Por este motivo, podem ser aplicadas nos mais diversos problemas.

A usabilidade pode ser um fator muito importante no sucesso de um software pois, caso ele possua uma usabilidade baixa, o usuário pode ficar bastante frustrado e simplesmente abandonar o produto. Técnicas de avaliação de usabilidade mostram-se bastante úteis para que um índice satisfatório seja atingido ao desenvolver um software.

Os objetivos definidos no anteprojeto deste trabalho foram atingidos. O objetivo geral foi definido como: criar um editor de redes bayesianas com foco em usabilidade que possa ser empregado em diferentes aplicações de diferentes domínios. Conforme relatado neste trabalho, o desenvolvimento do editor foi realizado e nele podem ser construídas diversas redes bayesianas. Pela própria definição destas redes, elas podem ser aplicadas a qualquer problema que envolva incerteza, portanto, o editor também se aplica a diferentes domínios.

Além disso, foram definidos objetivos específicos para realizar pesquisa bibliográfica a respeito de redes bayesianas e conceitos de usabilidade. Estas pesquisas foram realizadas e serviram como base durante todo o desenvolvimento do trabalho. Também foram feitas pesquisas sobre os editores de redes bayesianas existentes no mercado, o que foi bastante útil para encontrar pontos fortes e fracos nas soluções existentes, buscando manter as características positivas e eliminar as negativas na nova solução.

Tendo como base os resultados dessas pesquisas, foi feita uma especificação do novo editor utilizando como principal ferramenta os *wireframes*, além da escolha de tecnologias que estejam alinhadas com os objetivos do novo editor. Após essa especificação, foi feito o desenvolvimento deste novo editor. Estas duas etapas também estavam incluídas nos objetivos específicos do trabalho.

Por fim, o último objetivo específico definido foi a validação parcial do editor. Essa validação foi definida como parcial pois, conforme relatado no Capítulo 5, existem diversas outras possibilidades e rodadas de validação que poderiam ser realizadas, mas que não foram definidas como parte do escopo deste trabalho. Portanto, uma validação completa fica definida como um dos trabalhos futuros.

Como um dos resultados da validação realizada, obteve-se uma quantidade valiosa de melhorias que podem ser realizadas no editor. Estas melhorias serão feitas em uma versão futura e, após isso, o editor será submetido a mais rodadas de testes com o intuito de validar as soluções aplicadas.

Conforme relatado no Capítulo 5, para redes complexas o editor apresentou lentidão, principalmente pelo fato de que, para mostrar todos os resultados em tempo real, diversas inferências precisam ser feitas na mesma rede. O algoritmo *junction tree* é mais indicado para essa situação, pois é dividido em duas partes: a compilação da *junction tree* e a inferência na *junction tree*. A inferência na *junction tree* é bastante eficiente, e a mesma *junction tree* construída para uma rede pode ser reutilizada para todas as inferências nela (BEAL, 2003). Desta forma, fica compreendido como trabalho futuro o estudo e implementação deste algoritmo na biblioteca de inferência.

No Capítulo 3, foram apresentados itens referentes à identidade visual e estética do software. Estes itens estão fortemente ligados à usabilidade, e possuem impacto nela. Porém, este trabalho focou nas questões de simplificar o fluxo de trabalho e comportamentos do software. Portanto, um trabalho futuro da área de design para avaliar e definir as questões estéticas e de identidade visual do software poderia aumentar a usabilidade do mesmo.

Uma funcionalidade que facilitaria a migração de usuários de outros editores de redes bayesianas para o editor desenvolvido neste trabalho seria a importação e exportação de redes entre os editores. Desta forma, caso o usuário possua redes já salvas, não seria preciso recriá-las novamente. Para isso, um trabalho futuro para estudo dos formatos utilizados por estes editores e implementação dessas funcionalidades fica definido.

Conforme relatado no Capítulo 1, redes bayesianas podem ser construídas manualmente ou automaticamente a partir de dados. Este trabalho focou na construção manualmente por meio de um software editor. No futuro, poderia ser incluída a funcionalidade de gerar uma rede automaticamente a partir de alguma fonte de dados externa, com a possibilidade de o especialista fazer ajustes manuais após essa geração utilizando o editor desenvolvido.

Além destes trabalhos futuros, é importante ressaltar as principais contribuições deste trabalho. Como foi feito em estudo de usabilidade para vários editores de redes bayesianas, os resultados deste estudo podem ser utilizados para realizar melhorias nestes editores e também

podem servir para qualquer trabalho que possua o objetivo de estudar e melhorar e usabilidade de algum produto.

Na validação também podem ser encontrados diversos itens a respeito de usabilidade que podem ser úteis, não apenas para editores de redes bayesianas, mas também para diversos softwares. Estes itens mostraram as dificuldades que os usuários encontraram e também como foi importante realizar testes com eles.

Durante o desenvolvimento, também foi relatado um comparativo entre dois algoritmos para inferência, mostrando os resultados de cada um. Isto pode ser útil para quem deseja iniciar um projeto e não tem um conhecimento prático de quanto tempo cada algoritmo demora para realizar inferências em redes bayesianas.

Por fim, foi desenvolvido um editor de redes bayesianas que pode ser utilizado para criar e observar o resultado de redes para os mais diversos problemas. Também foi desenvolvida uma biblioteca para realizar a inferência em redes bayesianas que pode ser utilizada nos mais diversos softwares. Ambos os projetos foram desenvolvidos com código aberto e podem servir como referência para outros trabalhos. Um artigo sobre o conteúdo deste trabalho foi publicado no CAVA 2016 e está apresentado no apêndice E.

REFERÊNCIAS BIBLIOGRÁFICAS

BARROS, Paulo Ricardo Muniz et al. Um simulador de casos clínicos complexos no processo de aprendizagem em saúde. **RENOTE - Revista Novas Tecnologias na Educação**, Porto Alegre, v. 10, n. 1, p. 234, jun. 2012.

BEAL, Matthew James. **Variational algorithms for approximate Bayesian inference.** London: University of London, 2003.

FLORES, Cecília Dias et al. **Negociação pedagógica no ambiente AMPLIA.** In CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, v. 23, p. 617-626. 2003.

FONSECA, João Marcelo Lopes. **Descrição de um simulador baseado em redes bayesianas como método de avaliação do aprendizado de diretrizes clínicas em ensino a distância para medicina de família e comunidade.** 2013. 118 f. Dissertação (Mestrado em Ciências da Saúde) – Universidade Federal de Ciências da Saúde de Porto Alegre, Porto Alegre, RS, 2013.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **NBR ISO/IEC 9126-1** – Engenharia de Software – Qualidade de Produto - Parte 1: Modelo de Qualidade. Rio de Janeiro, 2003.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 9241-210** - Ergonomics of human–system interaction – Part 210: Human-centred design for interactive systems. 2010.

GARRETT, Jesse James. **The elements of user experience:** user-centered design for the web and beyond. 2. ed. Berkley: New Riders, 2011. 172 p.

HUGIN. **Company Information.** Disponível em: <http://www.hugin.com/company-information>. Acesso em: jun. 2016.

KRUG, Steve. **Não me faça pensar!:** uma abordagem de bom senso à usabilidade na web. 2. ed. Rio de Janeiro: Alta Books, 2006. 201 p.

JHA, Vikram; DUFFY, Sean. "Ten golden rules" for designing software in medical education: results from a formative evaluation of DIALOG. **Medical Teacher**, v. 24, n. 4, p. 417-421, 2002.

LIMA, Alessandro et al. Projeto para desenvolvimento do Simulador Health Simulator. **Anais do Computer on the Beach**, Florianópolis, 279-288. 2015.

LUCAS, Peter J. F; GAAG, Linda C. van der; ABU-HANNA, Ameen. Bayesian networks in biomedicine and health-care. **Artificial Intelligence in Medicine.** Elsevier Health. Cascais – Portugal, v. 30, n. 3, p. 201-214, 2004.

MARQUES, Roberto Ligeiro; DUTRA, Inês. **Redes Bayesianas:** o que são, para que servem, algoritmos e exemplos de aplicações. Coppe Sistemas – Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2002.

MARONI, Vinícius. **Construção de um motor de inferência para análise de desempenho em ambientes virtuais de aprendizagem aplicados ao ensino da medicina de família e comunidade.** 2013. 103 f. Dissertação (Mestrado em Ciências da Saúde) – Universidade Federal de Ciências da Saúde de Porto Alegre, Porto Alegre, RS, 2013.

MEMÓRIA, Felipe. **Design para a Internet:** projetando a experiência perfeita. Rio de Janeiro: Elsevier, 2005.

MADSEN, Anders L. et al. The Hugin tool for learning Bayesian networks. In: **Symbolic and quantitative approaches to reasoning with uncertainty.** Springer Berlin Heidelberg, 2003. p. 594-605.

MEURER, Heli. **Ferramenta de gerenciamento e recomendação como recurso na aprendizagem baseada em projetos em design.** 2014. 246 f. Teste (Doutorado em Informática na Educação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, 2014.

MEURER, Heli; SZABLUK, Daniela. Projeto E: aspectos metodológicos para o desenvolvimento de projetos dígito-virtuais. **Ação Ergonômica – ERGODESIGN II**, v. 5, n. 2, 2010.

NETICA. **Netica Application.** Disponível em: <https://www.norsys.com/netica.html>. Acesso em: jun. 2016.

NETICA. **Netica APIs (Application Programmer Interfaces).** Disponível em: <https://www.norsys.com/netica_api.html>. Acesso em: jun. 2016.

NILSSON, Nils J. **Artificial intelligence:** a new synthesis. San Francisco: Morgan Kaufmann, 1998. 513 p.

NORMANN, Donald A. **The Design of Everyday Things.** New York: Basic Books, 1988.

PREECE, Jennifer; ROGERS; Yvonne; SHARP, Helen. **Design de Interação:** Além da Interação Homem-Computador. Porto Alegre: Bookman, 2013. 600 p.

REISS, Eric. **Usable usability:** simple steps for making stuff better. Indiana: John Wiley & Sons, 2012. 230 p.

RUSSEL, Stuart; NORVIG, Peter. **Artificial Intelligence:** A modern approach. 3. ed. New Jersey: Prentice-Hall. 1995. 1132 p.

SANTOS, Felipe Scuciatto dos. **Desenvolvimento de módulo de redes bayesianas para o amplia.** 2009. 70 f. Trabalho de Conclusão de Curso (Monografia) – Curso de Ciência da Computação, Universidade Feevale, Novo Hamburgo, RS, 2009.

SEIXAS, Louise Jeanty de et al. **An architecture for an intelligent learning environment with a constructivist approach.** ITS2002. San Sebastian, Spain. 2002.

SEIXAS, Louise Marguerite Jeanty de. **Estratégias pedagógicas para um ambiente multi-agente probabilístico inteligente de aprendizagem – AMPLIA.** 2005. 147 f. Tese (Doutorado em Informática na Educação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, 2005.

UNBBAYES. **UnBBayes Overview.** Disponível em: <http://unbbayes.sourceforge.net/index.html>. Acesso em: jun. 2016.

WIEGERINCK, Wim; KAPPEN, Bert; BURGERS, Willem. **Bayesian networks for expert systems: Theory and practical applications.** In: Interactive collaborative information systems. Springer Berlin Heidelberg, 2010. p. 547-578.

APÊNDICE A – ALGORITMO DE ENUMERAÇÃO

```
// @flow

import equal from 'deep-equal';

type CptWithoutParents = {
  [key: string]: number
};

type CptWithParentsItem = {
  when: { [key: string]: string },
  then: { [key: string]: number }
};

type CptWithParents = CptWithParentsItem[];

type Node = {
  id: string,
  states: string[],
  parents: string[],
  cpt: CptWithoutParents | CptWithParents
};

type Network = {
  [key: string]: Node
};

type Combinations = {
  [key: string]: string
};

const combinationsCache = new WeakMap();

export function infer(network: Network, nodes: Combinations, giving:
?Combinations): number {
  let combinations: Combinations[] = combinationsCache.get(network);

  if (combinations === undefined) {
    combinations = buildCombinations(network);
    combinationsCache.set(network, combinations);
  }

  let filteredCombinations: Combinations[] =
filterCombinations(combinations, nodes);
  let probGiving: number = 1;

  if (giving) {
    filteredCombinations = filterCombinations(filteredCombinations,
giving);
    probGiving = infer(network, giving);
  }

  return calculateProbabilities(network, filteredCombinations) /
probGiving;
}
```

```
function buildCombinations(network: Network): Combinations[] {
  const combinations: Combinations[] = [];

  makeCombinations(Object.keys(network));

  return combinations;

  function makeCombinations(nodes: string[], acc: Combinations = {}):
void {
    if (nodes.length === 0) {
      combinations.push(acc);
      return;
    }

    const [ node: string, ...rest: string[] ] = nodes;
    const states: string[] = network[node].states;

    for (let i = 0; i < states.length; i++) {
      const state: string = states[i];

      makeCombinations(rest, {
        ...acc,
        [node]: state
      });
    }
  }
}

function filterCombinations(combinations: Combinations[],
nodesToFilter: Combinations): Combinations[] {
  const idsToFilter = Object.keys(nodesToFilter);

  return combinations.filter(row => {
    for (let i = 0; i < idsToFilter.length; i++) {
      const idToFilter = idsToFilter[i];

      if (row[idToFilter] !== nodesToFilter[idToFilter]) {
        return false;
      }
    }

    return true;
  });
}

function calculateProbabilities(network: Network, combinations:
Combinations[]): number {
  const rowsProducts: number[] = [];

  for (let i = 0; i < combinations.length; i++) {
    let rowProduct = 1;

    const row = combinations[i];
    const ids = Object.keys(row);

    for (let j = 0; j < ids.length; j++) {
      const nodeId = ids[j];
      const node = network[nodeId];
```

```
      const cpt = (node.cpt : any);

      if (node.parents.length === 0) {
        rowProduct *= cpt[row[nodeId]];
      } else {
        const when = {};

        for (let k = 0; k < node.parents.length; k++) {
          const parent = node.parents[k];
          when[parent] = row[parent];
        }

        for (let k = 0; k < cpt.length; k++) {
          const cptRow = cpt[k];
          if (equal(cptRow.when, when)) {
            rowProduct *= cptRow.then[row[nodeId]];
            break;
          }
        }
      }
    }

    rowsProducts.push(rowProduct);
  }

  let probability = 0;

  for (let i = 0; i < rowsProducts.length; i++) {
    probability += rowsProducts[i];
  }

  return probability;
}
```

APÊNDICE B – ALGORITMO DE ELIMINAÇÃO DE VARIÁVEIS

```
// @flow

type CptWithoutParents = {
  [key: string]: number
};

type CptWithParentsItem = {
  when: { [key: string]: string },
  then: { [key: string]: number }
};

type CptWithParents = CptWithParentsItem[];

type Node = {
  id: string,
  states: string[],
  parents: string[],
  cpt: CptWithoutParents | CptWithParents
};

type Network = {
  [key: string]: Node
};

type Combinations = {
  [key: string]: string
};

type FactorItem = {
  states: { [nodeId: string]: string },
  value: number
};

type Factor = FactorItem[];

export function infer(network: Network, nodes: Combinations, giving:
?Combinations): number {
  const variables = Object.keys(network);
  const variablesToInfer = Object.keys(nodes);
  const variablesGiving = giving ? Object.keys(giving) : [];

  const variablesToEliminate = variables
    .filter(x => !variablesToInfer.some(y => y === x) &&
!variablesGiving.some(y => y === x));

  const factors = variables
    .map(nodeId => buildFactor(network[nodeId], giving));

  while (variablesToEliminate.length > 0) {
    const varToEliminate = variablesToEliminate.shift();

    const factorsToJoin = factors.filter(factor => {
      return Object.keys(factor[0].states).some(nodeId => nodeId ===
varToEliminate);
    });
```

```
    const resultFactor = eliminateVariable(
      factorsToJoin.reduce((f1, f2) => joinFactors(f1, f2)),
      varToEliminate
    );

    for (let i = 0; i < factorsToJoin.length; i++) {
      factors.splice(factors.indexOf(factorsToJoin[i]), 1);
    }

    factors.push(resultFactor);
  }

  const joinedFactor = factors
    .filter(factor => Object.keys(factor[0].states).length > 0)
    .sort((f1, f2) => f1.length - f2.length)
    .reduce((f1, f2) => {
      return joinFactors(f1, f2);
    });

  const normalizedFactor = normalizeFactor(joinedFactor);

  const inferenceRow = normalizedFactor.find(row => {
    return variablesToInfer.every(v => row.states[v] === nodes[v]);
  });

  if (inferenceRow === undefined) {
    return 0;
  }

  return inferenceRow.value;
}

function buildFactor(node: Node, giving: ?Combinations): Factor {
  const factor = [];
  const cpt = (node.cpt : any);

  if (node.parents.length === 0) {
    for (let i = 0; i < node.states.length; i++) {
      const state = node.states[i];

      factor.push({
        states: { [node.id]: state },
        value: cpt[state]
      });
    }
  } else {
    for (let i = 0; i < cpt.length; i++) {
      for (let j = 0; j < node.states.length; j++) {
        const state = node.states[j];

        factor.push({
          states: { ...cpt[i].when, [node.id]: state },
          value: cpt[i].then[state]
        });
      }
    }
  }
```

```
  if (giving) {
    const givingIds = Object.keys(giving);

    for (let i = factor.length - 1; i >= 0; i--) {
      for (let j = 0; j < givingIds.length; j++) {
        const givingId = givingIds[j];

        if (factor[i].states[givingId] && factor[i].states[givingId]
!== giving[givingId]) {
          factor.splice(i, 1);
          break;
        }
      }
    }
  }

  return factor;
}

function joinFactors(f1: Factor, f2: Factor): Factor {
  const newFactor = [];

  for (let i = 0; i < f1.length; i++) {
    for (let j = 0; j < f2.length; j++) {
      const states = {
        ...f1[i].states,
        ...f2[j].states
      };

      const nodeIds = Object.keys(states);

      const alreadyExists = newFactor.some(x => {
        return nodeIds.every(nodeId => x.states[nodeId] ===
states[nodeId]);
      });

      if (!alreadyExists) {
        newFactor.push({ states, value: 0 });
      }
    }
  }

  const nodeIdsF1 = Object.keys(f1[0].states);
  const nodeIdsF2 = Object.keys(f2[0].states);

  for (let i = 0; i < newFactor.length; i++) {
    const rowNewFactor = newFactor[i];

    const rowF1 = f1.find(x => {
      return nodeIdsF1.every(nodeId => x.states[nodeId] ===
rowNewFactor.states[nodeId]);
    });

    const rowF2 = f2.find(x => {
      return nodeIdsF2.every(nodeId => x.states[nodeId] ===
rowNewFactor.states[nodeId]);
    });
```

```
    if (rowF1 === undefined || rowF2 === undefined) {
      throw new Error('Fatal error');
    }

    rowNewFactor.value = rowF1.value * rowF2.value;
  }

  return newFactor;
}

function eliminateVariable(factor: Factor, variable: string): Factor {
  const newFactor = [];

  for (let i = 0; i < factor.length; i++) {
    const states = { ...factor[i].states };

    delete states[variable];

    const nodeIds = Object.keys(states);

    const existingRow = newFactor.find(x => {
      return nodeIds.every(nodeId => x.states[nodeId] ===
states[nodeId]);
    });

    if (existingRow === undefined) {
      newFactor.push({
        states,
        value: factor[i].value
      });
    } else {
      existingRow.value += factor[i].value;
    }
  }

  return newFactor;
}

function normalizeFactor(factor: Factor): Factor {
  const total = factor.reduce((acc, row) => acc + row.value, 0);

  if (total === 0) {
    return factor;
  }

  return factor
    .map(row => ({
      states: { ...row.states },
      value: row.value / total
    }));
}
```

APÊNDICE C – SLIDES UTILIZADOS NA OFICINA

Introdução a Redes Bayesianas

Fernando Alex Helwanger

Roteiro

- Para que servem?
- Estrutura de uma rede bayesiana
- Tipos de inferências em redes bayesianas
- Hugin
- BayesEditor

Para que servem?

- Problemas que envolvem incerteza
  - Saúde
  - Diagnósticos em geral
  - Processamento de imagens
  - Direito
  - Seguros
  - Etc...
- Representação eficiente

Parte qualitativa – estrutura lógica da rede

Parte quantitativa – probabilidades associadas

C=SIM	C=NÃO
0,1	0,9

C	T=SIM	T=NÃO
SIM	0,8	0,2
NÃO	0,1	0,9

T	A=SIM	A=NÃO
SIM	0,3	0,7
NÃO	0,1	0,9

Observações

- Cada variável pode ter um número diferente de estados
  - {Sim, Não}
  - {Baixo, Médio, Alto}
  - {Chuvoso, Nublado, Ensolarado}
  - Etc...
- Cada linha das tabelas de probabilidades deve somar 1
- Rede não pode possuir ciclos

Evidências

- Ao utilizar a rede em um caso específico, é possível informar evidências
- Exemplo: dado que está chovendo, qual a probabilidade de que eu me atrase?
- As tabelas de probabilidades **não** mudam para cada caso

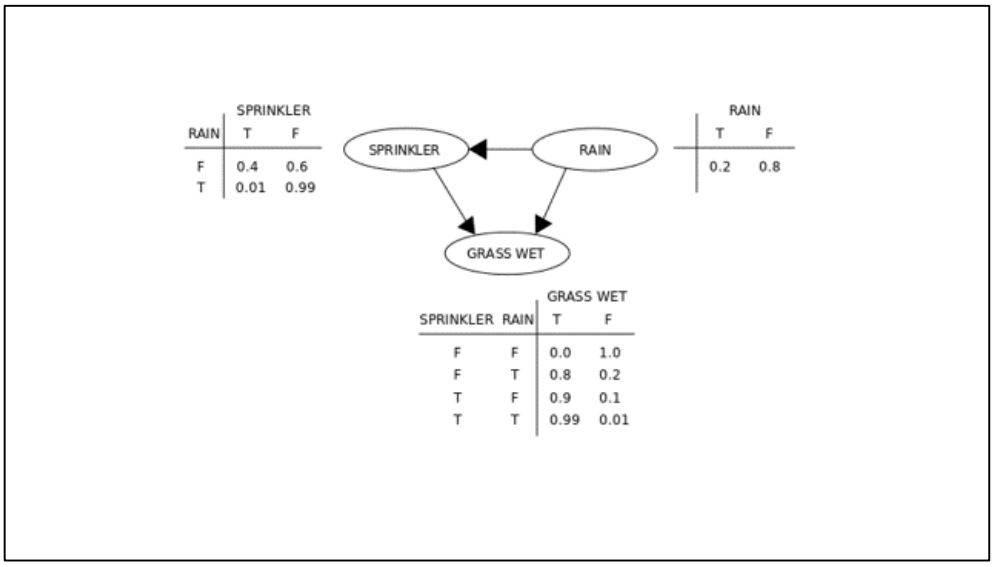
Tipos de inferências

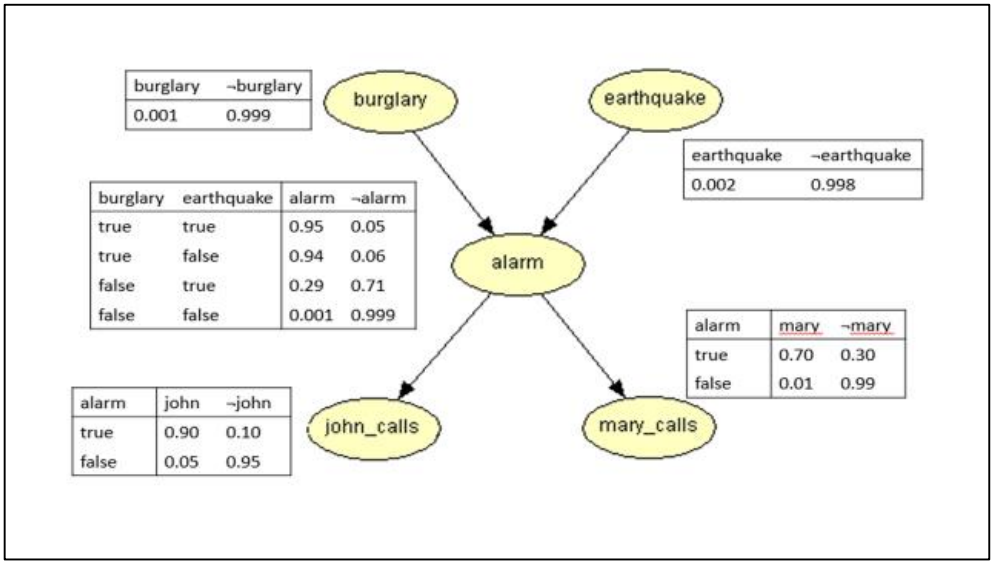
- P(C=Sim) = probabilidade de estar chovendo
- P(C=Sim, A=Não) = probabilidade de estar chovendo e não haver atraso acontecerem simultaneamente
- P(A=Sim | C=Sim) = probabilidade de haver atraso dado que está chovendo

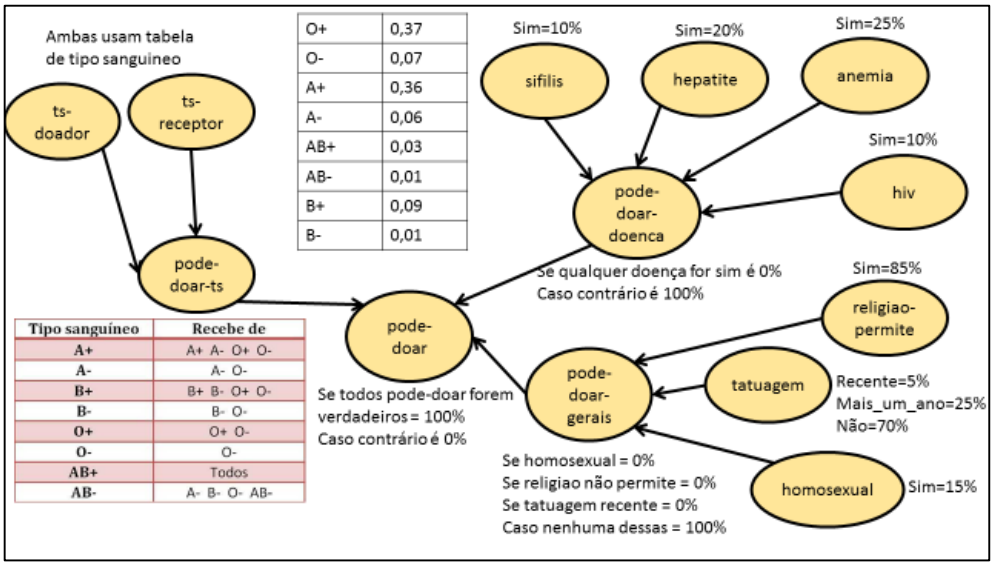
Passos para construção de redes bayesianas

1. Selecionar variáveis
2. Identificar relacionamentos entre as variáveis
3. Identificar probabilidades
4. Avaliar o resultado da rede

Exemplos


SPRINKLER
RAIN
SPRINKLER
RAIN
GRASS WET
RAIN
T F
0.2 0.8
SPRINKLER RAIN
T F
F 0.4 0.6
T 0.01 0.99
GRASS WET
SPRINKLER RAIN T F
F F 0.0 1.0
F T 0.8 0.2
T F 0.9 0.1
T T 0.99 0.01


burglary ¬burglary
0.001 0.999
burglary
earthquake
earthquake ¬earthquake
0.002 0.998
burglary earthquake alarm ¬alarm
true true 0.95 0.05
true false 0.94 0.06
false true 0.29 0.71
false false 0.001 0.999
alarm
alarm mary ¬mary
true 0.70 0.30
false 0.01 0.99
alarm john ¬john
true 0.90 0.10
false 0.05 0.95
john_calls
mary_calls


Ambas usam tabela de tipo sanguineo
ts-doador
ts-receptor
pode-doar-ts
O+ 0,37
O- 0,07
A+ 0,36
A- 0,06
AB+ 0,03
AB- 0,01
B+ 0,09
B- 0,01
Tipo sanguíneo Recebe de
A+ A+ A- O+ O-
A- A- O-
B+ B+ B- O+ O-
B- B- O-
O+ O+ O-
O- O-
AB+ Todos
AB- A- B- O- AB-
Sim=10%
sifilis
Sim=20%
hepatite
Sim=25%
anemia
Sim=10%
hiv
pode-doar-doenca
Se qualquer doença for sim é 0%
Caso contrário é 100%
Sim=85%
religiao-permite
pode-doar
Se todos pode-doar forem verdadeiros = 100%
Caso contrário é 0%
pode-doar-gerais
tatuagem
Recente=5%
Mais_um_ano=25%
Não=70%
Se homosexual = 0%
Se religiao não permite = 0%
Se tatuagem recente = 0%
Caso nenhuma dessas = 100%
homosexual
Sim=15%

APÊNDICE D – QUESTIONÁRIOS UTILIZADOS NA VALIDAÇÃO

Hugin

O objetivo deste questionário é avaliar qual o desempenho do software Hugin durante o seu teste de usabilidade.

1. **Nome do usuário**

   .....

2. **O editor fornece as funcionalidades necessárias para a criação e inferência em uma rede bayesiana**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

3. **O editor permite que a criação e inferência em uma rede bayesiana seja realizada com rapidez**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

4. **O editor possui um fluxo de trabalho que é claro e não gera confusões**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

5. **Os textos e mensagens utilizados no editor são claros**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

6. **O editor não apresenta erros durante a execução**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

7. **O editor informa ao usuário quando dados inválidos são informados**
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

8. **A interface gráfica utilizada pelo editor é intuitiva**
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

9. **É fácil de lembrar como executar as operações no editor**
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

10. **O editor fornece os resultados com tempo de resposta satisfatório**
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

11. **Caso possua alguma sugestão ou crítica em relação ao editor, preencha aqui**

BayesEditor

O objetivo deste questionário é avaliar qual o desempenho do software Hugin durante o seu teste de usabilidade.

1. **Nome do usuário**

   .....

2. **O editor fornece as funcionalidades necessárias para a criação e inferência em uma rede bayesiana**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

3. **O editor permite que a criação e inferência em uma rede bayesiana seja realizada com rapidez**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

4. **O editor possui um fluxo de trabalho que é claro e não gera confusões**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

5. **Os textos e mensagens utilizados no editor são claros**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

6. **O editor não apresenta erros durante a execução**
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

7. **O editor informa ao usuário quando dados inválidos são informados**
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

8. **A interface gráfica utilizada pelo editor é intuitiva**
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

9. **É fácil de lembrar como executar as operações no editor**
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

10. **O editor fornece os resultados com tempo de resposta satisfatório**
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo Totalmente	◯	◯	◯	◯	◯	Concordo Totalmente

11. **Caso possua alguma sugestão ou crítica em relação ao editor, preencha aqui**

APÊNDICE E – ARTIGO PUBLICADO NO CAVA

Proposta de um algoritmo de recomendação usando uma rede bayesiana no Health Simulator

Marta Bez, Paulo R. M. Barros, Alessandro Lima, Fernando Helwanger, Diego Reidel

Abstract— **This paper presents the Health Simulator project in the context of Artificial Intelligence in what regards to the storage of specialized knowledge in health care and educational strategy, that allows guiding students during learning process. Therefore, the paper covers the theory of Bayesian Networks, recommendation systems, the Health Simulator and finally a proposal for applying the techniques in the simulator environment.**

Index Terms—. **Recommendation Systems, Simulator, Health Simulator, Bayesian Networks**

Resumo— **Este artigo apresenta o projeto Health Simulator no contexto de inteligência artificial, no que se refere ao armazenamento do conhecimento especializado na área da saúde e estratégia pedagógica, que permite o auxílio ao aluno no seu processo de aprendizagem. Para tanto é apresentada a teoria de redes bayesianas, sistemas de recomendação, o projeto Health Simulator e a proposta de aplicação das técnicas utilizadas neste ambiente.**

Palavras Chave - Sistemas de recomendação, Simulador, Health Simulator, Redes Bayesianas

I. Introdução

Tem sido percebido um constante dinamismo e mudanças de conceitos nos ambientes estudantis. Novas métricas de ensino surgem de forma cada vez mais rápida e, junto a elas, as tecnologias educacionais. Segundo Botezatu et al. [1] essas, mais do que uma opção, são uma necessidade. Da mesma forma, as lacunas entre as atividades teóricas e as experiências clínicas que os estudantes da área da saúde vivenciam têm sido uma preocupação entre os educadores [2], pois os sistemas universitários fornecem uma estrutura envolvendo longos períodos de estudo intercalados com a prática clínica e os alunos, muitas vezes, não conseguem articular todos os conhecimentos e aplicar isso à prática [3], [4].

Buscando amenizar estes problemas, tem surgido diversos simuladores que permitem ao aluno articular a teoria e a prática. Exemplos desses simuladores são os desenvolvidos por [5], [1], [6], [7], entre tantos outros.

Simuladores de ensino médico podem ser compreendidos de forma ampla como ferramentas que permitam aos educadores manter o controle total em cenários clínicos pré-selecionados, descartando, nesta fase de aprendizagem, os riscos potenciais ao paciente [8]. Complementando, Kincaid e Hamilton [9], apresentam vantagens no uso de simuladores para o ensino médico, tais como: auxilia o aluno a compreender as relações complexas, que de outro modo exigiriam equipamentos caros ou experiências potencialmente perigosas; permite a aplicação de conhecimentos científicos e técnicos de forma integrada e simultânea; permite que o aluno busque novos métodos e estratégias para a solução de um mesmo caso do estudo; fornece um ambiente próximo da realidade para a formação e o reforço dos conhecimentos adquiridos; reduz o risco em situações autênticas.

Bradley [10] complementa, identificando outros benefícios, como: riscos para os pacientes e alunos são evitados; a interferência indesejada de fatores externos ao foco do ensino é reduzida; as habilidades podem ser praticadas repetidamente; o treinamento pode ser adaptado para os indivíduos; a retenção e a precisão são aumentadas; a transferência de treinamento da sala de aula para uma situação real é reforçada; normas de referência para avaliar o desempenho dos alunos e diagnosticar as necessidades educacionais são reforçadas.

Este artigo apresenta a proposta de um simulador do tipo Paciente Virtual denominado Health Simulator. Este se utiliza de dois conceitos importantes da área de Inteligência Artificial: Redes Bayesianas e Sistemas de Recomendação. Estes conceitos são apresentados na sequência. Ao final é apresentado o simulador Health Simulator e a integração de Sistemas de Recomendação como forma de indicar os próximos casos de estudos a serem resolvidos de acordo com a evolução do aluno na Rede Bayesiana.

II. Redes Bayesianas

O processo de diagnóstico, independente se for médico, de reparo automobilístico, ou qualquer outra área, quase sempre envolve incerteza. Neste processo, usando a área médica como exemplo, o médico avalia sinais e sintomas que indicam a probabilidade de o paciente possuir determinada doença. Neste cenário, as doenças são consideradas causas e os sinais e sintomas são considerados efeitos [11], [12].

Os sinais e sintomas geralmente não indicam com 100% de certeza a causa de uma única doença, mas sim qual a probabilidade de que o paciente possua cada doença presente em um determinado conjunto. Para a solução deste tipo de problema, não é possível a utilização de técnicas, como a lógica de primeira ordem, que tratem o resultado como uma única

possibilidade. É preciso utilizar técnicas que determinem o grau de confiança em cada um dos possíveis resultados [11], [13].

Para modelar este tipo de problema, é preciso identificar quais variáveis aleatórias tem relevância para o problema. Essas variáveis podem ser de diferentes tipos [13]:

1) *Booleanas:* variáveis que representem proposições, de domínio "verdadeiro" ou "falso".
2) *Numéricas:* variáveis que representem medidas físicas, como altura, largura, velocidade, etc.
3) *Categóricas:* variáveis que representem categorias, como cores, letras, entre outras.

Feita a definição das variáveis relevantes para o problema, uma maneira de resolvê-lo seria através da construção de uma tabela que considere todos os estados possíveis e atribua uma probabilidade a cada uma das possibilidades. Porém, para problemas reais que envolvam um número relativamente grande de variáveis, essa tabela pode ter um tamanho impraticável, já que seu crescimento é exponencial em relação ao número de variáveis envolvidas [11], [12], [13].

Outra maneira mais compacta para a representação deste tipo de problema é através da utilização de Redes Bayesianas. Nas Redes Bayesianas, a propriedade de independência condicional entre as variáveis é aproveitada para que o número de combinações entre elas seja diminuído. Uma variável A é dita condicionalmente independente de uma variável B caso, dado o valor de uma variável C, B não tenha nenhuma influência em A. Com essa diminuição de tamanho adquirida pela utilização de Redes Bayesianas, problemas que seriam impraticáveis tornam-se possíveis [11], [13].

Uma Rede Bayesiana envolve duas partes, uma qualitativa e outra quantitativa. A parte qualitativa envolve a estruturação da rede na forma de um grafo acíclico dirigido. Neste grafo, cada nodo representa uma variável aleatória e a topologia do grafo representa as relações de independência condicional entre as variáveis. Se houver uma aresta que vai do nodo A até o nodo B, o nodo A é dito pai do nodo B. Neste caso, B é condicionalmente independente de todos os outros nodos que não sejam descendentes de B, dado o valor de A [11], [13]. A parte qualitativa envolve a definição de uma tabela de distribuição de probabilidades condicionais para cada variável. Porém, ao invés de considerar todas as combinações possíveis entre todas as variáveis observadas, é considerada apenas a influência dos pais diretos de cada nodo. Com isto, não é necessário trabalhar com uma tabela normalmente grande, mas sim, com uma tabela menor associada a cada nodo, o que geralmente é mais simples [11], [13]. A definição das probabilidades para estas tabelas pode ser construída a partir da computação de dados, manualmente, por um especialista no domínio analisado ou ainda uma mistura entre as duas técnicas [14].

As arestas que conectam os nodos de uma Rede Bayesiana são normalmente interpretadas como uma relação de causa e efeito entre duas variáveis. Ou seja, se uma determinada doença causar um determinado sintoma, isso pode ser representado em uma Rede Bayesiana em que o nodo da doença é pai do nodo do sintoma, representando que ela é a causa do efeito. Isso se assemelha a maneira que humanos especialistas em domínios de conhecimento raciocinam, o que torna uma Rede Bayesiana uma representação natural do conhecimento [13].

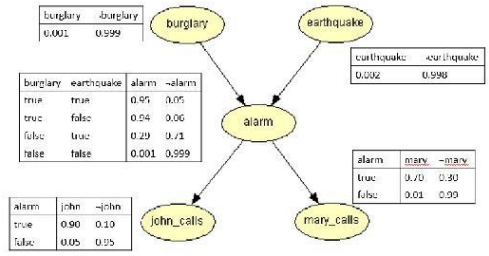


Fig. 1. Exemplo de Rede Bayesiana [11].

A Fig. 1, apresenta um exemplo de Rede Bayesiana composta por 5 nodos, juntamente com as tabelas de probabilidade condicional associadas a cada nodo. O problema modelado nesta rede envolve o acionamento do alarme de uma residência, que pode ter como causa um roubo ou um terremoto. O fato do alarme ser acionado pode causar mais dois efeitos, que seriam a probabilidade de dois vizinhos ligarem para o morador avisando do ocorrido. A partir dessa definição é possível realizar diversas inferências probabilísticas, como, por exemplo, dado que os dois vizinhos ligaram, qual a probabilidade de que tenha ocorrido um roubo?

O uso de software para a construção e o cálculo de Redes Bayesianas pode ser utilizado para auxiliar o processo de diagnóstico médico, tanto na prática quanto no ensino. No contexto do ensino, o aluno precisa praticar a construção de modelos hipotéticos que relacionem doenças com suas causas e sintomas. É preciso que ele possa avaliar a situação e tomar decisões durante este processo. Desta forma, o uso de simuladores proporciona ao aluno um ambiente em que ele possa praticar este tipo de raciocínio. Nestes simuladores, o uso de Redes Bayesianas provê uma maneira para a representação e inferência do conhecimento e raciocínio [15].

III. Sistemas de Recomendação

Um sistema de recomendação (SR) é uma ferramenta computacional que, dado um grupo extenso de itens recomendáveis, apresenta a um usuário aqueles que possam ser de seu maior interesse. Geralmente essa apresentação se dá na forma de uma recomendação simples ou como uma predição do valor da utilidade de um dado item para o usuário em questão [16]. O estudo desses sistemas ganhou destaque por volta da década de 90, principalmente quando cientistas do Palo Alto Research Center (PARC), na Califórnia, desenvolveram um algoritmo para filtragem de correio eletrônico [17], [16].

Em ferramentas ou sistemas com grande volume de itens, como fóruns de debate, bibliotecas virtuais, ou mesmo a Internet, uma sobrecarga de conteúdo pode acabar atrapalhando a busca de um usuário por aquilo que ele precisa ou acha mais relevante. Quando a informação que se busca é mais especializada, esse problema se acentua, de forma que frequentemente faz-se necessário que se dedique uma grande

porção de tempo garimpando as respostas obtidas em uma busca. Além disso, é possível que informações importantes se percam por estarem contidas em materiais indexados posteriormente [18]. Um sistema de recomendação é uma ferramenta de inteligência artificial que aprende as preferências de um ou mais usuários e lhe apresenta aquilo que se prevê ter maior valor, mitigando os problemas descritos anteriormente.

Cada técnica de recomendação traz consigo características distintas, porém grande parte dos estudos classificam os sistemas de recomendação em três categorias: *content-based filters* (filtragem baseada em conteúdo), *collaborative filtering* (filtragem colaborativa) e algoritmos híbridos, sendo o último o resultado da união das duas primeiras categorias em um mesmo algoritmo [16].

A. *Filtragem Colaborativa*

As técnicas de filtragem colaborativa baseiam-se em recomendar itens com base nas avaliações feitas por outros usuários do sistema. Nessa abordagem duas subcategorias se destacam: recomendação baseada no usuário (*user based*) e baseada no item (*item based*). Na abordagem do tipo *user based*, se cria um agrupamento de indivíduos com perfil similar ao do usuário ativo (vizinhos) e assume-se que a avaliação a um item pelo usuário ativo será bastante similar às avaliações realizadas por parte do seu grupo de vizinhos. Dessa forma, itens bem avaliados pelo grupo de vizinhos serão apresentados ao usuário ativo. Já aqueles que não tenham sido bem avaliados, terão menor importância para o usuário e por isso podem ser indexados posteriormente, por exemplo [19], [20]. Um elemento de grande importância nesse método é a escolha dos indivíduos que farão parte do grupo de vizinhos do usuário em questão [20]. Geralmente esse agrupamento é feito através de uma função de similaridade, como a correlação de Pearson. A equação (1) mostra uma fórmula da correlação de Pearson, que indica a similaridade *sim* entre dois usuários u_a e u_b, onde um grupo de m usuários u_k, com $k = 1, 2,...,m$, $U_m = \{u_1, u_2...,u_m\}$, realizaram uma avaliação $R(u_k,i_l)$ para um item i_l, onde $l = 1, 2,...,n$, $I_n = \{i_1,i_2,...,i_n\}$ é um grupo de itens [21], [22], [20]:

$$sim(u_a,u_b) = \frac{\sum_{l=1}^{n}\left(R(u_a,i_l)-\overline{R}(u_a)\right)\left(R(u_b,i_l)-\overline{R}(u_b)\right)}{\sqrt{\sum_{l=1}^{n}\left(R(u_a,i_l)-\overline{R}(u_a)\right)^2\sum_{l=1}^{n}\left(R(u_b,i_l)-\overline{R}(u_b)\right)^2}}. \quad (1)$$

Já algoritmos baseados em itens traçam similaridades entre itens usando as avaliações de usuários. Desta forma, quando o sistema detecta o interesse do usuário por um item, ele pode recomendar outro item similar. Se os usuários que avaliam o item *i* positivamente também avaliam positivamente o item *j,* é possível assumir que *i* e *j* são similares e, portanto, pode-se recomendar *j* a um usuário *u* que avalie *i* de forma positiva, ou vice-versa. Caso o contrário ocorra e um item *k* seja avaliado negativamente por usuários que bem avaliaram *i*, pode-se assumir que *k* não é similar a *i* e, portanto, não deve ser recomendado a *u* [21].

B. *Filtragem Baseada em Conteúdo*

Na abordagem de recomendação baseada em conteúdo busca-se identificar itens que possam ser de interesse do usuário tomando como base para a decisão avaliações que o usuário já tenha realizado a outros itens. Neste método cada item recomendável traz consigo atributos ou palavras-chave que descrevem o seu conteúdo. Além disso, busca-se construir um perfil para cada usuário com atributos que por ele já tenham sido avaliados. Ou seja, toda vez que o usuário *u* avalia um item *i*, o perfil que descreve as preferências de *u* é atualizado para cada um dos atributos de *i*. Para realizar uma recomendação, se busca por itens que apresentem atributos e características similares às quais o perfil do usuário mostre boa aceitação [21].

Existem diversas maneiras para captar o conteúdo de um item e identificar suas características, sendo uma das principais técnicas conhecida como *Term Frequency – Inverse Document Frequency* (TF-IDF). Essa técnica consiste em identificar termos frequentes em um documento que não sejam frequentes no restante do grupo de itens [23].

C. *Sistemas Híbridos*

As técnicas híbridas consistem em agrupar dois ou mais algoritmos distintos em uma mesma solução, com o intuito de maximizar a relevância da recomendação para o usuário. Cada método de recomendação apresenta forças e desvantagens. A recomendação baseada em conteúdo, por exemplo, é suscetível ao problema conhecido como superespecialização. Já a abordagem colaborativa sofre com o *cold-start* (falta de informações sobre um item ou usuário quando ele foi recém inserido no sistema). Ao utilizar um método híbrido, se busca usar o potencial de uma abordagem para minimizar as fraquezas de outra. Diversos algoritmos híbridos já foram propostos para as mais variadas áreas, demonstrando o potencial dessa técnica [24].

IV. HEALTH SIMULATOR

O Health Simulator é um jogo que simula um Paciente Virtual (PV), projetado para ser um recurso adicional na formação de estudantes da área de saúde. O seu desenvolvimento tem como foco os benefícios do uso de simuladores virtuais e as principais características dos jogos, que além de ser utilizados como recurso de diversão e interatividade de cunho colaborativo [26], permitem ao jogador entender a dinâmica, sistemática de ações, o ambiente e condições que lhe dão vitória ou perda, fazendo com que ele compreenda seus erros e acertos de forma segura e controlada. Outra característica interessante que se vislumbra, é permitir o entendimento de diversas relações complexas, sem a necessidade de equipamentos caros ou situações que possibilitem risco ou dano; permitindo a aplicação de conhecimentos científicos e técnicos de forma integrada e simultânea.

Deste modo, o Health Simulator apresenta-se como um ambiente complexo, sendo dividido em duas grandes áreas, em função das diferentes arquiteturas utilizadas, o aplicativo do jogo, chamado de Front-end, e o servidor, que é utilizado como um repositório de informações, chamado de Back-end.

A estrutura do simulador é apresentada na Fig. 2, e suas etapas serão especificadas na sequência.

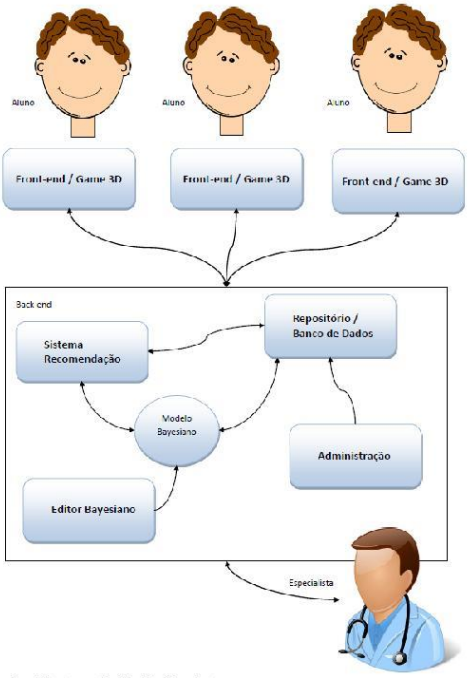


Figura 2 – Estrutura do Health Simulator

A. *Front-end*

O Front-end compõem a parte do simulador que será destinado aos alunos, apresentada em formato de um jogo sério, desenvolvido sob a plataforma da Unity, que permite que o jogo rode em diferentes plataformas e ambientes. O jogo possui cenários e personagens que representam profissionais da área da saúde e pacientes, reproduzindo da forma mais fiel possível um ambiente real.

O ambiente desenvolvido conta com diversos modelos de personagens (Fig. 3). São quatro categorias de personagens (médicos, pacientes, dentistas e enfermeiros), de ambos os gêneros, idades e etnias. Os cenários desenvolvidos (Fig. 4), são formandos por consultórios, hospitais das classes A, B ou C e um hospital do Sistema Único de Saúde (SUS). Destaca-se também o número elevado de *assets* (objetos) que foram desenvolvidos para compor os cenários. Exemplos de *assets* são cadeiras, mesas, poltronas, macas, estetoscópio, etc. Estes podem ser reaproveitados, compondo novos cenários de acordo com a necessidade.

Fig. 3. Modelos de enfermeiras e médicos nas etnias branco, negro e asiático.

Figura 4. Exemplo de cenário para consultório Classe C.

O projeto da interface do game está em fase final de desenvolvimento, a estrutura foi definida por turnos, onde, a cada turno, a *engine* envia informações ao Back-end, recebendo respostas da mesma. A troca de informações entre o jogo e o servidor ocorre de tempo em tempo, sendo representada como turno. A cada comunicação são inferidas novas variáveis em função das ações tomadas e, de acordo com o modelo bayesiano analisado. É apresentada uma resposta adequada, fornecendo ao jogador uma nova orientação.

B. *Back-end*

Para o *back-end*, tem-se uma estrutura complexa, sendo dividido em: Modelagem do conhecimento; Interface de administração e Sistema de recomendação, além da estrutura de comunicação.

A modelagem do conhecimento, primeira etapa, consiste em delimitar o conhecimento a ser representado pelo especialista. Para isto, ele faz uso de uma diretriz clínica, uma forma sistemática de orientação e delimitação do conteúdo a ser desenvolvida a partir de evidências, característica relatada como muito importante para um software educacional voltado ao ensino médico [27]. A partir deste momento, é necessário desenvolver um modelo estatístico de representação do conhecimento, através de uma rede bayesiana. A Fig. 5 apresenta uma rede bayesiana de cefaleia, baseada na diretriz clínica de cefaleia. Esta encontra-se disponível para acesso no site da Associação Médica Brasileira (http://www.projetodiretrizes.org.br/8_volume/16-Cefaleias.pdf). Cabe salientar que uma mesma Rede Bayesiana permite a geração de mais de cinquenta casos de estudo diferentes, dependendo da criatividade do professor.

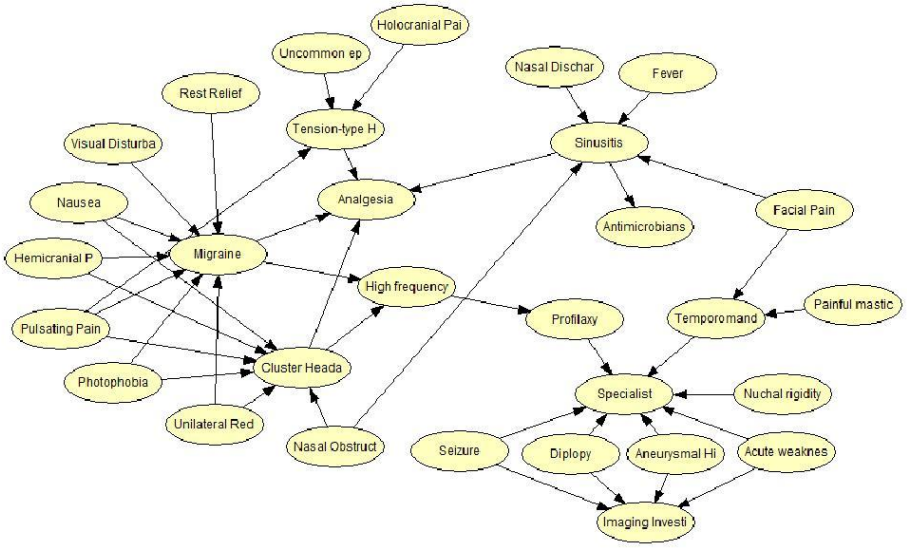


Fig. 5 – Rede Bayesiana de Cefaléia [28]

A interface de administração permite que o sistema seja gerenciado através da criação de usuários, cadastros principais (instituição, disciplina, avatar, tipo de exame, criação do caso clínico, entre outros).

A parte principal da interface de administração (Fig. 6), é um ambiente web onde o professor tem acesso aos sinais e sintomas presentes na rede, definidos na etapa anterior, possíveis históricos anteriores do paciente, bem como exames físicos e complementares. Conforme sua seleção, o sistema deve apresentar o desfecho do caso, apontando os diagnósticos e condutas, conforme as probabilidades são propagadas na rede, facilitando o processo e a elaboração de cada caso clínico. Esse processo pode ser observado a direita nos quadros Diagnóstico e Conduta. Com isso, o professor que desenvolverá os casos de estudo não necessita conhecer a rede bayesiana ou qualquer formalismo da área da informática.

Fig. 6. Cadastro de Avatar

Um caso clínico gerado no *Health Simulator* é armazenado no banco de dados, ficando disponível para utilização pelos alunos a partir do jogo através dos módulos de comunicação, e pelo módulo do sistema de recomendação, que representa o suporte pedagógico ao ambiente.

V. Proposta de uso de Sistemas de Recomendação no Health Simulator

O sistema de recomendação está em fase de prototipagem, e tem como objetivo auxiliar pedagogicamente o aluno, dando orientações mais amplas do que o simples resultado da inferência. Sua responsabilidade é muito maior, ele deve atuar de forma ativa durante a partida, analisando cada passo dado pelo aluno, coletando informações pertinentes ao seu desempenho e registrando estas informações para que sirvam de base de conhecimento para fornecer recomendação, colaborando com indicações pertinentes a sua fase de aprendizado. Desta forma, o objetivo é que este possa auxiliar o aluno na sua construção do conhecimento.

Manouselis et al. [19] apontam que os objetivos da recomendação no contexto da educação podem diferir daqueles nas demais áreas que fazem uso desse recurso, trazendo assim a necessidade de se mapear esses requisitos particulares, buscando a construção de ferramentas mais adequadas ao uso na educação. Estes autores mostram que tarefas recorrentes em sistemas de recomendação, tais como encontrar os melhores itens para um usuário ou prever quão relevante um determinado item pode ser para um usuário, são também aplicáveis ao contexto da educação. Entretanto, quando se busca a construção do aprendizado em um indivíduo, é possível explorar outros recursos que não estão presentes em sistemas de recomendação tradicionais. A Tabela I apresenta algumas tarefas que podem ser abordadas em sistemas de recomendação voltados à educação, conforme apontado por [19].

Tabela I
Tarefas que podem ser abordadas em Sistemas de Recomendação [19]

Tarefa	Descrição	Uso na educação
Encontrar novos itens	Recomendação de itens recém inseridos no sistema	Apontar itens recentes ou de conteúdo controverso
Encontrar parceiros	Recomendar usuários com interesses similares	Apontar estudantes de uma mesma classe, com preferências ou dificuldades similares
Encontrar bons caminhos	Recomendar uma sequência de itens	Apontar uma sequência de itens a serem estudados, buscando aprendizado em uma determinada área

É proposto aqui um algoritmo de recomendação híbrido para uso no Health Simulator. O principal objetivo do algoritmo é analisar o desempenho do usuário durante a partida, identificar o grau de entendimento no conteúdo apresentado e, com base nessa análise, recomendar os próximos materiais ou casos clínicos, de forma que o usuário exercite suas fraquezas, reforce seu conhecimento ou adquira novos conhecimentos.

A abordagem toma como entrada uma diretriz clínica, representada na forma de uma rede bayesiana, como descrito anteriormente. O algoritmo deve recomendar ao aluno uma maneira de explorar a diretriz clínica, indicando áreas da rede bayesiana na qual ele ainda apresenta dificuldade de aprendizado ou que não tenha sido estudada. Ou seja, se busca notar áreas da rede bayesiana da diretriz clínica na qual o aluno já tem forte conhecimento e então lhe induzir a casos similares que estejam em áreas que o aluno ainda não domina. Um exemplo disso seria, ao notar que o aluno tem experiência no diagnóstico de enxaqueca, recomendar-lhe algo que o faça

explorar mais a diretriz clínica, como um caso de sinusite. Busca-se, com isso, abordar o terceiro item listado por [19] na Tabela I, visto que o aluno terá uma indicação de caminho de aprendizagem personalizada de acordo com seus conhecimentos.

Dessa forma, vê-se a necessidade de dois passos para a recomendação. O primeiro é a identificação de quão longe na rede bayesiana se pode avançar. Ou seja, tendo em vista que o usuário ativo está estudando um caso clínico *x*, quanto é possível afastar-se de *x* para proporcionar ao aluno o melhor aprendizado? Essa identificação se dá através de uma avaliação do conhecimento do aluno no diagnóstico do caso clínico *x*. Tomando como partida o nodo da diretriz clínica que representa o diagnóstico de *x*, a rede bayesiana será expandida com o uso do algoritmo de busca em largura com um limite de nível. Esse limite de nível será determinado pela avaliação do conhecimento do aluno.

O segundo passo é identificar, dentro do grupo de itens nos níveis recomendáveis ao aluno, quais lhe trariam maior aprendizado. Para tal, decidiu-se utilizar o algoritmo descrito por [25]. Nesse algoritmo se identificam os alunos com melhor aprendizado, chamados de *good learners*. Um *good learner* é um aluno com aproveitamento maior que 80%. A avaliação desses alunos à itens e materiais é, então, levada em consideração para a recomendação. Entende-se que os itens melhor avaliados por bons alunos são mais indicados para os demais colegas. Essa técnica foi avaliada pelos autores e mostrou um aumento de 12,16% no desempenho dos alunos em comparação às técnicas tradicionais de recomendação [25]. Esse mesmo algoritmo pode ainda suprir as necessidades do segundo item descrito na Tabela I [19], recomendado um colega de estudos – um dos *good learners* – ao usuário ativo.

Espera-se que com o crescimento do uso do simulador, novos casos clínicos sejam gerados a partir da rede, que se tornará, eventualmente, bastante abrangente. Por isso a importância de um sistema de recomendação que deve não somente mostrar bons casos de estudo a um usuário, mas também guiá-lo pela rede bayesiana da diretriz clínica, proporcionando assim, maior oportunidade de aprendizado ao aluno.

Logo, a primeira fase da recomendação deve filtrar os casos clínicos tomando como base o nível do aluno e a área do caso de estudo. Como o ponto de partida para essa etapa da recomendação é baseado no conteúdo do caso inicial e então faz-se a seleção de outros casos similares, classificamos essa primeira etapa como uma recomendação baseada em conteúdo. Depois, a segunda etapa deve realizar um segundo filtro, buscando dentre os casos apontados pela primeira etapa os que tem maior aceitação da comunidade de estudantes, ou seja, um filtro colaborativo.

VI. Considerações Finais

Este artigo apresentou o simulador do tipo paciente virtual Health Simulator, que faz uso de dois formalismos de inteligência artificial: redes bayesianas e sistemas de recomendação. Para entendimento do seu funcionamento, os dois foram apresentados e exemplificados.

O Health Simulator foi dividido em duas partes distintas: *Front-end* e *Back-end*. O *front-end* foi desenvolvido no formato de um jogo sério, contendo mais de 150 personagens modelados e diversos cenários. Este está em fase de finalização e testes. O *back-end*, foco deste artigo, é composto pelo sistema que armazena o conhecimento do especialista, o sistema de recomendações e o sistema a ser utilizado pelos professores para a criação dos casos clínicos.

As redes bayesianas são utilizadas para o armazenamento do conhecimento do especialista. Essa representação parte das diretrizes clínicas a serem utilizadas pelos alunos para aprendizado em saúde. Essa mesma rede bayesiana é então empregada no sistema de recomendação, para seleção dos materiais a serem recomendados ao usuário ativo.

O simulador encontra-se em fase final de desenvolvimento. O sistema de recomendação a ser empregado está ainda em fase de prototipagem. Faz-se necessário ainda a identificação de como se dará a análise do nível de conhecimento do aluno em uma determinada área, bem como definir como isso será traduzido para uma representação numérica a ser utilizada como limite de nível para a busca em largura no grafo da rede bayesiana.

Após a implementação completa será possível realizar as primeiras avaliações com as duas etapas da recomendação: a busca em largura e o algoritmo descrito por [25]. Acredita-se que o sistema de recomendação proposto trará um aumento do desempenho dos alunos, se comparado ao aprendizado sem recomendação ou às técnicas comuns baseadas em conteúdo ou de filtragem colaborativa.

Referencias

[1] M. BOTEZATU, H. HULT, U. G. FORS. Virtual patient simulation: what do students make of it? A focus group study. BMC Medical Education, v. 10, n. 91, 2010.

[2] J. HIGGS, M. A. JONES, S. LOFTUS, H. CHRISTENSEN. Clinical reasoning in the health professions. 3.ed., China: Elsevier, 2008.

[3] M. FORTE, W. L. de SOUZA, A. F. PRADO. Portfólio Eletrônico Ubíquo no Aprendizado de Medicina. In: Congresso Brasileiro de Informática em Saúde – CBIS, 12., 2010, Recife/PE. Anais do XII Congresso Brasileiro de Informática em Saúde – CBIS 2010, São Paulo: Sociedade Brasileira de Informática em Saúde, 2010, p.1-6.

[4] S. D. BROOKFIELD. The power of critical theory: liberating adult learning and teaching. 1.ed., San Francisco: Jossey-Bass, 2005, 414p.

[5] A. HOLZINGER, M. D. KICKMEIER-RUST, S. WASSERTHEURER, M. HESSINGER, M. Learning performance with interactive simulations in medical education: Lessons learned from results of learning complex physiological models with the HAEMOdynamics SIMulator. Computer & Education, v. 52, n. 2, p.292-301, 2009.

[6] S. J. SMITH, C. J. ROEHRS. High-fidelity simulation: Factor correlated with nursing student satisfaction and self-confidence. Nursing Education Perspectives, v. 30, n. 2, p.77-78, 2009.

[7] P. R. BARROS, S. C. CAZELLA, M. B. BEZ, C. D. FLORES, A. DAHMER, J. B. MOSSMANN, J. M. FONSECA, V. MARONI. Um Simulador de Casos Clínicos Complexos no Processo de Aprendizagem em Saúde. RENOTE. Revista Novas Tecnologias na Educação, v. 12, p. 1-11, 2012.

[8] A. ZIV, S. BEN-DAVID, M. ZIV, M. Simulation Based Medical Education: an opportunity to learn from errors. Medical Teacher, v. 27, n. 3, p.193-199, 2005.

[9] J. P. KINCAID. Simulation in Education and Training. In: Modeling and Simulation: Theory and Applications, 1. ed., Boston: Kluwer, 2004, Cap.19, p. 273-280.

[10] P. BRADLEY. The history of simulation in medical education and possible future directions. Medical Education, v. 40, n. 3, p.254-262, 2006.

[11] S. RUSSELL, P. NORVIG. Artificial Intelligence: A modern approach. 3rd ed. New Jersey: Prentice-Hall. 1995. 1132 p.
[12] R. L. MARQUES, I. DUTRA. Redes Bayesianas: o que são, para que servem, algoritmos e exemplos de aplicações. Coppe Sistemas – Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2002.
[13] N. J. NILSSON. Artificial intelligence: a new synthesis. San Francisco: Morgan Kaufmann, 1998. 513 p.
[14] W. WIEGERINCK, B. KAPPEN, W. BURGERS. Bayesian networks for expert systems: Theory and practical applications. In: Interactive collaborative information systems. Springer Berlin Heidelberg, 2010. p. 547-578.
[15] L. J. SEIXAS et al. An architecture for an intelligent learning environment with a constructivist approach. ITS2002. San Sebastian, Spain. 2002.
[16] M. R. FAZIO. Previsão de avaliações em sistemas de recomendação para nichos de mercado. Ph.D. Dissertation, COPPE, UFRJ, Rio de Janeiro, RJ, Brazil, 2013.
[17] D. GOLDBERG, D. NICHOLS, B. M. OKI, D. TERRY. Using collaborative filtering to weave an information tapestry. Commun. ACM, vol. 35, no. 12, pp. 61-70, Dec. 1992.
[18] G. R. LOPES. Avaliação e recomendação de colaborações em redes sociais acadêmicas. M.S. thesis, PPGC, UFRGS, Porto Alegre, RS, Brazil, 2012.
[19] N. MANOUSELIS, H. DRACHSLER, R. VUORIKARI, H. HUMMEL, R. KOPER. Recommender systems in technology enhanced learning. In: Recommender Systems Handbook, 1st ed., F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, Ed. New York: Springer, 2010, pp. 387 – 409.
[20] G. G. TESTA. Uma abordagem híbrida para recomendação de parceiros em ambientes virtuais colaborativos de composição musical. Ph. D. dissertation, Prog. de Pós-Grad. em Comp., UFRGS, Porto Alegre, RS, Brazil, 2013.
[21] A. S. LAMPROPOULOS, G. A. TSIHRINTZIS. Review of previous work related to recommender systems. In: Machine Learning Paradigms, Applications in Recommender Systems, 1st ed., New York: Springer, 2015, pp. 13 – 30.
[22] B. SARWAR, G. KARYPIS, J. KONSTAN, J. RIEDL. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web. ACM, pp. 285-295.
[23] H. WU, Y. PEI, B. LI, Z. KANG, X. LIU, H. LI. Item recommendation in collaborative tagging systems via heuristic data fusion. Knowledge Based Systems, v. 75, pp. 124 – 140, Feb., 2015.
[24] F. RICCI, L. ROKACH, B. SHAPIRA, P. B. KANTOR. Introduction to recommender systems handbook. In: Recommender systems handbook, 1st ed., New York: Springer, 2010, pp. 1 – 38.
[25] K. I. GHAUTH, N. A. ABDULLAH. The effect of incorporating good learners' ratings in e-Learning recommender system. Educational Technology & Society, v. 14, n. 2, pp. 248 – 257, Feb., 2012.
[26] P. SCHUYTEMA. Design de Games: uma abordagem prática. São Paulo: Cengage Learning, 2008. p. 447.
[27] V. JHA, S. DUFFY. 'Ten golden rules' for designing software in medical education: results from a formative evaluation of DIALOG. Med Teach. 24(4), 417–421. 2002. DOI: 10.1080/0142159020145798.
[28] C. F. FLORES, J. M. FONSECA, M. R. BEZ, A. RESPÍCIO, H. COELHOMethod for Building a Medical Training Simulator with Bayesian Networks: SimDeCS. Studies in health technology and informatics, 207, 102-114. 2013.

Marta Rosecler Bez: Doutora em Informática na Educação pela Universidade Federal do Rio Grande do Sul (2013). Mestre em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul (2001) e graduada em Tecnólogo Em Processamento de Dados pela Universidade do Vale do Rio dos Sinos (1991). Atualmente é professora da Universidade Feevale nos cursos de Ciência da Computação, Sistemas de informação e no Mestrado de Indústria Criativa. Tem experiência na área de Ensino de Medicina com o uso de tecnologias, atuando principalmente nos seguintes temas: processamento de imagens, SAD, reconhecimento de padrões, tecnologias na educação. Possui projetos na área de desenvolvimento de material pedagógico mediado por tecnologia.

Paulo Ricardo Muniz Barros: Doutorando em Computação Aplicada na Universidade do Vale dos Sinos, Mestre em Ciências da Saúde: Educação e Informática em Saúde pela Universidade Federal de Ciências da Saúde de Porto Alegre (2013), Bacharel em Sistemas de Informação pela Universidade Feevale (2009). Atualmente é Docente na Universidade Feevale e Arquiteto de Software na RJM Informática Ltda, atuando com novas tecnologias, desenvolvimento de software e ministrando disciplinas ligadas ao desenvolvimento de Software/Jogos. Tem experiência na área de Ciência da Computação, com ênfase em Inteligência Artificial, Gerência de Projetos, Simuladores virtuais e Computação Gráfica.

Alessandro Peixoto de Lima: Mestre em Design Virtual pela UFRGS (2015), Pós-graduado em MBA Comunicação Estratégica e Branding pela Universidade Feevale (2011) e Graduado em Design Gráfico pela Universidade Uniritter Laureate (2010). Atua há mais de 13 anos na indústria de Computação Gráfica (CG), sendo 10 somente na indústria de Jogos Digitais com participação em mais de 30 projetos de jogos nacionais e internacionais publicados por empresas como Atari, Apple e Cartoon Network, dedicando suas habilidades ao desenvolvimento arte em jogos digitais. Hoje é docente na Universidade Feevale pelo curso de Jogos Digitais, pesquisador escritor de livros sobre arte tridimensional.

Diego Reidel: Graduando em Ciência da Computação pela Universidade Feevale. Atualmente é Desenvolvedor de Soluções na SAP Brasil, atuando com uma das ferramentas de comércio eletrônico mais utilizadas no mundo. Tem experiência com desenvolvimento de aplicações mobile, comércio eletrônico e educação à distância.

Fernando Alex Helwanger: Graduando em Ciência da Computação pela Universidade Feevale. Atualmente é Desenvolvedor de Soluções na Secullum Ponto e Acesso. Tem experiência em na área de Ciência da Computação com desenvolvimento de aplicações, participante de projetos de pesquisa com uso de simuladores virtuais com ênfase em Inteligência Artificial.