Um método para implementar redes bayesianas baseadas em nós ranqueados

João Batista Nunes Bezerra [1], Renan Willamy Bezerra Barbosa [2], Mirko Perkusich [3]

[1] lockenunes@gmail.com. [2] renawillamy2@gmail.com. [3] mirko.perkusich@ifpb.edu.br. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campus Monteiro.

RESUMO

Recentemente, redes Bayesianas estão se tornando populares para auxiliar na tomada de decisões. No entanto, ainda há desafios para sua aplicação prática em problemas de larga escala. Um dos desafios refere-se à definição das funções de probabilidade. Na literatura, utiliza-se o conceito de nós ranqueados baseado em distribuição Normal truncada para simplificar a definição de funções de probabilidade a partir do conhecimento de especialistas. Por outro lado, na literatura, não há detalhes de uma solução completa para nós ranqueados. Atualmente, esta solução está disponível apenas em uma ferramenta comercial. A contribuição principal deste trabalho refere-se à demonstração detalhada de um método para definição de funções de probabilidade para nós ranqueados, apresentando os passos necessários para misturar distribuições Normais truncadas e converter a distribuição resultante para uma tabela de probabilidade dos nós. Para validação, os resultados da nossa solução foram comparados com a solução disponível no mercado, sendo equivalentes, dentro de uma margem de erro de 5%. De acordo com os testes realizados, a solução proposta alcançou performance melhor que a solução comercial existente.

Palavras Chave: Distribuição Normal truncada. Modelo probabilístico. Nós ranqueados. Rede bayesiana. tabela de probabilidade dos nós.

ABSTRACT

Recently, Bayesian networks are increasingly being used to assist on decision-making. However, there are obstacles to its practical application in real world scenarios. In the literature, there is one solution that uses the concept of ranked nodes, which are based on the concept of truncated Normal distribution, to simplify the definition of probability functions through the elicitation of knowledge from experts. On the other hand, there are no details of a complete solution to implement ranked nodes. Nowadays, the solution is only available through a commercial tool. Our contribution is to present a method to define probability functions for ranked nodes by showing the steps to mix truncated Normal distributions and convert the resulting distribution into a node probability table. For validation purposes, we compared the results calculated using our method with the solution from a commercial tool. The results were promising, because the calculated data were equivalent, given a 5% margin of error. According to our tests, our solution has better performance than the existing available commercial tool.

Keywords: *Bayesian network. Node probability table. Probabilistic model. Ranked nodes. Truncated Normal distribution.*

1 Introdução

Rede bayesiana é um modelo matemático que representa as relações probabilísticas de variáveis aleatórias de forma gráfica e numérica por meio do teorema de Bayes. Atualmente, devido à evolução da capacidade computacional que possibilita o cálculo de redes bayesianas complexas, esta técnica está se tornando popular para auxílio na tomada de decisões (FENTON; NEIL; CABALLERO, 2007). Alguns exemplos de área de aplicação de redes bayesianas são: gerenciamento de projetos de desenvolvimento de software (PERKUSICH; ALMEIDA; PERKUSICH, 2013), projetos de engenharia de larga escala (LEE; PARK; SHIN, 2009) e previsão de sucesso de projetos de inovação (DE MELO; SANCHEZ, 2008).

Por outro lado, ainda há desafios relacionados à construção de redes bayesianas, os quais podem ser subdivididos em dois subproblemas: (i) definir o grafo acíclico direcionado e (ii) construir as Tabelas de Probabilidade dos Nós (TPN). Neste trabalho, foca-se no segundo subproblema.

Nos casos nos quais há um banco de dados com informações suficientes para o subproblema em questão, é possível automatizar o processo de construção da rede bayesiana, utilizando-se *batch learning* (HECKERMAN, 1995). Infelizmente, na prática, na maioria dos casos, não há dados suficientes, ou seja, é necessário coletar dados de especialistas e, manualmente, definir as TPN. Dado que a complexidade para definição de TPN cresce exponencialmente, para redes bayesianas de larga escala, torna-se impraticável a definição manual de todas as combinações possíveis das TPN (FENTON; NEIL; CABALLERO, 2007).

Neste artigo, é apresentado um método para definir TPN para nós ranqueados usando a função *WMEAN,* proposta por Fenton, Neil e Caballero (2007). O método foi implementado com funções em *C++*, e experimentos foram realizados para avaliar sua eficácia e eficiência em comparação com o *AgenaRisk*[1], única ferramenta existente com suporte a nós ranqueados. A implementação do método é disponibilizada por meio de um código *open source*. Dessa forma, os resultados deste trabalho contribuem para a pesquisa e aplicação de sistemas especialistas com redes bayesianas baseadas em nós ranqueados. Esclarecemos que, dos quatro tipos de funções para modelar nós ranqueados apresentados em Fenton, Neil e Caballero (2007), a solução aqui apresentada considera apenas um deles.

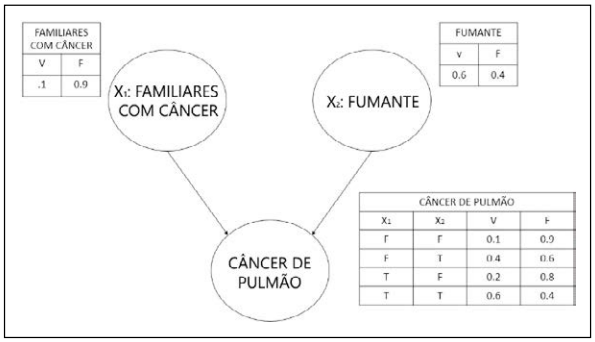
Este artigo é organizado da seguinte forma: na Seção 2, apresenta-se o conceito de redes bayesianas; na Seção 3, os resultados de pesquisas para definir TPN com foco em nós ranqueados; na Seção 4, detalhes acerca do método proposto e do algoritmo de análise de rede bayesiana; na Seção 5, os resultados são apresentados e discutidos; e, na Seção 6, conclusões, limitações e prospecção de trabalhos futuros.

2 Redes bayesianas

Redes bayesianas pertencem à família de modelos gráficos probabilísticos e são utilizadas para representar o conhecimento acerca de um domínio com incerteza. A rede bayesiana B refere-se a um grafo acíclico direcionado que representa a distribuição conjunta de probabilidade sobre um conjunto de variáveis aleatórias V (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997). A rede é definida pelo par B = {G, Θ}. No grafo acíclico direcionado G, as variáveis aleatórias são representadas pelos nós $X_1,\ldots,X_n$, e as dependências diretas entre as variáveis são representadas pelos arcos. As funções de probabilidade (i.e., tabelas de probabilidade dos nós) são representadas por Θ. Esse conjunto contém o parâmetro $\theta xi|\pi i = PB(xi|\pi i)$ para cada xi em Xi condicionado por πi, o conjunto de parâmetros de Xi em G. A distribuição conjunta de probabilidade definida por B em V é apresentada na Equação 1. Um exemplo de rede bayesiana é apresentado na Figura 1.

$$P_B(X_1,\ldots,X_n) = \prod_{i=1}^{n} P_B(x_i|\pi_i) = \prod_{i=1}^{n} \theta_{Xi|\pi i} \quad (1)$$

Figura 1 – Exemplo de Rede Bayesiana



Fonte: Elaborada pelo autor.

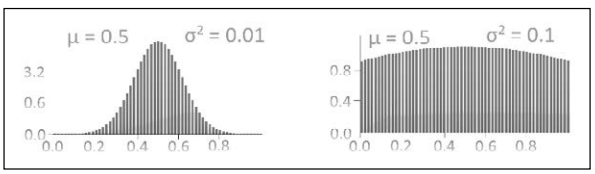
Neste caso, modela-se a probabilidade de uma pessoa ter câncer de acordo com duas variáveis: "Familiares com câncer" (X_1) e "Fumante" (X_2). O sentido das setas indica a dependência para definir as funções de probabilidade, e é assumido que todas as variáveis são *booleanas*. Neste caso, dado que "Câncer" é apontado por X_1 e X_2, sua função de probabilidade é composta de probabilidades para todas as combinações de estados possíveis de X_1 e X_2.

3 Trabalhos relacionados

O problema de definir TPN para redes bayesianas de larga escala a partir do conhecimento de especialistas já foi discutido no passado. *Noisy-OR* (HUANG; HENRION, 1996) e *Noisy-MAX* (DÍEZ, 1993) são dois métodos populares para codificar conhecimento em TPN. A desvantagem do Noisy-OR é que apenas se aplica a nós booleanos. De acordo com Fenton, Neil e Caballero (2007), a desvantagem do *Noisy-MAX* é que não modela a extensão de relacionamentos necessários para redes bayesianas de larga escala.

Uma solução para este problema foi apresentada por Fenton, Neil e Caballero (2007). Neste trabalho, apresenta-se o conceito de nós ranqueados, variáveis aleatórias ordinais representadas em uma escala contínua, ordenada monotonamente no intervalo [0, 1]. Por exemplo, para a escala ordinal ["Ruim", "Moderado", "Bom"], "Ruim" é representado pelo intervalo [0, 1/3]; "Moderado", pelo [1/3, 2/3]; e "Bom", pelo [2/3, 1]. Este conceito é baseado na distribuição Normal duplamente truncada (TNormal). Esta distribuição é composta de quatro parâmetros: μ, média (i.e., tendência central); σ^2, variância (i.e., confiança nos resultados); a, limite inferior (i.e., 0); e, b, limite superior (i.e., 1). Com esta distribuição, flexibiliza-se a modelagem ao se permitir a representação de uma variedade de curvas (i.e., relacionamentos entre variáveis), tais como a distribuição uniforme, quando $\sigma^2=\infty$, e curvas enviesadas, quando $\sigma^2 = 0$. Um exemplo de distribuição TNormal com mesma μ, mas com σ^2 diferente, é apresentado na Figura 2.

Figura 2 – Exemplo de distribuições TNormal



Fonte: Elaborada pelo autor.

Na solução apresentada por Fenton, Neil e Caballero (2007), μ é definida por uma função ponderada dos nós-pai. Há quatro tipos de função: média ponderada (*WMEAN*), mínimo ponderado (*WMIN*), máximo ponderado (*WMAX*) e uma mistura das funções clássicas *MIN* e *MAX* (*MIXMINMAX*). Na prática, estas expressões definem a tendência central do nó-filho para a combinação de estados dos nós-pais. O peso de cada nó-pai, que denota a influência exercida do nó-pai sobre o nó-filho, deve ser definido por uma constante w, na qual $w \in \mathbb{N}$.

A solução proposta por Fenton, Neil e Caballero (2007) foi validada por meio de estudos de caso em diferentes contextos no mundo real, tais como gerenciamento de recursos humanos em projetos de *software* (FENTON *et al.*, 2004), previsão da qualidade de *software* (FENTON *et al.*, 2007), controle de tráfego aéreo (NEIL; MALCOLM; SHAW, 2003) e gerenciamento operacional (NEIL; FENTON; TAILOR, 2005). A abordagem proposta também foi usada extensivamente em aplicações comerciais que não foram citadas pelos autores, devido a termos de confidencialidade. Concluiu-se, portanto, que, com estas funções, atende-se à necessidade de modelagem para definir TPN. Exemplos utilizando-se as funções *WMEAN*, *WMIN*, *WMAX* e *MIXMINMAX* são apresentados na Figura 3. Para *MIXMINMAX*, foi definido peso maior para a função *MAX*.

Por outro lado, em Fenton, Neil e Caballero (2007), a solução foi apresentada superficialmente, não contendo os detalhes necessários para, na prática, implementá-la. Além disso, atualmente, a solução está disponibilizada apenas por meio de uma ferramenta comercial, *AgenaRisk*.

Em Laitila (2013), algoritmos para implementar as funções *WMEAN*, *WMAX*, *WMIN* e *MIXMINMAX* foram apresentados. Neste artigo, complementa-se o trabalho de Fenton, Neil e Caballero (2007) e Laitila (2013), apresentando em detalhes um método para implementar a solução em *C++*, e dados de comparação entre a solução proposta e o *AgenaRisk*.

Figura 3 – Utilização das funções WMEAN, WMIN , WMAX e MIXMINMAX

Fonte: Elaborada pelo autor.

4 Método para construção de funções de probabilidade para nós ranqueados

Para construir a função de probabilidade (i.e., TPN) de um nó qualquer, independentemente do tipo, é necessário definir valores para todas as combinações possíveis de estados dos nós-pai. Com nós ranqueados, minimiza-se este esforço utilizando as funções definidas por Fenton, Neil e Caballero (2007).

Dada a complexidade exponencial do problema de definir TPN para redes Bayesianas de larga escala, é necessário aplicar estratégias de otimização. O algoritmo do método é apresentado na Figura 4, na qual temos os seguintes elementos dispostos: *repositório[]*, um vetor para armazenar as amostras dos estados-base para nós-pais; *pais[k]*, um vetor para armazenar referências para os nós-pais de cada nó-filho, em que *k* é o número de nós-pais; *estados[k]*, um vetor para armazenar os estados de cada nó-ranqueado; *m*, o número de combinações de estados possíveis dado o conjunto de nós-pais; *distribuição[k]*, um vetor para armazenar a distribuição resultante para cada combinação de estados dos nós-pais.

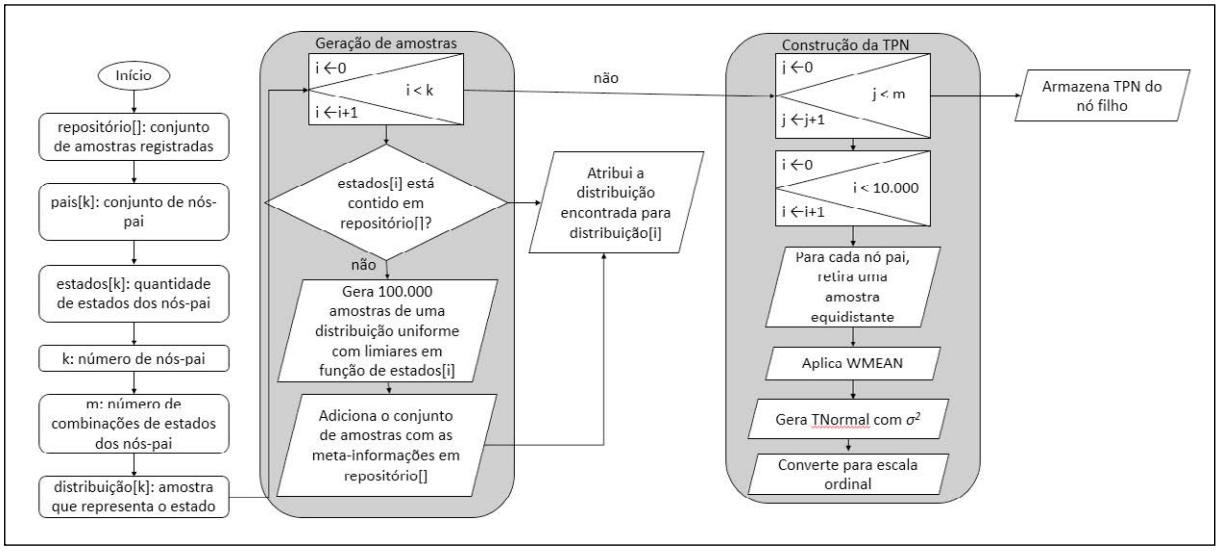
Primeiramente, registraram-se em memória (i.e., no *repositório[]*) as distribuições de amostras que representam os estados-base (i.e., 100% de probabilidade) possíveis dos nós-pais. Por exemplo, para um nó ranqueado composto dos estados ["Ruim", "Moderado", "Bom"], os estados-base são: 100% "Ruim", 100% "Moderado" ou 100% "Bom". Para se chegar a tais configurações, coletaram-se amostras de uma distribuição uniforme, com limites, em função

dos limiares da configuração em questão. Por exemplo, para a configuração 100% "Bom", coletaram-se amostras de uma distribuição uniforme limitada no intervalo [2/3, 1]. Empiricamente, foi definido que amostras com 100.000 elementos eram o suficiente para ter uma margem de erro menor que 0,1%. Cada amostra é registrada com metainformações acerca da sua configuração (i.e., número de estados e μ).

Os dados registrados em repositório[] são utilizados quando é necessário gerar amostras para um nó, ou seja, as amostras de estados-base são geradas apenas uma vez e reutilizadas quando possível. Neste processo, se ganha eficiência. Na Figura 4, esta lógica é representada pelo bloco "Geração de amostras".

O próximo passo, representado pelo bloco "Construção da TPN", na Figura 4, consiste em, para cada combinação de estados dos nós-pai, misturar as distribuições probabilísticas utilizando-se amostras equidistantes, coletadas, aleatoriamente, de cada nó-pai. Foi definido empiricamente que 10.000 amostras equidistantes são o suficiente para ter uma margem de erro menor que 0,1%. As amostras são misturadas utilizando-se a função WMEAN apresentada por Fenton, Neil e Caballero (2007).

Figura 4 – Algoritmo do método de definição de TPN para nós ranqueados



Fonte: Elaborada pelo autor.

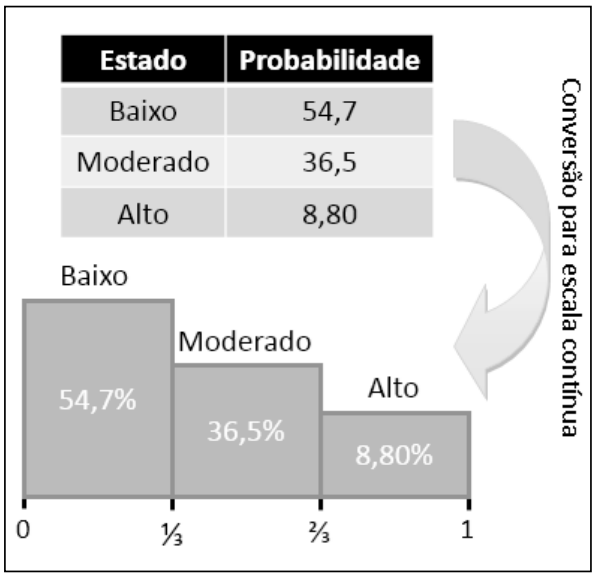
Mais detalhes sobre a implementação de WMEAN são apresentados na Seção 4.2. A distribuição resultante é utilizada, juntamente com a variância introduzida, como parâmetro para gerar uma nova TNormal. A distribuição resultante é convertida para a escala ordinal, representando uma coluna da TPN do nó-filho. Ao final deste passo, ao misturar todas as combinações possíveis, a TPN completa do nó-filho estará construída.

4.1 Conversão para escala contínua

Na prática, para aplicar o algoritmo apresentado na Figura 4, é necessário converter os dados da TPN de cada nó-pai da escala ordinal para a contínua, limitada em *[0, 1]*. Um exemplo de conversão é apresentado na Figura 5.

Representa-se a escala contínua como um conjunto de distribuições uniformes: □(0,1) = ρbaixo *□(0, ⅓) ∪ ρmoderado *□(⅓, ⅔) ∪ ρalto *□(⅔, 1), em que ρ é a densidade da distribuição. Para o caso apresentado na Figura 5, o conjunto de distribuições uniformes é composto pela união de três distribuições uniformes: □(0,1) = 54,7*□(0, ⅓) 36,5*□(⅓, ⅔) 8,80*□(⅔, 1), em que ρ é a densidade da distribuição uniforme. Numericamente, esta união é calculada a partir de amostras. Empiricamente, definiu-se que amostras com 100.000 elementos representam as distribuições com a margem de erro esperada. Por outro lado, dado que uma TNormal pode representar uma distribuição uniforme, na prática, é possível utilizá-la na modelagem.

Figura 5 – Exemplo de conversão de TPN na escala ordinal para escala contínua



Fonte: Elaborada pelo autor.

Desta forma, para representar a TPN apresentada na Figura 5, utilizam-se 54.700 amostras aleatórias de □(0, ⅓), 36.500 amostras aleatórias de □(⅓, ⅔) e 8.800 amostras aleatórias de □(⅔, 1). Ao final do processo, os dados coletados são aleatoriamente ordenados para representar □(0, 1).

4.2 Mistura das amostras

Depois da conversão para escala contínua, gerando-se amostras que representam distribuições de probabilidade, é necessário misturar as distribuições (i.e., amostras). Para tal processo, seleciona-se a função ponderada que melhor descreva as relações probabilísticas entre os nós-pai e o nó-filho. Neste trabalho, nos limitamos à função *WMEAN* que calcula a média ponderada dos estados dos nós-pai.

Para isso, aleatoriamente, retira-se um elemento equidistante das amostras dos nós-pai e calcula-se um elemento resultante, utilizando-se a função de mistura escolhida. Empiricamente, definiu-se que amostras com 10.000 elementos equidistantes são suficientes dentro da margem de erro esperada. Repete-se este passo até que 10.000 elementos das amostras de cada nó-pai sejam retirados. Depois, utiliza-se o conjunto de elementos calculados e σ^2 para gerar uma TNormal. Com este propósito, utilizou-se a biblioteca *TRUNCATED_NORMAL*[2].

4.3 Método para análise de redes bayesianas

Como dito anteriormente, redes bayesianas podem ser aplicadas em vários contextos, inclusive, no auxílio na tomada de decisões. Nem sempre, entretanto, seus usuários são capacitados para analisá-las, limitando sua aplicabilidade. Assim, para auxiliar na tomada de decisões, embora não seja necessário ter conhecimento profundo sobre rede Bayesiana, é importante fornecer, aos usuários, meios para analisar os resultados calculados pela aplicabilidade desse sistema. O processo implica, portanto, a definição de um algoritmo para identificar as variáveis de entrada que exercem maior influência sobre uma variável alvo, priorizando as de maior influência. O resultado obtido consiste de uma lista ordenada das variáveis de entrada que, se melhoradas, influenciarão a variável alvo de forma positiva.

De forma prática, o algoritmo é responsável por identificar todas as variáveis de entrada que exercem influência na variável alvo e realizar simulações para cada uma das variáveis de entrada, melhorando seu estado e calculando a distribuição resultante da variável alvo de acordo com a nova configuração. O passo final consiste em ordenar as variáveis de entrada de acordo com seu nível de influência na variável alvo.

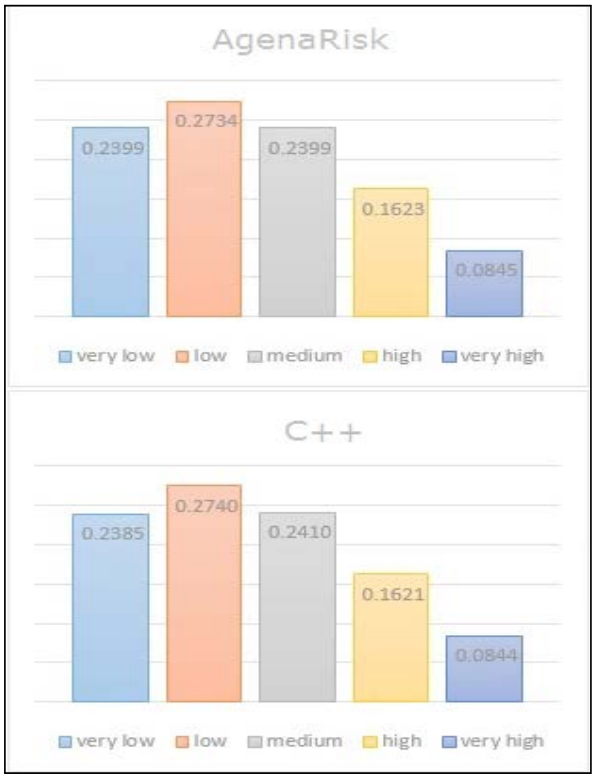
5 Resultados e discussão

O método foi implementado utilizando-se a linguagem de programação *C++*. A escolha da linguagem levou em consideração a existência da biblioteca *TRUNCATED_NORMAL*. A solução está disponibilizada no site de apoio[3] no formato de uma biblioteca.

Para validar a solução, esta foi comparada com o *AgenaRisk* com relação a dois fatores: similaridade dos resultados e performance.

Devido às diferenças na implementação da geração de amostras TNormal nos algoritmos da *TRUNCATED_NORMAL* e do *AgenaRisk*, para comparar a similaridade dos resultados, observou-se a necessidade de calibrar σ^2 para um valor equivalente (i.e., não necessariamente iguais) entre as soluções. Uma vez que ambos os algoritmos são capazes de gerar todos os tipos possíveis de curvas TNormal, isto não é considerado um problema. A calibração foi realizada apenas com o propósito de viabilizar a comparação dos resultados em cenários de teste equivalentes.

Figura 6 – Resultados para μ = 0,3 e σ2 = 0,15 no *AgenaRisk* e μ = 0,3 e σ2 = 0,39



Fonte: Elaborada pelo autor.

Dado que é recomendado melhorar o grafo da rede bayesiana, caso haja nós com mais do que três nós-pai (FENTON; NEIL, 2012), a similaridade dos resultados das soluções foi comparada em função de duas estruturas: um nó (i.e., caso mais simples), e um nó com três pais (i.e., caso mais complexo). Para a estrutura de um nó, foram executados dez cenários de teste, nos quais os valores de μ e σ^2 foram definidos aleatoriamente para o *AgenaRisk*. Para cada caso, empiricamente, definiu-se o valor equivalente de σ^2 na solução implementada em *C++*. Os resultados de três cenários de teste são apresentados nas Figuras 6 e 7. Os demais são apresentados no site de apoio. Os resultados obtidos na distribuição probabilística neste cenário apresentaram uma diferença percentual média de 0,6% entre o *C++* e o *AgenaRisk*, conforme apresentado na Tabela 1.

Para a estrutura com um nó e três nós-pai, a avaliação foi realizada com onze cenários de teste, utilizando-se uma rede bayesiana com a configuração *D = 3A + 2B + C*, em que *D* é o nó-filho e *A*, *B* e *C* são os nós-pai. Além disso, foi considerado que cada nó é composto de cinco estados. Os dados de entrada foram definidos aleatoriamente. Para o *AgenaRisk* foi utilizado $\sigma^2 = 0{,}0005$; para o *C++*, $\sigma^2 = 0{,}053$. Os resultados de um cenário de teste, no qual as evidências de *A*, *B* e *C* são, respectivamente, "muito baixo", "médio" e "muito alto", são apresentados na Figura 8. Os demais resultados são apresentados no site de apoio. A distribuição probabilística de nós para este cenário apresentou uma diferença percentual média de 1,2%, conforme apresentado na Tabela 2.

Tabela 1 – Resultados (resumidos) referentes a geração de nó simples

AgenaRisk		*C++*		Diferença percentual
μ	σ^2	μ	σ^2	
0,1	0,007	0,3	0,0842	0,9%
0,3	0,15	0,3	0,39	0,3%
0,5	0,15	0,5	0,39	0,4%
0,8	0,15	0,8	0,39	0,9%
0,9	0,007	0,5	0,0842	0,5%

Tabela 2 – Distribuição probabilística de nó com 3 pais. D = 3A+2B+C com σ2 = 0,053 no *C++* e 5,0E-4 no *AgenaRisk*

Evidência de nó pai			Diferença percentual da distribuição resultante					
A	B	C	MB	B	M	A	MA	TOTAL
MA	M	MA	0%	0%	0,1%	0,1%	0,1%	0,3%
MA	M	MB	0%	0%	0,2%	0,3%	0,1%	0,6%
MB	M	MA	0,1%	0,1%	0,1%	0%	0%	0,3%
MB	M	MB	0,2%	0,3%	0,1%	0%	0%	0,6%
MA	MA	MA	0 %	0%	0%	1,4%	1,4%	2,7%
MA	MA	MB	0%	0%	0,1%	0,1%	0,1%	0,3%
MB	MB	MA	0,2%	0,3%	0,1%	0%	0%	0,6%
MB	MB	MB	1,2%	1,2%	0%	0%	0%	2,4%
M	M	M	0%	1,3%	2,3%	1,1%	0%	4,7%
M	MA	M	0%	0%	0,2%	0,3%	0,1%	0,6%
M	MB	M	0,1%	0,1%	0,1%	0%	0%	0,3%

MB = *muito baixo*; B = *baixo*; M = *médio*; A = *alto*; MA = *muito alto*; A, B, C = *nós-pai*

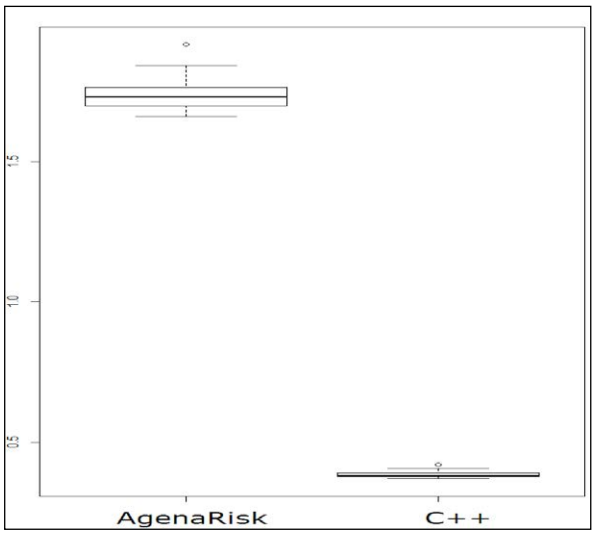
Para ambos os cenários, a diferença manteve-se na margem de diferença aceitável (i.e., abaixo de *5%*). Considerou-se abaixo de *5%* como margem de diferença aceitável, pois o objetivo da TPN é guiar um especialista na tomada de decisões. Então, na prática, como observado por Fenton, Neil e Caballero (2007), esta diferença percentual é irrelevante. Dessa forma, conclui-se que ambas as soluções possuem capacidade similar de modelagem.

Para avaliar a performance, foi comparado o tempo de execução para calcular uma TPN. Para tal, foi utilizada a estrutura de um nó com três nós-pai e variância fixa e equivalente nas duas soluções. Definiu-se $\sigma^2 = 0{,}0005$ no *AgenaRisk* e $\sigma^2 = 0{,}053$ no *C++*. A função de probabilidade do nó-filho foi calculada para vinte configurações, nas quais os coeficientes *k* dos nós-pai foram definidos aleatoriamente, em que $k \in [1, 5]$ e $k \in \mathbb{N}$. Os testes foram realizados em uma máquina *Intel Core i7-4510, 2GHz, 8GB* de *RAM* no sistema *Windows 10 Pro 64-bit*. O tempo médio de execução no *AgenaRisk* foi 1,738 segundos; no *C++* foi 0,3841 segundos. O *boxplot* com o tempo de execução das soluções é apresentado na Figura 8. Os dados coletados são apresentados no site de apoio.

Figura 7 – Resultados para μ = 0,9 e σ2 = 0,05 no *AgenaRisk* e μ = 0,9 e σ2 = 0,222

Fonte: Elaborada pelo autor.

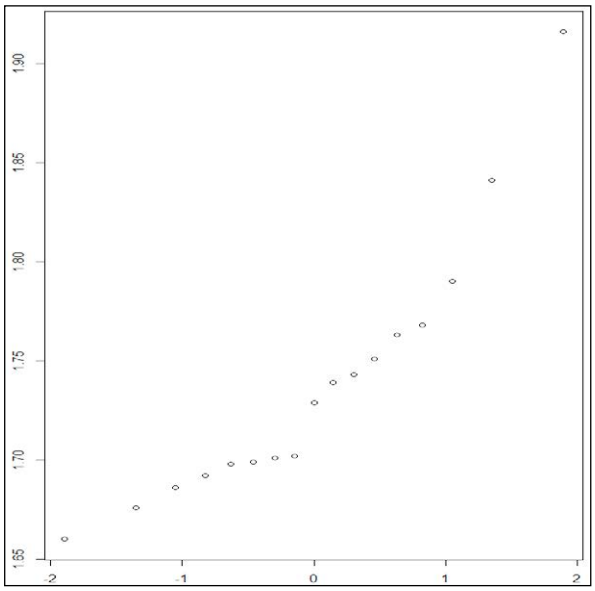
Figura 8 – *Boxplot* com o tempo de execução para os cenários de teste



Fonte: Elaborada pelo autor.

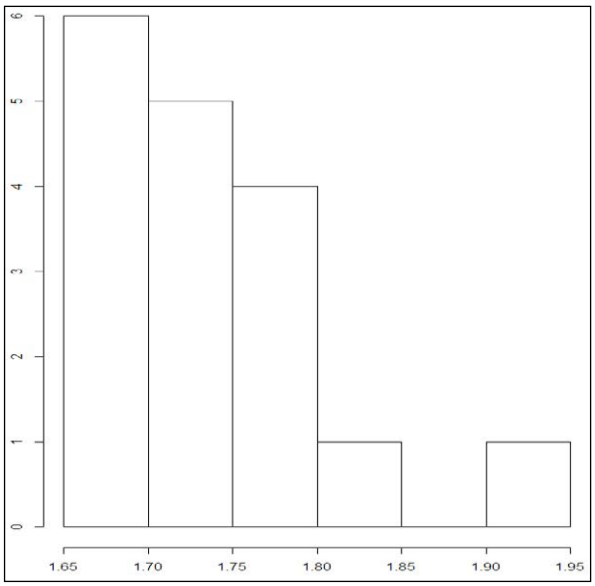
Ao analisar a Figura 8, visualmente, percebe-se que o tempo de execução do *C++* é menor que o de *AgenaRisk*, pois não há interseção entre os *boxplots*. De qualquer forma, para confirmar a suspeita, testes estatísticos foram realizados. Para isso, inicialmente, verificou-se a normalidade dos dados. Na Figura 9, o *QQ Plot* dos dados coletados do *AgenaRisk* é apresentado. Na Figura 10, o histograma dos dados coletados do *AgenaRisk* é apresentado.

Figura 9 – *QQ Plot* dos dados coletados do *AgenaRisk*



Fonte: Elaborada pelo autor.

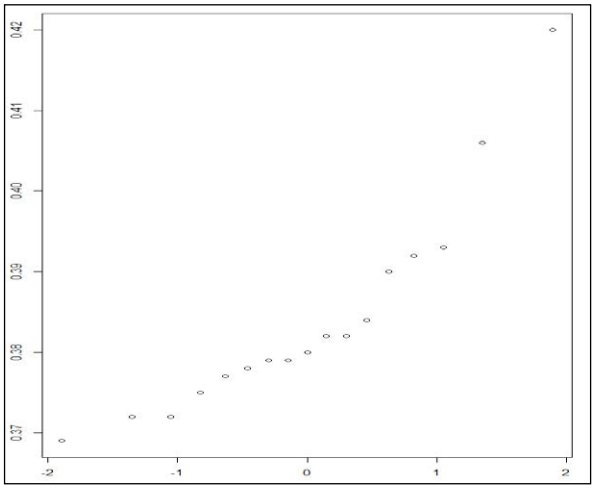
Figura 10 – Histograma dos dados coletados do *AgenaRisk*



Fonte: Elaborada pelo autor.

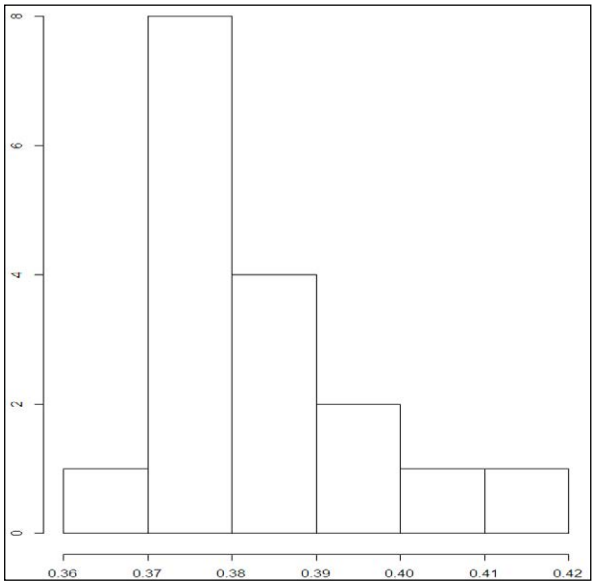
Ao se analisarem as Figuras 11 e 12, percebe-se que, aparentemente, os dados coletados do *AgenaRisk* não são normais. Para confirmar a conclusão, o teste de *Shapiro-Wilk* foi executado no R[4] com α = 0,05. O p-valor calculado foi *0,02675*. Dado que o p-valor < α, rejeita-se a hipótese nula, a qual afirma que a distribuição é normal. Desta forma, confirma-se que os dados coletados do *AgenaRisk* não são normais.

Figura 11 – *QQ Plot* dos dados coletados do *C++*



Fonte: Elaborada pelo autor.

Figura 12 – Histograma dos dados coletados do *C++*



Fonte: Elaborada pelo autor.

O mesmo processo foi seguido para os dados coletados do *C++*. O *QQ Plot* é apresentado na Figura 11. O histograma é apresentado na Figura 12. O p--valor calculado no teste de *Shapiro-Wilk* foi *0,01249*, ou seja, conclui-se que os dados coletados do *C++* também não são normais.

Desta forma, para comparar a tendência central de ambas as distribuições, o teste de *Wilcoxon* pareado foi utilizado com α = 0,05. Inicialmente, os dados foram comparados, a fim de se verificar se as tendências centrais das amostras são iguais. Como resultado, obteve-se *p-valor = 0,0003198*. Como o *p--valor* < α, conclui-se que as tendências centrais das amostras são diferentes.

Dado que, ao analisar o *boxplot* e as médias das amostras, aparentemente, a tendência central do *C++* é menor que do *AgenaRisk*, foi executado um teste de *Wilcoxon*, pareado com a hipótese alternativa de que a tendência central de *C++* é menor que a do AgenaRisk e α = 0,05. Como resultado, obteve-se *p--valor = 0,0001599*. Como o *p-valor* < α, conclui-se que a tendência central (i.e., tempo de execução) do *C++* é menor que a do *AgenaRisk*.

Apesar dos resultados demonstrarem que a solução em *C++* tem performance melhor que a do *AgenaRisk*, ressalta-se ser possível que o *AgenaRisk* execute processos em background que influenciem o tempo de sua execução. De qualquer forma, conclui--se que a solução em *C++* tem capacidade de modelagem similar ao *AgenaRisk* e performance aceitável para ser utilizada na prática.

6 Conclusões

Apesar da recente popularidade, a construção de redes bayesianas ainda apresenta alguns desafios. Um deles se refere à definição de TPN para redes bayesianas de larga escala. A automatização deste processo é possível com o *batch learning*. Por outro lado, este necessita de um banco de dados com informações suficientes para sua aplicação, o que, na prática, não é comum. A outra opção trata de coletar conhecimento de especialistas, manualmente, a qual é inviável para construção de redes de larga escala.

Em Fenton, Neil e Caballero (2007), uma solução baseada em nós ranqueados foi apresentada. Por outro lado, os detalhes necessários para aplicá-la na prática sem a utilização do *AgenaRisk*, que se trata de uma ferramenta paga, não foram apresentados. Em Laitila (2013), algoritmos para implementação das funções WMEAN, WMAX, WMIN e MIXMINMAX foram apresentados.

Neste trabalho, complementam-se os trabalhos de Fenton, Neil e Caballero (2007) e Laitila (2013) apresentando um método para implementar a solução com detalhes acerca do tamanho das amostras das distribuições probabilísticas e número de iterações e simulações necessárias para ter resultados aceitáveis. Os resultados foram promissores. A diferença nos resultados do nosso método, em comparação com o *AgenaRisk*, se manteve dentro da margem de erro aceitável (i.e., abaixo de 5%). A maior diferença entre a solução proposta e o *AgenaRisk* foi causada pelo uso diferenciado da variância no cálculo da TNormal, porém, em termos práticos, isso é irrelevante. A solução proposta limita-se à função *WMEAN*.

A solução utilizada foi disponibilizada em uma biblioteca em *C++*. Futuramente, serão implementadas as funções *WMAX*, *WMIN* e *MIXMINMAX*. Além disso, será definido um método para automatizar a configuração das expressões ponderadas de acordo com dados coletados de um especialista.

A solução foi avaliada em comparação com o *AgenaRisk* em função de dois fatores: similaridade dos resultados e performance. Com relação à similaridade dos resultados, concluiu-se que ambas as soluções são similares. Com relação à performance, concluiu-se que a solução em *C++* é melhor. Por outro lado, há ameaças à validade desta conclusão relativas à possibilidade de processos, que influenciem no tempo de execução, serem executados em *background* no *AgenaRisk*.

REFERÊNCIAS

DE MELO, A. C. V.; SANCHEZ, A. J. Software maintenance project delays prediction using Bayesian Networks. **Expert Systems with Applications:** An International Journal, v. 34, n. 2, p. 908-919, 2008.

DÍEZ, F. J. Parameter adjustment in Bayes networks: the generalized noisy or-gate. In: NINTH CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE. **Proceedings...** D. Heckerman and A. Mamdani, eds, p. 99-105, Washington D.C, 1993.

FENTON, N. *et al*. Making Resource Decisions for Software Projects, In: SOFTWARE ENGINEERING, 2004., **Proceedings...** 26., INTERNATIONAL CONFERENCE ON (ICSE2004). IEEE , 2004, p. 397-406.

FENTON, N.; NEIL, M. **Risk Assessment and Decision Analysis with Bayesian Networks**. Florida: CRC Press, 2012.

FENTON, N.; NEIL, M.; CABALLERO J. Using Ranked Nodes to Model Qualitative Judgements in Bayesian Networks. **IEEE Transactions on Knowledge and Data Engineering**, v. 19, n. 10, p. 1420-1432, 2007.

FENTON, N. *et al*. Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets, **Information & Software Technology**, v. 9, n. 1, p. 32-43, 2007.

FRIEDMAN, N.; GEIGER, D.; GOLDZSMITH, M. Bayesian network classifiers. **Machine Learning**, v. 20, p. 131-165, 1997.

HECKERMAN, D. A tutorial on learning with Bayesian networks. **Report MSR-TR-95-06, Microsoft Advanced Technology Division**, Microsoft Corporation, Seattle, Washington, 1995.

HUANG, K.; HENRION, M. Efficient Search-Based Inference for Noisy-OR Belief Networks. In: TWELFTH CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE. **Proceedings...** Portland, OR, p. 325-331, 1996.

LAITILA, P. **Improving the Use of Ranked Nodes in the Elicitation of Conditional Probabilities for Bayesian Networks**. Dissertação (Mestrado em Ciências em Tecnologia no Programa de Engenharia, Física e Matemática) – Aalto University, Espoo, Finlândia, 2013.

LEE, E.; PARK. Y.; SHIN, J. G. Large engineering project risk management using a Bayesian belief network. **Journal of Expert Systems and Applications**: An International Journal, v. 36, n. 3, p. 5880-5887, 2009.

NEIL, M.; MALCOLM B.; SHAW R. Modelling an Air Traffic Control Environment Using Bayesian Belief Networks. In: INTERNATIONAL SYSTEM SAFETY CONFERENCE, 21. **Proceedings...** Ottawa, Ontario, Canada, 2003.

NEIL, M.; FENTON N.; TAILOR M. Using Bayesian Networks to model Expected and Unexpected Operational Losses. **Risk Analysis: An International Journal**, v. 25, n. 4, p. 963-972, 2005.

PERKUSICH, M.; ALMEIDA, H.; PERKUSICH, A. A Model to Detect Problems on Scrum-Based Software Development Project. IN: ANNUAL ACM SYMPOSIUM ON APPLIED C, 28. **Proceedings...** 2013.

(Endnotes)

1 http://www.agenarisk.com

2 http://people.sc.fsu.edu/~jburkardt/cpp_src/truncated_normal/truncated_normal.html

3 https://bayesiannetwork.wordpress.com

4 https://www.r-project.org/

AGRADECIMENTOS

Ao IFPB pela bolsa PIBICT concedida como incentivo para realizar esta pesquisa.