FABIO HENRIQUE SANTANA MACHADO

ANÁLISE DE DESEMPENHO EM REDES BAYESIANAS COM LARGURA DE ÁRVORE LIMITADA

2017

FABIO HENRIQUE SANTANA MACHADO

ANÁLISE DE DESEMPENHO EM REDES BAYESIANAS COM LARGURA DE ÁRVORE LIMITADA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

2017

FABIO HENRIQUE SANTANA MACHADO

ANÁLISE DE DESEMPENHO EM REDES BAYESIANAS COM LARGURA DE ÁRVORE LIMITADA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Fabio Gagliardi Cozman

2017

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Machado, Fabio Henrique Santana
Análise de desempenho em redes bayesianas com largura de árvore limitada / F. H. S. Machado -- versão corr. -- São Paulo, 2017.
55 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Redes Bayesianas 2.Aprendizado Computacional 3.Inteligência Artificial 4.Programação Linear I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

AGRADECIMENTOS

Agradeço a meus pais, familiares e amigos pelo apoio e carinho. Em especial a Santiago e Leonardo pelo incentivo constante e companheirismo sincero. Agradeço também ao corpo docente, coordenadoria e funcionários da Pós Graduação do departamento Engenharia de Computação e Engenharia Elétrica da Escola Politécnica da USP e aos integrantes do Laboratório de Tomada de Decisões pela companhia, ainda que passageira. Por fim, agradeço principalmente ao professor e orientador Fabio Cozman pela paciência, incentivo e direção dos caminhos e também ao professor Denis Mauá pela orientação nas etapas iniciais além de sempre estar disponível para discussões a respeito deste trabalho.

Menciono também que este trabalho foi realizado com suporte do CNPq, processo de bolsa número 165873/2014-0 e os experimentos aqui apresentados foram realizados com recursos do LCCA - Laboratório de Computação Científica Avançada da Universidade de São Paulo.

Remember that all models are wrong;
the practical question is how wrong do
they have to be to not be useful.
George Box

RESUMO

Este trabalho fornece uma avaliação empírica do desempenho de Redes Bayesianas quando se impõe restrições à largura de árvore de sua estrutura. O desempenho da rede é visto especificamente pela sua capacidade de generalização e também pela precisão da inferência em problemas de tomada de decisão. Resultados preliminares sugerem que adicionar essa restrição na largura de árvore diminui a capacidade de generalização do modelo além de tornar a tarefa de aprendizado mais difícil.

Palavras-Chave – Largura de árvore. Aprendizado de Redes Bayesianas. Inferência em Redes Bayesianas.

ABSTRACT

This work provides an empirical evaluation of the performance of Bayesian Networks when treewidth is bounded. The performance of the network is viewed as its generalizability and also as the accuracy of inference in decision making problems. Preliminary results suggest that adding constraints to treewidth decreases the model performance on unseen data and makes the corresponding optimization problem more difficult.

Keywords – Treewidth. Learning Bayesian Networks. Inference in Bayesian Networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

- API - *Application Programming Interface*, um conjunto de padrões de programação para acesso a um aplicativo
- CPLEX - O método *simplex* de programação linear implementado na linguagem *C*
- CPU - *Central Processing Unit*, Unidade Central de Processamento
- ESS - *Equivalent Sample Size*, parâmetro de cálculo para o BD*eu*
- HPC - *High Performance Computing*, Computação de Alta Performance
- ILP - *Integer Linear Programming*, Programação Linear Inteira
- LP - *Linear Programming*, Programação Linear
- MILP - *Mixed Interger Linear Programming*, Programação Linear Inteira Mista
- TW - *Tree Width*, Largura de Árvore
- UCI - Universidade da Califórnia, Irvine
- WDBC - *Wisconsin Diagnostic Breast Cancer*, Base de dados relatando diagnósticos acerca de câncer de mama na cidade de Wisconcin, Estados Unidos

LISTA DE SÍMBOLOS

- $argmax$ - Operação que busca a combinação de parâmetros de uma função de forma a obter seu valor máximo.
- E - O conjunto de arestas de um grafo ou árvore
- F - Coleção de todos os conjuntos de pais candidatos para uma variável
- G - Um grafo
- $\mathcal{G}$ - Classes de todos os possíveis grafos G
- i,j,k,t - (Em minúsculo) Índices de um conjunto de elementos quaisquer
- N - O conjuntos de variáveis aleatórias de uma Redes Bayesiana
- T - Uma árvore
- Pa_x - Os pais de um nó x no grafo
- s - O *score* de uma variável
- V - O conjunto de vértices de um grafo ou árvore
- $\mathbf{X}$ - (Em maiúsculo) Um subconjunto de variáveis aleatórias de uma Rede Bayesiana
- x - (Em minúsculo) Uma variável aleatória de uma Rede Bayesiana
- w - A largura de árvore imposta para o problema de busca
- Θ - O conjunto de parâmetros de probabilidade de uma Rede Bayesiana
- θ - Um parâmetro de probabilidade de uma Rede Bayeisana
- π - Um conjunto de pais candidatos para uma variável

SUMÁRIO

1 INTRODUÇÃO

Redes Bayesianas são modelos gráficos probabilísticos usados para representar, de maneira eficiente, distribuições conjuntas de probabilidades e relações de dependência em problemas de incerteza com múltiplas variáveis [2]. Elas são compostas por uma estrutura (um grafo) que codifica as relações de (in)dependência entre as variáveis do problema e um conjunto de probabilidades condicionais, também chamados de parâmetros da rede. Juntos, eles representam a distribuição de probabilidade conjunta das variáveis do modelo.

Essas redes possuem diversas aplicações no contexto de inteligência artificial [3], como nos campos da estatística [4], biologia [5] e visão computacional [6]. Suas vantagens em relação a outros métodos de aprendizado de máquina incluem a interpretabilidade dos resultados, uma vez que o grafo da estrutura deixa claro as relações entre as variáveis do problema, além de permitirem incorporar facilmente conhecimentos *a priori*. Seu unico problema, no entanto, esta no fato de que sem os devidos cuidados essas redes podem apresentar problemas de escalabilidade [7]

As principais operações de uma Rede Bayesiana envolvem inferências arbitrárias sobre as variáveis do modelo, como configurar um estado para as variáveis de forma a maximizar uma distribuição de probabilidade condicional [8] ou ainda encontrar a probabilidade *a posteriori* de uma hipótese após a entrada de evidências. A literatura apresenta diversos algoritmos capazes de fazer inferência em Redes Bayesianas como passagem de mensagem [3], eliminação de variáveis [9] e propagação da decomposição em árvore [10].

Algoritmos de inferência em Redes Bayesianas tem seu pior caso exponencial no tamanho da rede, com exceção quando o grafo da rede é, na verdade, uma árvore, onde neste caso existem algoritmos de inferência com tempos polinomiais. Mais especificamente, pesquisas recentes apontam que há uma forte evidência de que esses algoritmos sejam NP-Difícil para redes arbitrárias e que sua complexidade é exponencial na largura de árvore do grafo da rede [11–13].

Construir uma Rede Bayesiana é uma tarefa difícil e tediosa e por essa razão é co-

mum utilizar métodos capazes de aprender as redes a partir de observações históricas das variáveis do problema [14]. Uma abordagem popular no aprendizado da estrutura baseia-se em busca e avaliação. Ela mede a qualidade de uma estrutura e usa uma função para atribuir uma pontuação a ela. Essa pontuação é utilizada por métodos de busca que encontram a melhor estrutura dentro do espaço de busca [15]. Essa função de avaliação favorece estruturas que se adequam melhor aos dados ao mesmo tempo que penaliza modelos de alta complexidade pelo princípio da navalha de Occam: quando diante de dois modelos que representem os dados com a mesma verossimilhança, escolha o modelo menos complexo. Maximizar a verossimilhança do modelo guia a busca para estruturas que atribuem maior probabilidade aos dados observados; penalizar a complexidade reduz a chances de sobreajustes (*overfitting*) no modelo, aumentando assim sua capacidade de generalização.

A maioria das funções de avaliação penalizam a complexidade do modelo baseado na quantidade de vértices no grafo da rede. Ainda que isso favoreça o aprendizado de grafos esparsos, que consequentemente tendem a obter um melhor desempenho em casos não observados, ela não equilibra a complexidade geral da Rede Bayesiana: redes com grafos esparsos ainda são capazes de representar distribuições de probabilidade muito complexas, que podem levar a *overfitting* do modelo quando aprendido de bases de dados muito pequenas. Além disso, há uma segunda razão pela qual penalizar a rede apenas pela quantidade de vértices não é suficiente.

Apenas manter o grafo esparso não necessariamente limita a largura de árvore, e isso pode levar a estruturas com baixa capacidade de generalização e necessidade de uso de inferência aproximada [1]. Também é comum limitar a quantidade de pais dos nós do grafo no espaço de busca. No entanto, isso não evita a geração de estrutura com largura de árvore limitada pois sabe-se que mesmo um grafo em *grid* possuindo no maximo 2 pais para cada nó e menos de $2n$ vértices, sua largura de árvore é $\sqrt{n}$ (onde n é a quantidade de variáveis do modelo) [16]. A Figura 1 exemplifica essa situação pois neste caso o *grid* possui 9 váriaveis, quantidade de pais limitados a 2 por nó e, mesmo assim, sua larugra de árvore é 3.

1.1 Motivação

Sendo assim, vários trabalhos sugerem restringir o espaço de busca no aprendizado de Redes Bayesianas a estruturas com baixa largura de árvore. Isso tem com o objetivo de aumentar o desempenho das redes aprendidas quando usadas em aplicações do mundo real

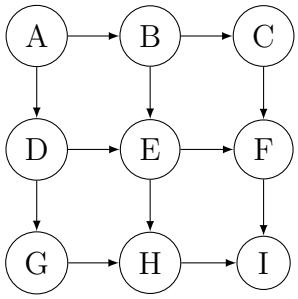


Figura 1: Um grafo direcionado do tipo *grid* com 9 nós. Cada nó neste grafo possui no máximo 2 pais. Mesmo assim, sua largura de árvore é 3.

[1, 17–20]. Uma vez que a tarefa de estimar a largura de árvore de um grafo é por si só uma problema NP-Difícil, estender as atuais abordagens de aprendizado de Redes Bayesianas nessa direção é um problema não trivial havendo várias pesquisas nessa área [1, 17–20]. Ainda que a maioria desses trabalhos sejam motivados pela suposição de que restringir o espaço de busca pode aumentar o desempenho (em termos de melhor capacidade de generalização) das redes aprendidas, a evidência empírica não é substancial. Menor ainda são as indicações do impacto que essas restrições tem na qualidade da inferência dessas redes quando aplicadas a problemas reais. Por fim, outro fator agravante e geralmente negligenciado é o custo adicional que as restrições de largura de árvore têm nos algoritmos de aprendizado.

Para ilustrar esse último caso, ressaltamos que as funções de avaliação podem ser decompostas sob a forma de uma soma de pontuações locais, que dependem somente da variável local e dos seus pais imediatos na estrutura. Assim, uma vez que a ordem topológica das variáveis é fixa, o problema de aprender uma estrutura que maximize esta pontuação se torna uma busca para o melhor conjunto de pais de cada nó, que pode ser resolvido de forma eficiente se assumirmos uma baixa interconectividade. Essa afirmação serve de base para muitas das abordagens de aprendizado de Redes Bayesianas como o método popular de [21]. Essa abordagem no entanto não pode ser aplicada quando se impõe um limite na largura de árvore, uma vez que ela torna os subproblemas (das pontuações locais) interdependentes entre si [2].

1.2 Objetivos

Este trabalho tem o principal objetivo de analisar empiricamente o efeito que restringir a largura de árvore exerce na capacidade de generalização da Redes Bayesianas aprendidas.

Especificamente buscamos responder a pergunta: "De que maneira se altera a verossi-

milhança da Rede Baysiana em relação a sua base de dados originária quando restringimos a largura de árvore da sua estrutura?"

1.3 Publicações

2 publicações decorreram deste trabalho.

Os resultados apresentados aqui foram publicados no Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2015). O artigo, entitulado "Bayesian Networks of Bounded Treewidth: A Performance Analysis" e publicado em Inglês, foi aceito e apresentado na conferência em Natal em Novembro de 2015. Na ocasião, o artigo recebeu a premicação de "Menção Honrosa", na categoria "*Best Paper* do ENIAC 2015".

Um resumo do trabalho também foi apresetado no IV Workshop de Pós-Graduação da Área de Concentração "Engenharia de Computação", da Escola Politécnica da Universidade de São Paulo em Setembro de 2015. Na ocasião, o artigo recebeu a premiação de "Melhor artigo de mestrado".

Os apêndices A e B apresentam o texto completo destes 2 artigos.

1.4 Organização do Texto

Para o problema de aprendizado com limite na largura de árvore, nós utilizamos de técnicas de Programação Inteira Mista (MILP - *Mixed Integer Linear Programing* na sigla em inglês) conforme descrito em [1]. O aprendizado é feito sobre bases de dados extraídas do repositório da UCI e a generalização dos modelos é calculada pela média de diversas instâncias do mesmo problema (através de *validação cruzada* medindo a verossimilhança do modelo em uma base de testes diferente da usada no aprendizado. Para cada base de dados, limitamos a largura de árvore em diferentes valores.

Nossos resultados sugerem que restringir a largura de árvore acaba por gerar redes com baixa capacidade de generalização. Com baixa restrições na largura de árvore (permitindo que o algoritmo aprenda redes com valores maiores) o desempenho da redes aprendias é compatível com os das redes aprendidas sem nenhum restrição. Além disso, é notório que a limitação acarreta em um grande aumento de complexidade para os algoritmos de aprendizado.

Dado isso, o restante deste trabalho segue a estrutura detalhada a seguir.

O Capítulo 2 revisa os conceitos e define as principais notações deste trabalho. A Seção 2.1 é dedicada aos tópicos sobre Teoria dos Grafos que são fundamentais nos algoritmos de aprendizado. A Seção 2.2 detalha a teoria sobre Redes Bayesianas.

No Capítulo 4 são detalhados os experimentos realizados até agora descrevendo a metodologia adotada (Seção 4.1) e os resultados em si (Seção 4.2). A implementação específica do algoritmo de aprendizado uzado aqui é apresentada no Capítulo 3.

Por fim, no Capítulo 5 são discutidos os resultados obtidos bem como as sugestões de trabalhos futuros.

2 PRELIMINARES TEÓRICOS

Neste capítulo são introduzidos os principais conceitos por trás deste trabalho bem como as notações usadas. Nós começamos relembrando os conceitos básicos sobre grafos e definindo o conceito mais importante de largura de árvore na seção 2.1 e, em seguida, a seção 2.2 revê a teoria a repeito de Redes Bayesiana e como é feito o aprendizado dessas redes.

2.1 Grafos

Um *grafo* é um par $G = (V, E)$, sendo V um conjunto de *vértices* e E um conjunto de *arestas*, usado para representar relações entre objetos. Os termos vértices e nós tem o mesmo significado neste trabalho, bem como arestas e arcos. Usamos $V(G)$ e $E(G)$ para nos referirmos ao conjunto de vértices e arestas do grafo G ou, se o contexto for claro, nos referimos apenas como V e E respectivamente.

Em um grafo dirigido seu conjunto de arestas é composto de pares ordenados de nós, ou seja, essas arestas possuem uma origem e apontam para seu destino. Os *pais* de um vértice i são todos os outros nós com arestas apontando para aquele vértice. Nós usamos Pa_i para nos referirmos ao conjunto de nós que são pais de um dado nó i. Da mesma forma, todos os nós aos quais as arestas de um vértice apontam, são seus *filhos* .

Definição 1. *Um* grafo moral *é o grafo obtido de um grafo direcionado conectando-se todos os pares de vértices que possuem filhos em comum e removendo a direção das arestas.*

A Figura 2 mostra o exemplo de um grafo moral.

Um *ciclo* em um grafo é uma sequencia de vértices que se inicia e termina no mesmo nó e onde quaisquer dois vértices consecutivos deste sequência estão conectados entre si respeitando-se a direção dos arcos. Uma *corda* é uma aresta que conecta quaisquer dois vértices em um ciclo e que não pertence a ele. Um *clique* é um subgrupo de vértices de um grafo onde todos estão conectados entre si.

(a) Grafo direcionado (b) Grafo Moral

Figura 2: Um grafo direcionado com 9 vértices na forma de um grid (2a) e seu respectivo grafo moral (2b). Arestas em vermelho foram adicionadas ao conectar-se os nós com filhos em comum.

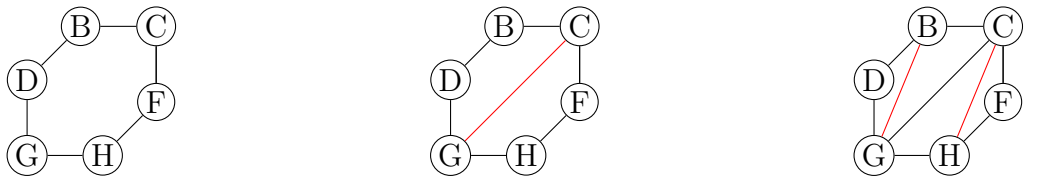
Definição 2. *Um* grafo cordal *é um grafo onde todo ciclo de tamanho 4 ou maior, possui uma corda.*

Definição 3. *Uma* ordem de eliminação perfeita *de um grafo é a enumeração de seus vértices de forma que, para todo nó, todos os vizinhos predecessores nessa ordem formam um clique.*

Um grafo tem uma única ordem de eliminação perfeita se, e somente se, ele for um grafo cordal [22]. É possível atribuir uma ordem de eliminação perfeita a qualquer grafo dirigido primeiro obtendo-se seu respectivo grafo moral e, em seguida, adicionando-se arestas extras aos seus ciclos de forma que ele se torne um grafo cordal.

A Figura 3 exemplifica o processo de obtenção um grafo cordal. Neste exemplo só há um ciclo sem cordas (o ciclo B-C-F-H-G-D-B). É possível ligar quaisquer dois vértices desse ciclo inserindo-se uma nova corda, por exemplo os vértices G e C. Ao fazer isso, dois novos ciclos sem cordas são gerados: G-D-B-C-G e G-H-F-C-G e duas novas cordas são inseridas, completando o Grafo Cordal.

A Figura 4 mostra o grafo final junto à sua a ordem de eliminação perfeita.



(a) Um ciclo sem cordas (b) Dois ciclos sem cordas (c) Grafo cordal

Figura 3: Um grafo com um ciclo e sua transformação em um Grafo Cordal.

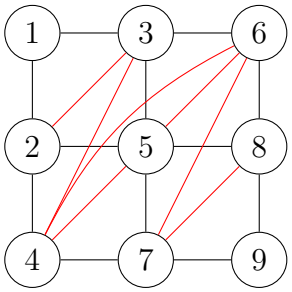


Figura 4: Um Grafo Cordal e sua ordem de eliminação perfeita.

2.1.1 Largura de Árvore

A largura de árvore de um grafo está relacionada com o tamamanho dos subconjuntos de vértices que compõem os nós da sua *decomposição em árvore*

Definição 4. *Uma* decomposição em árvore *de um grafo* $G = (V, E)$ *é um par* (T, X)*, onde* $T = (V, E)$ *é uma árvore e* $X = (X_t : t \in V(T))$ *é uma família de subgrupos de vértices de* G *sendo:*

1. $\cup_{i \in V(T)} X_i = V(G)$;
2. *Para cada* v*,* w *em* $V(G)$*, existe um nó* X_i *em* $V(T)$ *onde* $v, w \in X_i$;
3. *Para* $i, j, k \in V(T)$*, se* j *está no caminho em* T *entre* i *e* k *então* $X_i \cap X_k \subseteq X_j$.

A *largura* de uma decomposição em árvore se dá por $max_{t \in V(T)}(|X_t| - 1)$.

A largura de árvore de um grafo é a menor largura de todas as suas possíveis decomposições em árvore. Uma maneira alternativa de se obter a largura em árvore de um grafo é calculando-se o número máximo de vizinhos predecessores de acordo com a sua ordem de eliminação perfeita menos 1 [23].

Existem ainda outras forma de se calcular a largura em árvore de um grafo [24], mas estas não são relevantes neste trabalho.

2.2 Redes Bayesianas

Nesta seção são tratados os aspectos referentes à representação das Redes Bayesianas e suas propriedades. Também são apresentados os métodos de aprendizado de Redes Bayesianas, com foco nas técnicas utilizadas neste trabalho.

2.2.1 Representação

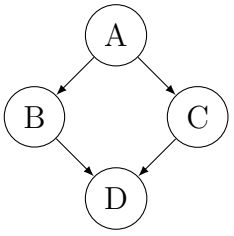
Formalmente, uma Rede Bayesiana é uma tripla (N, G, Θ) onde,

- $N = X_1, X_2, ..., X_n$ é um conjunto de variáveis aleatórias a respeito do domínio do problema. Aqui referenciamos como X_i (em maiúsculo) a i-ésima variável do conjunto, e como x_i (em minúsculo) o valor x da variável X_i. Apenas quando necessário diferenciar, representamos como x_i^j o j-ésimo valor da variável X_i;
- $G = (V, E)$ é um grafo dirigido que representa relações de dependências condicionais, com cada nó em V representando uma variável do problema;
- Θ é uma coleção de parâmetros condicionais que descrevem a distribuição de probabilidade conjunta das variáveis. Ele assume a forma $\Theta = \{\theta_i(x_i, y_{Pa_i})\}$, onde $\theta_i(x_i, y_{Pa_i}) = P(x_i, y_{Pa_i})$ para todo vértice i em G, valor x da variável X_i e configuração y para o conjunto de pais Pa_i da variável X_i.

Assume-se que uma variável é condicionalmente independente de todos os seus não descendentes dado seu conjunto de pais no grafo G. Por causa disso, uma Rede Bayesiana define a distribuição conjunta de todas as variáveis de V como a multiplicação de seus fatores em Θ, conforme mostrado pela Equação 2.1.

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} \theta_i(x_i, y_{Pa_i}) = \prod_{i=1}^{n} P(x_i | y_{Pa_i}). \tag{2.1}$$

A Figura 5 apresenta um exemplo de uma Rede Bayesiana sobre 4 variáveis.



(a) Estrutura G

$\theta_A(V) = P(A = V) = 0.1$
$\theta_C(V, V) = P(C = V | A = V) = 0.7$
$\theta_C(V, F) = P(C = V | A = F) = 0.2$
$\theta_B(V, V) = P(B = V | A = V) = 0.7$
$\theta_B(V, F) = P(B = V | A = F) = 0.2$
$\theta_D(V, V, V) = P(D = V | C = V, B = V) = 0.9$
$\theta_D(V, V, F) = P(D = V | C = V, B = F) = 0.7$
$\theta_D(V, F, V) = P(D = V | C = F, B = V) = 0.7$
$\theta_D(V, F, F) = P(D = V | C = F, B = F) = 0$

(b) Parâmetros Θ

Figura 5: Exemplo de uma Rede Bayesiana.

Aprender uma Rede Bayesiana consiste em estimar G e Θ a partir de um conjunto de dados contendo observações históricas das variáveis em N. Todos os métodos apresen-

tados aqui assumem uma base de dados completa e cujo os dados sejam independentes e identicamente distribuídos.

A estimativa dos parâmetros Θ é dada a partir de métodos estatísticos eficientes e bem estabelecidos na literatura, já o aprendizado da estrutura ainda é uma área em expansão e cuja pesquisa recente tem sido bastante ativa [19, 25, 26].

2.2.2 Aprendizado de Parâmetros

Dois métodos comuns usados no aprendizados de parâmetros são dados pela Máxima Verossimilhança e por Estimativa Bayesiana [27].

Considerando um conjunto de dados D e um modelo (com um determinado conjunto de parâmetros θ) de Rede Bayesiana M_θ, a verossimilhança do modelo aos dados pode ser apresentada sob a forma apresentada na Equação 2.2, sendo d cada observação das variáveis da rede naquela base de dados.

$$L(M_\theta|D) = \prod_{d \in D} P(d|M) \tag{2.2}$$

A máxima verossimilhança consiste então em encontrar um parâmetro θ que maximize a função de verossimilhança, como mostrado na equação 2.3, onde $argmax_\theta$ seja uma função que encontra a configuração θ dentro do modelo M que maximize a função de verossimilhança dentro da base de dados D.

$$\hat{\theta} = arg\ max_\theta\ (L(M_\theta|D)) \tag{2.3}$$

Devido às relações de independência entre as variáveis de uma Rede Bayesiana, a sua máxima verossimilhança pode ser decomposta como a soma da máxima verossimilhança de cada variável considerando apenas seus pais na estrutura. É possível reescrever a função de verossimilhança sob a forma da contagem de valores que cada variável combinada com seus pais assume na base de dados. Assim, o parâmetro de cada variável $X_i \in N$ e seu conjunto de pais Pa_i pode ser calculada pela equação 2.4 onde $\#N(x)$ é a contagem de vezes em que a configuração de valores x das variáveis aparecem na base de dados D.

$$\hat{\theta}_{X_i|Pa_i} = \frac{\#N(x_i, y_{Pa_i})}{\#N(y_{Pa_i})} \tag{2.4}$$

Já o método da Estimativa Bayesiana assume os parâmetros da rede como variáveis do

problema e calcula através do teorema de bayes a probabilidade a posteriori do parâmetro θ em relação à base de dados D. Isso permite incorporar conhecimento a priori na estimativa de parâmetros.

O cálculo dos parâmetros $P(\Theta|D)$ é dado pela equação 2.5.

$$P(M_\theta|D) = \frac{P(D|M_\theta)P(M_\theta)}{P(D)} \tag{2.5}$$

O cálculo de $P(D|M)$ nada mais é do que a verossimilhança e pode ser calculado da mesma forma descrita anteriormente, $P(D)$ é um fator de normalização e $P(M_\theta)$ é capaz de acrescentar o conhecimento a priori no aprendizado do parâmetro.

Uma maneira de encontrar essa distribuição a priori é utilizando a *distribuição beta*, caracterizada pela utilização de *hiperparâmetros* capazes de moldar a distribuição baseado em conhecimento a priori do problema. A distribuição beta é um caso específico da distribuição Dirichlet que se utiliza de apenas dois hiperparâmetros α e β.

Da mesma forma que a máxima verossimilhança, a estimativa bayesiana pode ser reescrita sob a forma de uma soma de estimativas locais de cada variável dada seus pais e pela contagem de valores assumidos por cada um deles na base de dados, a única diferença é que os parâmetros α e β são incluídos na contagem. A equação 2.6 apresenta esse cálculo [15].

$$\hat{\theta}_{X_i|Pa_i} = \frac{\#N(x_i, y_{Pa_i}) + \alpha_{X_i|Pa_i}}{\#N(y_{Pa_i}) + \beta_{Pa_i}} \tag{2.6}$$

2.2.3 Aprendizado de Estrutura

Existem diversas maneiras de se aprender a estrutura de uma Rede Bayesiana a partir de um conjunto de dados. Por exemplo, algumas abordagens baseadas em restrições consistem em encontrar, para cada variável do domínio, um outro conjunto (mínimo) de variáveis que a torne condicionalmente independente das demais. Uma abordagem popular, e por este motivo adotada neste trabalho, é baseada em busca e avaliação, onde se usa de uma função de avaliação para atribuir uma medida de qualidade da rede e então buscar-se no grupo de possíveis estruturas da rede, aquela que maximiza essa medida [28].

Uma função de avaliação popular baseia-se também numa estimativa bayesiana utilizando-se de uma distribuição Dirichlet chamada BDeu e inicialmente apresentada por [14]. No entanto, um problema com o uso da distribuição Dirichlet é que devido à quantidade de

redes a serem buscadas, a especificação de hiperparâmetros torna-se impraticável. Por essa razão essa avaliação assume algumas premissas:

- Grafos equivalentes: A premissa de que grafos dirigidos acíclicos são equivalentes se eles codificam a mesma distribuição de probabilidades conjuntas;
- Equivalência de verossimilhança: A premissa de que, se dois grafos são equivalentes a distribuição de probabilidade de seus parâmetros θ também serão as mesmas;
- Possibilidade de estrutura: O *esqueleto* de um grafo acíclico dirigido é o grafo obtido quando se ignora a direção das arestas, um grafo acíclico dirigido é *completo* se o seu esqueleto é completo. Essa premissa assume que para todo subgrafo acíclico dirigido completo, $P(G) > 0$.

Com essas premissas o *score* BDeu apresentado a seguinte forma:

$$BDeu(G, D) = P(G) + \sum_{i=1}^{N} \sum_{j=1}^{\Pi_i} \left(\frac{\Gamma(\frac{N'}{\Pi_i})}{\Gamma(\#N_{ij} + \frac{N'}{\Pi_i})}\right) + \sum_{k=1}^{V_i} \left(\frac{\Gamma(\#N_{ijk} + \frac{N'}{V_i\Pi_i})}{\Gamma(\frac{N'}{V_i\Pi_i})}\right),$$

que aparece quando:

$$P(X_i = x_{ik}, \Pi_i = w_{ij}|G) = \frac{1}{V_i\Pi_i}.$$

Essa função baseia-se unicamente em um hiperparâmetro, N'. Não há consenso sobre qual valor esse hiperparâmetro deve adotar, mas sabe-se que a função de score é altamente sensível a ele, sendo recomendado fazer testes iniciais para adaptar o aprendizado ao domínio do problema.

Uma vez calculado a avaliação local para todas as variáveis do problema, o aprendizado de estrutura passa a ser um problema de busca por todas as combinações possíveis de variáveis e seus conjuntos de pais, de forma a maximizar a soma dos scores.

Uma função de busca pode focar em encontrar uma solução ótima, ou seja, aquela que maximiza a avaliação do conjunto de pais dados pela função de score, ou também se basear em métodos heurísticos, que não garante a solução ótima, mas encontra uma solução boa em tempo factível.

A solução BDeu pode ser considerada um método híbrido. Ela ainda foca em encontrar a solução ótima, mas, devido à natureza da programação matemática, ela pode ser

interrompida a qualquer momento (ou após um período de tempo específico) e fornecer a melhor soluçao encontrada até aquele momento.

3 FORMULAÇÃO MILP PARA O APRENDIZADO DE ESTRUTURA DE REDES BAYESIANAS

Este capítulo chama a atenção para o problema de busca de uma estrutura de uma Rede Bayesiana que tenha uma determinada largura de árvore e que melhor descreva um conjunto de dados a partir de uma função de avaliação.

A *programação linear* é um método para se encontrar a melhor solução em um modelo matemático cujos requisitos são representados por uma série de relações lineares entre as variáveis do problema. A programação linear é um caso especial da *programação matemática*. Formalmente, a programação linear é a técnica de otimização de uma função objetivo linear, sujeita a restrições de igualdade e inigualdades lineares. A *programação inteira*, geralmente referenciada pela sigla em inglês ILP é um caso especial da programação linear onde as variáveis do problema estão restritas ao conjunto de números inteiros. Como o próprio nome sugere, a *programação linear inteira mista* ou MILP na sigla em inglês é aquela na qual as variáveis do problema assumem valores reais ou inteiros, dependendo das restrições aplicadas.

A função de avaliação escolhida foi a BDeu conforme apresentada no Capítulo 2. O problema de otimização é definido pela equação 3.1.

$$G^* = \underset{G \in \mathcal{G}_{n,w}}{argmax} \sum_{i \in N} s_i(Pa_i) \tag{3.1}$$

Na equação 3.1, $\mathcal{G}_{n,k}$ é o conjunto de todos os DAGs com n nós e largura de árvore menor ou igual a w. s_i é a função de avaliação da variável i considerando seus pais no grafo G.

Formulações MILP tem se mostrado bastante eficientes para o problema de busca de estrutura [29, 30], principalmente quando se faz necessário limitar a largura de árvore do grafo resultante.

A abordagem usada neste trabalho foi inicialmente proposta por [1]. Ela combina uma versão modificada do programa ILP de busca de estrutura de [31] (apresentada em detalhes na Seção 3.1 deste capítulo) com a solução ILP para limitar a largura de árvore de um grafo apresentada em [32] (descrito em detalhes na Seção 3.1 deste capítulo). Este capítulo finaliza com o programa MILP completo na Seção 3.3 e alguns detalhes de implementação e complexidade de algoritmo na Seção 3.4.

3.1 Programa MILP para a limitação da largura de árvore

Considerando-se um grafo $G = (V, E)$, podemos definir uma formulação MILP que estabelece todos os supergrafos de G e que tenham largura de árvore menor ou igual a uma constante especificada. Isso se dá estabelecendo um conjunto de regras de restrições que codificam todas as possíveis ordens de eliminação do grafo G que permitam que, para todo vértice do grafo, o número máximo de vizinhos que precedem essa ordem não seja maior que a largura requerida, obtendo-se então um grafo cordal $G' = (V, E')$ de G.

São definidas as seguintes variáveis de decisão:

- w: Constante que especifica a largura de árvore máxima do grafo resultante, ou seja, o grafo obtido terá largura de árvore menor ou igual a w.
- $\{x_i : i \in V\}$: Conjunto que especifica uma ordem de eliminação parcial e recebe valores reais no intervalo $[0, N]$. Se $x_i < x_j$ então a variável i é eliminada antes de j. Essa ordem de eliminação é parcial porque permite que $x_i = x_j$ uma vez que há casos onde múltiplas ordens de eliminação podem ser igualmente suficientes para produzir um grafo cordal G' de G. Quando isso ocorrer, o fato de $x_i = x_j$ indica que qualquer ordem de precedência desses dois nós resultará num grafo com a mesma largura de árvore.
- $\{y_{ij} : i, j \in V\}$: Matriz $[V \times V]$ binária (recebe valores entre 0 e 1) onde, se $y_{ij} = 1$ então há um arco entre i e j no grafo resultante e i precede j na ordem de eliminação.

O conjunto de restrições proposto pela Figura 6 codifica o problema proposto. Para efeito de simplificação n é o equivalente a $|V|$.

A seguir são detalhadas cada uma dessas restrições.

Figura 6: Formulação MILP para a limitação da largura de árvore [1].

$$\sum_{j \in V} y_{ij} \leq w \qquad \forall i \in V \qquad (3.2a)$$

$$(n+1) \,.\, y_{ij} \leq n + x_j - x_i \qquad \forall i, j \in V \qquad (3.2b)$$

$$y_{ij} + y + ji = 1 \qquad \forall (i,j) \in E \qquad (3.2c)$$

$$y_{ij} + y_{ik} - (y_{jk} + y_{kj}) \leq 1 \qquad \forall i, j, k \in V \qquad (3.2d)$$

$$x_i \in [0, n] \qquad \forall i \in V \qquad (3.2e)$$

$$y_{ij} \in \{0, n\} \qquad \forall i, j \in V \qquad (3.2f)$$

- Restrição 3.2a $\sum_{j \in V} \; y_{ij} \leq w \qquad \forall i \in V$:

  Especifica que, para todo nó i, a quantidade de vizinhos predecessores no grafo moral resultante (casos onde $y_{ij} = 1$)não deve ser maior que w. Limitando-se a largura de árvore.

- Restrição 3.2b $(n+1) \,.\, y_{ij} \leq n + z_j - z_i \qquad \forall i, j \in V$:

  Permite que y_{ij} assuma o valor 1 somente se j aparecer depois de i na ordem de eliminação (mais especificamente $x_j > x_i + 1$).

- Restrição 3.2c $y_{ij} + y + ji = 1 \qquad \forall (i,j) \in E$:

  Garante que G' é um supergrafo de G.

- Restrição 3.2d $y_{ij} + y_{ik} - (y_{jk} + y_{kj}) \leq 1 \qquad \forall i, j, k \in V$:

  Garante que a ordem de eliminação codificada por x é perfeita para G'. Se j e k são vizinhos de i e predecessores na ordem de eliminação, então j e k também são vizinhos em G' (y_{jk} ou y_{kj} precisam ter o valor 1).

- Restrições 3.2e $x_i \in [0, n] \qquad \forall i \in V$ e 3.2f $y_{ij} \in \{0, n\} \qquad \forall i, j \in V$:

  Definem o domínio das variáveis x e y. Sendo a primeira pertencendo ao domínio dos números reais no intervalo de 0 a $|V|$ (mas que, na prática, quase sempre assumirá valores inteiros nesse intervalo) e a segunda, variáveis binárias como descrito anteriormente.

A diferença da formulação apresentada por [1] daquela apresentada em [32] está no fato da variável x ser contínua e admitir uma ordem de eliminação parcial.

Proposição 1. *Sejam* $\{z_i, y_{ij} : i, j \in V\}$ *um conjunto de variáveis que satisfaçam as restrições impostas da Figura 6 então o grafo* $G' = (V, E')$*, onde* $E' = \{ij \in V \times V : y_{ij} =$

1 ou $y_{ji} = 1\}$ é o grafo cordal de G com largura de árvore no máximo w e qualquer ordem de eliminação consistente com x é perfeita para G'.

3.2 Programa MILP para o aprendizado de estrutura

Para o aprendizado de estrutura de uma Rede Bayesiana, considerando-se a existência de um grafo cordal (não dirigido) $G' = (V, E')$ obtido pelo programa MILP descrito na seção 3.1. Ou seja, ele possui uma ordem de eliminação perfeita para e uma matriz binária $\{y_{ij}; i, j \in V\}$ onde $y_{ij} = 1$ somente se E' contem o par (i, j) e i é eliminado antes de j.

Além da matriz binária y já descrita, esta formulação opera acerca das seguintes variáveis de decisão:

- $\{F_i : i \in V\}$: É a coleção de todos os conjuntos de pais aceitos como predecessores da variável i no grafo. Esses conjuntos podem ser manualmente especificados pelo usuário (permitindo-se aí a inclusão de conhecimento a priori do domínio do problema) ou apenas definido como sendo todos os subconjuntos de Ni.

- $\{\pi_{it} : i \in V, t \in F_i\}$: Variável binária que tem valor 1 se o t-ésimo conjunto de pais pertencente à coleção F_i da variável i for selecionado. 0 do contrário. t assume valores entre $1, 2, ..., |F|$

- $\{z_i : i \in V\}$: Conjunto que especifica a ordem topológica dos nós no grafo resultante. Se $z_i > z_j$ então j não é pai de i.

O conjunto de restrições apresentados na Figura 7 codifica o problema de aprendizado de estrutura. Para efeito de simplificação, o escopo $\forall t$ assume valores no conjunto $1, 2, ..., |F_i|$.

Cada restrição está detalhada a seguir:

- Restrição 3.3a $\sum_t \pi_{it} = 1 \qquad \forall i \in N$:

  Estabelece que somente um conjunto de pais candidato para a variável i será selecionado.

- Restrição 3.3b $(n + 1) . \pi_{it} \leq n + z_j - z_i \qquad \forall i \in N, \forall t, \forall j \in F_{it}$:

  Garante que o grafo obtido será acíclico. Forçando a ordem topológica das variáveis codificadas por z. Aqui também, múltiplas ordens podem ser aceitas ($z_i = z_j$).

Figura 7: Formulação MILP para o aprendizado de estrutura [1].

$$\sum_t \pi_{it} = 1 \qquad \forall i \in N \tag{3.3a}$$

$$(n+1)\,.\,\pi_{it} \leq n + z_j - z_i \qquad \forall i \in N, \forall t, \forall j \in F_{it} \tag{3.3b}$$

$$\pi_{it} \leq y_{ij} + y_{ji}, \qquad \forall i \in N, \forall t, \forall j \in F_{it} \tag{3.3c}$$

$$\pi_{it} \leq y_{jk} + y_{kj}, \qquad \forall i \in N, \forall t, \forall j, k \in F_{it} \tag{3.3d}$$

$$y_{ij} \in [0, n] \qquad \forall i, j \in N \tag{3.3e}$$

$$z_i \in [0, n] \qquad \forall i \in N \tag{3.3f}$$

$$\pi_{it} \in \{0, n\} \qquad \forall i \in N, \forall t \tag{3.3g}$$

- Restrições 3.3c $\pi_{it} \leq y_{ij}+y_{ji}, \qquad \forall i \in N, \forall t, \forall j \in F_{it}$ e 3.3d $\pi_{it} \leq y_{jk}+y_{kj}, \qquad \forall i \in N, \forall t, \forall j, k \in F_{it}$: Juntas, fazem com que um arco só apareça no grafo resultante se esse arco também existir no seu grafo moral (lembre-se que a variável y codifica as informações necessárias para se construir um grafo moral).

  Especificamente a restrição 3.3c garante que a variável i seja descendente de todas as variáveis contidas no conjunto escolhido por π_{ij}. Por sua vez a restrição 3.3d requer que também haja um arco entre todos os vizinhos predecessores de i.

- Restrições 3.3e $y_{ij} \in [0, n] \qquad \forall i, j \in N$, 3.3f $z_i \in [0, n] \qquad \forall i \in N$ e 3.3g $\pi_{it} \in [0, 1] \qquad \forall i \in N, \forall t$:

  Define o escopo das variáveis do problema. y e π continuam sendo matrizes binárias e z assume valores reais no intervalo 0 a $|V|$.

A largura de árvore do grafo aprendido será, no máximo, a mesma largura do grafo moral codificado por y.

Proposição 2. *Sejam* $\{z_i, \pi_{it} : i \in V, t = 1, 2, ..., |F_i|\}$ *variáveis que satisfazem as restrições impostas na Figura 7 então um grafo* $H = (V, A)$*, onde* $A = \{i \leftarrow j : i, j \in V, \pi_{it} = 1 \, e \, j \in F_{it}\}$ *é acíclico, consistente com todos os conjuntos em* F_i *e do qual o grafo moral é subgrafo de* G'*.*

3.3 Unificação dos dois programas

Ao unificar os dois programas definidos nas Seções 3.1 e 3.2 é possível obter-se um programa inteiro capaz de aprender um grafo acíclico não dirigido que possua largura de árvore no máximo w. O programa unificado é apresentado na Figura 8.

Figura 8: Formulação MILP para o aprendizado de redes bayesianas com limite na Largura de Árvore [1].

Maximize:

$$\sum_{it} \pi_{it}.s_i(F_{it}) \tag{3.4a}$$

Sujeito a:

$$\sum_{j \in V} y_{ij} \leq w \qquad \forall i \in V \tag{3.4b}$$

$$(n+1)\,.\,y_{ij} \leq n + x_j - x_i, \qquad \forall i, j \in V \tag{3.4c}$$

$$y_{ij} + y_{ik} - (y_{jk} + y_{kj}) \leq 1 \qquad \forall i, j, k \in V \tag{3.4d}$$

$$\sum_{t} \pi_{it} = 1 \qquad \forall i \in V \tag{3.4e}$$

$$(n+1)\,.\,\pi_{it} \leq n + z_j - z_i \qquad \forall i \in V, \forall t, \forall j \in F_{it} \tag{3.4f}$$

$$\pi_{it} \leq y_{ij} + y_{ji}, \qquad \forall i \in V, \forall t, \forall j \in F_{it} \tag{3.4g}$$

$$\pi_{it} \leq y_{jk} + y_{kj}, \qquad \forall i \in V, \forall t, \forall j, k \in F_{it} \tag{3.4h}$$

$$x_i \in [0, n] \qquad \forall i \in V \tag{3.4i}$$

$$y_{ij} \in \{0, n\} \qquad \forall i, j \in V \tag{3.4j}$$

$$z_i \in [0, n] \qquad \forall i \in V \tag{3.4k}$$

$$\pi_{it} \in \{0, n\} \qquad \forall i \in V, \forall t \tag{3.4l}$$

Nesta formulação unificada, foi incluída a função objetivo. Vale lembrar que a função de avaliação BDeu de um grafo pode ser decomposto pela soma das funções de avaliação a respeito de cada variável e seus pais no grafo de forma independente. Sendo assim, a função objetivo do programa apresentado na Equação 3.4a é dada pela soma da função de avaliação de cada conjunto de pais escolhido para a variável i.

Teorema 1. *Sendo* $\{x_i, y_{ij}, z_i, \pi_{ij} : i, j \in N, t = 1, 2, ..., |Pa_i|\}$ *variáveis que satisfazem as restrições 3.4a-3.4l e definindo-se um grafo* $G = (N, E)$*, onde* $E = \{i \leftarrow j : i \in N, j \in N, \exists t \pi_{it} = 1 \, e \, j \in F_{it}\}$*, então* G *é acíclico, consistente com os conjuntos de pais especificados para as variáveis em* N *e possui largura de árvore no máximo* w*.*

Um corolário deste teorema é que o grafo G definido acima é aceito como uma solução para o problema combinatório apresentado pela Equação 3.1.

3.4 Detalhes de Implementação

Uma das vantagens do programa MILP apresentado aqui é que ele é definido apenas como um programa linear "puro", diferenciando-se de outras implementações na literatura que fazem uso de funções de *callback* e outras modificações durante o processo de otimização. Isso permite que o programa seja enviado diretamente para um software *solver* como o CPLEX (que possuis licenças pagas mas que também disponibiliza licenças acadêmicas) ou SCIP (alternativa de software livre de melhor desempenho disponível).

As funções de restrição foram implementadas na linguagem Python e enviadas ao CPLEX pelas bibliotecas disponibilizadas pelo software.

Vale ressaltar que estes programas inteiros apresentam um número muito grande de restrições, de complexidade polinomial em relação ao número de variáveis da base de dados. As restrições de maior custo são as Restrições 3.2b e 3.3b de complexidade $O(N^3)$. Assume-se que as funções de avaliação para as variáveis foram previamente calculadas e podem ser recuperadas em tempo constante.

4 EXPERIMENTOS

Os experimentos realizados focam principalmente em avaliar a capacidade de generalização da rede. Nesta seção são apresentados esses experimentos bem como os resultados alcançados.

4.1 Metodologia

A capacidade de generalização das redes foi analisada comparando sua verossimilhança a um conjunto de dados. Foram selecionadas bases de dados do repositório UCI[1] com diferentes tamanhos (quantidade de variáveis) já utilizados em trabalhos anteriores com Redes Bayesianas[1, 18] conforme a Tabela 1. Uma vez que bases pequenas, por exemplo com até 15 variáveis, geram Redes Bayesianas com largura de árvore também pequena, avaliar o efeito do limite da largura tem pouco efeito nessas redes. Também, bases de dados muito grandes, por exemplo com até 100 variáveis, tornam a tarefa de aprendizado muito complexa e impraticável. Desta forma, foram selecionadas três bases de dados, dentro destes limites e com diferentes tamanhos, para os experimentos apresentados aqui.

Cada base de dados forma separada em uma porção de teste e outra de treinamento. Foi usada a formulação MILP descrita neste trabalho para aprender a estrutura da rede e seus parâmetros eram então aprendidos através de Estimativa Bayesiana.

A função de avaliação usada foi a BDeu com parâmetros de 1 e 0.5. Para cada base de dados, foram aprendidas redes com largura de árvore limitada em 3, 4, 5 e incrementos de 5 até a quantidade máxima de variáveis do conjunto de dados (que é o equivalente a não se impor nenhum limite na largura de árvore). Para cada iteração de largura de árvore foi usada validação cruzada, com 5 grupos.

Os experimentos foram escritos em Python V.2.7. Como solucionador dos problemas MILP foi utilizado o software CPLEX V.12.6.1, incorporado aos experimentos através de

[1] http://archive.ics.uci.edu/ml/

Tabela 1: Bases de Dados.

Base de Dados	Variáveis	Tamanho da amostra
Mushroom	22	8124
WDBC	31	569
Audio	62	200

sua *API* para Python. Foi usado também o utilitário LibTW para calcular a largura em árvore das redes sem limite, desta forma era possível saber a largura *inerente* daquela rede.

Cada experimentos de aprendizado foi chamado de uma tarefa. Por exemplo, o aprendizado do primeiro grupo de validação cruzada da base de dados *Mushroom* com parâmetro BDeu 1 e largura de árvore 3, era uma tarefa, e assim por diante. Os experimentos foram executados num *Cluster* HPC com 20 Intel ®Xeon®CPU E7-2870 2.4GHz e 512GB de RAM localizado no campus Universidade de São Paulo. Até 6 tarefas foram executadas em paralelo em cada nó do Cluster e, em cada tarefa, o CPLEX executou a otimização do programa inteiro utilizando-se de até 3 núcleos de processamento. A otimização foi executada por até 3 horas (tempo de CPU) e, se nenhuma solução ótima fosse encontrada até então, a melhor solução até aquele momento era selecionada juntamente com o *GAP* de erro da solução ótima fornecido pelo CPLEX. A função de avaliação foi executada previamente.

4.2 Resultados

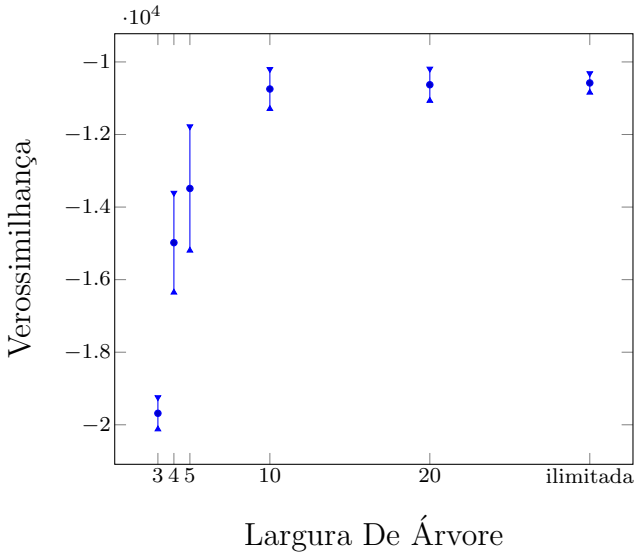
A largura das redes sem limite para o conjunto de dados *Mushroom* ficou entre 6 e 9. De fato, os resultados (Figuras 9 e 11) mostram que o desempenho da rede melhorou à medida que se aumenta a largura de árvore, atingindo um ponto máximo nas larguras entre 5 e 10. Permitir limites maiores na largura não teve impacto na verossimilhança. Ao verificar a lacuna de erro na tabela 2 é possível ver que aumentar o limite na largura de árvore teve impacto significativo no desempenho do programa inteiro. Nos gráficos 10 and 12 é possível ver que a métrica de avaliação da rede segue o mesmo comportamento da verossimilhança e, por isso, é possível afirmar que o desempenho meno na redes com limite de 3 se deve mais ao fato do programa de aprendizado não ter aprendido uma boa rede, do que á uma perda de generalização devido ao limite da largura de árvore.

A largura de árvore (sem limites) da base de dados WDBC manteve-se entre 7 e 9. Novamente, nas Figuras 13 e 15 é possível ver que limitar a largura ainda mais tem impacto

Tabela 2: Lacunas de erro em relação à solução ótima para todas as execuções MILP da base de dados Mushroom.

Limite de Pais	3	4	
ESS	1	0.5	1
	GAP Médio	GAP Médio	GAP Médio
largura 03	32.42%	62.98%	62.90%
largura 04	26.69%	49.54%	51.38%
largura 05	23.92%	44.06%	45.72%
largura 10	20.27%	32.57%	33.09%
largura 20	20.51%	33.61%	32.26%
largura ilimitada	20.10%	32.90%	31.88%

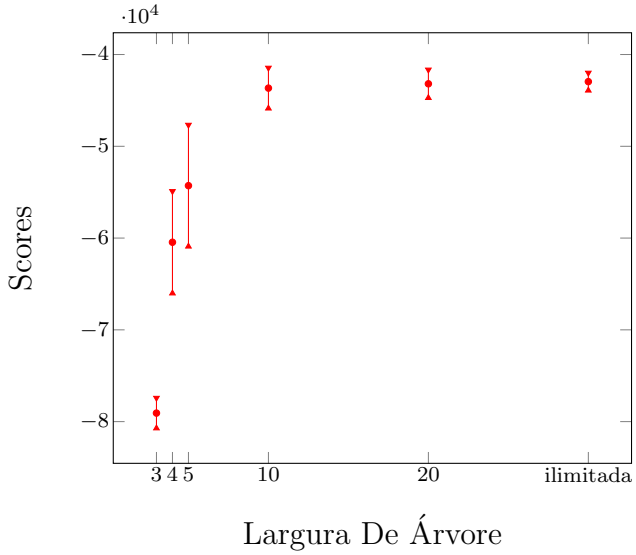
Figura 9: Verossimilhança das redes aprendidas da base de dados Mushroom, limitando-se a largura de árvore para um máximo de 4 e usando um parâmetro BDeu de 1.



negativa no desempenho da rede, sendo difícil dizer se a queda se deve à complexidade da tarefa de aprendizado (visto pelas lacunas de erro na tabela 3). A base de dados WDBC mostrou o mesmo comportamento assintótico da base Mushroom, mas com variância maior.

Devido ao seu tamanho, não foi possível calcular a largura em árvore da base de dados Audio. Como é possível ver na Figura 17 esta base não seguiu o mesmo comportamento assintótico e as redes aprendidas tiveram verossimilhança significativamente maior em larguras de árvore mais baixas. Isso mostra que, quando o algoritmo de aprendizagem é capaz de encontrar uma boa solução, restringir a largura de árvore pode levar a uma melhor generalização. Pelo mesmo motivo, quando o algoritmo de aprendizado não consegue encontrar boas soluções (por exemplo nas largura de 10 a 50 da Figura 17), a rede tem fraco desempenho não importando a largura.

Figura 10: Avaliação das redes aprendidas da base de dados Mushroom, limitando-se a largura de árvore para um máximo de 4 e usando um parâmetro BDeu de 1.



Foi testado o mesmo conjunto de dados Audio com um algoritmo que aprende a estrutura da rede sem limites na largura de árvore [29]. Sem essa restrição, é possível que a solução encontrada teria melhor desempenho e, de fato, nota-se que as métricas de verossimilhança foram até 30% melhores que no nosso algoritmo de aprendizado. Infelizmente, não é possível alcançar o mesmo desempenho ao se aprender redes com limite na largura.

Figura 11: Verossimilhança das redes aprendidas da base de dados Mushroom, limitando-se a largura de árvore para um máximo de 4 e usando um parâmetro BDeu de 0.5.

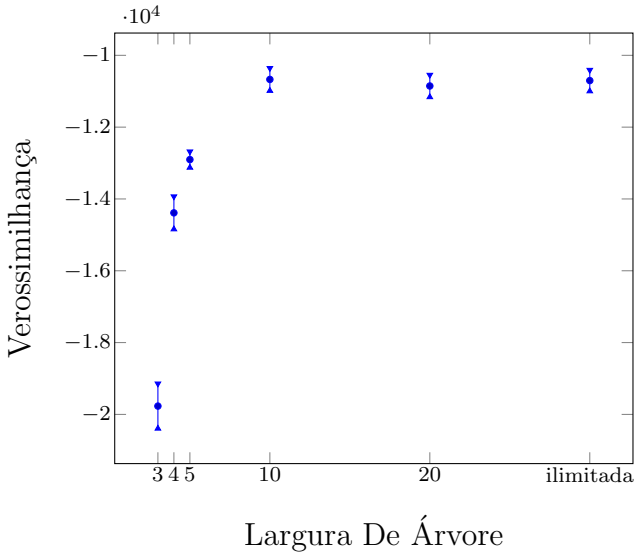


Figura 12: Avaliação das redes aprendidas da base de dados Mushroom, limitando-se a largura de árvore para um máximo de 4 e usando um parâmetro BDeu de 0.5.

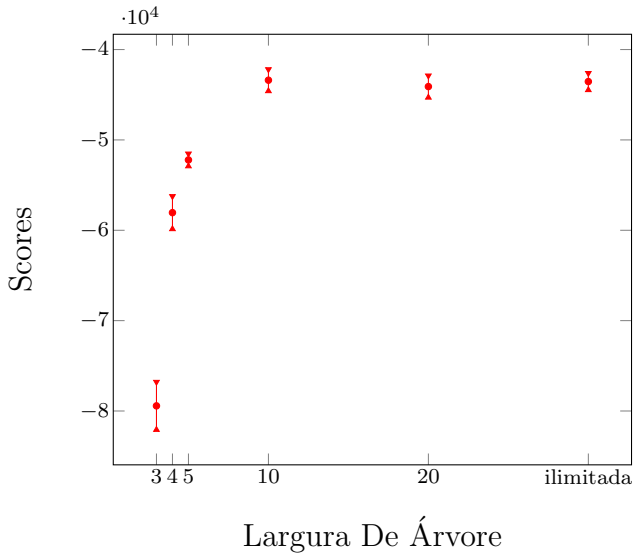


Tabela 3: Lacunas de erro em relação à solução ótima para todas as execuções do conjunto de dados WDBC.

Limite de Pais	3	4	
ESS	1	0.5	1
	GAP Médio	GAP Médio	GAP Médio
largura 03	25.11%	21.25%	27.32%
largura 04	22.71%	20.25%	25.46%
largura 05	23.09%	19.90%	22.82%
largura 10	19.90%	18.94%	20.54%
largura 15	18.98%	17.50%	19.06%
largura 20	18.48%	16.80%	19.10%
largura 25	18.42%	16.58%	18.11%
largura 30	18.26%	16.65%	18.27%
largura ilimitada	18.36%	16.54%	18.09%

Figura 13: Verossimilhança das redes aprendidas do conjunto de dados WDBC, limitando o número de pais em no máximo 4 e usado um parâmetros ESS de 1.

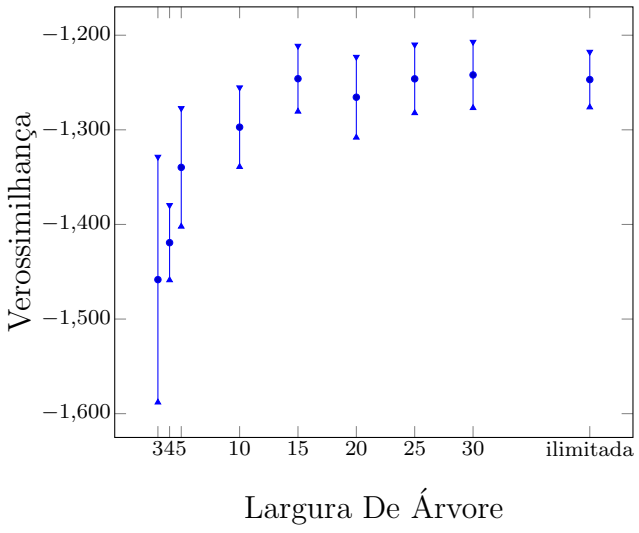


Figura 14: Medidas de avaliação das redes aprendidas do conjunto de dados WDBC, limitando o número de pais em no máximo 4 e usado um parâmetros ESS de 1.

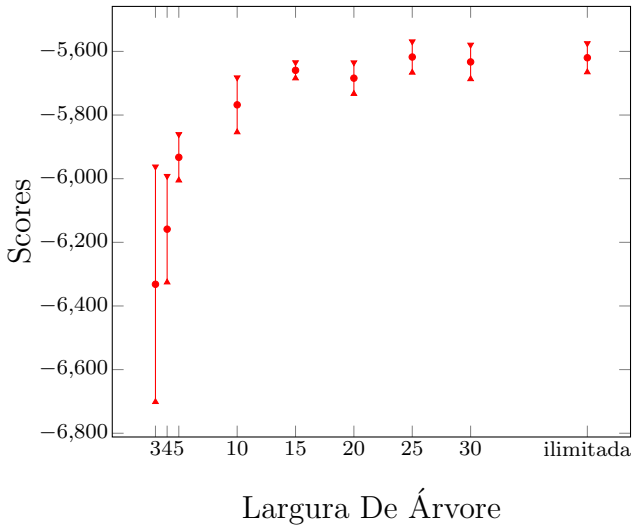


Figura 15: Verossimilhança das redes aprendidas do conjunto de dados WDBC, limitando o número de pais em no máximo 4 e usado um parâmetros ESS de 0.5.

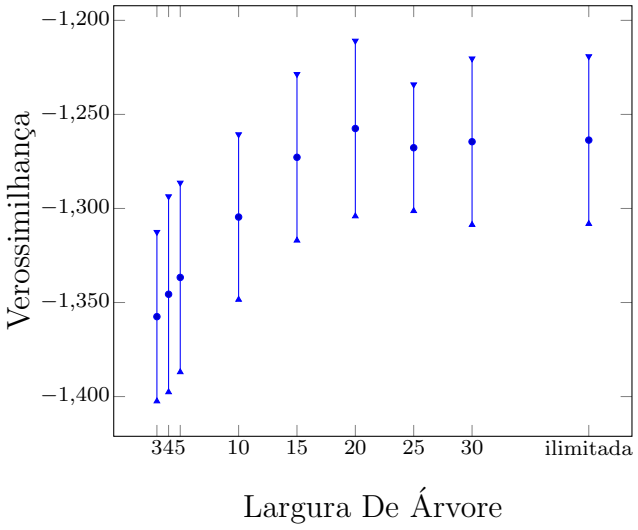


Figura 16: Medidas de avaliação das redes aprendidas do conjunto de dados WDBC, limitando o número de pais em no máximo 4 e usado um parâmetros ESS de 0.5.

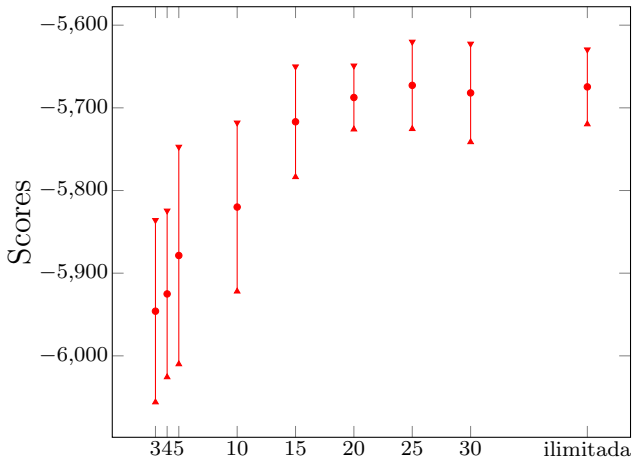


Tabela 4: Lacunas de erro em relação à solução ótima para todas as execuções do conjunto de dados Audio.

Limite de Pais	3	4
ESS	1	1
	GAP Médio	GAP Médio
largura 03	13.37%	19.35%
largura 04	14.44%	16.77%
largura 05	14.40%	20.47%
largura 10	28.40%	30.79%
largura 15	28.40%	30.72%
largura 20	28.40%	30.79%
largura 25	28.40%	30.79%
largura 30	25.42%	30.79%
largura 35	25.41%	30.79%
largura 40	22.30%	30.79%
largura 45	20.83%	30.79%
largura 50	17.96%	30.55%
largura 55	14.38%	23.64%
largura 60	12.02%	20.15%
largura ilimitada	20.92%	26.92%

Figura 17: Verossimilhança das redes aprendidas do conjunto de dados Audio, limitando o número de pais em no máximo 4 e usado um parâmetros ESS de 1.

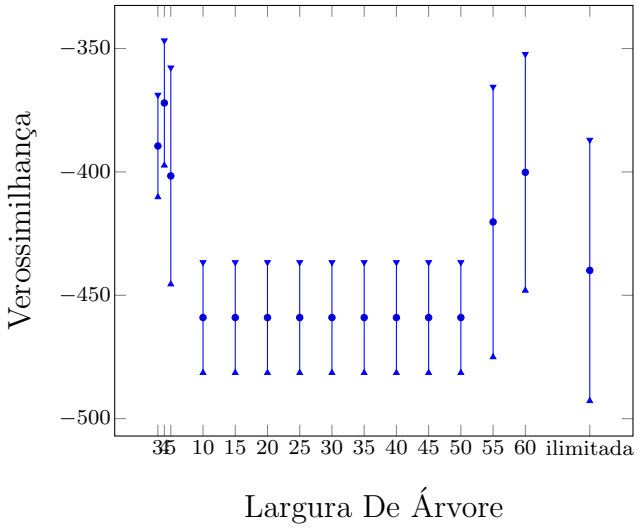
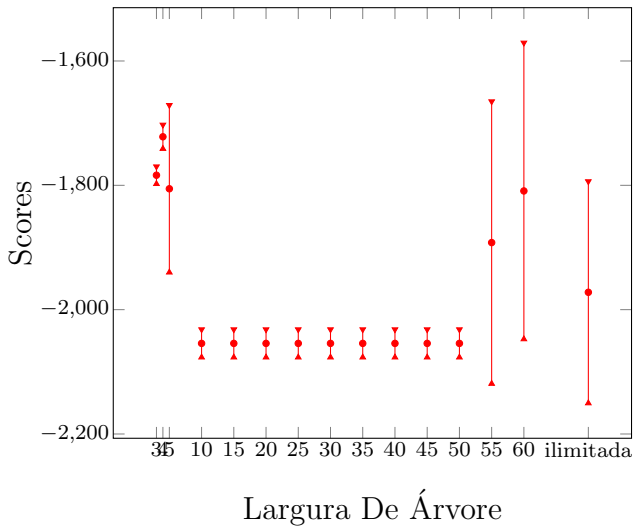


Figura 18: Medidas de avaliação das redes aprendidas do conjunto de dados Audio, limitando o número de pais em no máximo 4 e usado um parâmetros ESS de 1.



5 CONCLUSÕES

Esse trabalho buscou uma resposta para a influência da lagura de árvore sobre a verossimilhança da Rede Bayesiana em relação à base de dados da qual ela foi aprendida. Em outras palavras, como isso impacta na sua capacidade de generalização e representatividade. Esse questionamento motivou-se do fato de que os algoritmos de inferência usando Redes Bayesianas tem complexidade exponencial na largura de árvore do grafo e, por esta razão, aprender estruturas com baixa largura de árvore tornou-se favorável, ainda que se desconheça seus impactos.

Nossos resultados mostraram que, considerando-se apenas a verossimilhança das redes aprendidas em relação à base de testes, limitar a largura de árvore não acarretou em nenhuma melhora significativa de desempenho, como era esperado anteriormente. Especificamente era esperado que a verossimilhança da rede em relação à base de dados diminuísse à medida que se aumentava o limite de largura de árvore no processo de aprendizado, o que não ocorreu. Ao contrário, limitar demais a largura de árvore em redes com muitas variáveis pode acarretar em perda de informação.

Além disso, um fator ignorado inicialmente estava no impacto de limitar-se a largura de árvore no processo de aprendizado, mesmo sabendo-se de sua alta complexidade. Limitar a largura de árvore em limiares altos só faz sentido em bases de dados de alta dimensão (acima de 60 variáveis), o mesmo tipo de bases de dados sob as quais os atuais métodos de aprendizado com limite na largura de árvore se comportam de maneira precária. Se o objetivo de limitar a largura de árvore em uma aplicação de inferência em Redes Bayesianas é diminuir sua complexidade viabilizando essas operações, o mesmo limite pode tornar o processo de aprendizado da rede igualmente inviável.

Para avaliar de forma ainda melhor o desempenho dessas mesmas redes, sugerimos que sejam feitos experimentos medindo-se a sua acurácia de inferência bem como seu desempenho em problemas de classificação multilabel. Experimentos que permitam o aprendizado de estrutura ótimas e não somente aproximadas como ocorreram em alguns casos dos experimentos apresentados aqui, podem também ajudar a distinguir o efeito

que limitar a largura de árvore exerce sobre os algoritmos de aprendizado.

REFERÊNCIAS

1 NIE, S.; MAUA, D. D.; CAMPOS, C. P. de; JI, Q. Advances in learning bayesian networks of bounded treewidth. In: *Advances in Neural Information Processing Systems 27*. Montréal: Curran Associates, Inc., 2014. p. 23.

2 KOLLER, D.; FRIEDMAN, N. *Probabilistic Graphical Models: Principles and Techniques*. Cambridge: MIT Press, 2009. ISBN 0262013193.

3 PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann; 1 edition, 1988. 552 p. ISBN 1558604790.

4 GILKS, W. R.; RICHARDSON, S.; SPIEGELHALTER, D. J. *Markov Chain Monte Carlo in Practice*. [S.l.]: Chapman and Hall/CRC; Softcover reprint of the original 1st ed. 1996 edition, 1995. 512 p. ISBN 0412055511.

5 BALDI, P.; BRUNAK, S. *Bioinformatics: The Machine Learning Approach, Second Edition*. [S.l.]: A Bradford Book; second edition edition, 2001. 476 p. ISBN 026202506X.

6 COHEN, I.; SEBE, N.; COZMAN, F. G.; HUANG, T. S. Semi-supervised learning for facial expression recognition. In: *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval - MIR '03*. New York, New York, USA: ACM Press, 2003. p. 17. ISBN 1581137788.

7 TASKAR, B.; ABBEEL, P.; KOLLER, D. Discriminative probabilistic models for relational data. *UAI'02 Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., p. 485–492, aug 2002.

8 ANTONUCCI, A.; CORANI, G.; MAUÁ, D. D.; GABAGLIO, S. An ensemble of bayesian networks for multilabel classification. In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence IJCAI*. Beijing: AAAI Press, 2013. p. 1220–1225.

9 TAKIYAMA, F. I. *Algoritmos de Inferência Exata para Modelos de Primeira Ordem*. 131 p. Tese (Doutorado), 2013.

10 LAURITZEN, S.; SPIEGELHALTER, D. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, v. 2, n. 1, p. 157–224, 1988. Disponível em: ⟨http://www.jstor.org/stable/2345762⟩.

11 CHANDRASEKARAN, V.; SREBRO, N.; HARSHA, P. Complexity of inference in graphical models. In: *Proc. 24th Conf. on Uncertainty in AI*. Helsinki: AUAI Press, 2008. p. 70–78.

12 KWISTHOUT, J. H. P.; BODLAENDER, H. L.; GAAG, L. C. van der. The necessity of bounded treewidth for efficient inference in bayesian networks. In: *Proc. 19th European Conf. on AI*. Lisbon: IOS Press, 2010. p. 237–242.

13 KWISTHOUT, J. Treewidth and the computational complexity of MAP approximations. *Probabilistic Graphical Models*, v. 8754, p. 271–285, 2014.

14 HECKERMAN, D.; GEIGER, D.; CHICKERING, D. M. Learning bayesian networks: The combination of knowledge and statistical data. *Mach. Learning*, v. 20, n. 3, p. 197–243, 1995.

15 COOPER, G. F.; HERSKOVITS, E. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, v. 9, n. 4, p. 309–347, oct 1992. ISSN 0885-6125.

16 MAUÁ, D. D.; CAMPOS, C. P. de. Anytime marginal MAP inference. In: *Proc. 28th Int. Conf. on Mach. Learning*. [S.l.: s.n.], 2012. p. 1471–1478.

17 BERG, J.; JÄRVISALO, M.; MALONE, B. Learning optimal bounded treewidth bayesian networks via maximum satisfiability. In: *Proc. 17th Int. Conf. on AI and Stat.* Reykjavik: MIT Press, 2014. p. 86–95.

18 ELIDAN, G.; GOULD, S. Learning bounded treewidth bayesian networks. *J. of Mach. Learning Res.*, v. 9, p. 2699–2731, 2008.

19 KORHONEN, J. H.; PARVIAINEN, P. Exact Learning of Bounded Tree-width Bayesian Networks. In: *Proc. 16th Int. Conf. on AI and Stat.* Scottsdale: MIT Press, 2013. p. 370–378.

20 PARVIAINEN, P.; FARAHANI, H. S.; LAGERGREN, J. Learning bounded treewidth bayesian networks using integer linear programming. In: *Proc. 17th Int. Conf. on AI and Stat.* Reykjavik: MIT Press, 2014. p. 751–759.

21 TEYSSIER, M.; KOLLER, D. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In: *Proc. 21st Conf. on Uncertainty in AI*. Edinburgh: AUAI Press, 2005. p. 584–590.

22 Robin J. Wilson. *Introduction to Graph Theory*. 5th. ed. London: Pearson, 2012. 192 p. ISBN 978-0273728894.

23 ROBERTSON, N.; SEYMOUR, P. D. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, v. 7, n. 3, p. 309–322, 1986. ISSN 01966774.

24 SEYMOUR, P.; THOMAS, R. Graph Searching and a Min-Max Theorem for Tree-Width. *Journal of Combinatorial Theory, Series B*, Academic Press, v. 58, n. 1, p. 22–33, may 1993. ISSN 00958956.

25 BARTLETT, M.; CUSSENS, J. Integer linear programming for the bayesian network structure learning problem. *Artificial Intelligence*, p. 14, mar 2015. ISSN 00043702.

26 YUAN, C.; MALONE, B. An Improved admissible heuristic for learning optimal bayesian networks. In: *Proc. 28th Conf. on Uncertainty in AI*. California: MIT Press, 2012. p. 924–933.

27 HECKERMAN, D. A tutorial on learning with bayesian networks. In: *Innovations in Bayesian Networks*. Berlin, Heidelberg: Springer, 2008. p. 33–82. ISBN 978-3-540-85065-6.

28 TASKAR, B.; ABBEEL, P.; WONG, M.; KOLLER, D. Probabilistic relational models. In: *Introduction to Statistical Relational Learning*. Cambridge: The MIT Press, 2007. cap. 5, p. 129–174.

29 BARLETT, M.; CUSSENS, J. Advances in bayesian network learning using integer programming. In: *Proc. 29th Conf. on Uncertainty in AI*. Washington: AUAI Press, 2013. p. 182–191.

30 CUSSENS, J. Bayesian network learning with cutting planes. In: *Proc. 27th Conf. on Uncertainty in AI*. [S.l.: s.n.], 2011. p. 153–160.

31 CUSSENS, J.; BARTLETT, M.; JONES, E. M.; SHEEHAN, N. A. Maximum likelihood pedigree reconstruction using integer linear programming. *Genetic Epidemiology*, v. 37(1), p. 69–83, 2013.

32 GRIGORIEV, A.; HANS, E.; NATALYA, U. *Integer linear programming formulations for treewidth*. [S.l.], 2011.

APÊNDICE A – REPRODUÇÃO DE ARTIGO: BAYEISAN NETWORKS OF BOUNDED TREEWIDTH: A PERFORMANCE ANALYSIS

Publicação 1: Artigo em Conferência

Título: Bayeisan Networks of Bounded Treewidth: A Performance Analysis (8 Páginas)

Autores: Fabio H. S. Machado, Denis D. Mauá, Fábio G. Cozman

Conferência: Encontro Nacional de Inteligência Artifical e Computacional (ENIAC 2015)

Observações: O artigo recebeu o prêmio de "Menção Honrosa na categoria *Best Papers* do ENIAC 2015".

Bayesian Networks of Bounded Treewidth: A Performance Analysis

Fabio H. S. Machado
Escola Politécnica
Universidade de São Paulo

Denis D. Mauá
Instituto de Matemática e Estatística
Universidade de São Paulo

Fabio G. Cozman
Escola Politécnica
Universidade de São Paulo

***Abstract*—Bounding the treewidth of Bayesian Networks has been claimed to guarantee polynomial-time inference with little harm to accuracy. However, there has been little empirical evidence to support that claim. In this work we study empirically the effect of bounding treewidth on generalization ability. Our results suggest that adding a constraint to treewidth decreases the model performance on unseen data and makes the corresponding optimization problem more difficult.**

I. Introduction

Bayesian networks are graphical models used for efficiently representing dependency relations and joint probability distributions in multivariate uncertainty problems [1]. A Bayesian Network has three main components: a set of random variables, a directed acyclic graph (referred to as the network structure) representing (in)dependences between variables, and a collection of conditional probability values that jointly specify a joint probability distribution over the variables. Specifying a Bayesian Network is a daunting task, and practitioners often resort to automatic methods that "learn" both the structure and the conditional probabilities from data [2].

In this paper, we refer to "learning Bayesian Networks" specifically as the problem of learning a Bayesian Network structure from data; given a Bayesian Network structure, we can learn the parameters (i.e., the conditional probabilities) from a complete data set efficiently. A popular approach for learning Bayesian Networks, called score-based learning, uses a score function that assigns a quality measure to any given structure and searches for the score-optimizer structure within a group of candidate structures [3]. The score functions usually reward data fitness while penalizing the model complexity in a sort of Occam's razor policy: when faced with a choice between two models that equally fit the data, choose the least complex one. Maximizing the fitness of data guides the search towards structures that assign high probability to the observed data set. Penalizing complexity reduces model overfitting and increases the generalization ability.

Most score functions penalize a structure by the (weighted) number of edges in the network. Even though this favors sparse graphs that tend to perform better on unseen data, it does not balance for the overall complexity of the model: sparse networks can represent fairly complex distributions, which can still lead to overfitting when learning from small data sets. There is a second reason why penalizing the network edge density may not suffice. Very often, a Bayesian network is learned so that it can be used for drawing arbitrary inferences such as querying the posterior probability of a hypothesis after evidence is entered or selecting the states of a set of variables so as to maximize its conditional probability [4]. These inferences are all known to be NP-Hard to compute, and there is strong evidence that their complexity is exponential in the network treewidth, which is a measure of tree-likeness of a graph [5]–[7]. Controlling for sparsity does not limit the treewidth of the learned structures, and this can lead to learning models that perform poorly in applications due to poor generalization and the use of approximate inference [8], [9]. It is also very common to limit the number of parents (i.e., the in-degree) of nodes in the search. This again does not avoid generating structures of high treewidth. For example, a squared grid-like structure (such as the one in Figure 1) has maximum in-degree of two, less than $2n$ edges, and treewidth $\sqrt{n}$, where n is the number of nodes.

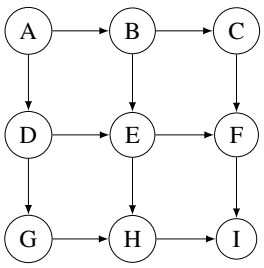


Fig. 1: A directed grid with 9 nodes. Each node in this graph has at most 2 parents. Still, the treewidth is 3.

With these issues in mind, many researchers have suggested constraining the search space, when learning Bayesian Networks, to structures of bounded treewidth, aiming at increased performance of the learned structure in real-world applications [9]–[13]. Since the task of estimating a network's treewidth is itself NP-Hard [14], extending current approaches to learning Bayesian networks in this direction is not trivial, and research on the topic has recently been very intense [9]–[13]. Even though most of these works are motivated by the claim that constraining the search space increases the performance of learned networks, the empirical evidence has not been substantial. Moreover, they have overlooked the additional burden that constraining the treewidth adds to search algorithms. To illustrate this point, note that score functions decompose as a sum of local score functions which depend only on a variable and its (immediate) parents. Hence, once a topological ordering among the variables is fixed, the problem of learning a score-maximizing structure breaks down to a greedy search for the

best parent set of each node; the latter can be solved efficiently if a low bound on in-degree is assumed. This fact is the base of many approaches to learning Bayesian Networks such as the popular Order-Based Sampling method of Teyssier and Koller [15]. Such an approach can no longer be applied if a bound on treewidth is imposed, since the constraint on treewidth makes the subproblems interdependent, and one often has to consider more complex structures (than orderings) such as (partial) k-trees in order to decompose the problem [9], [12].

Our main goal in this paper is to empirically analyze the effect that constraining the treewidth has on the generalization ability of learned Bayesian Networks. We do that by applying the mixed-integer linear programming approach to learning bounded treewidth Bayesian Networks described in [9] to a collection of real data sets from the UCI repository. We measure generalization ability by computing the cross-validated data likelihood of the model on held-out data. For each dataset we learn structures with different limits in its treewidth (usually 3, 4, 5 and increments of 5 until the dataset's maximum number of variables). The results suggest that constraining the search space actually leads to more difficult computational problems as the errors returned by programming solvers are often far larger for more constrained problems (ones with a smaller treewidth bound). The severe constraint on treewidth also tends to generate networks which generalize poorly on the test set. When the bound on treewidth is more loose, the performance of the learned networks is competitive with the unconstrained networks.

The remainder of this paper is structured as follows. We start with background knowledge on graph theory and Bayesian Networks (Section II), and then move to the presentation of the mixed-integer linear programming approach to learning bounded treewidth networks that we use (Section III). The experimental results, our main contribution, appear in Section IV, along with a discussion. Conclusions and comments on future work appear in Section V.

II. Preliminaries

In this section we present some necessary background on graph theory and Bayesian Networks.

A. Graph Theory

A Directed Acyclic Graph (DAG) is a graph where the edges have a direction associated with them such that there are no directed cycles (i.e., one cannot reach a node from itself following the direction of the edges). The parent set of a node i, denoted by Pa_i, is the set of all nodes that have an edge directed to this node. Similarly, the children set of a node is the set of all node that it points to. The descendants of a node comprises all the nodes that can be reached from it, the node's children, their children's children, and so on. The set of non-descendants of a node is the complement of its descendants.

The moral graph is the equivalent undirected form of a DAG and it is used by most operations in graphical models. It can be obtained by adding an edge between all pair of nodes that have a common child and dropping edge directions. Figure 2 show the moral graph of Figure 1.

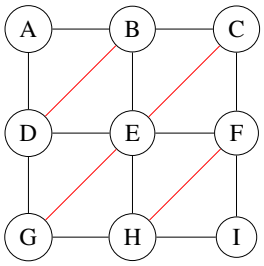
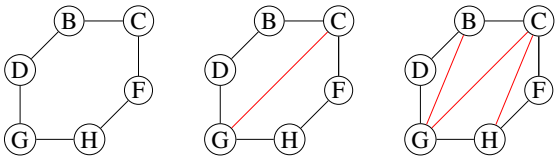


Fig. 2: The moral graph of the graph shown in 1. Red edges are the added edges by linking common parents.



(a) A Cycle (b) New Chord (c) New Chords

Fig. 3: The cicle B-D-G-H-F-C (3a) has no chord. This can be solved by inserting and edge, for example, from node G to C (3b). However this creates tow new chord-less cycles G-D-B-C and G-H-F-C. Again, two new edges are inserted (3c).

The treewidth of a DAG is the treewidth of its corresponding moral graph. The treewidth of any undirected graph can be found in the size of the cliques in a chordal graph that contains it (i.e. this graph is a subgraph of the given chordal graph). A chordal graph is a graph where all cycles of length four or more have a chord. A cycle in an undirected graph has a chord if it contains two nodes that are connected by an edge not in the cycle. Any undirected graph can be made chordal by inserting edges in it, a process called chordalization. A clique is a subset of vertices in an undirected graph that form a complete subgraph (i.e., all vertices are pairwise connected). Figure 3 shows a chordalization of the graph of Figure 2.

The treewidth of a undirected graph can be found in the maximum number of higher ordered neighbors according to its perfect elimination order. A perfect elimination order is a linear ordering of the nodes such that the higher ordered neighbors of each node form a clique. A graph can have a perfect elimination order if and only if it is chordal. Figure 4 shows a chordal graph obtained from Figure 2 and its corresponding perfect elimination order.

Thus, the treewidth of an undirected graph G is the minimum $w \geq 0$ such that G is a subgraph of a chordal graph with all cliques of size at most $w+1$ [16]. Alternatively, the treewidth of a graph can be computed as the minimum treewidth of a perfect elimination order. The perfect elimination order in Figure 4 has minimum treewidth 3, which is thus the treewidth of the DAG shown in Figure 1.

B. Bayesian Networks

Bayesian Networks are probabilistic graphical models that encode (in)dependences and joint probability distributions on

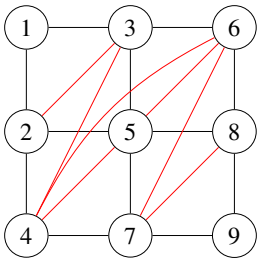
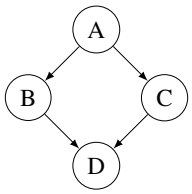


Fig. 4: A perfect elimination order for a chordal graph.



(a) Structure G

$\theta_A(T) = P(A = T) = 0.1$
$\theta_C(T,T) = P(C = T|A = T) = 0.7$
$\theta_C(T,F) = P(C = T|A = F) = 0.2$
$\theta_B(T,T) = P(B = T|A = T) = 0.7$
$\theta_B(T,F) = P(B = T|A = F) = 0.2$
$\theta_D(T,T,T) = P(D = T|C = T, B = T) = 0.9$
$\theta_D(T,T,F) = P(D = T|C = T, B = F) = 0.7$
$\theta_D(T,F,T) = P(D = T|C = F, B = T) = 0.7$
$\theta_D(T,F,F) = P(D = T|C = F, B = F) = 0$

(b) Parameters Θ

Fig. 5: Example of a Bayesian Network.

multivariate domains. Formally a Bayesian Network is a triple $\{V, G, \Theta\}$, where $V = \{X_1, X_2, ..., X_n\}$ is a finite set of random variables; $G = \{V, E\}$ is a DAG where the edge in E represents dependency relationships between variables in V; and $\Theta = \{\theta_i(x_i, x_{Pa_i}) = P(x_i|x_{Pa_i})\}$ is a collection of conditional probability values, one for each value x_i of variable X_i and configuration x_{Pa_i} of its parents Pa_i. A variable X_i is assumed independent of all its non-descendants given its parents Pa_i. Because of this, a Bayesian Network specifies a joint probability distribution over V as:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa_i) = \prod_{i=1}^{n} \theta_i(x_i, x_{Pa_i}). \quad (1)$$

To learn a Bayesian Network is to estimate Θ and G given a dataset D of historical observations of variables V. A possible way to estimate the structure G is to find, for each variable, another (minimal) set of variables that makes it conditionally independent of all the others. This is called constraint-based structure learning; we will not pursue such an approach here. Alternatively, one can formulate the problem as a combinatorial optimization where one searches for a high scoring structure among the set of all possible DAGs. Depending on the choice of scoring function, exact solutions of constrain-based and score-based approaches are equivalent [2].

A *scoring* function is the key element that makes a search method possible. It provides a measure of quality for a structure to describe datasets sampled from a hypothetical distribution (assumed to be the same used to sample the observed dataset). There are different approaches for deriving such metrics, but most of them focus on rewarding the structure by the log-likelihood of G according to D and penalizing the structure complexity.

Given a Bayesian Network with structure G and conditional probabilities Θ, and a data set D, we compute its data loglikehood $L_{G,D}$ as:

$$L_{G,D}(\Theta) = \log p(D|G, \theta) \quad (2)$$

$$= \sum_{i=1}^{n} \sum_{x_{Pa_i}} \sum_{x_i} n(x_i, x_{Pa_i}) \log \theta_i(x_i, x_{Pa_i}), \quad (3)$$

where $n(x_i, x_{Pa_i})$ indicates the number of instances (i.e., rows) of D where X_i takes values x_i and each $X_j \in Pa_i$ takes its corresponding value in x_{Pa_i}, and the inner sums are resp. over the values of X_i and Pa_i.

Two commonly used score functions are the BIC score and the BD score and its variants. The BIC score [17] was first proposed as a heuristic for avoiding overfitting introduced by maximizing likelihood and later provided a formal justification as an asymptotic approximation for the marginal posterior distribution of a graph. It is defined as:

$$s_{\text{BIC}}(G) = \max_{\theta} L_{G,D}(\Theta) - \frac{\log N}{2} \cdot |G|, \quad (4)$$

where N is the size of D (i.e., number of rows) and

$$|G| = \sum_{i=1}^{n} (|X_i| - 1) \prod_{X_j \in Pa(X_i)} |X_j| \quad (5)$$

is the number of free parameters ($|X_i|$ denotes the number of values X_i can assume). It is well known that $\arg\max_\theta L_{G,D}(\theta) = n(x_i, x_{Pa_i})/n(x_{Pa_i})$ (these are known as the maximum likelihood estimates of the conditional probability values).

The BD score [3] evaluates a structure by its marginal posterior probability $p(G|D)$ (up to a constant):

$$s_{\text{BD}}(G) = \log p(G) \int_{\theta} p(D|G, \theta) p(\theta|G). \quad (6)$$

The BDe score is a variant of the above metric which assigns a prior distribution over the network parameters that satisfy data likelihood equivalence: two structure with the same data likelihood are assigned the same (BDe) score. The BDeu score imposes an additional assumption that conditional probability distributions are sampled with uniform probability [2].

It is important to note that the BDeu score evaluates the *marginal* likelihood, as opposed to the maximum likelihood. By integrating the parameters out of the equation it measures the expected likelihood averaged over different possible choices of θ [1]. In this way, the BDe score helps avoiding overfitting even more by being less optimistic regarding a particular choice of parameters that maximizes the likelihood only on testing data.

Both these score functions satisfy the decomposability property that states that they can be written as a sum of local score functions $s_i(Pa_i)$. Furthermore, any of these local scores can be computed in time polynomial in the dataset (but exponential in the cardinality of Pa_i). Given the independence relations encoded in a Bayesian Network, the score of a complete network structure can be computed as a sum of scores of each variable given only its parents in the graph. This way, the goal of a search method is to find a structure G^* such that:

$$G^* = \arg\max_{G \in G_{n,k}} \sum_{i \in V} s_i(Pa_i), \tag{7}$$

where $G_{n,k}$ is the set of all possible DAGs with n nodes and treewidth at most k. This can be seen as a classic optimization problem and be solved using known search techniques such as Genetic Algorithms [18] or Hill Climbing [19]. Less obviously, the problem above can be cast as a mixed-integer linear programming optimization [9], [13], [20].

A linear program is a mathematical description of a constrained optimization problem involving linear inequalities over continuous variables and a linear objective. An Integer Linear Program (ILP) is a special case where the variables are restricted to take only integer values. When there is a mix of continuous and integer variables the problem is called a Mixed-Integer Linear Program (MILP). Casting the structure learning problem as a MILP problem allows us to benefit from highly optimized commercial solvers of mathematical programming, and it has been shown to be very effective. In fact, one of the most popular methods for learning Bayesian Networks with no constraint on treewidth is based on a MILP formulation [21], [22].

III. Mixed Integer Linear Program Formulations

In this section we present the MILP formulation for learning Bayesian Network structures proposed in [9]. We assume that the scores for all variables of the network are pre computed and can be retrieved in constant time (a common assumption). We choose using the MILP approach because (i) it is an anytime method, (ii) it is easy to relax the constraint on treewidth, (iii) it allows for other constraints be easily inserted, and (iv) it allows the use of commercial packages, which are readily available and easy-to-use.

Figure 6 gives the integer program formulation. Our constraints are divided in two groups: A group that enforces bounding treewidths (Constrs. 8b to 8d, 8i and 8k) and a group related to learning the network structure (Constrs. 8e to 8l).

A. Bounding treewidth

This formulation aims at encoding all possible elimination orders of a given graph $G = (V, E)$. If a solution exists, a chordalization of the graph with treewidth at most w can be obtained from the integer program.

Variable z_i, which takes real values (Constr. 8i), partially defines the elimination order in the formulation (partially because the formulation allows two nodes to have the same value of z, indicating that any order results in the same chordal graph). A variable i is eliminated before j if $z_i < z_j$.

Maximize:

$$\sum_{it} Pa_{it}.s_i(F_{it}) \tag{8a}$$

Subject to:

$$\sum_{j \in N} y_{ij} \leq w \qquad \forall i \in N \tag{8b}$$
$$(n+1).y_{ij} \leq n + z_j - z_i \qquad \forall i, j \in N \tag{8c}$$
$$y_{ij} + y_{ik} - (y_{jk} + y_{kj}) \leq 1 \qquad \forall i, j, k \in N \tag{8d}$$
$$\sum_{t} Pa_{it} = 1 \qquad \forall i \in N \tag{8e}$$
$$(n+1)Pa_{it} \leq n + v_j - v_i \qquad \forall i \in N, \forall t, \forall j \in F_{it} \tag{8f}$$
$$Pa_{it} \leq y_{ij} + y_{ji} \qquad \forall i \in N, \forall t, \forall j \in F_{it} \tag{8g}$$
$$Pa_{it} \leq y_{jk} + y_{kj} \qquad \forall i \in N, \forall t, \forall j, k \in F_{it} \tag{8h}$$
$$z_i \in [0, n] \qquad \forall i \in N \tag{8i}$$
$$v_i \in [0, n] \qquad \forall i \in N \tag{8j}$$
$$y_{ij} \in \{0, 1\} \qquad \forall i, j \in N \tag{8k}$$
$$Pa_{it} \in \{0, n\} \qquad \forall i \in N, \forall t \tag{8l}$$

Fig. 6: MILP formulation for learning Bayesian Networks of bounded treewidth

Variables y_{ij} are binary valued (Constr. 8k), and ensure that a node i is eliminated before j ($z_i < z_j$) and an edge existis among them in the resulting chordal graph.

Constraint 8b bounds the treewidth to be at most w by bounding the number of high ordered neighbors of every variable. By constraint 8c y_{ij} can only have value 1 if j appears after i in the elimination order. Constraint 8d guarantees the perfect elimination order of z because if variable j and k are higher ordered neighbors of i, then they are also neighbors.

B. Structure learning

Having a chordal graph $G' = (V, E')$, its perfect elimination order and a set of binary valued variables y_{ij} such that it is 1 only if E' contains ij and i is eliminated before j, the rest of our formulation specifies all DAGs over N for which the moral graph is a subgraph of G'.

The set F_i in constraints 8f to 8h is the set of all possible parents sets for that variable. This can be just the subset of $V \setminus \{i\}$, but normally scoring functions can limit this subset or it can even be specified by the user. An element F_{it} of this subset is a set of variables denoting one of those possible parents combinations. Each $\forall t$ in each constraint is for $1, 2, ..., |F_i|$.

The variable v_i, takes real values in $[0, n]$ (Constr. 8j) and specifies a topological order for the variables. If $v_i > v_j$ then j is not an ancestor of i. The $n$ary (constr. 8l) variable Pa_{it} denotes if the tth parent set of F_i were chosen for variable i.

Only one possible parent set can be chosen for each variable, as enforced by constraint 8e and those choices must be acyclic (Constr. 8f). Constraints 8g and 8h ensure that the edges of the found DAG have corresponding edges in G'.

Dataset	Variables	Sample Size
Mushroom	22	8124
WDBC	31	569
Audio	62	200

TABLE I: Datasets

C. Combining formulations

Having the set of variables $y_{ij}, z_i, v_i, Pa_{it}$ with $i, j \in V$, $t = 1, 2, ..., |Pa_i|$ and all satisfying constraints 8a to 8l. A Directed Graph $G = (V, E)$ where $E = \{i \leftarrow j : i, j \in V,\ \exists t\ s.t.\ Pa_{it} \text{ and } j \in F_{it}\}$ is acyclic, consistent with parents Pa_i and has treewidth at most w.

This formulation produces an optimal structure. However, both tasks are difficult tasks per se and their combination requires significant time and memory resources. Fortunately most MILP optimizers allow runs to be ended prematurely with a valid (perhaps not optimal) solution. In this way, this formulation can also be used to find approximate solution for learning Bayesian Networks. Some solvers even provide an outer error bound, showing the distance between the found solution for the maximum score. This is the approach used in the experiments reported in the next section.

IV. Experiments

In this section we present our methodology and results.

A. Methods

We empirically analyzed the performance of our networks by comparing their likelihood to the data. We used our MILP formulation on a training portion of each dataset to learn the structure and then we learned the network's parameters and calculated their log-likelihood to a test portion of each dataset. We have also tried using the alternative MILP formulation of Parviainen et al. [13], available through the authors' implementation, but we found it to be too slow.

We scored the networks using the BDeu function. We limited the maximum parents of each variable to 4 and used equivalent sample sizes (ESS) of 1.0 and 0.5. The two different values of ESS are used to isolate possible performance differences introduced by the score (and not by the constraint on treewidth). For each variation in each dataset, we learned networks of bounded treewidth, starting with treewidth 3, 4, 5 and then increments of 5 (or 10) until (and including) the dataset's maximum number of variables (which is equivalent as not imposing a bound to the networks's treewidth). For each iteration of treewidth, we ran a 5-fold cross validation to obtain a more reliable estimate of the expected test likelihood.

As mentioned before, our chosen datasets were selected from the UCI Repository[1]. We have selected datasets of different dimensions used in previous works on learning bounded treewidth Bayesian Networks [9], [11], but since our preliminary tests showed that datasets with few variables (say less than 15 variables) usually have "optimal" structures of very small treewidth, such datasets were discarded from this analysis. The datasets with 100 or more variables were also discarded for being computationally too costly to the MILP approach used. We were then left with 3 datasets of different dimensionality. These datasets are summarized in Table I. The experiments were performed by programs written in Python V.2.7. We used the CPLEX's Python API to interface with CPLEX version 12.6.1 for the MILP problems.

[1] http://archive.ics.uci.edu/ml/

Parents Limit	3	4	
ESS	1	0.5	1
	Avg. GAP	Avg. GAP	Avg. GAP
tw 03	32.42%	62.98%	62.90%
tw 04	26.69%	49.54%	51.38%
tw 05	23.92%	44.06%	45.72%
tw 10	20.27%	32.57%	33.09%
tw 20	20.51%	33.61%	32.26%
tw unlm.	20.10%	32.90%	31.88%

TABLE II: Remaining GAP values for the optimal solution for all MILP executions of the Dataset Mushroom

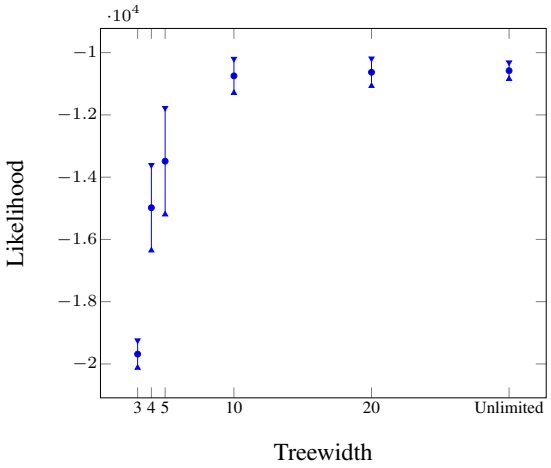


Fig. 7: Likelihood for the networks learned from dataset Mushroom, bounding the parent set to a maximum of 4 and using a ESS score of 1

We called each unique learning experiment a task. For example, the learning experiment of the first fold of the dataset Mushroom with maximum number of parents limited to 4 and the BDeu score's ESS parameter of 1.0 is one task. We ran our experiments in a HPC cluster with nodes of 20 Intel®Xeon®CPU E7-2870 of 2.40GHz and 512GB of available memory. Up to 6 tasks were allowed to run at the same time on each node and each CPLEX optimizer assigned to each task were given three cores to work in parallel. We allowed the optimizer to run up to two hours (CPU time) and we collected the found solution and the error gap of each task.

B. Results

We used the LibTW utility from [23] to calculate the treewidth of the unbounded networks so we can see the underlying treewidth of the "true" network of the training data. Even though learning the true network of the training data is not our main objective, we use this information so we can see the upper bound of the network's treewidth on that dataset.

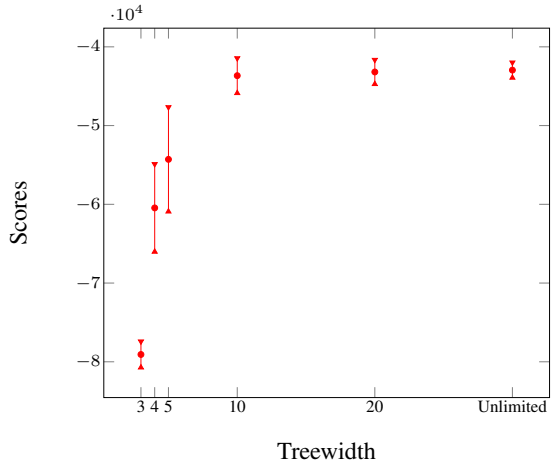


Fig. 8: Scores for the networks learned from dataset Mushroom, bounding the parent set to a maximum of 4 and using a ess score of 1

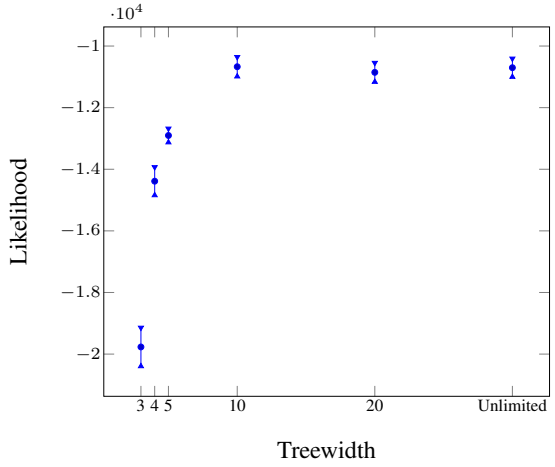


Fig. 9: Likelihood for the networks learned from dataset Mushroom, bounding the parent set to a maximum of 4 and using a ESS score of 0.5

The treewidth of the unbounded network for the dataset Mushroom stayed between 6 and 9 (we give an interval since different folds could yield different treewidths). In fact, our results (Figures 7 and 9) shows that the network‘s performance improves as the treewidth rises and stays at its best between treewidths 5 and 10. Allowing higher treewidths has no impact in the likelihood. By looking at the error gaps in Table II we can see that decreasing the bound on treewidth had a significant impact on the linear program’s performance. If we look at graphs on Figures 8 and 10 we can see that the score of the network follows the same behavior and because of that we can say that the poor performance of the network bounded on treewidth 3 is due to poor ability to learning a good network rather than poor generalization. Varying the ESS parameter on the score function had little differences in the learned networks.

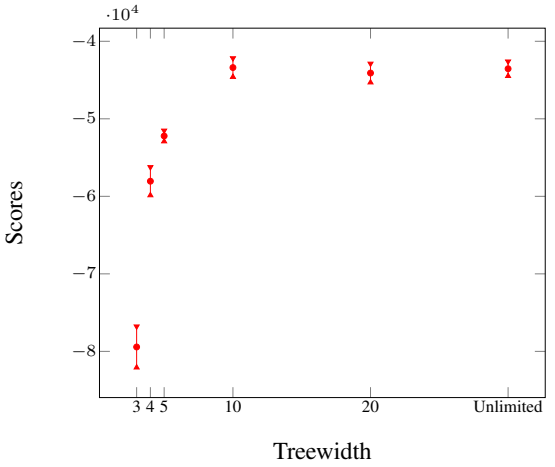


Fig. 10: Scores for the networks learned from dataset Mushroom, bounding the parent set to a maximum of 4 and using a ESS score of 0.5

Parents Limit	3	4	
ESS	1	0.5	1
	Avg. GAP	Avg. GAP	Avg. GAP
tw 03	25.11%	21.25%	27.32%
tw 04	22.71%	20.25%	25.46%
tw 05	23.09%	19.90%	22.82%
tw 10	19.90%	18.94%	20.54%
tw 15	18.98%	17.50%	19.06%
tw 20	18.48%	16.80%	19.10%
tw 25	18.42%	16.58%	18.11%
tw 30	18.26%	16.65%	18.27%
tw unlm.	18.36%	16.54%	18.09%

TABLE III: Remaining GAP values for the optimal solution for all MILP executions of the Dataset WDBC

The overall treewidth of the (unbounded) WDBC dataset stayed between 7 and 9. Again, in Figures 11 and 13 we can see that bounding the treewidth even further impacted the network‘s performance, although it is hard to say if this is due to poor performance of the network or high complexity in the learning task (see the gap values in Table III). The WDBC dataset showed the same assintotic behavior of Mushroom dataset, but with higher variance, and also, the likelihood of the network with an ESS parameter of 0.5 (Figure 13) showed similar result but a much higher variance.

We couldn‘t find the treewidth of unbounded networks for the audio dataset. As we can see from Figure 15 the audio dataset didn‘t followed the expected assintotic behavior and our learned networks had a significant higher likelihood with low treewidths, even better than the unbounded performance. This shows that, when the learning algorithm can find a good solution, constraining the treewidth can lead to a better generalization. For the same reason, when the learning algorithm can‘t find good solutions (i.e. treewidths 10 to 50 in Figure 15), the network have a poor behavior no matter the treewidth.

We tested the Audio training data with an algorithm that learns network structure with no bound on the treewidth [24]. Since it doesn‘t have this restriction it was expected that the found solution would have a better performance. Indeed,

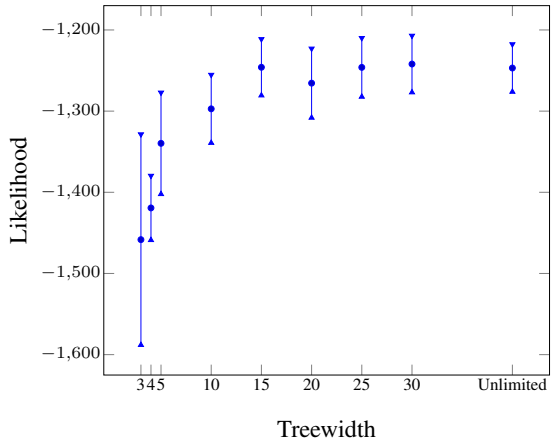


Fig. 11: Likelihood for the networks learned from dataset WDBC, bounding the parent set to a maximum of 4 and using a ESS score of 1

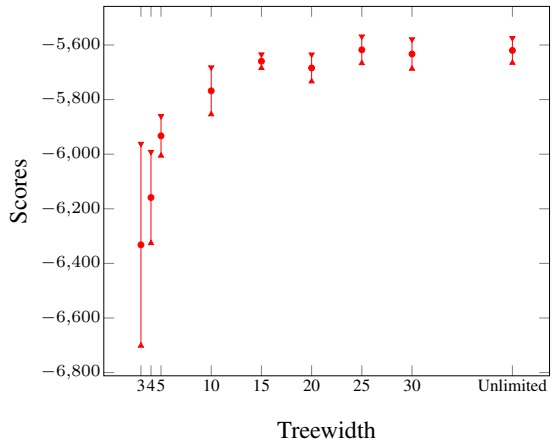


Fig. 12: Scores for the networks learned from dataset WDBC, bounding the parent set to a maximum of 4 and using a ESS score of 1

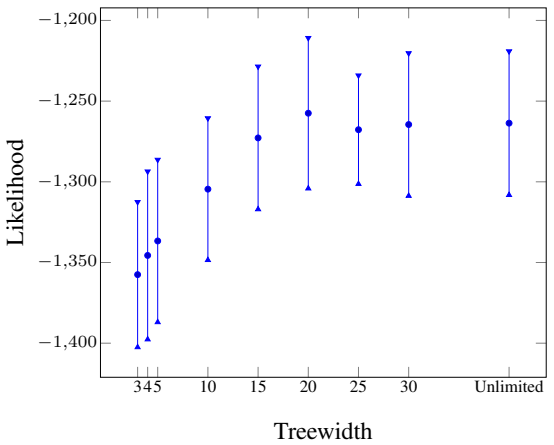


Fig. 13: Likelihood for the networks learned from dataset WDBC, bounding the parent set to a maximum of 4 and using a ESS score of 0.5

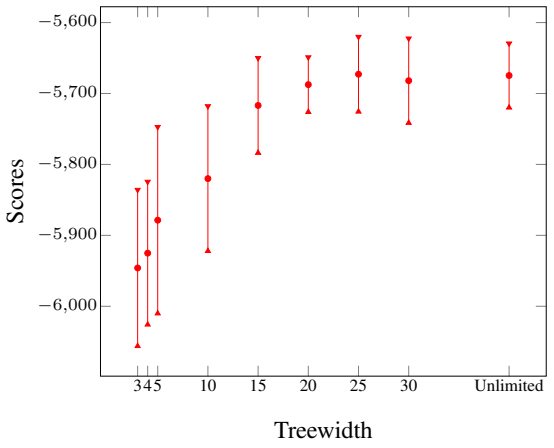


Fig. 14: Scores for the networks learned from dataset WDBC, bounding the parent set to a maximum of 4 and using a ESS score of 0.5

we found likelihood scores of up to 30% better than our learning algorithm but unfortunately, we can't achieve this same performance in leaning structures of bounded treewidth.

V. Conclusion

Overall our results showed that by considering only the likelihood of the learned network to the test data, limiting the treewidth have not outright improved the network's performance as expected. It also showed that the relevance of higher limits on treewidth only starts to matter with high dimension datasets, the same datasets for which the current methods of learning networks of bounded treewidth behave very poorly. Still, more experiments are necessary to distinguish the effect of bounded treewidht and the effect of poor performance of learning algorithms (i.e. experiments that allow a lower the gap on the found solutions). We also suggest a further evaluation of performance by measuring the posterior inference capabilities of the learned network, as well as its behavior in multi label classification problems.

Acknowledgment

The first author of this paper was supported by CNPq, grant 165873/2014-0. The third author is partially supported by CNPq. Experiments were made using resources of the LCCA — Laboratory of Advanced Scientific Computation of the University of São Paulo.

References

[1] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge: MIT Press, 2009.

[2] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning {B}ayesian networks: The combination of knowledge and statistical data," *Mach. Learning*, vol. 20, no. 3, pp. 197–243, 1995.

Parents Limit	3	4
ESS	1	1
	Avg. GAP	Avg. GAP
tw 03	13.37%	19.35%
tw 04	14.44%	16.77%
tw 05	14.40%	20.47%
tw 10	28.40%	30.79%
tw 15	28.40%	30.72%
tw 20	28.40%	30.79%
tw 25	28.40%	30.79%
tw 30	25.42%	30.79%
tw 35	25.41%	30.79%
tw 40	22.30%	30.79%
tw 45	20.83%	30.79%
tw 50	17.96%	30.55%
tw 55	14.38%	23.64%
tw 60	12.02%	20.15%
tw unlm.	20.92%	26.92%

TABLE IV: Remaining GAP values for the optimal solution for all MILP executions of the Dataset Audio

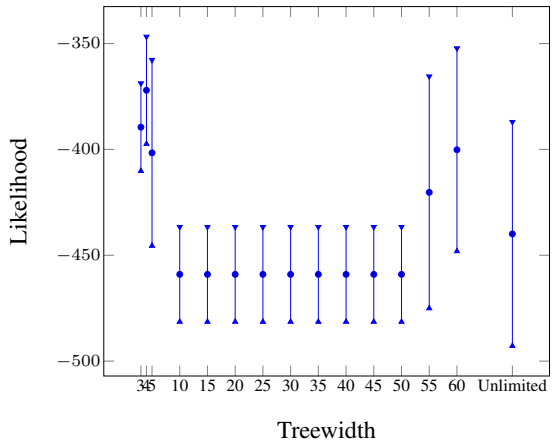


Fig. 15: Likelihood for the networks learned from dataset Audio, bounding the parent set to a maximum of 4 and using a ESS score of 1

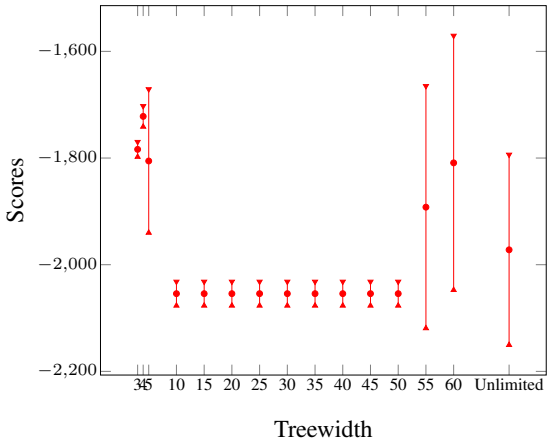


Fig. 16: Scores for the networks learned from dataset Audio, bounding the parent set to a maximum of 4 and using a ESS score of 1

[3] G. F. G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, no. 4, pp. 309–347, Oct. 1992.

[4] A. Antonucci, G. Corani, D. D. Mauá, and S. Gabaglio, "An Ensemble of {B}ayesian Networks for Multilabel Classification," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence ({IJCAI})*, 2013, pp. 1220–1225.

[5] V. Chandrasekaran, N. Srebro, and P. Harsha, "Complexity of Inference in Graphical Models," in *Proc. 24th Conf. on Uncertainty in {AI}*, 2008, pp. 70–78.

[6] J. H. P. Kwisthout, H. L. Bodlaender, and L. C. van der Gaag, "The Necessity of Bounded Treewidth for Efficient Inference in Bayesian Networks," in *Proc. 19th European Conf. on AI*, 2010, pp. 237–242.

[7] J. Kwisthout, "Treewidth and the Computational Complexity of MAP Approximations," *Probabilistic Graphical Models*, vol. 8754, pp. 271–285, 2014.

[8] R. Gens and D. Pedro, "Learning the Structure of Sum-Product Networks," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 873–880. [Online]. Available: http://machinelearning.wustl.edu/mlpapers/papers/icml2013_gens13

[9] S. Nie, D. D. Maua, C. P. de Campos, Q. Ji, D. D. Mauá, C. P. de Campos, and Q. Ji, "Advances in Learning Bayesian Networks of Bounded Treewidth," in *Advances in Neural Information Processing Systems 27*, Jun. 2014, p. 23.

[10] J. Berg, M. Järvisalo, and B. Malone, "Learning Optimal Bounded Treewidth {B}ayesian Networks via Maximum Satisfiability," in *Proc. 17th Int. Conf. on {AI} and Stat.*, 2014, pp. 86–95.

[11] G. Elidan and S. Gould, "Learning Bounded Treewidth Bayesian Networks," *J. of Mach. Learning Res.*, vol. 9, pp. 2699–2731, 2008.

[12] J. H. Korhonen and P. Parviainen, "Exact Learning of Bounded Tree-width {B}ayesian Networks," in *Proc. 16th Int. Conf. on {AI} and Stat.*, 2013, pp. 370–378.

[13] P. Parviainen, H. S. Farahani, and J. Lagergren, "Learning Bounded Tree-width Bayesian Networks using Integer Linear Programming," in *Proc. 17th Int. Conf. on {AI} and Stat.*, 2014, pp. 751–759.

[14] S. Arnborg, D. Corneil, and A. Proskurowski, "Complexity of finding embeddings in a k-tree," *SIAM J. on Matrix Analysis and Applications*, vol. 8, no. 2, pp. 277–284, 1987.

[15] M. Teyssier and D. Koller, "Ordering-based search: A simple and effective algorithm for learning Bayesian networks," in *Proc. 21st Conf. on Uncertainty in {AI}*, 2005, pp. 584–590.

[16] N. Robertson and P. Seymour, "Graph minors. II. Algorithmic aspects of tree-width," *Journal of algorithms*, vol. 7, no. 3, pp. 309–322, 1986.

[17] G. Schwarz, "Estimating the dimension of a model," *Annals of Stat.*, vol. 6, no. 2, pp. 461–464, 1978.

[18] P. Larranaga and C. Kuijpers, "Learning Bayesian network structures by searching for the best ordering with genetic algorithms," *Systems, Man and Cybernetics*, vol. 26, no. 4, pp. 487–493, 1996.

[19] I. Tsamardinos, L. Brown, and C. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," *Machine learning*, vol. 65, no. 1, pp. 31–78, 2006.

[20] A. Grigoriev, E. Hans, and U. Natalya, "Integer linear programming formulations for treewidth," Maastricht University, Maastricht Research School of Economics of Technology and Organization (METEOR), Tech. Rep., 2011.

[21] M. Bartlett and J. Cussens, "Integer Linear Programming for the Bayesian network structure learning problem," in *Artificial Intelligence*, Mar. 2015, p. 14.

[22] J. Cussens, "Bayesian network learning with cutting planes," in *Proc. 27th Conf. on Uncertainty in AI*, 2011, pp. 153–160.

[23] T. van Dijk, J. van den Heuvel, and W. Slob, "Computing treewidth with LibTW," University of Utrecht, Tech. Rep., 2006.

[24] M. Barlett and J. Cussens, "Advances in Bayesian Network Learning using Integer Programming," in *Proc. 29th Conf. on Uncertainty in AI*, 2013, pp. 182–191.

APÊNDICE B – REPRODUÇÃO DE ARTIGO: ANÁLISE DE PERFORMANCE EM REDES BAYESIANAS COM LARGURA DE ÁRVORE LIMITADA

Publicação 2: Resumo em Workshop

Título: Análise de Performance em Redes Bayesianas com Largura de Árvore Limitada (2 páginas)

Autores: Fabio H. S. Machado

Conferência: IV Workshop de Pós-Graduação da Área de Concentração "Engenharia de Computação", WPG-EC 2015

Observações: O artigo recebeu o prêmio de "Melhor trabalho de mestrado".

Análise de Desempenho em Redes Bayesianas com Largura de Árvore Limitada

Fabio H. S. Machado
Escola Politécnica
Universidade de São Paulo

Fabio G. Cozman
Escola Politécnica
Universidade de São Paulo

***Abstract*—Bounding the treewidth of Bayesian Networks has been claimed to guarantee polynomial-time inference with little harm to accuracy. However, there has been little empirical evidence to support that claim. In this work we study empirically the effect of bounding treewidth on generalization ability and inference quality. Our preliminary results suggest that adding a constraint to treewidth may decrease the model performance on unseen data.**

I. Objetivos e Motivação

Muitos trabalhos na literatura sugerem que restringir o espaço de busca no aprendizado de Redes Bayesianas, especificamente limitando a largura de árvore da sua estrutura, pode levar no aumento do desempenho dessas redes quando utilizadas em aplicações reais, contribuindo tanto para a sua capacidade de generalização, quanto para otimizar o tempo de execução dos algoritmos de inferência que usam essas redes [1], [2]. No entanto, há pouca evidências empírica que corroborem com estas afirmações. o objetivo deste trabalho é analisar empiricamente o desempenho das redes aprendidas considerando três condições: sua capacidade de generalização, a qualidade da inferência e a velocidade da inferência.

II. Revisão da Literatura

Um *grafo acíclico dirigido* (ou *DAG*, da sua sigla em inglês) é um grafo cuja estrutura é composta por vértices direcionados e o qual não apresenta ciclos dirigidos. Um grafo moral é o grafo não dirigido obtido de um DAG ao conectar-se nós que possuam descendentes em comum e retirando-se a direção dos vértices. Um grafo possui uma "corda" quando há um ciclo neste grafo e dois de seus nós são ligados por um vértice que não pertença ao ciclo. Um grafo é dito "cordal" quando todos os seus ciclos de tamanho 4 ou maiores possuem uma corda. Todo grafo pode se tornar um grafo cordal adicionando-se novos vértices. Um *clique* é um subconjunto dos nós de um grafo não direcionado que formam um subgrafo completo (todos os nós estão conectados par a par). Uma ordem de eliminação perfeita é a ordenação dos nós de um grafo de forma que seus vizinhos ascendentes sempre formam um clique. Um grafo só possui uma ordem de eliminação perfeita se, e somente se, ele for um grafo cordal. A largura de árvore de um grafo é calculada como a maior quantidade de vizinho ascendentes (com base na sua ordem de eliminação perfeita) dos nós de um grafo cordal.

Redes Bayesianas são modelos gráficos probabilísticos que codificam relações de (in)dependência e distribuições conjuntas de probabilidade em um conjunto de variáveis aleatórias. Formalmente, uma Rede Bayesiana é uma tripla $\{V, G, \Theta\}$, onde $V = \{X_1, X_2, ..., X_n\}$ é um conjunto finito de variáveis aleatórias; $G = \{V, E\}$ é um DAG cujas arestas E representam relações de dependência entre as variáveis V; e $\Theta = \{\theta_i(x_i, x_{Pa_i}) = P(x_i|x_{Pa_i})\}$ é uma coleção de distribuição de probabilidades condicionais, uma para cada valor x_i da variável X_i e configuração x_{Pa_i} do conjunto de pais Pa_i daquela variável no grafo. Assume-se que uma variável X_i é independente de todos os seus não descendentes dados seus pais no grafo [3]. A largura de árvore de uma Rede Bayesiana é a largura de árvore de sua estrutura G. A figura 1 exemplifica a estrutura de uma Rede Bayesiana sobre 9 variáveis. A figura 2 mostra o grafo cordal, e sua ordem de eliminação perfeita, obtido da estrutura mostrada na figura 1.

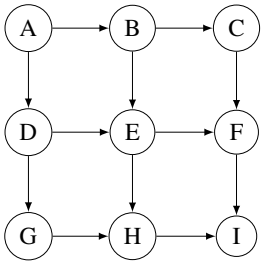


Fig. 1: Uma estrutura com 9 variáveis.

Aprender uma Rede Bayesiana consiste em estimar Θ e G dado um conjunto de dados que contenha observações históricas de V [4]. Devido à complexidade de se aprender uma estrutura gráfica a partir dos dados, os algoritmos estados da arte em aprendizado de Redes Bayesianas utilizam-se de técnicas de programação inteira (MILP, ou *Mixed Integer Linear Programing* na sigla em Inglês) modelando o problema sob a forma de equações matemáticas que devem ser otimizadas.

III. Metodologia e Desenvolvimento

O objetivo deste trabalho é avaliar o real efeito que limitar a largura da árvore de uma rede exerce em seu desempenho. Por desempenho de um rede, no referimos a três fatores: sua capacidade de generalização, calculada através da sua verossimilhança a uma base de testes (diferente da usada no treinamento); a qualidade da inferência, em termos de quantas inferências corretas ela é capaz de prever em modelos conhecidos; e, por fim, o real ganho de velocidade de inferência das redes aprendidas em comparação com os modelos sem limite de largura de árvore.

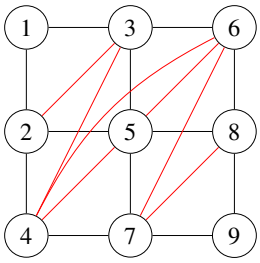


Fig. 2: Uma ordem de elminação perfeita para o grafo cordal obtido da Fig. 1, vértices em vermelho são vértices adicionados para transformar a estrutura em um grafo cordal.

Nome do conjunto	Quantidade de variáveis	Amostras
Mushroom	22	8124
WDBC	31	569
Audio	62	200

TABLE I: Conjuntos de Dados

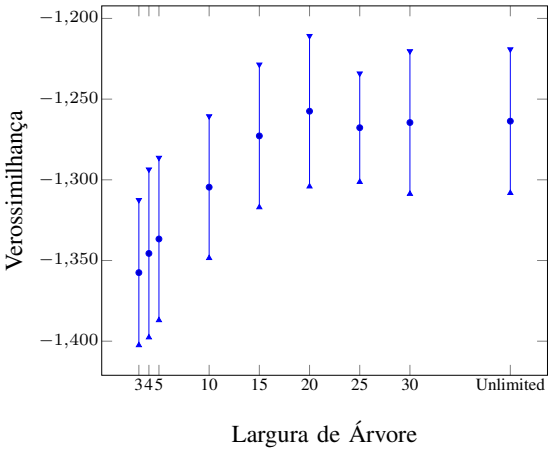


Fig. 3: Comparação da verossimilhança das redes aprendidas para o conjunto de dados WDBC.

Foram escolhidos alguns conjuntos de dados do repositório da UCI [1]. Em testes preliminares, percebemos que conjuntos muito pequenos (com 15 variáveis ou menos) geralmente possuem uma estrutura "ótima" com largura de árvore muito baixa, e assim, limitar a largura nesse tipo de modelo não gerava nenhum impacto. Ao mesmo tempo, conjuntos de dados muito grandes (com 100 ou mais variáveis) tornam a abordagem computacionalmente infactível. Sendo assim selecionamos três conjuntos de dados dentro destas restrições, conforme mostrados na tabela I. Para cada conjunto de dados, foram aprendidas redes cuja largura máxima de árvore era limitada em 3, 4, 5 e incrementos de 5 até o número máximo de variáveis daquele dataset. Para cada iteração de largura de árvore nós aplicamos avaliação cruzada com 5 subconjuntos a fim de obter uma estimativa melhor de qual a verossimilhança esperada daquele caso de teste. O aprendizado das Redes Bayesianas com limite de largura de árvore foi formulado através de um programa MILP conforme descrito em [5].

IV. Experimentos e resultados preliminares

O gráfico da figura 3 mostra a comparação do desempenho de diferentes redes aprendidas utilizando o conjunto de dados WDBC. No geral, as redes aprendidas sem limite da largura de árvore, tiveram largura em torno 7 e 9, bem menor que o número de variáveis do conjunto. É importante notar que com essa quantidade de variáveis no conjunto de dados, os algoritmos de aprendizado não são capazes de encontrar soluções ótimas, e essa complexidade aumenta ainda mais à medida que se limita mais a largura de árvore do grafo. Sendo assim, ainda é difícil afirmar se o baixo desempenho em redes com largura muito limitada se deve à alta complexidade do problema de aprendizado ou à real precariedade de generalização da rede nessas condições. Mesmo assim, os gráficos apontam que limitar demais a largura de árvore acarreta em perda de desempenho quando medimos a sua capacidade de generalização.

[1] http://archive.ics.uci.edu/ml/

V. Conclusão

Nossos resultados até agora apontam que, considerando-se apenas a verossimilhança das redes aprendidas em dados de teste, limitar a largura de árvore não acarreta em um ganho imediato de desempenho. Eles também mostram que limitar a largura de árvore só começa a ter relevância em bases de dados com um grande número de variáveis, o mesmo tipo de base de dados nas quais os atuais algoritmos de aprendizagem encaram grandes problemas de complexidade. Assim, novos experimentos são necessários para se distinguir o baixo desempenho das redes de fato ao limite da largura de árvore ou à desempenho do algoritmo de aprendizagem. Também é objetivo da pesquisa medir o desempenho das redes aprendidas em relação à sua capacidade de fazer inferências de distribuições de probabilidades sob as variáveis.

Agradecimentos

O mestrando é suportado pelo CNPq, bolsa de mestrado, processo 165873/2014-0. Os experimentos foram realizados utilizando-se de recursos do LCCA - Laboratório de Computação Científica Avançada da USP.

References

[1] J. Kwisthout, "Treewidth and the Computational Complexity of MAP Approximations," *Probabilistic Graphical Models*, vol. 8754, pp. 271–285, 2014.

[2] J. H. P. Kwisthout, H. L. Bodlaender, and L. C. van der Gaag, "The Necessity of Bounded Treewidth for Efficient Inference in Bayesian Networks," in *Proc. 19th European Conf. on AI*, 2010, pp. 237–242.

[3] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge: MIT Press, 2009.

[4] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning {B}ayesian networks: The combination of knowledge and statistical data," *Mach. Learning*, vol. 20, no. 3, pp. 197–243, 1995.

[5] S. Nie, D. D. Maua, C. P. de Campos, Q. Ji, D. D. Mauá, C. P. de Campos, and Q. Ji, "Advances in Learning Bayesian Networks of Bounded Treewidth," in *Advances in Neural Information Processing Systems 27*, Jun. 2014, p. 23.