

# Geração uniforme de $k$ -trees para aprendizado de redes bayesianas

Tiago Madeira  
<madeira@ime.usp.br>

Supervisor: Prof. Dr. Denis Deratani Mauá

Bacharelado em Ciência da Computação  
Instituto de Matemática e Estatística  
Universidade de São Paulo

Novembro de 2016

# No que consiste o trabalho?

Estudo sobre amostragem uniforme de  $k$ -trees e seu uso no aprendizado da estrutura de redes bayesianas com *treewidth* limitado.

# Por que estudar $k$ -trees?

Há interesse considerável em desenvolver ferramentas eficientes para manipular  $k$ -trees, porque **problemas NP-difíceis são resolvidos em tempo polinomial** em  $k$ -trees e subgrafos de  $k$ -trees.

Alguns exemplos<sup>1</sup>:

- Encontrar tamanho máximo dos conjuntos independentes;
- Computar tamanho mínimo dos conjuntos dominantes;
- Calcular número cromático;
- Determinar se tem um ciclo hamiltoniano.

---

<sup>1</sup>ARNBORG S., PROSKUROWSKI A. Linear time algorithms for NP-Hard problems restricted to partial  $k$ -trees. *Discrete Applied Mathematics*, 23:11–24, 1989.

# Por que gerar $k$ -trees?

Há muitas razões, como por exemplo para testar a eficácia de algoritmos aproximados.

O problema que desperta nosso interesse é o **aprendizado de redes bayesianas**.

# O que foi feito?

- Implementação do algoritmo de Caminiti *et al.* (2010)<sup>2</sup> para **codificar  $k$ -trees de forma bijetiva em tempo linear**.
- Implementação de algoritmo para **amostrar  $k$ -trees uniformemente** e testes para comprovar seu funcionamento.
- Estudo sobre **aprendizado de redes bayesianas com treewidth limitado** por meio da amostragem uniforme de  $k$ -trees conforme artigo de Nie *et al.* (2014)<sup>3</sup>.
- **Comparação deste método** para aprender redes bayesianas com o estado da arte.

---

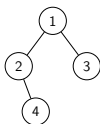
<sup>2</sup>CAMINITI S., FUSCO E. G., PETRESCHI R. Bijective linear time coding and decoding for  $k$ -trees. *Theory of Computing Systems*, 46:284–300, 2010.

<sup>3</sup>NIE S., MAUÁ D. D., CAMPOS C. P., JI Q. Advances in learning bayesian networks of bounded treewidth. *CoRR*, abs/1406.1411, 2014.

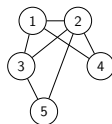
# Primeiramente, o que são $k$ -trees?

Uma  $k$ -tree é definida da seguinte forma recursiva<sup>4</sup>:

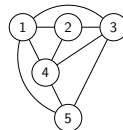
- Um grafo completo com  $k$  vértices é uma  $k$ -tree.
- Se  $T'_k = (V, E)$  é uma  $k$ -tree,  $K \subseteq V$  é um  $k$ -clique e  $v \notin V$ , então  $T_k = (V \cup \{v\}, E \cup \{(v, x) \mid x \in K\})$  é uma  $k$ -tree.



(a)



(b)



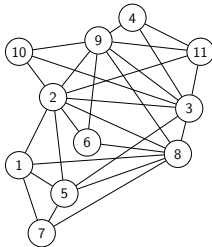
(c)

**Figura:** (a) Uma 1-tree (ou seja, uma árvore comum) com 4 vértices. (b) Uma 2-tree com 5 vértices. (c) Uma 3-tree com 5 vértices.

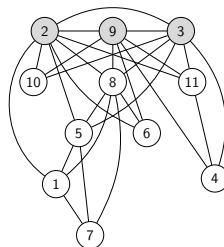
<sup>4</sup>HARARY F., PALMER E. M. On acyclic simplicial complexes. *Mathematika*, 15:115–122, 1968.

## $k$ -trees enraizadas

Uma  $k$ -tree **enraizada** é uma  $k$ -tree com um  $k$ -clique destacado  $R = \{r_1, r_2, \dots, r_k\}$  que é chamado de **raiz** da  $k$ -tree enraizada.



(a)



(b)

**Figura:** (a) Uma 3-tree  $T_3$  com 11 vértices. (b) A mesma 3-tree ( $T_3$ ) enraizada no 3-clique  $\{2, 3, 9\}$ .


## $k$ -tree e $treewidth$

Dado um grafo  $G = (V, E)$ , seu  **$treewidth$**  é um inteiro definido da seguinte forma:

- Se  $G$  é um **grafo cordal**, então seu  $treewidth$  é o tamanho do seu maior clique menos 1.
- Se  $G$  é um **grafo não-dirigido arbitrário**, então seu  $treewidth$  é o mínimo entre os  $treewidth$  de todas as suas cordalizações.
- Se  $G$  é um **DAG**, então seu  $treewidth$  é o  $treewidth$  do seu grafo moral.

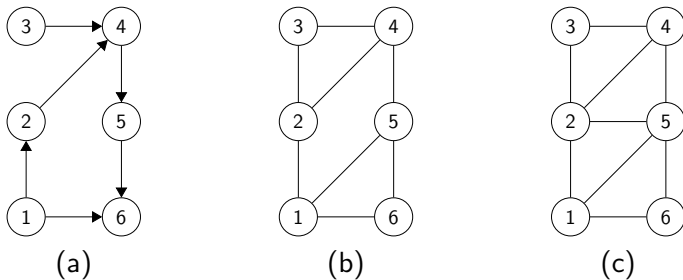
Um subgrafo de uma  $k$ -tree é chamado de **partial  $k$ -tree**. Um grafo é uma *partial  $k$ -tree* se e só se ele tem  $treewidth$  menor ou igual a  $k^5$ .

---

<sup>5</sup>BODLAENDER, H. L.  $Treewidth$ : Structure and algorithms. *Structural Information and Communication Complexity*, 4474:11–25, 2007. 



## Ilustração de *treewidth*



**Figura:** (a) Um grafo acíclico dirigido  $G$ . (b) Grafo moral  $G'$  de  $G$ , obtido conectando-se todo par de vértices com um filho em comum e retirando-se a direção das arestas. (c) Um dos grafos cordais obtidos por meio da cordalização de  $G'$ . O *treewidth* dos três grafos mostrados na figura é 2.

# A relação entre geração e codificação

O problema de **gerar**  $k$ -trees está intimamente relacionado ao problema de **codificá-las e decodificá-las**.

Se há uma codificação bijetiva que associa  $k$ -trees a *strings*, basta gerar *strings* uniformemente aleatórias para gerar  $k$ -trees uniformemente aleatórias.

# Codificação de $k$ -trees

- Em 1889, Cayley<sup>6</sup> demonstrou que para um conjunto de  $n$  vértices existem  $n^{n-2}$  árvores possíveis. Desde lá, foram criados vários códigos para árvores, como o de Prüfer<sup>7</sup>.
- Em 1970, Rényi e Rényi apresentaram uma codificação redundante (ou seja, não bijetiva) para um subconjunto de  $k$ -trees rotuladas que chamamos de  $k$ -trees de Rényi<sup>8</sup>.  
Definição: Uma  **$k$ -tree de Rényi**  $R_k$  é uma  $k$ -tree enraizada com  $n$  vértices rotulados em  $[1, n]$  e raiz  $\{n - k + 1, \dots, n\}$ .

---

<sup>6</sup>CAYLEY, A. A theorem on trees. *Quart J. Math*, 23:376–378, 1889.

<sup>7</sup>PRÜFER, H. Neuer beweis eines satzes über permutationen. *Archiv der Mat. und Physik*, 27:142–144, 1918.

<sup>8</sup>RÉNYI, C., RÉNYI, A. The prüfer code for  $k$ -trees. *Combinatorial Theory and its Applications*, 945–971, 1970.

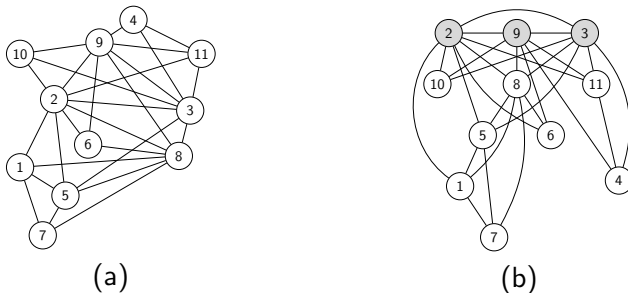
## A solução de Caminiti *et al.*

- Apenas em 2008 surgiu um código bijetivo para  $k$ -trees com algoritmos lineares de codificação e decodificação. Esses algoritmos, propostos por Caminiti *et al.*, foram implementados neste trabalho.
- O código é formado por uma permutação de tamanho  $k$  e uma generalização do *Dandelion Code*<sup>9</sup>. A codificação das  $k$ -trees associa elementos em  $\mathcal{T}_k^n$  (conjunto das  $k$ -trees com  $n$  vértices) com elementos em:

$$\mathcal{A}_k^n = \binom{[1, n]}{k} \times (\{(0, \varepsilon)\} \cup ([1, n - k] \times [1, k]))^{n-k-2}$$

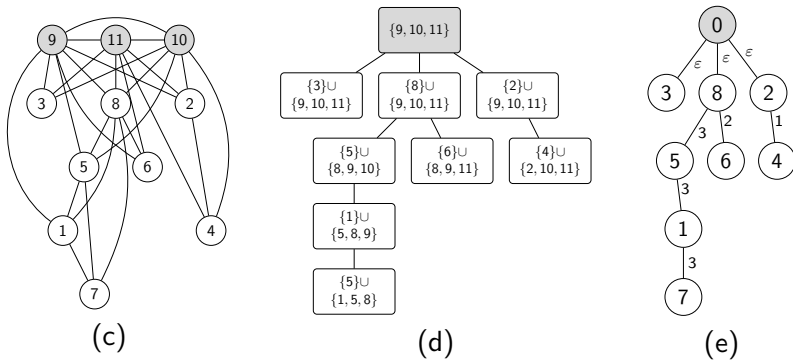
<sup>9</sup>EĞECIOĞLU Ö., REMMEL J. B. Bijections for cayley trees, spanning trees, and their  $q$ -analogues. *Journal of Combinatorial Theory*, 42:15–30, 1986.

## Transformações de Caminiti *et al.* (1/3)



**Figura:** (a) Uma 3-tree  $T_3$  com 11 vértices. (b) A mesma 3-tree ( $T_3$ ) enraizada no 3-clique  $\{2, 3, 9\}$ .

## Transformações de Caminiti *et al.* (2/3)



**Figura:** (c)  $R_3$ , 3-tree de Rényi gerada por meio da re-rotulação de  $T_3$ .  
 (d) O esqueleto de  $R_3$ . (e) A árvore característica de  $R_3$ .

## Transformações de Caminiti *et al.* (3/3)

*Dandelion Code* generalizado correspondente a  $T_3$ :

$$\{2, 3, 9\}, [(0, \varepsilon), (2, 0), (8, 2), (8, 1), (1, 2), (5, 2)]$$

## Geração uniforme de $k$ -trees

Geramos  $k$ -trees uniformemente por meio da geração uniforme de *strings* em:

$$\mathcal{A}_k^n = \binom{[1, n]}{k} \times (\{(0, \varepsilon)\} \cup ([1, n - k] \times [1, k]))^{n-k-2}$$

A biblioteca que desenvolvemos<sup>10</sup> tem três utilitários que rodam na linha de comando:

- `code_ktree` (para codificar  $k$ -trees)
- `decode_ktree` (para decodificar *Dandelion Codes*)
- `generate_ktree` (para gerar uma  $k$ -tree uniformemente)

---

<sup>10</sup>Implementada em Go e disponível em <https://github.com/tmadeira/tcc>



## Testes (1/2)

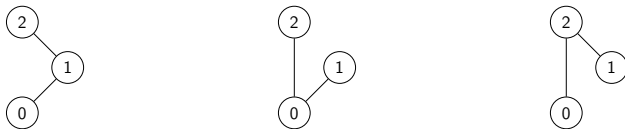
- Todos os pacotes desenvolvidos neste trabalho possuem testes unitários que podem ser executados usando o utilitário `go test`. **96% das linhas do código são cobertas por testes<sup>11</sup>.**
- Para mostrar que nossa implementação gera  $k$ -trees aleatórias **corretamente e uniformemente**, realizamos dezenas de milhares de testes com  $n$  e  $k$  pequenos.

---

<sup>11</sup>O relatório de cobertura da ferramenta *Coveralls* pode ser visto em:  
<https://coveralls.io/github/tmadeira/tcc?branch=master>

## Testes (2/2)

**Exemplo:** Com  $n = 3$ ,  $k = 1$  existem 3  $k$ -trees rotuladas distintas.



**Figura:** Representação das três 1-trees rotuladas distintas com  $n = 3$  vértices.

Ao executar 10 mil gerações com  $n = 3$  e  $k = 1$  esperamos que as três 1-trees (árvores) apareçam com frequência similar (aprox. 3333). As frequências que obtivemos:

- 3320
- 3335
- 3345

# O que são redes bayesianas?

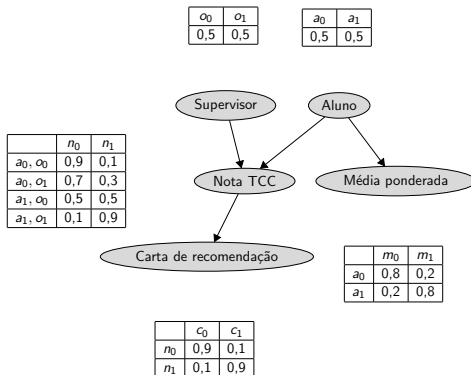
**Redes bayesianas** são modelos probabilísticos gráficos<sup>12</sup> que representam distribuições de probabilidade conjunta e são usados para raciocinar em situações com incerteza.

Formalmente: Seja  $N = \{1, \dots, n\}$  e seja  $X = \{X_i : i \in N\}$  um conjunto de variáveis aleatórias  $X_i$  tomando valores em conjuntos finitos  $\mathcal{X}_i$ . Uma **rede bayesiana** é uma tripla  $(X, G, \theta)$ , onde  $G = (V, E)$  é um DAG (grafo acíclico dirigido, que chamamos de **estrutura** da rede bayesiana) cujos vértices correspondem a variáveis em  $X$  e  $\theta = \{\theta_i(x_i, x_{\pi_i})\}$  é um conjunto de parâmetros numéricos especificando valores de probabilidade condicional  $\theta_i(x_i, x_{\pi_i}) = P(x_i | x_{\pi_i})$  para todo vértice  $i \in V$ , valor  $x_i \in \mathcal{X}_i$  e atribuição  $x_{\pi_i}$  para os pais  $\pi_i$  de  $X_i$  (em  $G$ ).

---

<sup>12</sup>KOLLER D., FRIEDMAN N. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

# Exemplo de rede bayesiana



**Figura:** Exemplo de rede bayesiana com distribuições de probabilidade condicional.

# Aprendizado de redes bayesianas (1/2)

- Aprender uma rede bayesiana se refere ao processo de **produzir a sua estrutura** (i.e., seu DAG) a partir de dados.
- **Inferência em rede bayesiana é NP-difícil até mesmo aproximadamente** e todos os algoritmos conhecidos (exatos e comprovadamente bons) têm complexidade no pior caso exponencial no *treewidth*.
- Resultados empíricos sugerem que **limitar o treewidth pode melhorar a performance** dos modelos e não causa perdas significativas na sua expressividade.
- Por isso estamos interessados em fixar  $k$  e aprender redes bayesianas cujo **DAG tem treewidth limitado a  $k$** .

## Aprendizado de redes bayesianas (2/2)

Função de *score*  $s(G)$  atribui pontuação para cada DAG  $G$  e pode ser escrita como soma de funções de *score* locais:

$$s(G) = \sum_{i \in N} s_i(X_{\pi_i}).$$

Para cada variável, sua pontuação só depende do seu conjunto de pais. Portanto, nosso problema é encontrar  $G^*$  tal que

$$G^* = \arg \max_{G \in \mathcal{G}_{n,k}} \sum_{i \in N} s_i(\pi_i),$$

onde  $\mathcal{G}_{n,k}$  é o conjunto de todos os DAGs de *treewidth* não maior que  $k$ . Esse problema é NP-difícil<sup>13</sup>.

<sup>13</sup>KORHONEN J. H., PARVIAINEN P. Exact learning of bounded tree-width bayesian networks. *Proceedings of the 16th International Conference on AISTATS*, 2013.

## Aprendizado por amostragem de $k$ -trees

A ideia para aprender um DAG por meio da amostragem de  $k$ -trees baseia-se em, para cada  $k$ -tree  $T_k$  amostrada, construir uma ordem parcial  $\sigma$  dos vértices e fazer com que o DAG  $G$  seja consistente com ela e com  $T_k$ .

Construímos a ordem parcial  $\sigma$  a partir do enraizamento da  $k$ -tree num  $k$ -clique qualquer. Em particular, podemos escolher usar a raiz da  $k$ -tree de Rényi que aparece durante a decodificação de um *Dandelion Code* em uma  $k$ -tree.

# Algoritmo para aprender estrutura

**Entrada:**  $n$ ,  $k$  e função de *score*  $s_i$  para cada  $i \in [0, n]$

**Saída:** um DAG  $G^{\text{melhor}}$

- 1 Inicializar  $G^{\text{melhor}}$  como um grafo com  $s(G^{\text{melhor}}) = -\infty$ .
- 2 Repetir até atingir um determinado número de iterações:
  - 1 Gerar  $(Q, S) \in \mathcal{A}_k^n$  e decodificar na árvore característica  $T$ ;
  - 2 Sortear ordem pros vértices do  $k$ -clique raiz e usar função de *score* para calcular os melhores pais para cada um deles;
  - 3 Percorrer  $T$  a partir dos vértices ligados ao  $k$ -clique raiz: para cada  $v$ , é sorteado um lugar para ele em  $\sigma$  e selecionado seu melhor conjunto de pais dentre os vértices predecessores adjacentes, assim como são atualizados os melhores pais dos vértices sucessores adjacentes;
  - 4 Se  $\left(\sum_{i \in [0, n]} s_i(\pi_i^G)\right) = s(G) > s(G^{\text{melhor}})$ , atualiza  $G^{\text{melhor}} = G$ .



# Experimentos

- Algoritmo foi **implementado por João de Santana Brito Junior**, aluno de mestrado do Prof. Denis. Faz uso da biblioteca desenvolvida neste trabalho para amostrar  $k$ -trees.
- Usamos implementação para testar aprendizado por amostragem uniforme de  $k$ -trees com **cinco conjuntos de dados reais** contendo de 64 a 1556 variáveis.
- **Comparamos resultados** com os obtidos por Perez e Mauá<sup>14</sup> para o *Acyclic Selection Order-Based Search (ASOBS)* com *Best First-Based initialization heuristic (BFT)*, abordagem com resultados comparáveis ao estado da arte.

---

<sup>14</sup>PEREZ W., MAUÁ, D. D. Improving acyclic selection order-based bayesian network structure learning. *XIII Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, 169–180, 2016.

# Conjuntos de dados

<i>Dataset</i>	<i>n</i>	<i>N</i>
kdd	64	234954
tretail	135	29387
cr52	889	9100
bbc	1058	2225
ad	1556	3279

**Tabela:** Características dos conjuntos de dados utilizados;  $n$  é o número de variáveis e  $N$  é o número de instâncias.

# Resultados obtidos

Dataset	$k = 4$		$k = 10$		Perez e Mauá	
	Máx.	Média	Máx.	Média	Máx.	Média
kdd	0,0755	$0,0691 \pm 0,0022$	0,1037	$0,1007 \pm 0,0016$	0,1468	$0,1447 \pm 0,0007$
tretail	0,0188	$0,0156 \pm 0,0019$	0,0289	$0,0249 \pm 0,0017$	0,0447	$0,0444 \pm 0,0002$
cr52	0,0203	$0,0171 \pm 0,0011$			0,1617	$0,1598 \pm 0,0010$
bbc	0,0059	$0,0054 \pm 0,0002$			0,0777	$0,0770 \pm 0,0004$
ad	0,0448	$0,0400 \pm 0,0023$			0,7114	$0,7089 \pm 0,0013$

**Tabela:** Performance do aprendizado de redes bayesianas por amostragem de  $k$ -trees e comparação dele com os resultados obtidos por Perez e Mauá.

## Considerações finais (1/2)

- **Principal produto do trabalho:** biblioteca para codificação e geração uniforme de  $k$ -trees em tempo linear. Experimentos no aprendizado de redes bayesianas mostram como pode ser usada na prática. O método é eficiente e funciona com grandes domínios e limite alto de *treewidth*.
- Muitos dos **conceitos estudados são recentes e pouco explorados**. Os dois principais artigos (Caminiti *et al.* e Nie *et al.*) foram publicados respec. em 2010 e 2014. O segundo ressalta que simultaneamente ao seu desenvolvimento foram publicados independentemente outros trabalhos intimamente relacionados.

## Considerações finais (2/2)

- Extensão interessante deste trabalho seria estudar a **geração de DAGs de treewidth limitado de maneira uniforme**. A geração uniforme de  $k$ -trees não resolve esse problema.
- Há **outras amostragens (não uniformes)** que podem ser testadas e comparadas em trabalhos futuros. Outro artigo de Nie *et al.* (2015)<sup>15</sup> sugere um *Distance Preferable Sampling*.
- Embora nova, a linguagem Go se mostrou satisfatória. Acreditamos que suas convenções facilitem o **reaproveitamento futuro do código desenvolvido**.

---

<sup>15</sup>NIE S., CAMPOS C. P., JI Q. *Learning Bounded Treewidth Bayesian Networks via Sampling*, 387–396. Springer International Publishing, Cham, 2015.

# Obrigado.

Esta apresentação, a monografia e os códigos desenvolvidos estão disponíveis em <https://github.com/tmadeira/tcc/>

Perguntas?