Comparing Binary and Standard Probability Trees in Credal Networks Inference

Andrés Cano, Manuel Gómez-Olmedo, Andrés R. Masegosa, Serafín Moral

Department of Computer Science and Artificial Intelligence
University of Granada, Spain

ISIPTA '11 - July 27th, 2011

Research group

- I am associate professor at the University of Granada.
- I am a member of the *Uncertainty Treatment in Artificial Intelligence* research group (dpt. Computer Science and Artificial Intelligence).
  - Probabilistic graphical models: Learning and inference
  - Imprecise probabilities
  - Data mining
  - Information retrieval
- I am interested in topics like inference in probabilistic graphical models and supervised classification.

Research group

- I am associate professor at the University of Granada.
- I am a member of the *Uncertainty Treatment in Artificial Intelligence* research group (dpt. Computer Science and Artificial Intelligence).
    - Probabilistic graphical models: Learning and inference
    - Imprecise probabilities
    - Data mining
    - Information retrieval
- I am interested in topics like inference in probabilistic graphical models and supervised classification.

Research group

- I am associate professor at the University of Granada.
- I am a member of the *Uncertainty Treatment in Artificial Intelligence* research group (dpt. Computer Science and Artificial Intelligence).
    - Probabilistic graphical models: Learning and inference
    - Imprecise probabilities
    - Data mining
    - Information retrieval
- I am interested in topics like inference in probabilistic graphical models and supervised classification.

Goal of the paper

- The goal of this paper is to use Binary Probability Trees in the problem of credal networks inference and to compare them with the use of Standard Probability Trees.
- We are interested in algorithms for inference in the *strong extension* of a **credal network**.

$K(Y)$	y_1	y_2
r_1	0.1	0.9
r_2	0.2	0.8

$K(X\|Y=y_1)$	x_1	x_2
p_1	0.2	0.8
p_2	0.3	0.7

$K(X\|Y=y_2)$	x_1	x_2
q_1	0.4	0.6
q_2	0.6	0.4

$\Rightarrow p(X|\mathbf{E})?$

- We use a credal set $K(X_i|\Pi_i=\pi_i)$ for each configuration π_i of Π_i (*separately specified credal sets*).

Goal of the paper

- The goal of this paper is to use Binary Probability Trees in the problem of credal networks inference and to compare them with the use of Standard Probability Trees.
- We are interested in algorithms for inference in the *strong extension* of a **credal network**.

$K(Y)$	y_1	y_2
r_1	0.1	0.9
r_2	0.2	0.8

$K(X\|Y=y_1)$	x_1	x_2
p_1	0.2	0.8
p_2	0.3	0.7

$K(X\|Y=y_2)$	x_1	x_2
q_1	0.4	0.6
q_2	0.6	0.4

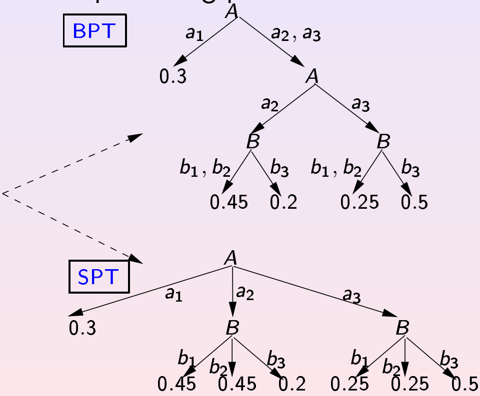
$\Rightarrow p(X|\mathbf{E})$?

- We use a credal set $K(X_i|\Pi_i = \pi_i)$ for each configuration π_i of Π_i (*separately specified credal sets*).

Standard and Binary Probability Trees

- Standard probability trees (SPTs) and Binary probability trees (BPTs) are suitable data structures for representing potentials.

$P(A\|B)$	b_1	b_2	b_3
a_1	0.3	0.3	0.3
a_2	0.45	0.45	0.2
a_3	0.25	0.25	0.5

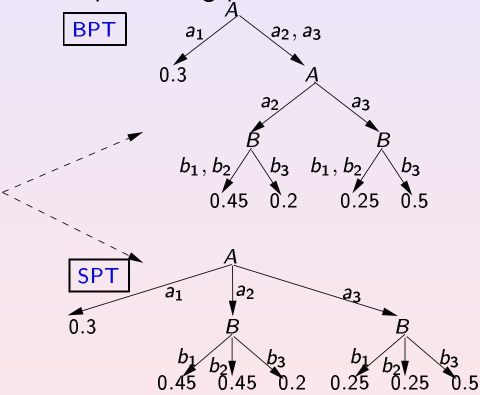


- They allow to control the **accuracy** of inference algorithms by means of the pruning operation.

Standard and Binary Probability Trees

- Standard probability trees (SPTs) and Binary probability trees (BPTs) are suitable data structures for representing potentials.

$P(A \mid B)$	b_1	b_2	b_3
a_1	0.3	0.3	0.3
a_2	0.45	0.45	0.2
a_3	0.25	0.25	0.5



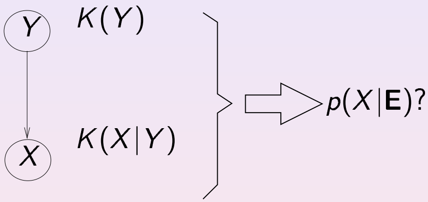
- They allow to control the **accuracy** of inference algorithms by means of the pruning operation.

Operations with Standard and Binary Probability Trees

- Inference algorithms require three operations over potentials.
- In previous works we have defined detailed algorithms for making these operations directly over SPTs and BPTs.
    - **Restriction** ($\mathcal{BT}^{R(\mathbf{x}_J)}$): Get the part of $\mathcal{BT}$ consistent with configuration $\mathbf{x}_J$
    - **Combination** ($\mathcal{BT}1 \otimes \mathcal{BT}2$): Get a BPT that represents the product of the potentials represented by $\mathcal{BT}1$ and $\mathcal{BT}2$
    - **Marginalization**: $\mathcal{BT}^{\downarrow \mathbf{X}_I \setminus \{X_j\}}$: Sum over the variable X_j

Inference in credal networks

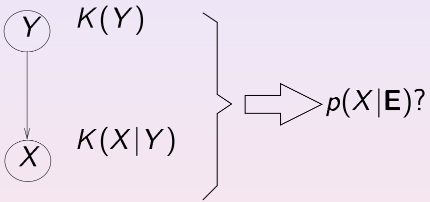
- Using an extensive conditional credal set at each variable of the network we can apply propagation algorithms like *Variable elimination*



- This algorithm begins with a set of potentials: the set of extensive conditional credal sets.
- It iterativelly eliminates variables from the set of potentials by using operations of *combination* and *marginalization* until only the queried variable remains.
- To eliminate a variable Y from the set of potentials, it combines all the potentials that contains Y and then Y is removed by marginalization.

Inference in credal networks

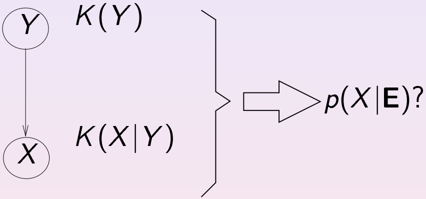
- Using an extensive conditional credal set at each variable of the network we can apply propagation algorithms like *Variable elimination*



- This algorithm begins with a set of potentials: the set of extensive conditional credal sets.
- It iterativelly eliminates variables from the set of potentials by using operations of *combination* and *marginalization* until only the queried variable remains.
- To eliminate a variable Y from the set of potentials, it combines all the potentials that contains Y and then Y is removed by marginalization.

Inference in credal networks

- Using an extensive conditional credal set at each variable of the network we can apply propagation algorithms like *Variable elimination*



- This algorithm begins with a set of potentials: the set of extensive conditional credal sets.
- It iterativelly eliminates variables from the set of potentials by using operations of *combination* and *marginalization* until only the queried variable remains.
- To eliminate a variable Y from the set of potentials, it combines all the potentials that contains Y and then Y is removed by marginalization.

Using BPTs for inference in credal networks I

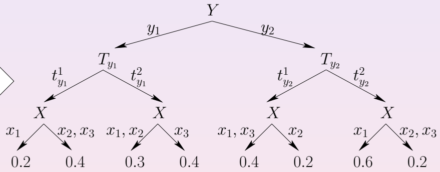
- **Extensive conditional credal sets** can be represented using *Binary probability trees* or *Standard probability trees* with the help of *transparent variables*:

$K(X|Y = y_1)$

	x_1	x_2	x_3
p_1	0.2	0.4	0.4
p_2	0.3	0.3	0.4

$K(X|Y = y_2)$

	x_1	x_2	x_3
q_1	0.4	0.2	0.4
q_2	0.6	0.2	0.2



- The representation of $K(X|Y)$ with a table need a bigger size:

$K(X\|Y)$	x_1, y_1	x_2, y_1	x_3, y_1	x_1, y_2	x_2, y_2	x_3, y_2
p_1, q_1	0.2	0.4	0.4	0.4	0.2	0.4
p_1, q_2	0.2	0.4	0.4	0.6	0.2	0.2
p_2, q_1	0.3	0.3	0.4	0.4	0.2	0.4
p_2, q_2	0.3	0.7	0.6	0.6	0.2	0.2

Using BPTs for inference in credal networks II

- For each variable X_i of the credal network we consider its **Extensive conditional credal set** $K(X_i|\Pi_i)$ represented with a BPT.
- The *Variable elimination* is then applied until only the queried variable remains.
- After an operation of combination or marginalization we can apply a pruning of the tree to reduce the size of the resulting BPT.
- We can also apply a reordering of the tree so that the most informative variables appears in the upper levels.

Using BPTs for inference in credal networks II

- For each variable X_i of the credal network we consider its **Extensive conditional credal set** $K(X_i|\Pi_i)$ represented with a BPT.
- The *Variable elimination* is then applied until only the queried variable remains.
- After an operation of combination or marginalization we can apply a pruning of the tree to reduce the size of the resulting BPT.
- We can also apply a reordering of the tree so that the most informative variables appears in the upper levels.

Using BPTs for inference in credal networks II

- For each variable X_i of the credal network we consider its **Extensive conditional credal set** $K(X_i|\Pi_i)$ represented with a BPT.
- The *Variable elimination* is then applied until only the queried variable remains.
- After an operation of combination or marginalization we can apply a pruning of the tree to reduce the size of the resulting BPT.
- We can also apply a reordering of the tree so that the most informative variables appears in the upper levels.

Using BPTs for inference in credal networks II

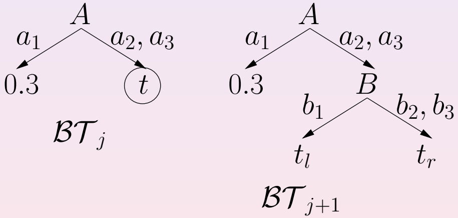
- For each variable X_i of the credal network we consider its **Extensive conditional credal set** $K(X_i|\Pi_i)$ represented with a BPT.
- The *Variable elimination* is then applied until only the queried variable remains.
- After an operation of combination or marginalization we can apply a pruning of the tree to reduce the size of the resulting BPT.
- We can also apply a reordering of the tree so that the most informative variables appears in the upper levels.

Constructing or reordering a Binary Probability Tree

- The process to build a BPT from a potential p uses an iterative process where at step j, $\mathcal{BT}_j$ is transformed into $\mathcal{BT}_{j+1}$ with:

$$\mathcal{BT}_{j+1} = \mathcal{BT}_j(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i})$$

- At each step, we have to look for the **best partition** $(X_i,\Omega^{t_l}_{X_i},\Omega^{t_r}_{X_i})$ of a leaf node t



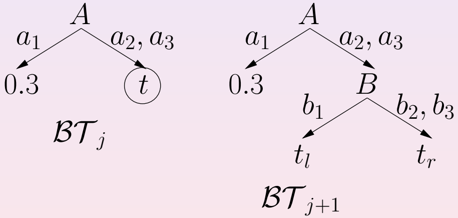
- The selection of the **best partition** tries to minimized the Kullback-Leibler divergence between the exact potential and the one associated to the tree

Constructing or reordering a Binary Probability Tree

- The process to build a BPT from a potential p uses an iterative process where at step j, $\mathcal{BT}_j$ is transformed into $\mathcal{BT}_{j+1}$ with:

$$\mathcal{BT}_{j+1} = \mathcal{BT}_j(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i})$$

- At each step, we have to look for the **best partition** $(X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i})$ of a leaf node t

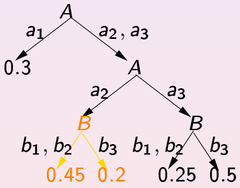


- The selection of the **best partition** tries to minimized the Kullback-Leibler divergence between the exact potential and the one associated to the tree

Pruning a Binary Probability Tree

- The pruning algorithm is a recursive algorithm with Δ, $\Delta \geq 0$, as input parameter (the threshold for pruning):
- It is a process where at each step a *terminal tree* is replaced by the average of values it represents if:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq \Delta$$

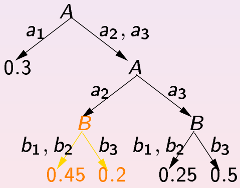


$P(A\|B)$	b_1	b_2	b_3
a_1	0.3	0.3	0.3
a_2	0.45	0.45	0.2
a_3	0.25	0.25	0.5

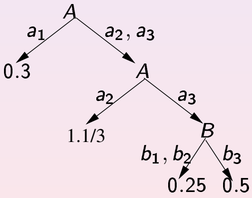
Pruning a Binary Probability Tree

- The pruning algorithm is a recursive algorithm with Δ, $\Delta \geq 0$, as input parameter (the threshold for pruning):
- It is a process where at each step a *terminal tree* is replaced by the average of values it represents if:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq \Delta$$



$P(A\|B)$	b_1	b_2	b_3
a_1	0.3	0.3	0.3
a_2	0.45	0.45	0.2
a_3	0.25	0.25	0.5



$P(A\|B)$	b_1	b_2	b_3
a_1	0.3	0.3	0.3
a_2	1.1/3	1.1/3	1.1/3
a_3	0.25	0.25	0.5

Experiments I

- We have compare propagation using SPTs and BPTs in credal networks obtained from the Alarm and Insurance topology.
- Queries for different variables have been performed.
- For each experiment (Ex):
  - $|\mathbf{E}|$ is the number of observed variables in the credal network (randomly choosen)
  - nvpc is the number of vertices per each separately specified credal set.
  - per the percentage of configurations of Ω_{Π_i} that will contain nv vertices in the credal sets $K(X_i|\pi_i)$ (for the rest we use 1 vertex).
  - n_v is the potential size of the strong extension of the CN.

Ex	Var	Network	$\lvert\mathbf{E}\rvert$	nvpc	per	n_v
1	Venttube	Alarm	0	3	90	354294
2	Expco2	Alarm	0	3	17	177147
3	RiskAversion	Insurance	0	3	70	177147
4	DrivHist	Insurance	0	3	31.5	177147
5	Venttube	Alarm	6	3	12.25	354294
6	DrivHist	Insurance	9	3	12	944784

Experiments II

- We have run the variable elimination algorithm with SPTs and BPTs using several values for the Δ threshold in the interval $[10^{-7}, 10^{-2}]$.
- For each run we have measured:
  - Maximum required size of SPTs and BPTs during the propagation (biggest tree used in the computations).
  - The *mean square error* for the a posteriori bounds of the queried variable.

$$\sqrt{\frac{\sum_{x_q \in \Omega_{\mathbf{X_q}}} ((\underline{P}^*(x_q|\mathbf{e}) - \underline{P}(x_q|\mathbf{e}))^2 + (\overline{P}^*(x_q|\mathbf{e}) - \overline{P}(x_q|\mathbf{e}))^2)}{2 \cdot |\Omega_{X_q}|}} \quad (2)$$

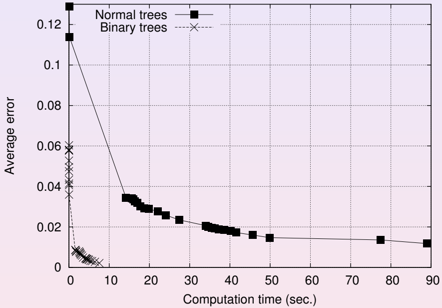
  - The running time used by the propagation algorithm.

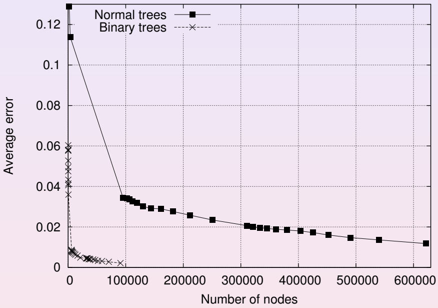
Experiments III

- We have compared average mean square error versus largest tree size required in the two versions of the propagation algorithm (using SPTs and BPTs). We also have compared average mean square error versus computing time.
- As expected with both kind of trees, high values of Δ will cause large errors but require lower computing time and smaller trees.
- Small values of Δ will give small errors but require a high computing time and large trees.

Experiments IV

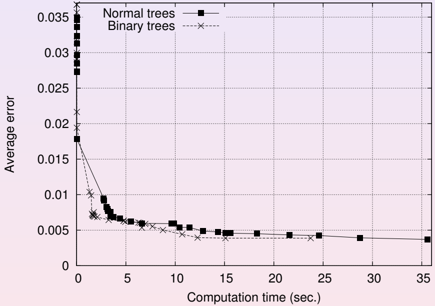
- In some cases we obtain a noticeable reduction in the size and required time using BPTs with respect to SPTs (Experiments 1, 3 and 5):

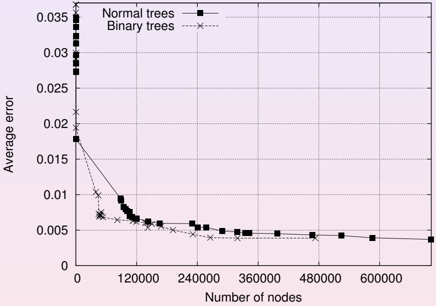




Experiments V

- In other cases, a similar performance is obtained (Experiments 2 and 6):





- So, we conclude that BPTs are a better representation for the potentials of a credal network than SPTs.