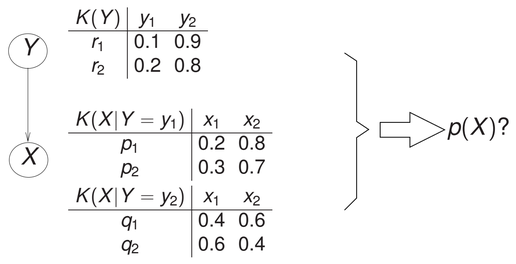
Comparing Binary and Standard Probability Trees in Credal Networks Inference

Andrés Cano, Manuel Gómez-Olmedo, Andrés R. Masegosa, Serafín Moral
{acu,mgomez,andrew,smc}@decsai.ugr.es
Department of Computer Science and Artificial Intelligence. University of Granada, Spain

ISIPTA '11 - July 27th, 2011

1.- Introduction

- 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

- Particularly we are interested in obtaining *a posteriori intervals* for each queried variable X_q given a set of observed variables $\mathbf{X}_E$:

$$\overline{p}(X_q = x_q^i|\mathbf{x}_E) \text{ and } \underline{p}(X_q = x_q^i|\mathbf{x}_E)$$

$$\text{if } \mathbf{X}_E = \emptyset \text{ then } p(X) = \begin{cases} [0.3,0.45] \text{ if } X = x_1 \\ [0.55,0.7] \text{ if } X = x_2 \end{cases}$$

2.- Credal networks

- A *credal network* (CN) is a DAG where each node represents a conditional credal set $K(X_i|\Pi_i)$.
- We use a credal set $K(X_i|\Pi_i = \pi_i)$ for each configuration π_i of Π_i (*separately specified credal sets*).
- The topology of a CN represents independece relations between variables using *d-separation criterion*.
- The meaning of such independences depends on which concept of independence is adopted: here we use *strong independence*.
- The *strong extension* $K(\mathbf{X})$ of a CN can be obtained as the convex hull of the collection of joint mass functions that can be obtained with every possible combination of the vertices of the set of the separately specified credal sets $K(X_i|\pi_i)$:

$$K(\mathbf{X}) = \text{CH}\{P(\mathbf{X}) : P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi_i), \\ \forall \mathbf{x} \in \Omega_{\mathbf{X}}, \forall \pi_i \in \Omega_{\Pi_i},\ P(X_i|\pi_i) \in K(X_i|\pi_i)\} \quad (1)$$

- A CN can be regarded as a collection of BNs with the topology of the CN.
  - The joint probability of each BN is defined by each one of the vertices of $K(\mathbf{X})$.
  - So, the CN defines a collection of joint probabilities:

$$\mathbf{P}(\mathbf{X}) = \{P_k(\mathbf{X})\}_{k=1}^{n_v} \quad (2)$$

- From the set of separately specified credal sets $K(X|\pi_i), \forall \pi_i \in \Omega_{\Pi_i}$, we can obtain the *extensive conditional credal set* $K(X_i|\Pi_i)$ with:

$$K(X_i|\Pi_i) = \{P|P(x_i, \pi_i) \in K(X_i|\Pi_i = \pi_i), \forall \pi_i \in \Omega_{\Pi_i}\}$$

- Example:

$K(X\|Y=y_1)$	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

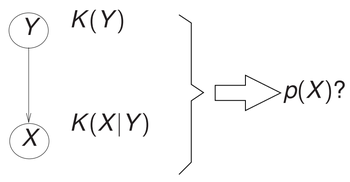
$K(X\|Y)$	x_1, y_1	x_2, y_1	x_1, y_2	x_2, y_2
p_1, q_1	0.2	0.8	0.4	0.6
p_1, q_2	0.2	0.8	0.6	0.4
p_2, q_1	0.3	0.7	0.4	0.6
p_2, q_2	0.3	0.7	0.6	0.4

- $K(\mathbf{X})$ can also be defined as the multiplication (combination) of all the (extensive) conditional credal sets $K(X_i|\Pi_i)$ in the credal network:

$$K(\mathbf{X}) = \prod_{i=1}^{n} K(X_i|\Pi_i) \quad (3)$$

Inference in credal networks

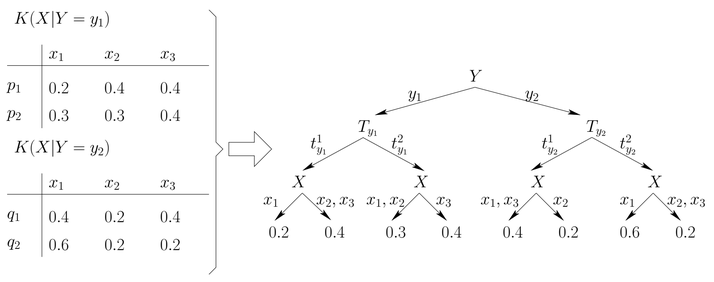
- Using an extensive conditional credal set at each variable of the network we can apply propagation algorithms like *Variable elimination*



- This algorithm begins with a set of potentials: the set of extensive conditional credal sets.
- It iteratively 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 potential that contains Y and then Y is removed by marginalization.

3.- Using BPTs for inference in credal networks

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

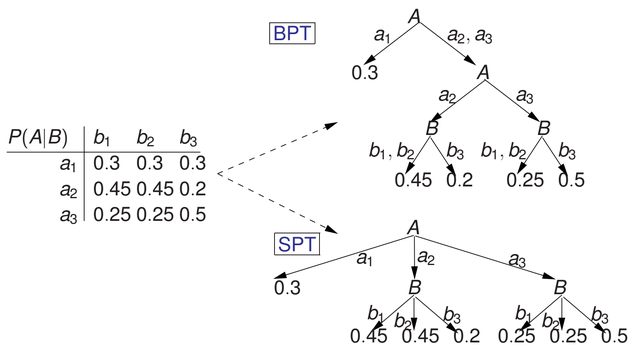


- Here, the representation of $K(X|Y)$ with a table need a bigger size:

$K(X\|Y)$	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

4.- Standard and Binary Probability Trees

- Standard probability trees (SPTs) and Binary probability trees (BPTs) are suitable data structures for representing potentials.
- They allow to control the accuracy of inference algorithms by means of the pruning operation with a given threshold parameter Δ.
- SPTs has been previously used for inference in credal networks. So, we give more details about BPTs.
- A binary probability tree (BPT) is a labeled tree:
  - Each internal node is labeled with a variable and it has exactly two children
  - Each leaf node is labeled with a real number
  - Outgoing arcs of a node labeled with X_i are labeled with a subset of Ω_{X_i}
- A variable can appear more than once labeling the nodes in the path from the root to a leaf node
- A BPT $\mathcal{BT}$ can be used to represent a probability potential for $\mathbf{X}_I$
- BPTs allows to represent fine-grained *context specific independences* than probability trees when some variables contain more than two states.



$P(A\|B)$	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

Operations with Standard and Binary Probability Trees

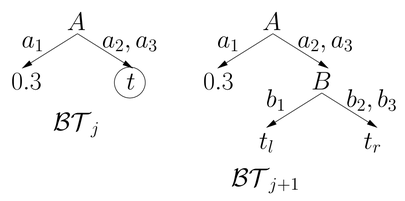
- Inference algorithms require three operations over potentials
  - **Restriction** ($\mathcal{BT}^{R(\mathbf{x}_J)}$): Get the part of $\mathcal{BT}$ consistent with configuration $\mathbf{x}_J$
  - **Combination** ($\mathcal{BT}1 \otimes \mathcal{BT}2$): Get a BPT that represents the product of the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

5.- Constructing or reordering a Binary Probability Tree

- The process to build a BPT from a potential p is similar to the one for inducing *classification trees* from a dataset.
- We use an iterative process where at step j, $\mathcal{BT}_j$ is transformed into $\mathcal{BT}_{j+1}$ with:

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

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



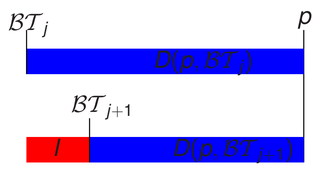
- The definition of the splitting criterion requires a distance to measure the goodness of the approximation of a $\mathcal{BT}$ to a potential p. We use the Kullback-Leibler divergence between the normalized potentials:

$$D(p, \mathcal{BT}) = \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} \overline{p}(\mathbf{x}_I) \log \frac{\overline{p}(\mathbf{x}_I)}{\overline{\mathcal{BT}}(\mathbf{x}_I)} \quad (4)$$

- At step j, for expanding node t we select the partition $X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}$ that maximizes the *information gain*:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = D(p, \mathcal{BT}_j) - D(p, \mathcal{BT}_j(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}))$$

- By maximazing the *information gain* we achieve to minimize the Kullback-Leibler's distance between potential p and the new $\mathcal{BT}_{j+1}$.



- The *information gain* can be efficiently calculated in BPTs as we have shown in previous works.

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) = \text{sum}(p^{R(\mathbf{A}^t)}) \cdot \log(|\Omega_{X_i}^{t_l}|/\text{sum}(p^{R(\mathbf{A}^t)})) \\ + \text{sum}(p^{R(\mathbf{A}^{t_l})}) \cdot \log(\text{sum}(p^{R(\mathbf{A}^{t_l})})/|\Omega_{X_i}^{t_l}|) \\ + \text{sum}(p^{R(\mathbf{A}^{t_r})}) \cdot \log(\text{sum}(p^{R(\mathbf{A}^{t_r})})/|\Omega_{X_i}^{t_r}|)$$

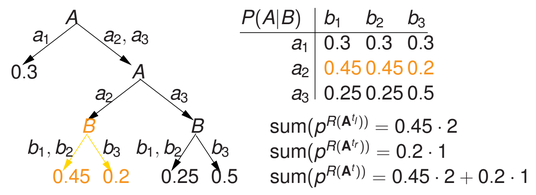
6.- Pruning a Binary Probability Tree

- Pruning is applied to reduce the size of a BPT
- The pruning algorithm is a recursive algorithm with Δ as input parameter (the threshold for pruning):
  boolean prune(double Δ)
- It is a process where at each step a *terminal tree t* is pruned (replaced by a node labeled with the average of values it represents) if:

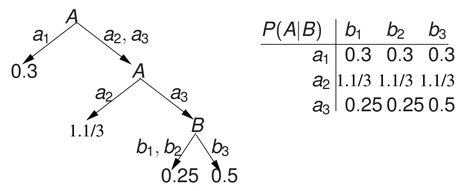
$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq \Delta$$

- $I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r})$ can be again locally computed at node t.

Example



$P(A\|B)$	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

7.- The variable elimination algorithm using BPTs

Input : $\mathbf{K} = \{K(X_i|\pi_i) : i = 1, \ldots, n\}$ the set of separately specified credal sets in the CN; $\mathbf{e}$ the set of observed values, $\mathbf{e} \in \Omega_{\mathbf{E}}$; a variable of interest X_q, $X_q \in \mathbf{X}_N \setminus \mathbf{E}$; and Δ the threshold for pruning

Output: $\underline{P}(x_q|\mathbf{e})$ and $\overline{P}(x_q|\mathbf{e})$ for each $x_q \in \Omega_{X_q}$, $X_q \in \mathbf{X}_N \setminus \mathbf{E}$

Get the set $S_{\mathcal{BT}}$ of binary trees, building each binary tree $\mathcal{BT}_i$ from the credal sets $K(X_i|\pi_i), \forall \pi_i \in \Omega_{\Pi_i}$

Transform each $\mathcal{BT}_i$ into $\mathcal{BT}_i^{R(\mathbf{e})}$ (restrict to evidence)

Reorder variables and split sets in every $\mathcal{BT}_i$

Prune each $\mathcal{BT}_i$ with the Δ threshold

foreach $Y \in \mathbf{X}_N \setminus (\mathbf{E} \cup \{X_q\})$ **do**

  Let $\mathbf{S}_Y = \{\mathcal{BT}_i | Y \in s(\mathcal{BT}_i)\}$

  Calculate $\mathcal{BT}_{prod} = \prod_{\mathcal{BT}_i \in \mathbf{S}_Y} \mathcal{BT}_i$

  Calculate $\mathcal{BT}_{sum} = \mathcal{BT}_{prod}^{\downarrow s(\mathcal{BT}_{prod})\setminus Y}$

  Reorder variables and split sets in $\mathcal{BT}_{sum}$

  Prune $\mathcal{BT}_{sum}$ using the Δ threshold

  $S_{\mathcal{BT}} = \{(S_{\mathcal{BT}} \setminus \mathbf{S}_Y) \cup \mathcal{BT}_{sum}$

end

Calculate $\mathcal{BT}_q = \prod_{\mathcal{BT}_i \in S_{\mathcal{BT}}} \mathcal{BT}_i$

Get $\underline{P}(x_q|\mathbf{e})$ and $\overline{P}(x_q|\mathbf{e})$ by normalizing the vertices in $\mathcal{BT}_q$

8.- Experimental results

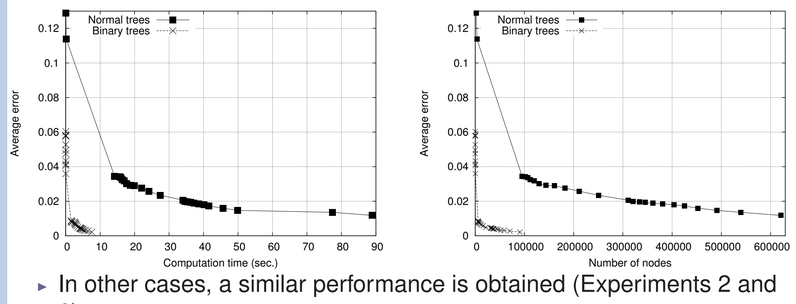
- We have compare propagation using SPTs and BPTs in credal networks obtained from the Alarm and Insurance topology.
- Queries for different variables have been permormed.
- For each experiment (Ex):
  - $|\mathbf{E}|$ is the number of observed variables in the credal network (randomly choosen)
  - nvpc 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	$\|\mathbf{E}\|$	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

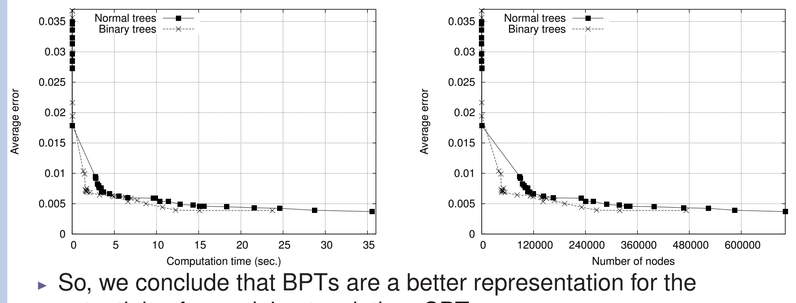
- We have run the variable elimination algorithm with SPTs and BPTs using several values for the Δ threshold in the interval $[10^{-7}, 10^{-2}]$.
- For each run we have measured:
  - Maximum required size of SPTs and BPTs during the propagation (biggest tree used in the computations).
  - The *mean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

$$\sqrt{\frac{\sum_{x_q \in \Omega_{X_q}} ((\underline{P}^*(x_q|\mathbf{e}) - \underline{P}(x_q|\mathbf{e}))^2 + (\overline{P}^*(x_q|\mathbf{e}) - \overline{P}(x_q|\mathbf{e}))^2)}{2 \cdot |\Omega_{X_q}|}} \quad (5)$$

  - The running time used by the propagation algorithm.
- We have compare average mean square error versus largest tree size required in the two versions of the propagation algoritm (using SPTs and BPTs). We also compare 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.
- In some cases we obtain a noticeable reduction in the size and required time using BPTs with respect to SBTs (Experiments 1, 3 and 5):



- 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.

Acknowledgements

This work has been supported by the Spanish Ministry of Science and Innovation, under projects TIN2007-67418-C03-03, TIN2010-20900-C04-01 and by the European Regional Development Fund (FEDER). We are also very grateful to the anonymous reviewers for their valuable comments and suggestions.

Essential references

- A. Cano, M. Gómez, S. Moral, and J. Abellán. Hill-climbing and branch-and-bound algorithms for exact and approximate inference in credal networks. *International Journal of Approximate Reasoning*, 44:261–280, 2007.
- A. Cano, M. Gómez-Olmedo, and S. Moral. Approximate inference in Bayesian networks using binary probability trees. *International Journal of Approximate Reasoning*, 52:49–62, 2011.
- A. Cano, M. Gómez-Olmedo, and S. Moral. Approximate inference in Bayesian networks using binary probability trees. *International Journal of Approximate Reasoning*, 52:49–62, 2011.
- A. Salmerón, A. Cano, and S. Moral. Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413, 2000.