Approximate Probability Propagation with Mixtures of Truncated Exponentials [⋆]

Rafael Rumí [∗] and Antonio Salmerón

Dpt. Statistics and Applied Mathematics
University of Almería
La Cañada de San Urbano s/n
04120 Almería, Spain

Abstract

Mixtures of truncated exponentials (MTEs) are a powerful alternative to discretisation when working with hybrid Bayesian networks. One of the features of the MTE model is that standard propagation algorithms can be used. However, the complexity of the process is too high and therefore approximate methods, which tradeoff complexity for accuracy, become necessary. In this paper we propose an approximate propagation algorithm for MTE networks which is based on the Penniless propagation method already known for discrete variables. We also consider how to use Markov Chain Monte Carlo to carry out the probability propagation. The performance of the proposed methods is analysed in a series of experiments with random networks.

Key words: Hybrid Bayesian networks, Mixtures of Truncated Exponentials, continuous variables, probability propagation, penniless propagation, MCMC.

1 Introduction

A Bayesian network is an efficient representation of a joint probability distribution over a set of variables, where the network structure encodes the independence relations among the variables. Bayesian networks are commonly

[⋆] This work has been supported by the Spanish Ministry of Education and Science, project TIN2004-06204-C03-01, and by FEDER funds
[∗] Corresponding author
Email addresses: rrumi@ual.es (Rafael Rumí), Antonio.Salmeron@ual.es (Antonio Salmerón).

used to make inferences about the probability distribution on some variables of interest, given that the values of some other variables are known. This task is usually called *probabilistic inference* or *probability propagation*.

Much attention has been paid to probability propagation in networks where the variables are discrete with a finite number of possible values. Several exact methods have been proposed in the literature for this task [1–4], all of them based on *local computation*. Local computation means to calculate the marginals without actually computing the joint distribution, and is described in terms of a message passing scheme over a structure called *join tree*. Also, approximate methods have been developed with the aim of dealing with complex networks [5–10].

In *hybrid Bayesian networks*, where both discrete and continuous variables appear simultaneously, it is possible to apply local computation schemes similar to those for discrete variables. However, the correctness of exact inference depends on the model.

This problem was deeply studied before, but the only general solution is the discretisation of the continuous variables [11,12] which are then treated as if they were discrete, and therefore the obtained results are approximate. Exact propagation can be carried out over hybrid networks when the model is a conditional Gaussian distribution [13,14], but in this case, discrete variables are not allowed to have continuous parents. This restriction was overcome in [15] using a mixture of exponentials to represent the distribution of discrete nodes with continuous parents, but the price to pay is that propagation cannot be carried out using exact algorithms: Monte Carlo methods are used instead.

The Mixture of Truncated Exponentials (MTE) model [16] provide the advantages of the traditional methods and the added feature that discrete variables with continuous parents are allowed. Exact standard propagation algorithms can be performed over them [17], as well as approximate methods. In this work, we introduce an approximate propagation algorithm for MTEs based on the idea of Penniless propagation [5], which is actually derived from the Shenoy-Shafer [4] method. We also show how the propagation can be carried out using the Markov Chain Monte Carlo methodology suggested in [16].

This paper continues with a description of the MTE model in section 2. The representation based on mixed tress is studied in section 3. Section 4 contains the application of Shenoy-Shafer algorithm to MTE networks, while in section 5 the Penniless algorithm is presented. Section 6 is devoted to explain how Markov Chain Monte Carlo simulation can be applied to propagate with MTEs. The performance of the proposed algorithms is analysed through some experiments in section 7. The paper ends with conclusions in section 8.

2 The MTE model

Throughout this paper, random variables will be denoted by capital letters, and their values by lowercase letters. In the multi-dimensional case, boldfaced characters will be used. The domain of the variable $\mathbf{X}$ is denoted by $\Omega_{\mathbf{X}}$. The MTE model is defined by its corresponding potential and density as follows [16]:

Definition 1 (MTE potential) *Let $\mathbf{X}$ be a mixed n-dimensional random vector. Let $\mathbf{Y} = (Y_1, \ldots, Y_d)$ and $\mathbf{Z} = (Z_1, \ldots, Z_c)$ be the discrete and continuous parts of $\mathbf{X}$, respectively, with $c + d = n$. We say that a function $f : \Omega_{\mathbf{X}} \mapsto \mathbb{R}_0^+$ is a* Mixture of Truncated Exponentials potential (MTE potential) *if one of the next conditions holds:*

i. $\mathbf{Y} = \emptyset$ *and f can be written as*

$$f(\mathbf{x}) = f(\mathbf{z}) = a_0 + \sum_{i=1}^{m} a_i \exp\left\{\sum_{j=1}^{c} b_i^{(j)} z_j\right\} \tag{1}$$

for all $\mathbf{z} \in \Omega_{\mathbf{Z}}$, where a_i, $i = 0, \ldots, m$ and $b_i^{(j)}$, $i = 1, \ldots, m$, $j = 1, \ldots, c$ are real numbers.

ii. $\mathbf{Y} = \emptyset$ *and there is a partition $D_1, \ldots, D_k$ of $\Omega_{\mathbf{Z}}$ into hypercubes such that f is defined as*

$$f(\mathbf{x}) = f(\mathbf{z}) = f_i(\mathbf{z}) \quad \text{if} \quad \mathbf{z} \in D_i \ ,$$

where each f_i, $i = 1, \ldots, k$ can be written in the form of (1).

iii. $\mathbf{Y} \neq \emptyset$ *and for each fixed value $\mathbf{y} \in \Omega_{\mathbf{Y}}$, $f_{\mathbf{y}}(\mathbf{z}) = f(\mathbf{y}, \mathbf{z})$ can be defined as in* ii.

Definition 2 (MTE density) *An MTE potential f is an* MTE density *if*

$$\sum_{\mathbf{y} \in \Omega_{\mathbf{Y}}} \int_{\Omega_{\mathbf{Z}}} f(\mathbf{y}, \mathbf{z}) d\mathbf{z} = 1 \ .$$

In a Bayesian network, two types of densities can be found:

(1) For each variable X which is a root of the network, a density $f(x)$ is given.
(2) For each variable X with parents $\mathbf{Y}$, a conditional density $f(x|\mathbf{y})$ is given.

A *conditional MTE density* $f(x|\mathbf{y})$ is an MTE potential $f(x, \mathbf{y})$ such that fixing $\mathbf{y}$ to each of its possible values, the resulting function is a density for X.

3 Mixed trees

In [16] a data structure was proposed to represent MTE potentials: The so-called *mixed probability trees* or mixed trees for short. The formal definition is as follows:

Definition 3 (Mixed tree) *We say that a tree $\mathcal{T}$ is a* mixed tree *if it meets the following conditions:*

i. *Every internal node represents a random variable (either discrete or continuous).*
ii. *Every arc outgoing from a continuous variable Z is labeled with an interval of values of Z, so that the domain of Z is the union of the intervals corresponding to the arcs Z-outgoing.*
iii. *Every discrete variable has a number of outgoing arcs equal to its number of states.*
iv. *Each leaf node contains an MTE potential defined on variables in the path from the root to that leaf.*

Mixed trees can represent MTE potentials defined by parts. Each entire branch in the tree determines one sub-region of the space where the potential is defined, and the function stored in the leaf of a branch is the definition of the potential in the corresponding sub-region. An example of a mixed tree is shown in Fig. 1.

Definition 4 *The* label *of a node in a mixed tree is defined as the random variable it represents, if it is an inner node, and the MTE potential it contains, if it is a leaf node.*

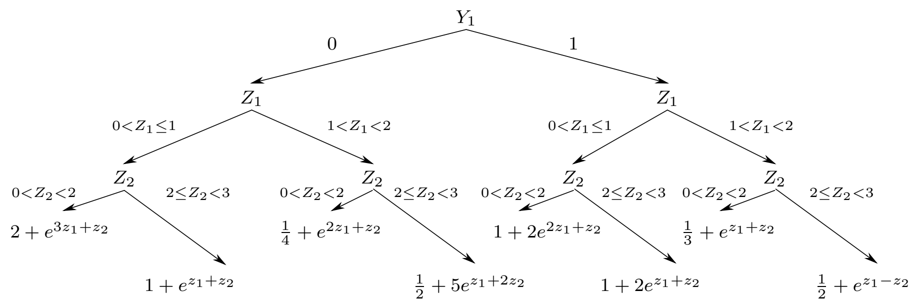

Fig. 1. A mixed tree representing an MTE potential.

The operations required for probability propagation in Bayesian networks (restriction, marginalisation and combination) can be carried out by means of algorithms very similar to those described, for instance, in [12,9]. We refer to [16] for a formal definition of the basic operations over MTE potentials.

3.1 Implementation of the basic operations over mixed trees

The simplest operation is the *restriction*. This operation is covered by the concept of *restricted tree*, which is defined as follows.

Definition 5 (Restricted tree) *Let $\mathcal{T}$ be a mixed tree and X a variable of $\mathcal{T}$.*

(1) If X is discrete, the restricted tree *of $\mathcal{T}$ for a value $x \in \Omega_X$, denoted as $\mathcal{T}^{R(X=x)}$ is the tree obtained from $\mathcal{T}$ by replacing each node labeled with X by its child corresponding to value x.*

(2) if X is continuous,the restricted tree *of $\mathcal{T}$ for an interval $(a,b) \in \Omega_X$, denoted as $\mathcal{T}^{R(X\in(a,b))}$, is the tree obtained from $\mathcal{T}$ by repeating the next procedure for each node labeled with X:*

- *If there is an outgoing arc of X which labeled with an interval that contains (a,b), then replace X by the child of X corresponding to that arc.*
- *Otherwise, remove all the children of X corresponding to intervals whose intersection with (a,b) is empty, and replace the labels of the remaining arcs by their intersection with (a,b).*

The *combination* of two mixed trees can be implemented recursively. The idea is that the root of one of the trees is taken as root of the new tree, and each child of this root is replaced by the product of that child and the restriction of the other tree to the interval associated to child in question. The recursion continues until the leaves are reached. The details are given in the following pseudo-code.

COMBINE($\mathcal{T}_1$,$\mathcal{T}_2$)

```
INPUT: Two mixed trees $\mathcal{T}_1$ and $\mathcal{T}_2$.
OUTPUT: The combination of $\mathcal{T}_1$ and $\mathcal{T}_2$.

(1) Create a node $\mathcal{T}_r$ without label.
(2) Let $L_1$ and $L_2$ be the labels of the root nodes of $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively.
(3) If $L_1$ and $L_2$ are MTE potentials, make $L_1 \cdot L_2$ be $\mathcal{T}_r$ label.
(4) If $L_1$ is an MTE potential, but $L_2$ is a variable,
    (a) Make $L_2$ be the label of $\mathcal{T}_r$.
    (b) For each tree $\mathcal{T}$ child of the root node of $\mathcal{T}_2$,
            set $\mathcal{T}_h$ := **COMBINE($\mathcal{T}_1$,$\mathcal{T}$)** as a child of $\mathcal{T}_r$.
(5) If $L_1$ is a variable, let $X$ be that variable.
    (a) Make $X$ be the label of $\mathcal{T}_r$.
    (b) If $X$ is discrete,
        (i) For each $x \in \Omega_X$,
            • set $\mathcal{T}_h$ := **COMBINE($\mathcal{T}_1^{R(X=x)}$,$\mathcal{T}_2^{R(X=x)}$)** as a child of $\mathcal{T}_r$.
    (c) If $X$ is continuous,
        (i) For each interval $(a,b)$ belonging to outgoing arcs of $X$,
            • Set $\mathcal{T}_h$ := **COMBINE($\mathcal{T}_1^{R(X\in(a,b))}$,$\mathcal{T}_2^{R(X\in(a,b))}$)** as a child of
              $\mathcal{T}_r$.
(6) RETURN $\mathcal{T}_r$.
```

A variable is marginalised out from a mixed tree by summing over all its possible values, if it is discrete, or by integrating over its entire domain otherwise, as stated in the next algorithm.

MARGINALISE_OUT($\mathcal{T}$,X_i)

INPUT: A mixed tree $\mathcal{T}$ and a variable X_i.
OUTPUT: The marginal of $\mathcal{T}$ for variables in $\mathcal{T}$ except X_i (i.e., X_i is marginalised out from $\mathcal{T}$).

```
(1) If $X_i$ is discrete, $\mathcal{T}_r$ := **SUM_OUT($\mathcal{T}$,$X_i$)**.
(2) Else
        Let $(a,b)$ be the range of values of $X_i$.
        $\mathcal{T}_r$ := **INTEGRATE_OUT($\mathcal{T}$,$X_i$,$a$,$b$)**.
(3) RETURN $\mathcal{T}_r$.
```

Procedure **SUM_OUT($\mathcal{T}$,X_i)** recursively searches for X_i, and when it is found, it is replaced by the sum of its children. The sum of two mixed trees is exactly the same as the combination, but changing products by sums. The details are given in the next algorithm.

SUM_OUT($\mathcal{T}$,X_i)

INPUT: A mixed tree $\mathcal{T}$ and a variable X_i.
OUTPUT: A tree obtained from $\mathcal{T}$ removing X_i summing the subtrees corresponding to its children.

```
(1) Let $L$ be the label of the root node of $\mathcal{T}$, and let $X$ be the variable
    corresponding to label $L$.
(2) If $X$ is discrete,
    (a) If $X = X_i$,
          • Let $\mathcal{T}_1, \ldots, \mathcal{T}_s$ be the children of the root node of $\mathcal{T}$.
          • $\mathcal{T}_r := \mathcal{T}_1$.
          • For $i := 2$ to $s$, $\mathcal{T}_r :=$ SUM($\mathcal{T}_r$,$\mathcal{T}_i$).
    (b) Else
          • Create a node $\mathcal{T}_r$ with label $X$.
          • For each $x \in \Omega_X$, set $\mathcal{T}_h :=$ SUM_OUT($\mathcal{T}^{R(X=x)}$,$X_i$) as the
            next child of $\mathcal{T}_r$.
(3) If $X$ is continuous,
    (a) Create a node $\mathcal{T}_r$ with label $X$.
    (b) For each interval $(a, b)$ corresponding to outgoing arcs of $X$,
          • Set $\mathcal{T}_h :=$ SUM_OUT($\mathcal{T}^{R(X\in(a,b))}$,$X_i$) as the next child of $\mathcal{T}_r$.
(4) RETURN $\mathcal{T}_r$.
```

In the algorithm above, **SUM($\mathcal{T}_r$,$\mathcal{T}_i$)** can be implemented as **COMBINE($\mathcal{T}_r$,$\mathcal{T}_i$)**, changing the product in step 2 by a sum. Therefore, we skip the detailed algorithm here.

Finally, the next procedure implements the elimination of a continuous variable through integration. The basis of this algorithm is similar to **SUM_OUT**. The details are given below.

INTEGRATE_OUT($\mathcal{T}$,X_i,a,b)

INPUT: a mixed tree $\mathcal{T}$, a variable X_i and two real numbers a and b.
OUTPUT: a tree obtained from $\mathcal{T}$ where X_i has been removed integrating over (a, b).

```
(1) Let $L$ be the label of the root node of $\mathcal{T}$.
(2) If $L$ is an MTE potential $\phi$, create a node $\mathcal{T}_r$ with label $\int_a^b \phi(x)\mathrm{d}x$.
(3) Else, let $X$ be the variable corresponding to label $L$.
(4) If $X$ is discrete,
    (a) Make $X$ be $\mathcal{T}_r$ label.
    (b) For each child $\mathcal{T}_h$ of the root node of $\mathcal{T}$,
            • Make $\mathcal{T}'_h$ := INTEGRATE_OUT($\mathcal{T}_h$,$X_i$,$a$,$b$) be a $\mathcal{T}_r$ child.
(5) If $X$ is continuous,
    (a) If $X = X_i$,
        (i) Label $\mathcal{T}_r$ with a null potential.
        (ii) For each interval $(\alpha,\beta)$ corresponding to outgoing arcs of $X$,
            • Make $\mathcal{T}_h$ be the corresponding tree to that arc.
            • $(\alpha',\beta') := (\alpha,\beta) \cap (a,b)$.
            • If $(\alpha',\beta') \neq \emptyset$,
                · $\mathcal{T}_r$ := SUM($\mathcal{T}_r$, INTEGRATE_OUT($\mathcal{T}_h$,$X_i$,$\alpha'$,$\beta'$))
    (b) Else
        (i) Make $X$ be $\mathcal{T}_r$ label.
        (ii) For each child $\mathcal{T}_h$ of the root node of $\mathcal{T}$,
            • make $\mathcal{T}'_h$ := INTEGRATE_OUT($\mathcal{T}_h$,$X_i$,$a$,$b$) be a $\mathcal{T}_r$ child.
(6) RETURN $\mathcal{T}_r$.
```

4 Shenoy - Shafer propagation algorithm with MTEs

In [16] it was shown that MTE potentials are closed for restriction, combination and marginalisation. It means that probability propagation can be carried out in networks with MTEs using the Shenoy-Shafer algorithm [4], and furthermore, mixed trees can be used as a data structure, since Shenoy-Safer method can be implemented using the algorithms described in section 3.1.

The Shenoy-Shafer propagation algorithm requires an adequate order of elimination of the variables to get the join tree, since different orders may result in join trees of distinct sizes, and the efficiency of probability propagation depends on the complexity of the join tree. This problem has been widely studied for discrete networks [18,19]. Here we propose a one-step lookahead strategy to determine the elimination order. We will choose the next variable to eliminate according to the size [1] of the potential associated with the resulting clique.

The decision on which variable to select next time, requires the knowledge about the size of the clique that would result from combining all the potentials defined for the chosen variable. In the case of some MTE networks, it is possible

[1] The *size* of an MTE potential is defined as the number of exponentials terms, including the independent term, out of which the MTE potential is composed. For instance, the potential represented in Fig. 1 has size equal to 16, because it has 8 leaves, and in each one an independent term, and one exponential term, so $8 \times (1 + 1) = 16$.

to estimate it beforehand. If the MTE potentials are such that for each of them, the number of exponential terms in each leaf is the same, and the number of splits of the domain of the continuous variables also coincides, and only one variable appears in the MTE functions stored in the leaves of the mixed tree (the rest of the variables are used just to split the domain), as in [16] and [20], then there is an upper bound on the potential size, which is given in the next proposition.

Proposition 6 *Let $\mathcal{T}_1, \ldots, \mathcal{T}_h$ be h mixed probability trees, $\mathbf{Y}_i$, $\mathbf{Z}_i$ the discrete and continuous variables of each of them, and n_i the number of intervals into which the domain of the continuous variables of T_i is split. Let Ω_{Y_i} be the set of possible values of the discrete variable Y_i. It holds that*

$$size(\mathcal{T}_1 \times \mathcal{T}_2 \times \ldots \times \mathcal{T}_h) \leq \left(\prod_{Y_i \in \overset{h}{\underset{i=1}{\cup}} \mathbf{Y}_i} |\Omega_{Y_i}| \right) \times \left(\prod_{j=1}^{h} n_j^{k_j} \right) \times \left(\prod_{j=1}^{h} t_j \right) ,$$

where t_j is the number of exponential terms in each leaf of $\mathcal{T}_j$, and k_j is the number of continuous variables in $\mathcal{T}_j$.

PROOF. Each discrete variable in a mixed tree, $Y_i \in \overset{h}{\underset{i=1}{\cup}} \mathbf{Y}_i$ has as many children as states. Continuous variables in tree $\mathcal{T}_i$ have n_i children. Nevertheless, when combining the trees we must take into account the intersections of the intervals in which the same variable is defined in different trees, i.e., the same continuous variable, $Z_i \in \overset{h}{\underset{i=1}{\cup}} \mathbf{Z}_i$ may have different intervals splitting its domain in each of the trees. On each tree $\mathcal{T}_i$, the joint domain of the continuous variables, $\mathbf{Z}_i$, is divided in $n_i^{k_i}$ pieces, so when combining h trees the joint domain of the continuous variables will be divided, at most, in $\prod_{j=1}^{h} n_j^{k_j}$ pieces.

When combining discrete and continuous variables, there will be an upper bound of $\prod_{Y_i \in \overset{h}{\underset{i=1}{\cup}} \mathbf{Y}_i} |\Omega_{Y_i}| \times | \overset{h}{\underset{i=1}{\cup}} \mathbf{Z}_i|^{\sum n_i}$ leaf nodes in the tree, and each one of them will be the result of combining the MTE functions defined over the initial leaves. In $\mathcal{T}_j$ these functions have the form $k + \sum_{i=1}^{t_j} a_i exp(b_i z_j)$ for a continuous variable Z_j and a real number p_i, for the discrete variables, so when combining them the outcome has at most, on each leaf, $\prod_{j=1}^{h} t_j$ terms.

5 Penniless propagation with MTEs

Using the Shenoy-Shafer algorithm, it is usual in large discrete networks that the size of the potentials involved grow so much that the propagation becomes infeasible. In the case of MTE networks, the complexity is higher, since the potentials are larger in general.

To overcome this problem in the discrete case, the Penniless propagation algorithm was proposed [5]. This propagation method is based on the Shenoy-Shafer method, but modifying it so that the results are approximations of the actual marginal distributions in exchange of lower time and space requirements.

The Shenoy-Shafer algorithm operates over the join tree built from the original network using a message passing scheme between adjacent nodes. Between every pair of adjacent nodes C_i and C_j there is a *mailbox* for the *messages* from C_i to C_j and another one for the messages from C_j to C_i. Sending a message from C_i to C_j can be considered as transferring the information contained in C_i that is relevant to C_j. Messages stored in both mailboxes are potentials defined for $C_i \cap C_j$. Initially these mailboxes are empty and once a message is stored it is full. A node C_i is allowed to send a message to its neighbour C_j if and only if every mailbox for messages arriving to C_i is full except the one from C_j to C_i.

The propagation is organised in two steps: in the first one, messages are sent from leaves to a previously selected root node, and in the second one the messages are sent from the root to the leaves.

The message from C_i to C_j is recursively defined as follows:

$$\phi_{C_i \to C_j} = \left\{ \phi_{C_i} \cdot \left(\prod_{C_k \in ne(C_i) \setminus \{C_j\}} \phi_{C_k \to C_i} \right) \right\}^{\downarrow C_i \cap C_j} , \qquad (2)$$

where ϕ_{C_i} is the original potential defined over C_i, $ne(C_i)$ is the set of adjacent nodes of C_i and superscript $\downarrow C_i \cap C_j$ indicates the marginal over $C_i \cap C_j$.

The main feature of the Penniless algorithm is that the messages are approximated, decreasing their size. This approximation [5,7] is performed after every combination and marginalisation in (2), and also when obtaining the posterior marginals. It consists of reducing the size of the probability trees used to represent the potentials by *pruning* some of their branches (namely, those that are more similar). The same approach can be taken within the MTE framework, with the difference that, instead of probability trees, the potentials are represented as mixed trees. Let us consider now how the pruning operation can be carried out over mixed trees.

5.1 Pruning a mixed tree

The size of an MTE potential (and consequently the size of its corresponding mixed tree) is determined by the number of leaves it has and the number of exponential terms in each leaf. Thus, a way of decreasing the size of the MTE potentials is decreasing each one of these two quantities. However, every pruning has an error associated with it. We propose to measure it in terms of divergence between the mixed trees before and after pruning.

Definition 7 (Divergence between mixed trees) *Let $\mathcal{T}$ be a mixed tree representing an MTE potential ϕ defined for $\mathbf{X} = (\mathbf{Y}, \mathbf{Z})$. Let $\mathcal{T}^*$ be a subtree of $\mathcal{T}$ with root $Z \in \mathbf{Z}$ where every child of Z is an MTE potential (i.e. a leaf node). Let ϕ_1 be the potential represented by $\mathcal{T}^*$. Let $\mathcal{T}_P^*$ be a tree obtained from $\mathcal{T}^*$ replacing ϕ_1 by the potential ϕ_2 for which it holds that $\int_{\Omega_\mathbf{Z}} \phi_1 d\mathbf{z} = \int_{\Omega_\mathbf{Z}} \phi_2 d\mathbf{z}$. The* divergence *between $\mathcal{T}^*$ and $\mathcal{T}_P^*$ is defined as*

$$D(\mathcal{T}^*, \mathcal{T}_P^*) = E_{\phi_1^*}[(\phi_1^* - \phi_2^*)^2] = \int_{\Omega_\mathbf{Z}} \frac{\phi_1(\mathbf{z})}{\Delta}\left(\frac{\phi_1(\mathbf{z})}{\Delta} - \frac{\phi_2(\mathbf{z})}{\Delta}\right)^2 d\mathbf{z},$$

where ϕ_i^* is the normalisation of ϕ_i and Δ is the total weight of ϕ:

$$\Delta = \sum_{\mathbf{Y}} \int_{\Omega_\mathbf{Z}} \phi(\mathbf{y}, \mathbf{z}) d\mathbf{z}.$$

We have considered three different kinds of pruning that are described in the next subsections.

5.1.1 Removing exponential terms.

In each leaf of the mixed tree, the exponential terms that have little impact on the density function could be removed and the resulting potential would be rather similar to the original one.

Let $f(\mathbf{z}) = k + \sum_{i=1}^{n} a_i e^{b_i \mathbf{z}}$ be the potential stored in a leaf. The goal is to detect those exponential terms $a_i e^{b_i \mathbf{z}}$ having little influence on the entire potential and to eliminate them. With this aim, a threshold α is established and the terms with lower absolute weight[2], $|p_i|$ are removed until only α terms remain in the potential . Whenever a term is removed, the resulting potential is updated by adding the maximum value of the term to the independent term k, and finally the potential is normalised in order to make it integrate up to the total weight of the original potential. The reason why the maximum of the potential

[2] The *weight* of an exponential term is $p_i = \int_{\Omega_\mathbf{Z}} a_i e^{b_i \mathbf{z}} d\mathbf{z}$.

is added to the independent term is to avoid negative points in the resulting potential. Here is the detailed procedure.

PRUNE_TERMS($\mathcal{T}$,α)

INPUT: A mixed tree $\mathcal{T}$ whose children are leaves (i.e. potentials). The maximum number of terms (α) in each potential.
OUTPUT: The pruned tree.

```
(1) Let $X$ be the label of the root node. For each child of $X$:
    (a) Let $f(\mathbf{z}) = k + \sum_{i=1}^{m} a_i e^{b_i \mathbf{z}}$ be its MTE function.
    (b) Let $p = \int_{\Omega_\mathbf{Z}} f(\mathbf{z}) d\mathbf{z}$.
    (c) Repeat
        (i) For i:=1 to no. terms in $f$, $p_i = \int_{\Omega_\mathbf{Z}} a_i e^{b_i \mathbf{z}} d\mathbf{z}$.
        (ii) Let $j = \min_i |p_i|$.
        (iii) $k' := k + \max_{\mathbf{z} \in \mathbf{Z}} \{a_j e^{b_j \mathbf{z}}\}$.
        (iv) Let $f_P(\mathbf{z}) = k' + \sum_{i \neq j} a_i e^{b_i \mathbf{z}}$.
        (v) $f := f_P$.
        (vi) Normalise $f_P$ to integrate up to $p$.
    (d) Until no. terms in $f \leq \alpha$.
(2) RETURN $\mathcal{T}$.
```

5.1.2 Merging intervals.

Let $\mathcal{T}$ be a mixed tree whose root node, X, is continuous, and its children are MTE functions. The domain of X is divided into intervals, I_j, and for each of those intervals, a potential $f_j(\mathbf{z}) = k^j + \sum_{i=1}^{n} a_i^j e^{b_i^j \mathbf{z}}$ is defined. If the potentials in two consecutive intervals are very similar, the intervals could be merged and therefore the same potential would be defined over the resulting interval with little loss of information. Two intervals I_{j_1} and I_{j_2} are merged by replacing the potentials $f_{j_1}(\mathbf{z})$ and $f_{j_2}(\mathbf{z})$ by another potential $f(\mathbf{z})$, defined for over $I_{j_1} \cup I_{j_2}$.

We propose to compute $f(\mathbf{z})$ as follows. Let $p_{j_1} = \int_{\Omega_\mathbf{Z}} f_{j_1}(\mathbf{z}) d\mathbf{z}$ and $p_{j_2} = \int_{\Omega_\mathbf{Z}} f_{j_2}(\mathbf{z}) d\mathbf{z}$ be the weights of $f_{j_1}(\mathbf{z})$ and $f_{j_2}(\mathbf{z})$ respectively. The new function is proportional to

$$f(\mathbf{z}) = \frac{p_{j_1} f_{j_1}(\mathbf{z}) + p_{j_2} f_{j_2}(\mathbf{z})}{p_{j_1} + p_{j_2}} \ .$$

Since both functions must integrate up to the same quantity over $I_{j_1} \cup I_{j_2}$, a constant K must be found such that $\int_{\Omega_{\mathbf{Z}}} Kf(\mathbf{z})d\mathbf{z} = p_1 + p_2$, which implies that $K = (p_1 + p_2)/(\int_{\Omega_{\mathbf{Z}}} f(\mathbf{z})d\mathbf{z})$. The intervals are actually merged if the divergence between the trees before and after replacing $f_{j_1}(\mathbf{z})$ and $f_{j_2}(\mathbf{z})$ by $f(\mathbf{z})$ is lower than a given threshold.

PRUNE_MERGE($\mathcal{T}$,ϵ)

INPUT: A mixed tree $\mathcal{T}$ whose root node is a continuous variable, whose children are MTE functions, and a divergence threshold ϵ.
OUTPUT: The pruned tree.

```
(1) Let $X$ be the label of the root node.
(2) If $X$ has more than one child:
    • Repeat
        (a) Let $f_1(\mathbf{z})$ and $f_2(\mathbf{z})$ be the potentials associated with two consecutive children of variable $X$, defined on intervals $I_1$ and $I_2$ respectively.
        (b) Let $p_1 = \int_{\Omega_{\mathbf{Z}}} f_1(\mathbf{z})$ and $p_2 = \int_{\Omega_{\mathbf{Z}}} f_2(\mathbf{z})$.
        (c) Let
            $f(\mathbf{z}) = \frac{p_1 f_1(\mathbf{z}) + p_2 f_2(\mathbf{z})}{p_1 + p_2} \quad \forall \mathbf{z} \in I_1 \cup I_2$ .
        (d) Let $\mathcal{T}_P$ be the tree result of replacing in $\mathcal{T}$ these two children of the root node by a single node with label $f(\mathbf{z})$.
        (e) If $D(\mathcal{T}, \mathcal{T}_P) < \epsilon$, make $\mathcal{T} := \mathcal{T}_P$.
    • Until every pair of neighbour children of $X$ have been explored.
(3) RETURN $\mathcal{T}$.
```

5.1.3 Pruning discrete variables.

Assume Y is a discrete variable in a node in a mixed tree, whose children are leaves. These leaf nodes are real numbers rather than MTE potentials, and therefore the pruning procedure can be the same as the one used for discrete networks in [9].

This *discrete pruning* procedure is carried out as follows. Assume $\mathcal{T}$ is a tree representing a potential ϕ whose root node is a discrete variable Y, and its children are real numbers, $p_i \in \mathbb{R}$ for $i = 1, \ldots, d$. Let $\overline{p} = (\sum_i p_i)/d$. Node Y is replaced by $\overline{p}$ if $D(\phi, \overline{p}) < \xi$, where ξ represents the threshold for the divergence increase. In this case, the Kullback-Leibler divergence [21] is used. The idea behind this pruning is that distributions which are very close to the uniform can be replaced by the average value without much loss of information.

Rather than establishing threshold ξ directly, we think it is more intuitive to determine it as in [9]. Instead, a value $\epsilon < 0.5$ is given, which represent the absolute deviation in probability from a binary uniform distribution. Then, ξ can be computed as the entropy of the distribution that varies an amount of ϵ from the uniform, i.e. $\xi = -((0.5 - \epsilon)\log(0.5 - \epsilon) + (0.5 + \epsilon)\log(0.5 + \epsilon))$. We will call **PRUNE_DISCRETE($\mathcal{T}$,ϵ)** the procedure that carries out the discrete pruning, which takes as input a tree with discrete root and leaf nodes as children, and the threshold ϵ, and returns the pruned tree.

Finally, the global pruning procedure can be described in terms of the three methods defined above as follows.

PRUNE($\mathcal{T}$,α,ϵ_{join},ϵ_{dis})

INPUT: A mixed tree, $\mathcal{T}$, a threshold weight α, a threshold divergence for merging intervals, ϵ_{join}, and a threshold for discrete pruning, ϵ_{dis}.
OUTPUT: The pruned tree.

```
(1) Let $X$ be the label of the root node of $\mathcal{T}$.
  (a) If the children of $X$ are leaves:
      (i) If $X$ is continuous:
          (A) $\mathcal{T}$ := PRUNE_MERGE($\mathcal{T}$,$\epsilon_{join}$).
          (B) $\mathcal{T}$ := PRUNE_TERMS($\mathcal{T}$,$\alpha$).
      (ii) If $X$ is discrete
          (A) $\mathcal{T}$ := PRUNE_DISCRETE($\mathcal{T}$,$\epsilon_{dis}$).
  (b) Else:
      (i) If $X$ is continuous, for each child of $X$:
            • Let $max$ and $min$ be the borders of the interval corresponding to this child.
            • $\mathcal{T}$ := PRUNE($\mathcal{T}^{R(X\in(min,max))}$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
      (ii) Else, for each child of $X$:
            • Let $a$ be the value of $X$ for that child.
            • $\mathcal{T}$ := PRUNE($\mathcal{T}^{R(X=a)}$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
  (c) RETURN $\mathcal{T}$.
```

5.2 The propagation algorithm

In this point we are ready to formulate the Penniless propagation algorithm for networks with MTEs. Like the Shenoy-Shafer, the algorithm starts selecting a root node in the join tree, and then in a first stage the messages are sent from the leaves to the root, and in a second stage the messages are distributed from the root to the leaves. The pseudo code for the Penniless algorithm is as follows, where it is assumed that the potentials in the nodes of the join tree and the messages are represented as mixed trees.

```
Penniless($J$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$)

INPUT: A join tree $J$. The pruning parameters $\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$
OUTPUT: The join tree $J$ after propagation.

(1) Choose a root node $R$.
(2) Initialise the clique potentials as in Shenoy-Shafer propagation.
(3) For each $C \in ne(R)$ [3],
      • PropagateUp($R$,$C$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
(4) For each $C \in ne(R)$,
      • $\phi$ :=COMBINE($\phi_R$,$\left(\prod_{C_k \in ne(R)\setminus C} \phi_{C_k \rightarrow R}\right)$).
      • $\phi$:= PRUNE($\phi$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
      • For all $X$ not in $R \cap C$, $\phi$ :=MARGINALISE_OUT($\phi$,$X$).
      • Compute the message $\phi_{R \rightarrow C}$:=PRUNE($\phi$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
      • PropagateDown($R$,$C$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
(5) RETURN $\mathcal{T}$.
```

```
PropagateUp($C_1$,$C_2$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$)

(1) For each $C \in ne(C_2)\setminus C_1$,
      • PropagateUp($C_2$,$C$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$)
(2) $\phi$ :=COMBINE($\phi_{C_2}$,$\left(\prod_{C_k \in ne(C_2)\setminus C_1} \phi_{C_k \rightarrow C_2}\right)$).
(3) $\phi$:= PRUNE($\phi$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
(4) For all $X$ not in $C_1 \cap C_2$, $\phi$ :=MARGINALISE_OUT($\phi$,$X$).
(5) Compute the message $\phi_{C_2 \rightarrow C_1}$:=PRUNE($\phi$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
```

[3] $ne(R)$ denotes the neighbour nodes of R in join tree J.

PropagateDown(C_1,C_2,α,ϵ_{join},ϵ_{dis})

```
(1) For each $C \in ne(C_2) \backslash C_1$,
    • $\phi$ :=COMBINE($\phi_{C_2}$,$\left(\prod_{C_k \in ne(C_2) \backslash C_1} \phi_{C_k \to C_2}\right)$).
    • $\phi$:= PRUNE($\phi$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
    • For all $X$ not in $C \cap C_2$, $\phi$ :=MARGINALISE_OUT($\phi$,$X$).
    • Compute the message $\phi_{C_2 \to C}$:=PRUNE($\phi$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
    • PropagateDown($C_2$,$C$,$\alpha$,$\epsilon_{join}$,$\epsilon_{dis}$).
```

6 Propagation algorithm based on MCMC

The Penniless algorithm is deterministic, which means that, for the same input parameters, the results are always the same. However, it is very common to find propagation algorithms with random character, based on Monte Carlo methods. In this work we have considered how the propagation with MTEs can be carried out using Monte Carlo simulations. More precisely, we have used the Markov Chain Monte Carlo (MCMC) algorithm outlined in [16].

Unlike the Penniless algorithm, MCMC proceeds by generating a sample of the variables in the network, and once the sample is generated, the posterior distributions are learnt from it. The way in which MTE densities can be estimated from data has been studied in [16,20,25].

The sample is generated starting from an initial configuration of the variables, which can be obtained by means of *forward sampling* [22].

Once this initial configuration has been obtained, a new one is generated simulating each variable X_i according to its distribution conditional on its *Markov blanket*, $\mathbf{W}_{X_i}$, restricted to the values of the current configuration.

The simulation procedure described above can be applied to MTE networks; the only item to specify is how to simulate a value from an MTE distribution [16].

When simulating a variable X_i, from its conditional distribution $f_i(x_i|\mathbf{W}_{X_i})$ given its Markov blanket restricted to the values $\mathbf{w}_{x_i}$ in the current configuration, we are simulating from a distribution depending only on X_i. If X_i is discrete, it is easy to simulate a value for it: a random number is generated and the inverse transform method is applied [23].

If X_i is continuous, the inverse transform method is not, in general, valid for MTE densities, since the inverse of the distribution function cannot be computed. However, the *composition method* [23] can be applied. The composition method consists of expressing the target density as a convex combination of densities for which the inverse transform method, or the acceptance-rejection technique can be applied.

Assume that the density of the variable to simulate is defined as:

$$f(x) = f_i(x) \quad \text{if} \quad \alpha_i \leq x < \beta_i, \quad i = 1, \ldots, k \ ,$$

where every function f_i is like:

$$f_i(x) = a_0 + a_1 e^{k_1 x} + a_2 e^{k_2 x} + \cdots + a_n e^{k_n x} \quad \alpha_i \leq x < \beta_i \ . \tag{3}$$

The way to simulate from $f(x)$ using the composition method consists of choosing one of the functions f_i with probability equal to $\int_{\alpha_i}^{\beta_i} f_i(x)dx$ and then simulate a value inside the interval (α_i, β_i) from a density proportional to f_i:

$$f_i^*(x) = \frac{f_i(x)}{\int_{\alpha_i}^{\beta_i} f_i(x)dx} \quad \alpha_i \leq x < \beta_i \ ,$$

which is also a function like (3).

There are two possible scenarios:

- **Every a_i in (3) is positive.** Then the composition method can again be applied as follows.

(1) The density f_i^* is decomposed as the following weighted sum of densities.

$$f_i^*(x) = p_1 f_1'(x) + \cdots + p_m f_m'(x) \tag{4}$$

where $\sum_{j=1}^m p_j = 1$.

(2) Two random numbers u_1 and u_2 are generated. u_1 is used to choose one of the f_j' functions with probability p_j, and u_2 is used to obtain a value for X applying the inverse transform method to the distribution corresponding to f_j'.

- **At least one a_i in (3) is negative.** In this case, the method proposed in [24] is used:

(1) Decompose the density f_i^* as a weighted sum of densities

$$f_i^*(x) = p_1 f_1'(x) + \cdots + p_m f_m'(x) \tag{5}$$

where $\sum_{j=1}^m p_j = 1$.

(2) Let N be the set of all i such that $p_i < 0$. Then

$$\begin{aligned} f_i^*(x) &= \left(\sum_{i\in N} p_i\right)\left(\frac{1}{\sum_{i\in N} p_i}\sum_{i\in N} p_i f_i'(x)\right) + \sum_{i\notin N} p_i f_i'(x) \\ &= \left(\sum_{i\in N} p_i\right) g(x) + \sum_{i\notin N} p_i f_i'(x) \ . \end{aligned} \tag{6}$$

(3) Simulate a value x^* from density $g(x)$ by the inverse transform method.

(4) Generate a random number u.

(5) If $u \leq \dfrac{f_i'(x^*)}{\sum_{i\in N} p_i f_i'(x^*)}$, accept x^* as the value generated for X, Else repeat from step 3.

The *efficiency* [23] of this method is ef $= 1/(\sum_{i\notin N} p_i)$, which means that each value x^* generated has a probability ef of being accepted as a value for X.

The decomposition in (4) and (6) must be such that the inverse of the distribution function corresponding to each f_j' can be easily computed. Such a decomposition can be obtained as follows. Define $c_j = \int_{\alpha_i}^{\beta_i} e^{k_j x} dx$, $j = 1, \ldots, n$. Then,

$$f_j'(x) = \frac{1}{c_j} e^{k_j x}, \quad j = 1, \ldots, n$$

is a density function over (α_i, β_i).

For $j = 0$, $c_0 = \int_{\alpha_i}^{\beta_i} dx = \beta_i - \alpha_i$ and hence

$$f_0(x) = \frac{1}{c_0}$$

is a density over (α_i, β_i).

So, multiplying and dividing each term by the corresponding c_j we get

$$f_i^*(x) = \frac{1}{\int_{\alpha_i}^{\beta_i} f_i(x)dx}\left(a_0 c_0 \frac{1}{c_0} + a_1 c_1 \frac{1}{c_1} e^{k_1 x} + \cdots + a_n c_n \frac{1}{c_n} e^{k_n x}\right) \quad \alpha_i \leq x \leq \beta_i$$

so that we can select as weights $p_j = \frac{a_j c_j}{\int_{\alpha_i}^{\beta_i} f_i(x)dx}$, $j = 0, \ldots, n$, which actually sum up to one:

$$\begin{aligned}\sum_{j=0}^{n} p_j &= \frac{1}{\int_{\alpha_i}^{\beta_i} f_i(x)dx} (a_0c_0 + a_1c_1 + \cdots + a_nc_n) \\ &= \frac{1}{\int_{\alpha_i}^{\beta_i} f_i(x)dx} \left(a_0 \int_{\alpha_i}^{\beta_i} dx + a_1 \int_{\alpha_i}^{\beta_i} e^{k_1x}dx + \cdots + a_n \int_{\alpha_i}^{\beta_i} e^{k_nx}dx \right) \\ &= \int_{\alpha_i}^{\beta_i} f_i^*(x)dx = 1 \ .\end{aligned}$$

Finally the inverse of the distribution function of each f_j' is computed. If $j = 0$, the distribution function is uniform over (α_i, β_i). If $j > 0$, for $x \in (\alpha_i, \beta_i)$, the distribution function is

$$F_j'(x) = \int_{-\infty}^{x} f_j'(t)dt = \int_{\alpha_i}^{x} \frac{1}{c_j} e^{k_jt}dt = \frac{1}{c_jk_j} \left(e^{k_jx} - e^{k_j\alpha_i} \right) \ .$$

Hence, given a random number u, $0 < u < 1$, its inverse is computed as:

$$\begin{aligned} u = \frac{1}{c_jk_j} \left(e^{k_jx} - e^{k_j\alpha_i} \right) &\Rightarrow e^{k_jx} = c_jk_ju + e^{k_j\alpha_i} \Rightarrow \\ k_jx = \log \left(c_jk_ju + e^{k_j\alpha_i} \right) &\Rightarrow x = \frac{1}{k_j} \log \left(c_jk_ju + e^{k_j\alpha_i} \right) \ . \end{aligned}$$

So, for $j > 0$, $F_j'^{-1}(u) = \frac{1}{k_j} \log \left(c_jk_ju + e^{k_j\alpha_i} \right) \quad 0 < u < 1.$

7 Experimental evaluation of the algorithms

In order to test the performance of the Penniless and the MCMC algorithms, we have carried out a series of experiments aimed to validate the accuracy of the proposed algorithms (Penniless and MCMC) and also to show the advantage of the MTE approach versus the alternative of discretising the continuous variables in order to treat them as discrete. We have used three artificial hybrid networks, denoted as Net1, Net2 and Net3. Net1 has 42 continuous variables and 3 discrete, Net2 has 77 and 8 and finally Net3 has 86 and 11.

Three networks have been generated following these restrictions:

(1) The number of parents of each variable follows a Poisson distribution with mean 0.8. Once that number is determined, the actual parents are chosen at random.
(2) Discrete variables:

- Its number of states is simulated from the distribution showed in Table 1.
- The probability value of each state is simulated from a Negative Exponential distribution with mean 0.5.

(3) Continuous variables:

- The number of splits of the variable in a potential is simulated from the distribution showed in Table 1.
- Every MTE potential has an independent term which is simulated from a Negative Exponential distribution with mean 0.01 and a number of exponential terms determined by the distribution showed in Table 1.
- In every exponential term, $a \exp\{bx\}$ the coefficient a is a real number following a Negative Exponential distribution with mean 1, and the exponent b is a real number determined by a standard Normal distribution (mean 0 and standard deviation 1).

After simulating the parameters of the potentials, they are normalised in order to guarantee that the potentials are density functions.

Table 1
Distributions used for generating the artificial networks

States	2	3	4
Prob.	1/3	1/3	1/3

Splits	1	2	3
Prob.	0.2	0.4	0.4

Exp. terms	0	1	2
Prob.	0.05	0.75	0.20

For each network, the 30% of its variables are observed at random. The corresponding evidence is inserted in the network by restricting the potentials to the observed values.

Table 2
Different pruning parameters evaluated.

Setting	A	B	C	D	E
ϵ_{join}	0	0.005	0.005	0.05	0.05
ϵ_{dis}	0	0	0.01	0	0.01

We have considered five setting of the pruning parameters for thee Penniless algorithm. These settings are labeled as A,B,C, D and E, and the corresponding values for discrete pruning and merging intervals are shown in Table 2. We will refer to each setting as Penni A, ..., Penni E. ϵ_{join} is the maximum error allowed for joining two intervals, while ϵ_{disc} indicates that discrete distributions that differ less than the value of the parameter with respect to a uniform distribution, in terms of entropy, are pruned. The maximum number of exponential terms is set to 2 in all the cases (i.e. $\alpha = 2$).

With the aim of comparing the MTE framework with the discretisation, the results of the propagation are compared with those provided by Shenoy-Shafer propagation for the discretisation obtained by replacing every MTE potential $f(\mathbf{z}) = k + \sum_{i=1}^{n} a_i e^{b_i \mathbf{z}}$ by a constant function $f^*(\mathbf{z}) = k^*$ such that $\int_{\Omega_{\mathbf{Z}}} f(\mathbf{z}) d\mathbf{z} = \int_{\Omega_{\mathbf{Z}}} f^*(\mathbf{z}) d\mathbf{z}$. This method will be denoted as Disc from now on.

After each propagation, the following statistics are computed:

- The maximum size of the potential needed to compute the marginal distribution. It is achieved after combining all the messages sent to the clique that contains the variable in the join tree.
- The error attached to it, according to Definition 7.

For each network, the mean of these quantities is computed for all the unobserved variables.

Regarding the MCMC algorithm, we have used samples of size 5000, and the propagation has been repeated ten times for each network. For each sample, three MTE densities are estimated for each unobserved variable, splitting the domain of the variable in 2, 3 and 4 pieces. We will denote these alternatives as MCMC 2, 3, 4. So, given a non-observed variable and the number of splits, we have ten different errors attached to it (one for each repetition of the experiment) , and we use the average of these errors as the *global* error of the variable for the given number of splits.

The summary of the obtained results is shown in Tables 3 and 4, where the best result for each network is marked in boldface and the worst result is underlined. Table 3 contains the average and maximum global error of all the variables in each network, while Table 4 displays the average and maximum sizes of the potentials obtained when computing the marginals for each variable. Notice that Table 4 does not have any entry for MCMC, since this algorithm computes the marginals directly from a sample and therefore it is not derived from other potentials.

The results of the experiments show that the use of MTEs instead of discretisation provides more accurate results. It is not surprising, since the discretisation is just a particular case of the MTE framework (a discretised density is an MTE density with one independent term an none exponential terms). However, it is important to point out that the increase in space required by the MTEs is significantly lower than the gain in accuracy, which means that the tradeoff space/accuracy, according to the evidence provided by the experiments reported here, is favourable to the MTE. The tradeoff between space and accuracy can be controlled using the pruning parameters, as shown by the different settings of the Penniless algorithm considered here.

Algorithm MCMC, however, is not competitive attending to the obtained results. It is not surprising, since the same happens in discrete networks, where more sophisticated simulation methods, like importance sampling, are used instead.

Furthermore, the Penniless algorithm is more robust than MCMC and Disc concerning the variability of the errors for the different variables in a network.

This claim is clearly supported by the box and whisker charts displayed in Figures 2, 3 and 4, that provide a representation of the vector of errors for each algorithm in each network. Notice that the outliers have not been represented in these charts.

Table 3
Errors in the approximations provided by the tested algorithms.

	Net1		Net2		Net3	
	Mean	Max	Mean	Max	Mean	Max
Penni A	**0.0031**	**0.0296**	0.0079	0.1283	**0.0051**	**0.04137**
Penni B	0.0041	0.0326	**0.0078**	**0.0921**	0.0071	0.09219
Penni C	0.0042	0.0326	0.0090	0.1176	0.0074	0.1215
Penni D	0.0112	0.0839	0.0235	0.1478	0.0195	0.2564
Penni E	0.0111	0.0804	0.0228	0.1392	0.0188	0.2323
Disc	0.0293	0.3465	0.0410	0.4512	0.0440	0.7509
MCMC 2	0.2654	3.2530	0.4739	17.2756	0.1956	2.7858
MCMC 3	0.2532	3.3459	0.3743	12.6620	0.1664	3.2575
MCMC 4	0.2586	3.4432	0.4060	14.6387	0.1637	3.2730

Table 4
Mean and maximum potential sizes.

	Net1		Net2		Net3	
	Mean	Max	Mean	Max	Mean	Max
Penni A	41	288	67.5370	432	86.65	1536
Penni B	33	216	32.7592	138	50.2333	477
Penni C	27.8965	144	29.4815	138	49.0833	477
Penni D	26.06897	180	24.90741	**108**	39.2833	**390**
Penni E	21.7931	108	**21.9630**	**108**	38.1333	**390**
Disc	**14.2759**	**96**	22.7963	144	**31.0833**	512

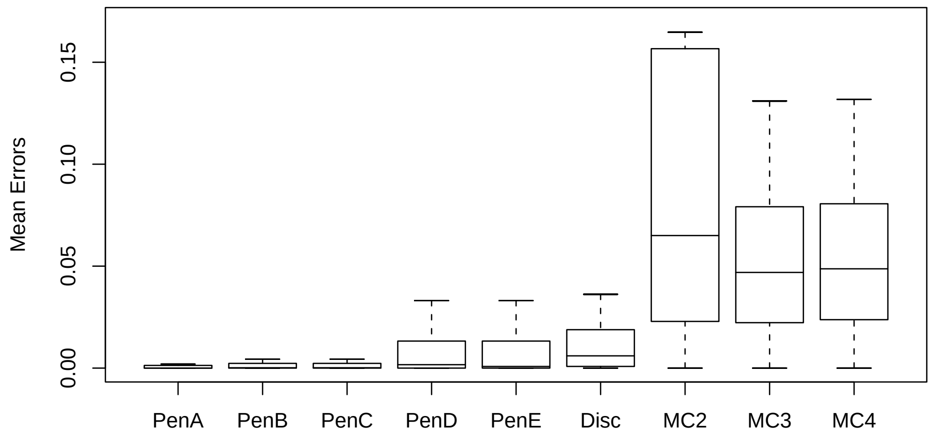

Fig. 2. Box and whisker chart of the errors for Net1

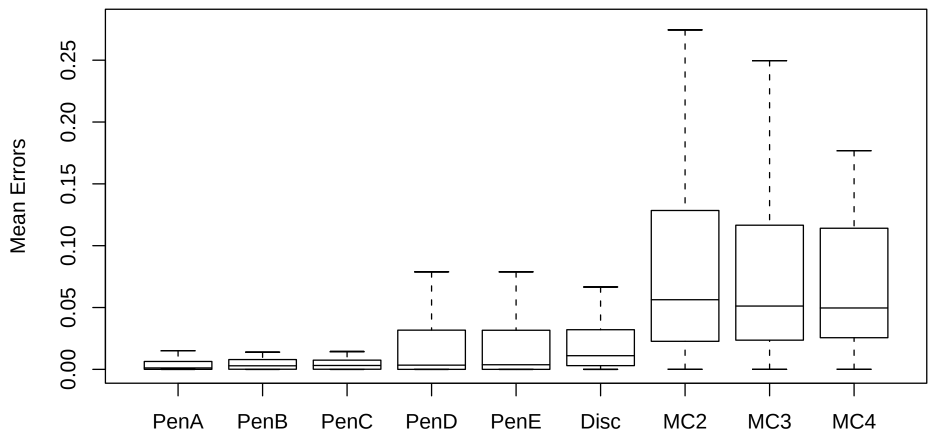

Fig. 3. Box and whisker chart of the errors for Net2

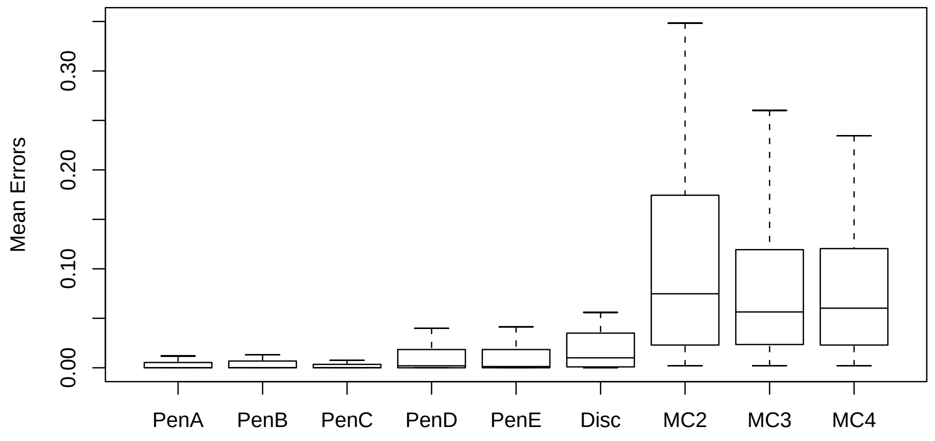

Fig. 4. Box and whisker chart of the errors for Net3

8 Conclusions

Previously to this work, some propagation methods had been successfully applied to MTE networks, for instance Shenoy-Shafer propagation [17], but so far they were not able to overcome the problem of the exponential increase of the sizes of the potentials involved in the propagation, specially when evidence is entered. In this paper we have presented a method to apply Penniless propagation to MTE networks, so that the sizes of the potentials are reduced because of the pruning operation.

The performance of the method has been tested on three artificial networks. The results of the experiments suggest that the Penniless algorithm is appropriate for MTE models, since the tradeoff between space requirements and accuracy is better than the one obtained with the discretisation.

We have also tested the use of Markov Chain Monte Carlo for solving the propagation problem, but the experiments support the conclusion that this method is not competitive.

The ideas contained in this paper can be extended to other propagation methods, specially the Lazy propagation and the class of Importance Sampling propagation algorithms, since these methods can take advantage of the reduction of the sizes of the potentials after pruning.

References

[1] F. Jensen, S. Lauritzen, K. Olesen, Bayesian updating in causal probabilistic networks by local computation, Computational Statistics Quarterly 4 (1990) 269–282.

[2] S. Lauritzen, D. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, Journal of the Royal Statistical Society, Series B 50 (1988) 157–224.

[3] A. Madsen, F. Jensen, Lazy propagation: a junction tree inference algorithm based on lazy evaluation, Artificial Intelligence 113 (1999) 203–245.

[4] P. Shenoy, G. Shafer, Axioms for probability and belief function propagation, in: R. Shachter, T. Levitt, J. Lemmer, L. Kanal (Eds.), Uncertainty in Artificial Intelligence 4, North Holland, Amsterdam, 1990, pp. 169–198.

[5] A. Cano, S. Moral, A. Salmerón, Penniless propagation in join trees, International Journal of Intelligent Systems 15 (2000) 1027–1059.

[6] A. Cano, S. Moral, A. Salmerón, Lazy evaluation in Penniless propagation over join trees, Networks 39 (2002) 175–185.

[7] A. Cano, S. Moral, A. Salmerón, Novel strategies to approximate probability trees in Penniless propagation, International Journal of Intelligent Systems 18 (2003) 193–203.

[8] F. Jensen, S. Andersen, Approximations in Bayesian belief universes for knowledge-based systems, in: Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence, 1990, pp. 162–169.

[9] A. Salmerón, A. Cano, S. Moral, Importance sampling in Bayesian networks using probability trees, Computational Statistics and Data Analysis 34 (2000) 387–413.

[10] E. Santos, S. Shimony, E. Williams, Hybrid algorithms for approximate belief updating in Bayes nets, International Journal of Approximate Reasoning 17 (1997) 191–216.

[11] A. Christofides, B. Tanyi, D. Whobrey, N. Christofides, The optimal discretization of probability density functions, Computational Statistics and Data Analysis 31 (1999) 475 – 486.

[12] D. Kozlov, D. Koller, Nonuniform dynamic discretization in hybrid networks, in: D. Geiger, P. Shenoy (Eds.), Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence, Morgan & Kaufmann, 1997, pp. 302–313.

[13] S. Lauritzen, Propagation of probabilities, means and variances in mixed graphical association models, Journal of the American Statistical Association 87 (1992) 1098–1108.

[14] K. Olesen, Causal probabilistic networks with both discrete and continuous variables, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (1993) 275–279.

[15] D. Koller, U. Lerner, D. Anguelov, A general algorithm for approximate inference and its application to hybrid Bayes nets, in: K. Laskey, H. Prade (Eds.), Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, Morgan & Kaufmann, 1999, pp. 324–333.

[16] S. Moral, R. Rumí, A. Salmerón, Mixtures of truncated exponentials in hybrid Bayesian networks, in: Lecture Notes in Artificial Intelligence, Vol. 2143, 2001, pp. 135–143.

[17] B. Cobb, P. Shenoy, R. Rumí, Approximating probability density functions with mixtures of truncated expoenntials, in: Proceedings of the Tenth International Conference IPMU'04, Perugia (Italy), 2004.

[18] A. Cano, S. Moral, Heuristic algorithms for the triangulation of graphs, in: B. Bouchon-Meunier, R. Yager, L. Zadeh (Eds.), Advances in Intelligent Computing, Springer Verlag, 1995, pp. 98–107.

[19] U. Kjærulff, Optimal decomposition of probabilistic networks by simulated annealing, Statistics and Computing 2 (1992) 1–21.

[20] S. Moral, R. Rumí, A. Salmerón, Estimating mixtures of truncated exponentials from data, in: Proceedings of the First European Workshop on Probabilistic Graphical Models, 2002, pp. 156–167.

[21] S. Kullback, R. Leibler, On information and sufficiency, Annals of Mathematical Statistics 22 (1951) 76–86.

[22] M. Henrion, Propagating uncertainty by logic sampling in Bayes' networks, in: J. Lemmer, L. Kanal (Eds.), Uncertainty in Artificial Intelligence, Vol. 2, North-Holland (Amsterdam), 1988, pp. 317–324.

[23] R. Rubinstein, Simulation and the Monte Carlo Method, Wiley (New York), 1981.

[24] A. Bignami, A. Matties, A note on sampling from combinations of distributions, J.Inst. Maths Applics 8 (1971) 80 – 81.

[25] R. Rumí, Kernel methods in bayesian networks, in: Proceedings of the International mediterranean congress of mathematics, 2005, pp. 135 – 149.