Different Strategies to Approximate Probability Trees in Penniless Propagation*

Andrés Cano, Serafín Moral,
Dpt. Computer Science and A. I.
University of Granada. Spain
e-mail: {acu,smc}@decsai.ugr.es

Antonio Salmerón
Dpt. Statistics and A. M.
University of Almería. Spain
e-mail: Antonio.Salmeron@ual.es

Abstract: *In this paper we propose some modifications over the Penniless algorithm. We use an improved measure of information to calculate the error of the approximations, what leads to a new way of pruning several values in a probability tree by a single one, computed from the value stored in the tree being pruned but taking into account the message stored in the opposite direction. Also, we have considered the possibility of replacing small probability values by zero. Locally, this is not an optimal approximation strategy, but in this problem a lot of different local approximations are carried out to obtain a final approximate value. In some experiments we will show that approximating by zeros we improve the quality of the final approximations.*

Keywords: Bayesian networks, Penniless propagation, join tree, probability trees.

1 Introduction

Bayesian networks are graphical models for efficiently handling uncertainty in probabilistic expert systems (expert systems where uncertainty is measured in terms of probability). A *Bayesian network* is a directed acyclic graph where each node represents a random variable, and the topology of the graph encodes the independence relations among the variables, according to the d-separation criterion. Associated with the graph, there is a probability distribution for each node conditioned on its parents, such that the joint distribution over all the variables in the network factorizes as the product of those conditional distributions.

The reasoning task, also called *probability propagation*, consists in the computation of the posterior marginals over some variables of interest given that the value of some other

*Partially supported by Junta de Andalucía, research groups TIC103 and FQM244

variables are known. Propagation algorithms have been proposed for computing posterior conditional probabilities without actually computing the joint distribution; instead, they operate over an auxiliary structure called *join tree* by means of local computations [JLO90, She97, MJ99]. However, in large networks the use of these algorithms usually becomes infeasible.

In order to deal with such large networks, approximate algorithms can be used, that provide results (though inexact) in a lower time. Some of these methods are based on Monte Carlo simulation [SCM00], while other are based on deterministic procedures. One of the more recent contributions within the group of deterministic algorithms is the so-called *Penniless propagation algorithm* [CMS00], which is based on Shenoy-Shafer propagation over binary join trees [She97], but representing probabilistic information by means of *probability trees* [SCM00]. The use of probability trees allows to approximate big probabilistic potentials by smaller ones, pruning the branches of the tree, making propagation be feasible even under limited resources (RAM and CPU).

In this paper we propose some modifications over the Penniless propagation algorithm. We use an improved measure of information to calculate the error of the approximations. This new measure suggests a novel way of pruning several values in a probability tree by a single one, computed from the value stored in the tree being pruned but taking into account the message stored in the opposite direction.

Also, we have considered the possibility of replacing small probability values by zero, with the aim of speeding up computations and controlling the complexity of the trees used to represent potentials. We will show a theorem showing that the best approximation of a potential conditional to other one, is obtained by substituting a node such that its children are leaves by a weighted average of the values on the leaves. However, here we will consider a different strategy: making the substitution by zero when the sum of the values of the children is very low. Though this is not an optimal strategy, it has a potential advantage. The penniless algorithm carries out several consecutive approximation steps. Each message is approximated and then it is used for further computations (multiplications and marginalizations). In these operations the complexity of the results is in the worst case exponential relative to the complexity of the operands, being the multiplication operation the main source of more complex potentials (the size of the frame is increased, whereas in marginalization is decreased). If we approximate a branch by a zero instead of a number different from zero, then the complexity of the representation of potentials in the zero parts does not increase by multiplication. The result of the multiplication by a zero value in a leaf of a tree representing a potential is always zero for each value of the other potential, and the result can be represented again using the same node. In this way, though the approximation is not optimal, we obtain simpler approximation problems in subsequent steps which may be, in some situations, better for the final approximate value. This fact is corroborated by the experiments in the paper.

We start off with a brief explanation on Shenoy-Shafer propagation in section 2 and Penniless propagation in section 3, where we also introduce the new contributions in this paper in subsection 3.2. The experiments carried out with the resulting algorithms are described in section 4 and the paper ends with conclusions in section 5.

2 Propagation over join trees

In all of this paper we will consider a Bayesian network defined for a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, each variable X_i taking values on a finite set U_i containing $|U_i|$ elements. If $I \subseteq N = \{1, \ldots n\}$ is a set of indices, we will write $\mathbf{X}_I$ for the set of variables $\{X_i | i \in I\}$, defined on $U_I = \times_{i \in I} U_i$. Given $\mathbf{x} \in U_I$ and $J \subseteq I$, $\mathbf{x}_J$ will denote the element of U_J obtained from $\mathbf{x}$ dropping the coordinates not in J. Given $\mathbf{x} \in U_J$ and $J \subseteq I$, we will denote by $A^I_{\mathbf{x}}$ the set of values $\mathbf{y} \in U_I$ such that $\mathbf{y}_J = \mathbf{x}$, i.e. the set of elements in U_I coinciding with $\mathbf{x}$ in the coordinates in J. If ϕ is a potential[1] defined on U_I, $\mathrm{dom}(\phi)$ will denote the set of indices of the variables for which ϕ is defined (i.e. $\mathrm{dom}(\phi) = I$).

The *marginal* of a potential ϕ for a set of variables $\mathbf{X}_J$ with $J \subseteq I$ is denoted by $\phi^{\downarrow J}$ and it is a function defined for variables $\mathbf{X}_J$ as $\phi^{\downarrow J}(\mathbf{y}) = \sum_{\mathbf{x}_J = \mathbf{y}} \phi(\mathbf{x})$ for all $\mathbf{y} \in U_J$.

The *combination or product* of two potentials ϕ and ϕ' is a new potential $\phi \cdot \phi'$ defined for variables $\mathbf{X}_{\mathrm{dom}(\phi) \cup \mathrm{dom}(\phi')}$ and obtained by point-wise multiplication.

The conditional distribution of each variable X_i, $i = 1, \ldots, n$, given its parents in the network, $\mathbf{X}_{pa(i)}$, is denoted by a potential $p_i(x_i | \mathbf{x}_{pa(i)})$ defined over $U_{\{i\} \cup pa(i)}$, and the joint probability distribution for the n-dimensional random variable $\mathbf{X}_N$ can be expressed as

$$p(\mathbf{x}) = \prod_{i \in N} p_i(x_i | \mathbf{x}_{pa(i)}) \quad \forall \mathbf{x} \in U_N \ . \tag{1}$$

If we denote by $\mathbf{e}$ the values of the observed variables, and by E their indices, the task of probability propagation can be seen as calculating the posterior probability function $p(x'_k | \mathbf{e}) = p(x'_k, \mathbf{e}) / p(\mathbf{e})$, for every $x'_k \in U_k$, $k \in \{1, \ldots, n\} \setminus E$.

In terms of potential notation, if we call H the set of potentials corresponding to the conditional distributions in the network, restricted to the observed values $\mathbf{e}$, the goal of probability propagation is to obtain, for each variable of interest X_k,

$$\phi^m_{X_k} = \left(\prod_{\phi \in H} \phi \right)^{\downarrow k} \ , \tag{2}$$

where superscript m indicates *posterior marginal*. Afterwards, the conditional distribution can be computed by normalizing $\phi^m_{X_k}$.

The computation of $\phi^m_{X_k}$ can be organized in a join tree, which is a tree where each node V is a subset of $\mathbf{X}_N$, and such that if a variable is in two distinct nodes, V_1 and V_2, then it is also in every node in the path between V_1 and V_2. A join tree is called *binary* if every node has no more than three neighbors. Every potential in the set of initial potentials, $\phi \in H$, is assigned to a node V_j such that $\mathbf{X}_{\mathrm{dom}(\phi)} \subseteq V_j$. In this way, attached to every node V_i there will be a potential ϕ_{V_i} defined over the set of variables V_i and that is equal to the product of all the potentials assigned to it. The Penniless algorithm operates over a binary join tree [CMS00, She97], and is based on the Shenoy-Shafer propagation algorithm, that we briefly describe now.

[1] A potential is a non negative function representing a conditional, joint, or marginal distribution

The *Shenoy-Shafer propagation algorithm* is carried out by sending messages in the two directions of each edge of the join tree. The *messages* between two adjacent nodes V_i and V_j are potentials defined on $V_i \cap V_j$ (see [SS90] for the details). The message V_i-outgoing and V_j-incoming is computed as

$$\phi_{V_i \to V_j} = \left\{ \phi_{V_i} \cdot \left(\prod_{V_k \in ne(V_i) \setminus \{V_j\}} \phi_{V_k \to V_i} \right) \right\}^{\downarrow V_i \cap V_j} , \tag{3}$$

where ϕ_{V_i} is the initial probability potential on V_i reduced to the observations $\mathbf{e}$, $\phi_{V_k \to V_i}$ are the messages V_k-outgoing and V_i-incoming and $ne(V_i)$ are the neighbor nodes of V_i.

The propagation is organized in two stages. In the first one, messages are sent from the leaves to a previously selected root node (*upward propagation*), and in the second stage, messages are sent from the root to the leaves (*downward propagation*). After these two stages, in order to compute the posterior marginal for variable X_k, we first determine a node V_i containing X_k and compute $\phi^m_{V_i} = \phi_{V_i} \cdot (\prod_{V_k \in ne(V_i)} \phi_{V_k \to V_i})$. The conditional distribution given $\mathbf{e}$ for X_k can be calculated marginalizing $\phi^m_{V_i}$ down to X_k (obtaining $\phi^m_{X_k}$) and normalizing the result.

3 Penniless propagation

Penniless propagation [CMS00] is a deterministic approximate propagation algorithm based on Shenoy-Shafer's method, which aim is to provide (approximate) results under limited resources. One of the main characteristics of this method is the use of *probability trees* [BFGK96], which allow to represent potentials in an approximate way within a given maximum number of values [CM97, Koz98, SCM00].

Since Penniless algorithm is based on Shenoy-Shafer's, it operates over binary join trees, since this kind of propagation was shown to be more efficient over this structure [She97].

One of the added features of Penniless is that the messages sent during the propagation are approximated in order to reduce their size. Another difference with respect to Shenoy-Shafer's algorithm is the number of stages of the propagation: Penniless propagation may perform more than two stages, in which messages are gradually improved taking into account the information flowing across the join tree. Thus, the basis of Penniless propagation is the use of probability trees as an approximate representation of the messages, and the incremental improvement on the quality of the approximations as the number of propagations is increased.

3.1 Probability trees

A *probability tree* [BFGK96, CM97, SCM00] is a directed labeled tree, where each internal node represents a variable and each leaf node represents a probability value. The number of leaves of a tree $\mathcal{T}$ is its *size*. Each internal node has as many outgoing arcs as states the variable it represents has.

A probability tree $\mathcal{T}$ on variables $\mathbf{X}_I$ represents a potential ϕ if for each $\mathbf{x}_I \in U_I$ the value $\phi(\mathbf{x}_I)$ is the number stored in the leaf node that is reached starting in the root node and selecting for each internal node labeled with X_i the child corresponding to value x_i.

Two important features of probability trees are that they can represent the same information as a probability table, but using less values, and that they can approximate the original tree by substituting some values by a single one (see figure 1).

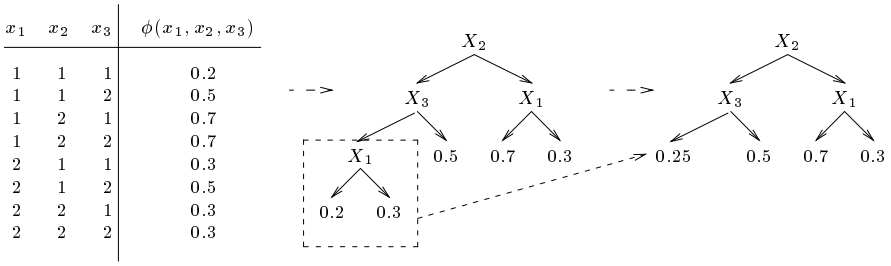

Figure 1. A potential, a probability tree representing it and an approximation of the exact tree. Arcs outgoing from a variable from left to right correspond to the values of the variable in lexicographical order.

The operations involved in propagation algorithms (combination, marginalization and restriction), can be carried out over the probability tree representation (see [CMS00, SCM00] for the details). In the case of Penniless propagation, another operation is particularly important: the approximation operation. So, we will concentrate on it.

3.2 Approximate probability trees

By approximating a tree $\mathcal{T}_1$ representing a potential ϕ we mean to obtain a tree $\mathcal{T}$ smaller than $\mathcal{T}_1$, but trying to keep a close representation of potential ϕ. One way of obtaining that approximate tree is by pruning the original one. A *pruning* of a tree consists in selecting a node such that all its children are leaves and replacing it and its children by one node containing a number. In the general case, the optimum number to be placed in that node is the average of the values of the leaf nodes being removed (this minimizes the Kullback-Leibler divergence [KL51] between the original tree and the approximate one [CM97, SCM00]).

However, in the Penniless propagation scheme, trees representing messages through an edge are approximated taking into account the message (a probability tree) in the opposite direction in the same edge. More precisely, the goal is to approximate a potential ϕ represented by a tree $\mathcal{T}$, by another potential ϕ' represented by another tree $\mathcal{T}'$ of smaller size, conditional on another potential ψ. In this section we will assume all the potentials to be defined on frame U_I. For a potential ϕ, let us consider the following notation:

- $\text{sum}(\phi|A) = \sum_{\mathbf{x}\in A} \phi(\mathbf{x})$, where $A \subseteq U_I$.

- $\text{sum}(\phi) = \text{sum}(\phi|U_I) = \sum_{\mathbf{x}\in U_I} \phi(\mathbf{x})$.
- If $\text{sum}(\phi) \neq 0$, then $N(\phi) = \phi/\text{sum}(\phi)$.

We will measure the distance between two potentials ϕ and ϕ' conditional on ψ by the Kullback-Leibler divergence between the normalized potentials:

$$D(\phi, \phi'|\psi) = \sum_{\mathbf{x}\in U_I} N(\phi(\mathbf{x})\psi(\mathbf{x})) \log\left(\frac{\phi(\mathbf{x})\text{sum}(\phi'\cdot\psi)}{\phi'(\mathbf{x})\text{sum}(\phi\cdot\psi)}\right) \ . \tag{4}$$

Since there is no difference between the distances $D(\phi, \phi'|\psi)$ and $D(\phi, \phi''|\psi)$ if $N(\phi') = N(\phi'')$, i.e. the distance is independent of the normalization factor, then ϕ' will be determined up to a normalization value. In [CMS00] it was assumed that ϕ' and ϕ were such that $\text{sum}(\phi') = \text{sum}(\phi)$, but here we will assume that $\text{sum}(\phi'\cdot\psi) = \text{sum}(\phi\cdot\psi)$. The selection of a normalization factor does not have any effect in the quality of the approximation, but with this assumption the results are simpler to express and prove.

As it has been reported in [BFGK96, CM97], the difficulty of the approximation lies in finding the *structure* of the tree, i.e. the same tree without numbers on the leaves. In [CMS00] we assumed that given a structure $\mathcal{S}$ we can build an approximate tree denoted by $\mathcal{T}_\mathcal{S}$ from ϕ by assigning to each leave characterized by configuration $\mathbf{X}_J = \mathbf{x}_J$, the average of potential ϕ in points in $A^I_{\mathbf{x}_J}$ (points in U_I for which $\mathbf{X}_J = \mathbf{x}_J$). However, this strategy is not optimal; it is appropriate when we do not have a conditioning potential, ψ, or when this potential is equal to 1, but not with general conditional information. This problem can be stated in the following general way: we have a potential ϕ defined on U_I and a partition $\mathcal{A}$ of this frame. We want to find a potential ϕ' which is constant in each set $A \in \mathcal{A}$ and such that the distance of ϕ to ϕ' conditioned to ψ is minimal. In our case, given a tree structure $\mathcal{S}$ the elements of the partition are defined by the leaves of the structure. If a leaf is characterized by configuration $\mathbf{X}_J = \mathbf{x}_J$, then this leaf defines the set $A = A^I_{\mathbf{x}_J}$. Then we can prove the following result, showing that now, the optimal strategy is to assign to the elements from A the average of ϕ weighted by the values of ψ.

Theorem 1 *If ϕ is a potential defined on U_I and $\mathcal{A}$ is a partition of U_I, then the potential ϕ' which is constant in the elements of each set $A \in \mathcal{A}$, with $\text{sum}(\phi\cdot\psi) = \text{sum}(\phi'\cdot\psi)$ and minimizing the distance (4) from ϕ to ϕ' given ψ is given by the potential ϕ' assigning to every element $\mathbf{x} \in A$ the value*

$$\phi'(\mathbf{x}) = \frac{\text{sum}(\phi\cdot\psi|A)}{\text{sum}(\psi|A)} \ . \tag{5}$$

Proof: Let us call $\phi'(A)$ to the constant value of ϕ' in the elements of A. We have,

$$D(\phi, \phi'|\psi) \ = \ \sum_{\mathbf{x}\in U_I} N(\phi(\mathbf{x})\psi(\mathbf{x})) \log\left(\frac{\phi(\mathbf{x})\text{sum}(\phi'\cdot\psi)}{\phi'(\mathbf{x})\text{sum}(\phi\cdot\psi)}\right) = \sum_{\mathbf{x}\in U_I} \frac{\phi(\mathbf{x})\psi(\mathbf{x})}{\text{sum}(\phi\cdot\psi)} \log\left(\frac{\phi(\mathbf{x})}{\phi'(\mathbf{x})}\right)$$

$$= \frac{1}{\text{sum}(\phi \cdot \psi)} \sum_{A \in \mathcal{A}} \sum_{\mathbf{x} \in A} \phi(\mathbf{x})\psi(\mathbf{x}) \log\left(\frac{\phi(\mathbf{x})}{\phi'(A)}\right)$$
$$= \frac{1}{\text{sum}(\phi \cdot \psi)} \sum_{A \in \mathcal{A}} \left(\sum_{\mathbf{x} \in A} \phi(\mathbf{x})\psi(\mathbf{x}) \log(\phi(\mathbf{x})) - (\sum_{\mathbf{x} \in A} \phi(\mathbf{x})\psi(\mathbf{x})) \log(\phi'(A))\right) \ .$$

Without taking into account constant parts not depending on ϕ', we have that minimizing this quantity is equivalent to maximizing

$$\sum_{A \in \mathcal{A}} (\sum_{\mathbf{x} \in A} \phi(\mathbf{x})\psi(\mathbf{x})) \log(\phi'(A)) = \sum_{A \in \mathcal{A}} \text{sum}(\phi \cdot \psi | A) \log(\phi'(A)) \ .$$

Adding the constant value $\sum_{A \in \mathcal{A}} \text{sum}(\phi \cdot \psi | A) \log(\text{sum}(\psi | A))$, which does not depend on ϕ', we get that we have to maximize

$$\sum_{A \in \mathcal{A}} \text{sum}(\phi \cdot \psi | A) \log(\phi'(A) \cdot \text{sum}(\psi | A)) \ . \tag{6}$$

Now, we have that being ϕ' constant in A, $\text{sum}(\phi' \cdot \psi) = \sum_{A \in \mathcal{A}} \phi'(A)\text{sum}(\psi | A)$, and as $\sum_{A \in \mathcal{A}} \text{sum}(\phi \cdot \psi | A) = \text{sum}(\phi \cdot \psi)$, and as $\text{sum}(\phi \cdot \psi) = \text{sum}(\phi' \cdot \psi)$, we have that $\sum_{A \in \mathcal{A}} \text{sum}(\phi \cdot \psi | A) = \sum_{A \in \mathcal{A}} \phi'(A)\text{sum}(\psi | A)$, and, by Gibbs' lemma,

$$\sum_{A \in \mathcal{A}} \text{sum}(\phi \cdot \psi | A) \log(\phi'(A) \cdot \text{sum}(\psi | A)) \leq \sum_{A \in \mathcal{A}} \text{sum}(\phi \cdot \psi | A) \log(\text{sum}(\phi \cdot \psi | A)) \ ,$$

which means that equation (6) is maximized for $\phi'(A) = \text{sum}(\phi \cdot \psi | A)/\text{sum}(\psi | A)$, and therefore the distance is minimized for this value. □

If we have a structure $\mathcal{S}'$, $\mathcal{S}''$ is the structure obtained by pruning $\mathcal{S}'$ and ϕ' and ϕ'' are the potentials associated to trees $\mathcal{T}_{\mathcal{S}'}$ and $\mathcal{T}_{\mathcal{S}''}$ respectively, then pruning is carried out trying to minimize $D(\phi', \phi'' | \psi)$.

It involves the computation of the Kullback-Leibler distance from ϕ to ϕ' given a third potential ψ. The value $\phi'(\mathbf{x})$ is equal to $\phi(\mathbf{x})$ in all the points of U_I, except for a subset $A \subseteq U_I$ in which $\phi'(\mathbf{x}) = \text{sum}(\phi \cdot \psi | A)/\text{sum}(\psi | A)$. In this case, the set A corresponds to all the values $\mathbf{x}_I \in U_I$ such that following the path associated to it we arrive to the node after pruning. Making some easy calculations, this distance, $D(\phi, \phi' | \psi)$, can be calculated according to the following formula:

$$\frac{1}{\text{sum}(\phi \cdot \psi | A)} \left(\left(\sum_{\mathbf{x} \in A} \phi(\mathbf{x})\psi(\mathbf{x}) \log(\phi(\mathbf{x}))\right) + \text{sum}(\phi \cdot \psi | A) + \log\left(\frac{\text{sum}(\psi | A)}{\text{sum}(\phi \cdot \psi | A)}\right)\right) \ . \tag{7}$$

This new formula is much easier than the one used in the original Penniless algorithm:

$$\frac{(\sum_{\mathbf{x} \in A}(\phi(\mathbf{x})\psi(\mathbf{x}) \log(\phi(\mathbf{x}))) - \text{sum}(\phi \cdot \psi | A) \log(\text{sum}(\phi | A)/|A|))}{\text{sum}(\phi \cdot \psi)} +$$

$$\log\left(\frac{\text{sum}(\phi\cdot\psi)-\text{sum}(\phi\cdot\psi|A)+(\text{sum}(\phi|A)\text{sum}(\psi|A))/|A|}{\text{sum}(\phi\cdot\psi)}\right) \quad . \tag{8}$$

Let ϕ a potential represented by a tree $\mathcal{T}$ and assume that we want to make an approximation conditioned to the values of a potential ψ. Consider a node in a tree such that all its children are leaves. Let X_k be the variable stored on it and $(\mathbf{X}_J = \mathbf{x}_J)$ the configuration of values defining the path from the root to this node. We have considered different ways of actually carrying out the pruning of a probability tree:

1. Consider a threshold $\Delta > 0$ and then approximate the children of X_k by its average if the value of formula (8), with $A = A^I_{\mathbf{x}_J}$, is less than Δ. This is the original penniless algorithm, denoted by `penni`. We have also considered the same scheme but replacing by the weighted sum in (5) with $A = A^I_{\mathbf{x}_J}$, instead of the average. This algorithm will be denoted by `new-penni`.

2. Consider a value $0 < \epsilon < 1$ and then prune node X_k if $\text{sum}(\phi\cdot\psi|A^I_{\mathbf{x}_J}) \leq \epsilon\cdot\text{sum}(\phi\cdot\psi)$, i.e. we prune every node such that beneath it, the proportion of the entire probability mass of the product of potentials is lower than ϵ. We have considered two possibilities here: replacing the deleted values by the weighted sum in (5), denoted as `new-penni-av`, or replacing the deleted values by zero, denoted as `new-penni-ze`. The aim of this way of approximating is to avoid investing much effort on dealing with not significant values. Replacing by zero in some sense is inspired in simulation algorithms, in which configurations with low probability are usually assigned probability zero, since they have tendency not to appear in any sample.

Though above criteria are expressed in terms of potentials, the computations can be carried out directly in the tree representations of them, being the number of computations a function of the structures of the trees and not of the sizes of the frames in which the potentials are defined.

The approximation steps are done in a recursive way, starting in the nodes whose children are leaves and going back to the root node. In this way, if all the children of an internal node are leaves or have been previously pruned to a number, then this node is considered again for approximation.

4 Experimental work

We have carried out two different experiments. The first was devoted to test the appropriateness of the new way of computing the divergence between the exact and approximate potentials displayed in equation (7), that is, to test `penni` vs. `new-penni`. The second experiment is designed to compare the effects of replacing small probability values by zero or by the weighted sum in equation (5), i.e. `new-penni-av` vs. `new-penni-ze`.

In the first experiment we have chosen a large pedigree network (441 variables) with many observed variables (166). The reason to make this choice is because in networks without many observations, replacing by the average or by the weighted sum does not

Table I. Results (time in seconds and K-L divergence) of `penni` and `new-penni` for a pedigree network with many observed nodes.

Δ	4 stages		5 stages		6 stages	
	`penni`	`new-penni`	`penni`	`new-penni`	`penni`	`new-penni`
0.01	1.16 - 0.004434	1.11 - 0.004366	1.18 - 0.003754	1.14 - 0.003609	1.24 - 0.003739	1.17 - 0.003575
0.005	1.20 - 0.001004	1.14 - 0.003376	1.21 - 0.001004	1.20 - 0.002932	1.24 - 0.001004	1.18 - 0.001162
0.001	1.22 - 1.505E-4	1.18 - 4.375E-5	1.22 - 1.505E-4	1.17 - 4.816E-5	1.24 - 1.505E-4	1.17 - 4.375E-5
0.0005	1.23 - 4.042E-5	1.19 - 1.978E-5	1.25 - 4.042E-5	1.19 - 1.883E-5	1.28 - 4.042E-5	1.20 - 1.978E-5

Table II. Results (time in seconds and K-L divergence) of the second experiment for the pedigree.

ϵ	4 stages		5 stages		6 stages	
	`new-penni-av`	`new-penni-ze`	`new-penni-av`	`new-penni-ze`	`new-penni-av`	`new-penni-ze`
0.01	1.22 - 1.392E-4	1.16 - 1.742E-4	1.18 - 1.383E-4	1.15 - 1.744E-4	1.18 - 1.392E-4	1.16 - 1.742E-4
0.006	1.18 - 6.391E-5	1.17 - 2.862E-4	1.17 - 6.310E-5	1.16 - 2.715E-4	1.18 - 6.391E-5	1.18 - 2.862E-4
0.003	1.20 - 2.959E-5	1.24 - 3.015E-5	1.18 - 2.550E-5	1.19 - 2.749E-5	1.19 - 2.959E-5	1.22 - 3.015E-5
0.001	1.20 - 1.978E-5	1.20 - 1.914E-5	1.19 - 2.017E-5	1.18 - 1.813E-5	1.20 - 1.978E-5	1.20 - 1.914E-5

make a big difference, since in the way Penniless propagation operates, many upward messages can be constantly equal to one, in which case both substitution schemes are the same. We have performed trials with 4, 5 and 6 propagation stages, and the accuracy of the approximations has been controlled by parameter Δ. The results of this experiment are displayed in table I. These results suggest that `new-penni` usually provides equal or better results in lower time.

For the second experiment we have chosen the same network as in the first one and also another network with less variables (189 with 8 observations) but higher complexity (bigger potential sizes). This new network is called Munin 1. Both networks have been borrowed from the Decision Support Systems Group at Aalborg University (Denmark).

In this second experiment we have fixed a value for parameter $\Delta = 0.001$ and then the algorithms have been tested varying the value of parameter ϵ. The results of this experiment are displayed in tables II and III. Attending to these results, the use of `new-penni-ze` does not seem to provide any improvement, but in the very complex network Munin 1, the computing time is considerably reduced and the quality of the approximations improved. We have performed many other experiments that are not reported here, suggesting the same conclusion.

The algorithms have been implemented in Java 2 version 1.3 and integrated in the Elvira system. Trials have been run on an AMD K7 (800 MHz) computer with 512MB of RAM and operating system Linux RedHat with kernel 2.2.16.22.

Table III. Results (time in seconds and K-L divergence) of the second experiment for Munin 1.

ϵ	4 stages		5 stages		6 stages	
	`new-penni-av`	`new-penni-ze`	`new-penni-av`	`new-penni-ze`	`new-penni-av`	`new-penni-ze`
0.01	202.8 - 0.16978	102.1 - 0.1942	204.2 - 0.1782	107.0 - 0.1933	284.4 - 0.1450	139.1 - 0.1766
0.006	310.4 - 0.1232	167.8 - 0.0857	274.6 - 0.1526	178.5 - 0.0889	403.5 - 0.1390	236.8 - 0.0954
0.003	440.6 - 0.0849	236.8 - 0.0606	444.4 - 0.0851	233.8 - 0.0647	650.5 - 0.0851	312.4 - 0.0888
0.001	2436.8 - 0.0375	639.7 - 0.0136	2418.5 - 0.0376	646.4 - 0.0135	3777.0 - 0.0375	920.3 - 0.0664

5 Conclusions

We have introduced in this paper new ways of doing approximations in Penniless propagation. Under the assumptions of theorem 1 we have shown that replacing by the weighted sum is optimal. Besides, replacing small probability values by zero allows the Penniless scheme to acquire a very good feature of Monte Carlo algorithms: speed and low space requirements, and in some complex cases we can also improve the quality of the final approximations.

References

[BFGK96] J. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In E. Horvitz and F.V. Jensen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 115–123. Morgan & Kauffman, 1996.

[CM97] A. Cano and S. Moral. Propagación exacta y aproximada con árboles de probabilidad. In *Actas de la VII Conferencia de la Asociación Española para la Inteligencia Artificial*, pages 635–644, 1997.

[CMS00] A. Cano, S. Moral, and A. Salmerón. Penniless propagation in join trees. *Int. Journal of Intelligent Systems*, 15:1027–1059, 2000.

[JLO90] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, 4:269–282, 1990.

[KL51] S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:76–86, 1951.

[Koz98] A.V. Kozlov. *Efficient inference in Bayesian networks*. PhD thesis, Stanford University, 1998.

[MJ99] A.L. Madsen and F.V. Jensen. Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113:203–245, 1999.

[SCM00] A. Salmerón, A. Cano, and S. Moral. Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413, 2000.

[She97] P.P. Shenoy. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *Int. Journal of Approximate Reasoning*, 17:239–263, 1997.

[SS90] P.P. Shenoy and G. Shafer. Axioms for probability and belief function propagation. In R.D. Shachter, T.S. Levitt, J.F. Lemmer, and L.N. Kanal, editors, *Uncertainty in Artificial Intelligence 4*, pages 169–198. North Holland, Amsterdam, 1990.