An efficient factorization for the noisy MAX[*]

Francisco J. Díez and Severino F. Galán
Dpto. Inteligencia Artificial. UNED
Juan del Rosal, 16. 28430 Madrid
{fjdiez,seve}@dia.uned.es
http://www.ia.uned.es/{~fjdiez,~seve}

March 12, 2007

Abstract

Díez's algorithm for the noisy MAX is very efficient for polytrees, but when the network has loops it has to be combined with local conditioning, a suboptimal propagation algorithm. Other algorithms, based on several factorizations of the conditional probability of the noisy MAX, are not as efficient for polytrees, but can be combined with general propagation algorithms, such as clustering or variable elimination, which are more efficient for networks with loops. In this paper we propose a new factorization of the noisy MAX that amounts to Díez's algorithm in the case of polytrees and at the same time is more efficient than previous factorizations when combined with either variable elimination or clustering.

1 INTRODUCTION

A Bayesian network is a probabilistic model which consists of an acyclic directed graph (ADG), in which each node represents a variable, plus a conditional probability table (CPT), $P(v_i|pa(v_i))$, for each node, V_i, given its parents in the graph, $pa(V_i)$. We follow the convention of representing variables by uppercase letters and their values by lowercase letters; thus $Pa(V_i)$ represents a set of nodes, while $pa(v_i)$ represents a particular configuration of them. A family consists of a node and its parents: $Fam(V_i) = \{V_i\} \cup Pa(V_i)$. The joint probability for the set of all variables is the product of the CPT's:

$$P(v_1, \ldots, v_n) = \prod_i P(v_i|pa(v_i)) \; . \tag{1}$$

Any other marginal or conditional probability can be obtained from it. In particular, it is possible to obtain the a posteriori probability of any variable given a certain evidence: $P(v_i|\mathbf{e})$. However, the straightforward method that computes the conditional probabilities by previously expanding the joint probability has a computational complexity that grows exponentially with the number of nodes. For this reason, there has been a lot of research on algorithms that can compute $P(v_i|\mathbf{e})$ more efficiently by taking profit of the conditional independencies of the joint probability. The two standard exact algorithms for Bayesian networks are variable elimination and clustering (see Sec. 1.1).

The probabilities that make up a CPT are usually obtained from a database or from human experts' judgement. However, since the number of parameters for a family grows

[*]Extended version of the paper published as F. J. Díez and S. Galán. Efficient computation for the noisy MAX. *International Journal of Intelligent Systems*, **18** (2003) 165-177. Section 4 did not appear in the journal version.

exponentially with the number of parents, it is usually unreliable if not infeasible to directly obtain the conditional probability of a family having more than three or four parents. For this reason, it is beneficial to apply *canonical models* [6], arising from different causal assumptions, which only require a few parameters per link.

The use of canonical models not only simplifies the construction of Bayesian networks and influence diagrams, but can also lead to more efficient computations. Unfortunately, most of the software packages that deal with the OR/MAX expand it into an explicit CPT, $P(y|\mathbf{x})$, before doing inference, which is very inefficient. In the case of a polytree, the computational complexity of the computation for a MAX family having N parents is $O(n_Y \cdot {n_X}^N)$, where n_X is the domain size of the parents, assuming all of them have the same number of values. In the case of a clustering algorithm, the computational cost not only depends on the size of $P(y|\mathbf{x})$ but also on the size of the other potentials that are assigned to same cluster as $P(y|\mathbf{x})$.

Several authors have proposed different techniques for avoiding this exponential complexity. The most significant proposals are Díez's [4] algorithm, whose complexity is $O(n_Y \cdot n_X \cdot N)$ for a noisy MAX at a polytree, and the multiplicative factorization by Takikawa and D'Ambrosio [28], whose complexity is $O(\max(2^{n_Y}, n_Y \cdot n_X \cdot N))$ for a polytree, but has the advantage that it permits efficient elimination orderings for the variable-elimination algorithm and efficient triangulations for clustering algorithms [18]. The purpose of our research is to develop a new algorithm that combines the advantages of both methods: the efficiency of Díez's algorithm for the noisy MAX itself and the flexibility of multiplicative factorizations.

The organization of this paper is as follows. In Section 1.1 we briefly review the two standard exact algorithms for Bayesian networks, namely variable elimination and clustering. We also review the noisy MAX in Section 1.2. We expose our algorithm in Section 2 by introducing a new factorization of the MAX CPT and showing how it can be integrated with variable elimination (Sec. 2.1) and with clustering (Sec. 2.2). Section 3 compares this method with previous algorithms, Section 4 proposes a refinement of our algorithm, which makes it essentially identical to Díez's [4], and Section 5 summarizes the conclusions.

1.1 STANDARD ALGORITHMS FOR BAYESIAN NETWORKS

The joint probability of a Bayesian network can be computed as the product of all the conditional probabilities of the network. Any other marginal or conditional probability can be obtained from it. Unfortunately, the complexity of this brute-force method grows exponentially with the size of the network. One of the ways of avoiding this problem is to sum out some variables before multiplying all the potentials (each probability table is a potential and the result of multiplying two potentials is a new potential [15]). For instance, given the network in Figure 1, the a posteriori probability of A given the evidence $\{H = h_k\}$, $P(a|h_k)$, can be computed as follows:

$$P(a, h_k) = \sum_b \sum_d \sum_f \sum_g P(a, b, d, f, g, h_k)$$

$$= \sum_b \sum_d \sum_f \sum_g P(a) \cdot P(b|a) \cdot P(d|a) \cdot P(f) \cdot P(g|b, d, f) \cdot P(h_k|g) \tag{2}$$

$$= P(a) \cdot \sum_b P(b|a) \cdot \sum_d P(d|a) \cdot \sum_f P(f) \cdot \sum_g P(g|b, d, f) \cdot P(h_k|g) \tag{3}$$

$$P(a|h_k) = \frac{P(a, h_k)}{P(h_k)} = \frac{P(a, h_k)}{\sum_a P(a, h_k)} \, . \tag{4}$$

Please note that Equation 3 implies a more efficient computation than Eq. 2. This process of "ordering" the potentials and the sums is the fundamental of variable elimination algorithms (see for instance [1, 3]). Naturally, in networks with loops, the main difficulty of this algorithm consists is finding the optimal elimination order, which is of crucial importance for the efficiency of the computation of probability [17].

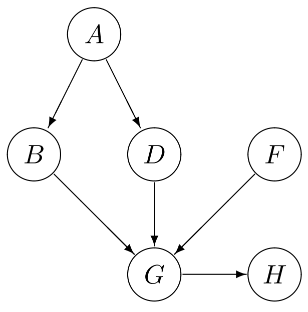

Figure 1: An example Bayesian network. We assume that there is a noisy MAX at $Fam(G)$.

The fundamental of clustering algorithms is essentially the same. One of the basic steps of this method is the construction of a join tree having the following properties:

Definition 1 *A* join tree *associated with a Bayesian network is an undirected tree such that*

- *each node in the join tree represents a cluster, i.e., a subset of variables of the Bayesian network;*
- *for each family in the Bayes net, there is at least one cluster containing all the variables of that family;*
- *if a variable appears in two clusters C_i and C_j then it must also appear in all the clusters that are in the path between C_i and C_j in the tree; this is called the* join tree property*.*

The standard way of building the join tree consists in moralizing the graph of the Bayesian network (by "marrying" the parents of each node) and triangulating it; each clique in the triangulated moral graph becomes a cluster in the join tree, which is called in this case a *clique tree* or a *junction tree*. After assigning each CPT to one cluster and introducing the evidence, the computation of the posterior probability takes place by an exchange of messages between neighboring clusters [14, 15, 26]. In essence, each join tree performs the same operations as variable elimination with a specific elimination ordering. The join tree is a structure that helps in storing and combining the potentials and caching the intermediate results. For this reason, variable elimination is more efficient for answering single queries, while clustering is more efficient for computing the posterior probability of each variable [31].

1.2 THE NOISY MAX

The most widely used canonical model is the noisy OR, introduced by Good [10] and further studied by Pearl [22]. Henrion [12] generalized the model to non-binary variables. Díez formalized Henrion's model, coined the term "MAX gate" and developed an evidence propagation algorithm whose complexity is linear in the number of parents [4]. In this paper, we concentrate on the noisy MAX model, since the noisy OR is a particular case of the former and the study of the noisy AND/MIN is very similar.[1]

[1]There is another generalization of the noisy OR, proposed by Srinivas [27, 6], which is very different from the noisy MAX and, to the best or our knowledge, can not be factored like the noisy MAX.

The noisy MAX consists of a child node, Y, taking on n_Y possible values that can be labeled from 0 to $n_Y - 1$, and N parents, $Pa(Y) = \{X_1, \ldots, X_N\}$, which usually represent the causes of Y. Each X_i has a certain zero value, so that $X_i = 0$ represents the absence of X_i. Two basic **axioms** define the noisy MAX [4]:

1. When all the causes are absent, the effect is absent:
$$P(Y = 0|X_i = 0_{\,[\forall i]}) = 1 \ . \tag{5}$$

2. The degree reached by Y is the **maximum** of the degrees produced by the X's if they were acting independently:
$$P(Y \leq y|\mathbf{x}) = \prod_i P(Y \leq y|X_i = x_i, X_j = 0_{\,[\forall j,\, j \neq i]}) \ . \tag{6}$$
where $\mathbf{x}$ represents a certain configuration of the parents of Y, $\mathbf{x} = (x_1, \ldots, x_N)$.

The **parameters** for link $X_i \to Y$ are the probabilities that the effect assumes a certain value y when X_i takes on the value x_i and all the other causes of Y are absent:
$$c_y^{x_i} = P(Y = y|X_i = x_i, X_j = 0_{\,[\forall j,\, j \neq i]}) \ . \tag{7}$$
If X_i has n_{X_i} values, the number of parameters required for the link $X_i \to Y$ is $(n_{X_i} - 1) \times (n_Y - 1)$—because of Equation 5. Since all the variables involved in a noisy OR are binary, this model only requires one parameter per link. Alternatively, it is possible to define new parameters:[2]
$$C_y^{x_i} = P(Y \leq y|X_i = x_i, X_j = 0_{\,[\forall j,\, j \neq i]}) = \sum_{y'=0}^{y} c_{y'}^{x_i} \ , \tag{8}$$
so that Equation 6 can be rewritten as
$$P(Y \leq y|x_1, \ldots, x_n) = \prod_i C_y^{x_i} \ . \tag{9}$$
The CPT is obtained by taking into account that
$$P(y|\mathbf{x}) = \begin{cases} P(Y \leq 0|\mathbf{x}) & \text{if } y = 0 \\ P(Y \leq y|\mathbf{x}) - P(Y \leq y-1|\mathbf{x}) & \text{if } y > 0 \end{cases} \ . \tag{10}$$

2 A NEW ALGORITHM FOR THE NOISY MAX

The departure point for our algorithm is the realization that Equation 10 can be represented as a product of matrices. For instance, when $n_Y = 3$,
$$\begin{pmatrix} P(Y=0|\mathbf{x}) \\ P(Y=1|\mathbf{x}) \\ P(Y=2|\mathbf{x}) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} P(Y \leq 0|\mathbf{x}) \\ P(Y \leq 1|\mathbf{x}) \\ P(Y \leq 2|\mathbf{x}) \end{pmatrix} \ .$$
In general,
$$P(y|\mathbf{x}) = \sum_{y'} \Delta_Y(y, y') \cdot P(Y \leq y'|\mathbf{x}) \ , \tag{11}$$

[2]Since the c parameters are more easily elicited from a database or from a human expert than the C's, a Bayesian network tool should display the former in its user interface, but internally it may use the latter for the sake of efficiency in the propagation of evidence. In any case, it is easy to convert from the ones to the others.

where Δ_Y is an $n_Y \times n_Y$ matrix given by

$$\Delta_Y(y, y') = \begin{cases} 1 & \text{if } y' = y \\ -1 & \text{if } y' = y - 1 \\ 0 & \text{otherwise} \end{cases} . \tag{12}$$

Because of Equation 9,

$$P(y|\mathbf{x}) = \sum_{y'} \Delta_Y(y, y') \cdot \prod_i C_{y'}^{x_i} . \tag{13}$$

This factorization of $P(y|\mathbf{x})$ is the keystone of our algorithm. We will now see how to integrate it with the standard algorithms.

2.1 INTEGRATION WITH VARIABLE ELIMINATION

In the factorization of the joint probability, each CPT $P(y|\mathbf{x})$ corresponding to a MAX family can be replaced with its equivalent given by Equation 13, and variable elimination can proceed in the usual way, as if Y' were an ordinary variable.[3] It might happen that the procedure that determines the elimination order decides to sum out Y' before than Y and its parents, which would be equivalent to expanding the whole CPT for this family. In this case, the algorithm we propose would only mean a saving of storage space at the cost of an increase in computational time. However, in general the elimination of variable Y' can be deferred until other variables have been eliminated, thus leading to a saving of both time and space, as shown in the following example.

Let us assume that, given the network in Figure 1, we are interested in the posterior probability of A given the evidence $\{H = h_k\}$. The computation can proceed as follows:

$$\begin{aligned} P(a, h_k) =& \sum_b \sum_d \sum_f \sum_g \sum_{g'} P(a) \cdot P(b|a) \cdot P(d|a) \\ & \cdot P(f) \cdot \Delta_G(g, g') \cdot C_{g'}^b \cdot C_{g'}^d \cdot C_{g'}^f \cdot P(h_k|g) \\ =& P(a) \cdot \sum_{g'} \left[\left(\sum_b P(b|a) \cdot C_{g'}^b \right) \cdot \left(\sum_d P(d|a) \cdot C_{g'}^d \right) \right. \\ & \left. \cdot \left(\sum_f P(f) \cdot C_{g'}^f \right) \cdot \left(\sum_g \Delta_G(g, g') \cdot P(h_k|g) \right) \right] . \end{aligned} \tag{14}$$

It is easy to verify that this equation is much more efficient than Equation 3 as a consequence of the factorization given by Equation 13. In Section 3 we discuss other network structures for which this new algorithm leads to significant savings of both time and space.

2.2 INTEGRATION WITH CLUSTERING ALGORITHMS

We have seen that the integration of our method with variable elimination is immediate. We will now explain how to integrate it with clustering algorithms. The process is as follows:

Algorithm 2 *Construction of the join tree:*

1. *Build an auxiliary (undirected) graph as follows:*

   (a) *Create a graph with the same set of nodes as the Bayes net, but without any link.*

[3] Y' is not a true random variable because its values are not exclusive. Please note that $P_{Y'}(y_{\max}) = P(Y \leq y_{\max}|\mathbf{x}) = 1$ and therefore $\sum_{y'} P_{Y'}(y') = \sum_y P(Y \leq y|\mathbf{x}) \geq 1$.

(b) For each node V:

i. If V is the child of a noisy MAX family, add a new node V' with $dom(V') = dom(V)$ to the auxiliary graph, add a link V'–V, and then a link U_i–V' for each link $U_i \rightarrow V$ in the Bayes net (i.e., for each parent U_i of V).

ii. Otherwise, add a link U_i–V for each link $U_i \rightarrow V$ in the network, and marry these parents with one another.

2. Triangulate the auxiliary graph.

3. Arrange the cliques of the triangulated graph in a join tree.

4. Assign the potentials to the cliques.

For example, the auxiliary graph for the network in Figure 1 is given in Figure 2.

There are two possible ways to triangulate this graph: the addition of an edge B–D and the addition of an edge A–G'. The latter is more efficient if $n_B \times n_D > n_A \times n_G$ —since $n_{G'} = n_G$— and leads to the junction tree displayed in Figure 3.

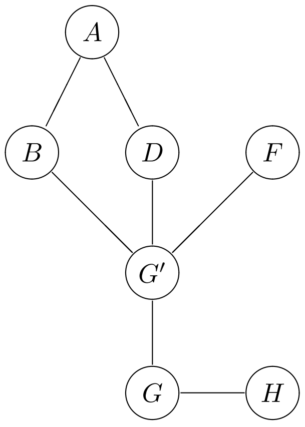

Figure 2: Auxiliary graph for the Bayesian network in Figure 1.

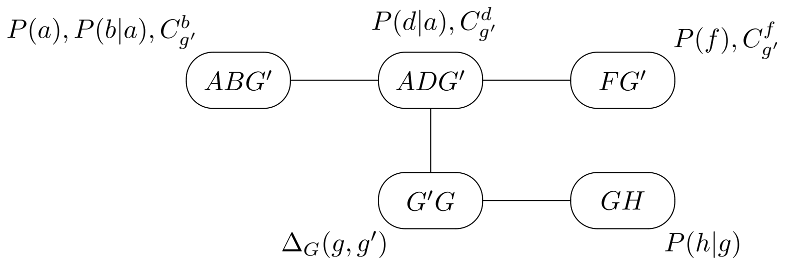

Figure 3: A junction tree for the graph in Figure 2. It shows the potentials assigned to each clique.

We would like to make the following comments about this algorithm:

- Step 1 in the algorithm guarantees that it is always possible to assign the potentials of the network to the join tree. For a explicit CPT $P(y|\mathbf{x})$, there is at least one clique containing all the nodes of that family (step 1(b)ii of the algorithm). If the interaction for a node Y is given by a noisy MAX, there is at least one clique containing both Y and Y' (step 1(b)i), and this clique can be assigned the potential $\Delta_Y(y, y')$. For the same reason, it is also possible to assign each $C_y^{x_i}$ to a clique containing both X_i and Y'. In the case of a leaky MAX [4, 6], the potential C_y^* can be assigned to any clique containing Y.

- The auxiliary graph is not necessarily a moral graph, because the parents involved in a noisy MAX do not need to be married. This "immorality" may be surprising for those who are familiar with clustering algorithms, but we argued in Section 1.1 that the main role of the join tree is to serve as a structure for storing and combining the potentials. Therefore, if the CPT for a family can be decomposed into a set of potentials, it is not necessary to have a cluster including all the nodes of that family—it suffices that, *for each potential,* there is a cluster containing all its variables. This property replaces the second condition in Definition 1 (join tree associated with a Bayesian network).

- Steps 2 and 3 of the algorithm are the same as in other clustering algorithms. There are several possibilities for triangulating the graph: heuristic search, simulated annealing, genetic algorithms, etc., and several types of join trees: junction trees, binary join trees, etc.

- The propagation of evidence consists of the exchange of messages between the nodes of the join tree. Shenoy-Shafer propagation [26] and lazy propagation [19, 20] are compatible with the factorization we propose (cf. Eq. 13)). However, the algorithms that divide the messages exchanged between nodes, such as Lauritzen-Spiegelhalter propagation [15], HUGIN propagation [14] or cautious propagation [13], are incompatible with our factorization, because these algorithms rely on the fact that, in the case of non-negative potentials, a certain value of a message is 0 only when all the corresponding values in the potential of the cluster that sends the message are 0; however this property does not hold for potentials including both positive and negative values.

  This is not a drawback of our algorithm since, contrary to the common belief, Shenoy-Shafer propagation with binary join trees is more efficient than the HUGIN algorithm in general [16, 25], and so is lazy propagation [19, 20]. Also, variable elimination is generally more efficient than HUGIN propagation in many practical situations [31].

3 RELATED WORK

As mentioned above, the complexity of computing the probability for a family having a child Y that takes on n_Y values, and N parents $\{X_1, \ldots, X_n\}$ that take on n_X values each, is $O(n_Y \cdot {n_X}^N)$. The first algorithm that took advantage of canonical models to simplify the computation of probability in Bayesian networks was Pearl's method for the noisy OR [22, sec. 4.3.2], whose complexity was $O(N)$. Díez [4] proposed a similar algorithm for the noisy MAX, whose complexity was $O(n_Y \cdot n_X \cdot N)$; however, for networks with loops this algorithm needs to be integrated with local conditioning [5], a method that in general is not as efficient as clustering methods based on near-optimal triangulations.

Parent divorcing [21] and sequential decomposition [11] are two algorithms that transform the original network by introducing $N-2$ auxiliary nodes, so that Y and each of the auxiliary nodes has exactly two parents;[4] the complexity of the computation in the transformed network is $O(n_Y \cdot {n_X}^3 \cdot N)$ for a noisy MAX in a polytree. The main shortcoming of these methods

[4]It is not difficult to prove that the number of auxiliary nodes is always $N-2$ for this kind of transformations: if there were no auxiliary nodes, Y could have only two parents. The addition of each auxiliary node allows Y

is the difficulty of choosing an efficient structure of auxiliary nodes when the noisy MAX family makes part of a network with loops—see the analysis in [18]. Other representations of the noisy MAX conditional probability, such as the additive factorization [2] and the heterogeneous factorization [30, 32, 33], usually increase the efficiency of variable elimination algorithms, but they are still inefficient for some large networks.

In 1999 Takikawa and D'Ambrosio [28] proposed a multiplicative factorization for the noisy MAX. Its integration with SPI [24] —which is in essence a variable elimination algorithm— leads to a complexity of $O(\max(2^{n_Y}, n_Y \cdot n_X \cdot N))$ for the polytree. Given that in practical situations n_Y is usually small and n_X may be large, this algorithm is in general more efficient than previous methods. This multiplicative factorization can also be integrated with clustering algorithms by introducing $n_Y - 1$ nodes representing binary variables [18]—see below.

The algorithm we propose in this paper is also based on a multiplicative factorization, given by Equation 13, which summarizes Equations 9 and 10.[5] The main difference is that for each MAX family $Fam(Y)$, the factorization used by Takikawa, D'Ambrosio and Madsen introduces $n_Y - 1$ auxiliary variables $\{Y'_0, \ldots, Y'_{n_Y-2}\}$, all of which are parents of Y and children of $Pa(Y)$, and for this reason, the complexity of their algorithm grows exponentially with n_Y. In contrast, our algorithm only introduces one auxiliary node Y', which leads to a complexity $O({n_Y}^2)$. This difference becomes especially significant when n_Y is large, for instance when a noisy MIN/MAX is used to represent a temporal noisy OR/AND; in this model, the temporal variables have as many values as time intervals considered [8, 9].

A second difference is that our algorithm does not require to marry the parents of each family. For instance, given an isolated noisy MAX having a child Y and N parents $\{X_1, \ldots, X_N\}$, the method by Madsen and D'Ambrosio would produce two cliques: $\{X_1 \ldots X_N Y'_0 \ldots Y'_{n_Y-2}\}$ and $\{Y'_0 \ldots Y'_{n_Y-2} Y\}$. The state size of the first clique grows exponentially with N, but the application of lazy propagation keeps the complexity proportional to N by conserving a list of potentials —the factorized conditional probability— rather than expanding the potential of the clique. The complexity $O(2^{n_Y})$ is due to the conditional probability table $P(y|y'_0, \ldots, y'_{n_Y-2})$ associated with the second clique. In contrast, our algorithm would produce a chain of N cliques of the form $\{X_i, Y'\}$ plus one clique $\{Y', Y\}$; this clique is responsible for the $O({n_Y}^2)$ complexity.

Analogously, given the graph in Figure 4 with a noisy MAX at Y, the algorithm by Madsen and D'Ambrosio would produce three cliques: $\{UX_1 \ldots X_N\}$, $\{X_1 \ldots X_N Y'_0 \ldots Y'_{n_Y-2}\}$ and $\{Y'_0 \ldots Y'_{n_Y-2} Y\}$. In contrast, our algorithm can triangulate the auxiliary graph by adding a link U–Y', which leads to a chain of N cliques of the form $\{UX_iY'\}$ plus one clique $\{Y'Y\}$.

As a third example, given the graph in Figure 5 with two noisy MAX's at Y and Z, the algorithm by Madsen and D'Ambrosio would produce four cliques: $\{X_1 \ldots X_N Y'_0 \ldots Y'_{n_Y-2}\}$, $\{X_1 \ldots X_N Z'_0 \ldots Z'_{n_Z-2}\}$, $\{Y'_0 \ldots Y'_{n_Y-2} Y\}$, and $\{Z'_0 \ldots Z'_{n_Z-2} Z\}$. Our algorithm can triangulate the associated graph by adding a link Y'–Z', thus producing a chain of N cliques of the form $\{X_iY'Z'\}$ plus two cliques $\{Y'Y\}$ and $\{Z'Z\}$. This triangulation of the network is also possible even if the models at Y and Z are different, provided that their conditional probabilities admit multiplicative factorizations; for instance, there may be a noisy OR/MAX at Y and a noisy AND/MIN at Z—see [29] for the multiplicative factorization of other CPT's. However if the CPT for Z, $P(z|\mathbf{x})$ is not factored, there must be a clique containing the whole family

to have one more parent in the noisy MAX to be represented; therefore, $N - 2$ auxiliary nodes are necessary to represent a noisy MAX having N parents.

[5]These two equations, which are a particular case of Equations (39) and (38) in [4], were also used by Pradham et al. [23] to compute the CPT's in their medical expert system; as a consequence, the use of noisy MAX gates simplified the acquisition of knowledge for their network, but had no benefit for the propagation of evidence. In contrast, the application of the more general equations led to computational savings in Díez's algorithm [4] when applied to the DIAVAL expert system [7].

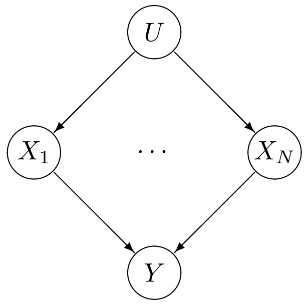

Figure 4: An example Bayesian network in which the noisy MAX at Y makes part of several loops.

$Fam(Z)$, and in this case the factorization of $P(y|\mathbf{x})$ is useless, or even counterproductive.

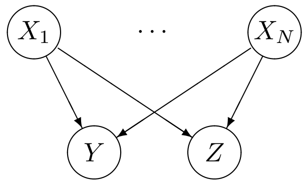

Figure 5: Another Bayesian network. We assume there is a noisy MAX at Y. In the text we discuss the cases in which the family of Z is modelled by a noisy MAX, a noisy AND/MIN and a general CPT.

3.1 Experimental comparisons

We have performed some experiments in order to measure the efficiency of our algorithm on random networks. In the first experiment, we selected 25 binary variables and drew links by randomly selecting 25 pairs of nodes. For a pair (X_i, X_j), the link was drawn from X_i to X_j if $i < j$, and vice versa; this way we made sure that the network was acyclic. We assigned a noisy MAX distribution to each family (in fact, it was a noisy OR, since all the variables were binary) and propagated evidence in a Shafer-Shenoy architecture, first with expanded CPT's and then with factorized conditional probabilities. Of course, the join tree was generally different in both cases. We repeated the experiment for a total of 50 random networks and then increased the number of links to 30, 35, ... , 80, randomly generating 50 networks in each case. Figure 6 represents the average time spent as a function of the number of links.

We performed a similar experiment with 10 five-valued variables. The number of links ranged from 10 to 24. All the families interacted through noisy MAX's. Again we generated 50 random networks for each number of links. The results are displayed in Figure 7. The comparison of these two figures confirms that the computational savings increase with the domain size of the variables.

Additionally, according with a comparison carried out by Dr. Takikawa (private communication), the most difficult marginal query for CPCS, a medical Bayesian network, required 639,637 multiplications and a total of 0.29 seconds with the factorization by Takikawa and D'Ambrosio, while it only required 62,796 multiplications and 0.05 with our factorization.

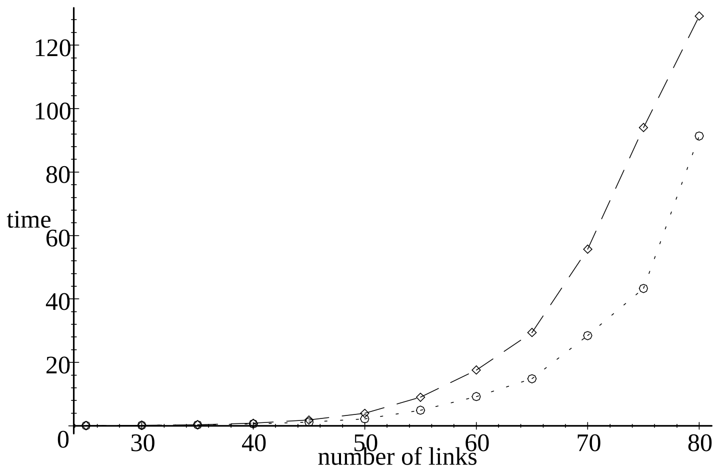

Figure 6: Average time (in seconds) spent for a set of 50 random networks, each consisting of 25 binary variables, as a function of the number of links. The propagation is performed on a Shafer-Shenoy architecture with expanded CPTs ($\diamond$) and with factorized noisy-MAX distributions ($\circ$).

There was also a significant reduction in memory space: the size of the largest intermediate table, which is analogous to the size of the largest clique, was reduced from 49,152 to 2,304 floating point numbers. It must be noted that previous methods, such as the additive factorization [2] and the heterogeneous factorization [30, 32, 33, 34] ran out of memory when trying to solve this and other queries for the CPCS network.

In the future, we will implement the lazy propagation algorithm and compare three methods: (1) our factorization combined with Shafer-Shenoy propagation on binary join trees, (2) our factorization combined with lazy propagation, and (3) the factorization of Takikawa and D'Ambrosio combined with lazy propagation, as done in [18]. We expect that the second method will be the most efficient on average. It would also be possible to compare variable elimination with clustering algorithms, but in this case the result will depend on the task to be performed: as mentioned in Section 1.1, variable elimination is more efficient when answering single queries, while clustering algorithms are more efficient for computing the posterior probability of each variable [31].

4 A REFINEMENT OF THE ALGORITHM

In the previous section we showed that in the case of a polytree having a noisy MAX at node Y, the time complexity for the cluster $\{Y'Y\}$ is ${n_Y}^2$, and the complexity for each cluster $\{X_iY\}$ is $n_X \cdot n_Y$—assuming that all the X_i's have the same number of values, n_X. Therefore the time complexity of the above version of our algorithm is $O(\max({n_Y}^2, n_Y \cdot n_X \cdot N))$, which is lower than that of previous factorizations, but still higher than that of Díez's [4] algorithm, $O(n_Y \cdot n_X \cdot N)$. For this reason, we tried to further reduce the complexity of our algorithm from quadratic to linear in n_Y in order to obtain a new version that is optimal for polytrees and, at the same time, can be combined with variable elimination (with efficient elimination orderings) and clustering (with efficient triangulations).

We can accomplish this goal by taking profit of the particular form of $\Delta_Y(y, y')$, given by Equation 12. The integration of our triangulation with variable elimination (cf. Sec. 2.1)

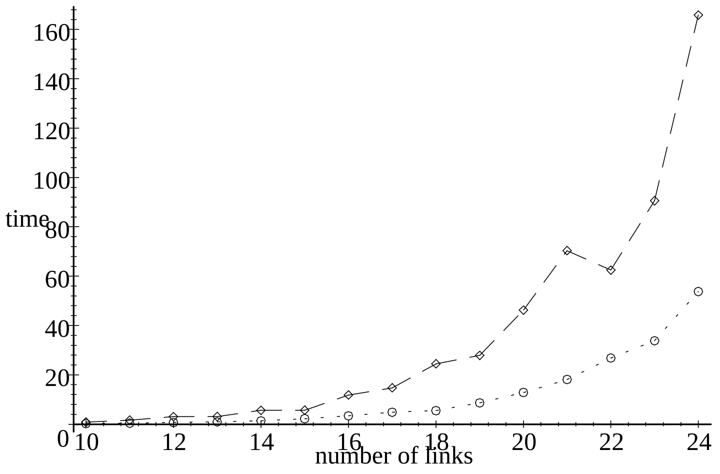

Figure 7: Average time (in seconds) spent for a set of 50 random networks, each consisting of 10 five-valued variables, as a function of the number of links. The propagation is performed on a Shafer-Shenoy architecture with expanded CPTs ($\diamond$) and with factorized noisy-MAX distributions ($\circ$).

requires to sum out Y and Y' for each noisy MAX family $Fam(Y)$. If Y is to be eliminated before Y', the computation can be performed more efficiently by multiplying together all the potentials that depend on y (except $\Delta_Y(y, y')$) into a potential $\Psi(y, \mathbf{v})$ —where $\mathbf{V}$ represents a set of other variables— and then eliminating Y by the following equation:

$$\sum_y \Delta_Y(y, y') \cdot \Psi(y, \mathbf{v}) = \begin{cases} \Psi(y', \mathbf{v}) & \text{for } y' = y_{\max} \\ \Psi(y', \mathbf{v}) - \Psi(y'+1, \mathbf{v}) & \text{for } y' < y_{\max} \end{cases} . \tag{15}$$

Analogously, if we decide to eliminate Y' before than Y, all the potentials that depend on y' (except $\Delta_Y(y, y')$) must be multiplied into a potential $\Psi'(y', \mathbf{v}')$ and the equation to be used is

$$\sum_{y'} \Delta_Y(y, y') \cdot \Psi'(y', \mathbf{v}') = \begin{cases} \Psi'(y, \mathbf{v}') & \text{for } y = 0 \\ \Psi'(y, \mathbf{v}') - \Psi'(y-1, \mathbf{v}') & \text{for } y > 0 \end{cases} . \tag{16}$$

The time-complexity induced by the left-hand sides of these equations is $O({n_Y}^2)$, while that induced by the right-hand sides is $O(n_Y)$. Therefore, the complexity of the algorithm may be improved if, instead of including matrix Δ_Y in the list of potentials, the algorithm uses special rules for the elimination of Y and Y'.

As an example, let us apply this method to Equation 14, in which G is eliminated before G'. In this case, there is only one potential that depends on g, namely $P(h_k|g)$, and it does not depend on any other variable. Therefore $\Psi(g, \mathbf{v}) = \Psi(g) = P(h_k|g)$, and because of Equation 15,

$$\sum_g \Delta_G(g, g') \cdot P(h_k|g) = \begin{cases} P(h_k|g') & \text{for } g' = n_G - 1 \\ P(h_k|g') - P(h_k|g'+1) & \text{for } g' < n_G - 1 \end{cases} . \tag{17}$$

This refinement of the algorithm can be also applied to clustering algorithms, since the propagation of messages between clusters essentially consists in multiplying potentials and eliminating variables. For instance, given the junction tree in Figure 3, the message sent from clique $\{G'G\}$ to clique $\{ADG'\}$ is

$$M_{\{G'G\}\rightarrow\{ADG'\}}(g') = \sum_{g} \Delta_G(g, g') \cdot M_{\{GH\}\rightarrow\{G'G\}}(g) \ , \tag{18}$$

but instead of having a potential Δ_G, we can eliminate g in a special way, by applying Equation 15:

$$M_{\{G'G\}\rightarrow\{ADG'\}}(g') = \begin{cases} M_{\{GH\}\rightarrow\{G'G\}}(g') & \text{for } g = g_{\max} \\ M_{\{GH\}\rightarrow\{G'G\}}(g) - M_{\{GH\}\rightarrow\{G'G\}}(g+1) & \text{for } g < g_{\max} \end{cases} \tag{19}$$

In the same way, the message from $\{G'G\}$ to $\{GH\}$,

$$M_{\{G'G\}\rightarrow\{GH\}}(g) = \sum_{g'} \Delta_G(g, g') \cdot M_{\{ADG'\}\rightarrow\{G'G\}}(g') \ , \tag{20}$$

which sums out G', can be computed by applying Equation 16:

$$M_{\{G'G\}\rightarrow\{GH\}}(g) = \begin{cases} M_{\{ADG'\}\rightarrow\{G'G\}}(g) & \text{for } g = 0 \\ M_{\{ADG'\}\rightarrow\{G'G\}}(g) - M_{\{ADG'\}\rightarrow\{G'G\}}(g) & \text{for } g > 0 \end{cases} \tag{21}$$

5 CONCLUSION

In Section 2 we have proposed a new algorithm for the noisy MAX based on a factorization of its CPT, given by Equation 13. This factorization can be easily integrated with variable elimination and clustering algorithms. In this case, instead of building a moral graph, the construction of the join tree is based on an auxiliary graph in which the parents of a noisy MAX are not necessarily married. The time complexity for a polytree having a noisy MAX at node Y is $O(\max({n_Y}^2, n_Y \cdot n_X \cdot N))$, where n_X is the domain size of each parent of Y, and N is the number of parents. The time complexity of the most efficient factorization obtained previously —by Takikawa and D'Ambrosio [28]— was $O(\max(2^{n_Y}, n_Y \cdot n_X \cdot N))$. Additionally, we believe that our algorithm is easier to understand, because it is not based on local expressions [2].

It is also possible to refine our algorithm in order to achieve the same time-complexity as Díez's [4] algorithm, namely $O(n_Y \cdot n_X \cdot N)$, as explained in Section 4. This refined version performs optimally for polytrees and, at the same time, can be integrated with variable elimination (with efficient elimination orderings) and with clustering methods (with efficient triangulations). The negative side of the refined version is that its implementation is more complex, because it requires a special treatment of some variables. Does this effort pay off? It depends on the number of noisy MAX gates to be dealt with, and for each MAX family $Fam(Y)$, the domain size of Y and the size of the cluster that contains the potential $\Delta_Y(y, y')$.

The method proposed in this paper can be applied not only to the OR, AND, MAX and MIN models, but to any noisy model built as shown in [6], since Vomlel [29] has proved recently that every deterministic CPT, i.e., a CPT consisting of only 0's and 1's, admits a multiplicative factorization. However, it must be pointed out that the factorization of the noisy MAX/MIN is virtually always advantageous for the computation of probability, while in other cases it is more efficient to work with a deteministic CPT than with its factorization.

Acknowledgments

This research was supported by the Spanish CICYT, under grant TIC97-1135-C04. The basic ideas emerged during a stay of the first author at the University of Pittsburgh, as a consequence of discussions with Marek Druzdzel, Yan Lin and Tsai-Ching Lu. This paper benefitted from the comments by Serafín Moral, Marek Druzdzel, Masami Takikawa, and the reviewers of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000), the Conference of the Spanish Association for Artificial Intelligence (CAEPIA-2001), and of this special issue of the International Journal of Intelligent Systems. We especially thank Masami Takikawa for sharing with us the results of his experiments with our method.

References

[1] F. G. Cozman. Generalizing variable elimination in Bayesian networks. In *Proceedings of the IBERAMIA/SBIA 2000 Workshops (Workshop on Probabilistic Reasoning in Artificial Intelligence)*, pages 27–32, São Paulo, Brazil, 2000.

[2] B. D'Ambrosio. Local expression languages for probabilistic dependence. *International Journal of Approximate Reasoning*, 13:61–81, 1995.

[3] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI'96)*, pages 211–219, Portland, OR, 1996. Morgan Kaufmann, San Francisco, CA.

[4] F. J. Díez. Parameter adjustment in Bayes networks. The generalized noisy OR–gate. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence (UAI'93)*, pages 99–105, Washington D.C., 1993. Morgan Kaufmann, San Mateo, CA.

[5] F. J. Díez. Local conditioning in Bayesian networks. *Artificial Intelligence*, 87:1–20, 1996.

[6] F. J. Díez and M. J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical Report 2005-01, CISIAD, UNED, Madrid, 2005. In preparation.

[7] F. J. Díez, J. Mira, E. Iturralde, and S. Zubillaga. DIAVAL, a Bayesian expert system for echocardiography. *Artificial Intelligence in Medicine*, 10:59–73, 1997.

[8] S. F. Galán, F. Aguado, F. J. Díez, and J. Mira. NasoNet. modelling the spread of nasopharyngeal cancer with temporal Bayesian networks. *Artificial Intelligence in Medicine*, 25:247–254, 2002.

[9] S. F. Galán and F. J. Díez. Networks of probabilistic events in discrete time. *International Journal of Approximate Reasoning*, 30:181–202, 2002.

[10] I. Good. A causal calculus (I). *British Journal of Philosophy of Science*, 11:305–318, 1961.

[11] D. Heckerman. Causal independence for knowledge acquisition and inference. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence (UAI'93)*, pages 122–127, Washington D.C., 1993. Morgan Kaufmann, San Mateo, CA.

[12] M. Henrion. Some practical issues in constructing belief networks. In L. N. Kanal, T. S. Levitt, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 161–173. Elsevier Science Publishers, Amsterdam, The Netherlands, 1989.

[13] F. V. Jensen. Cautious propagation in Bayesian networks. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, pages 323–328, Montreal, Canada, 1995. Morgan Kaufmann, San Francisco, CA.

[14] F. V. Jensen, K. G. Olesen, and S. K. Andersen. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20:637–660, 1990.

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

[16] V. Lepar and P. P. Shenoy. A comparison of Lauritzen-Spiegelhalter, HUGIN, and Shenoy-Shafer architectures for computing marginals of probability distributions. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, pages 328–337, Madison, WI, 1998. Morgan Kaufmann, San Francisco, CA.

[17] Z. Li and B. D'Ambrosio. Efficient inference in Bayes nets as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11:55–81, 1994.

[18] A. L. Madsen and B. D'Ambrosio. A factorized representation of independence of causal influence and lazy propagation. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 8:151–165, 2000.

[19] A. L. Madsen and F. V. Jensen. Lazy propagation in junction trees. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, pages 362–369, Madison, WI, 1998. Morgan Kaufmann, San Francisco, CA.

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

[21] K. G. Olesen, U. Kjærulff, F. Jensen, F. V. Jensen, B. Falck, S. Andreassen, and S. K. Andersen. A MUNIN network por the median nerve. A case study on loops. *Applied Artificial Intelligence*, 3:385–403, 1989.

[22] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988. Revised second printing, 1991.

[23] M. Pradham, G. Provan, B. Middleton, and M. Henrion. Knowledge engineering for large belief networks. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 484–490, Seattle, WA, 1994. Morgan Kaufmann, San Francisco, CA.

[24] R. D. Shachter, B. D'Ambrosio, and B. A. Del Favero. Symbolic probabilistic inference in belief networks. In *Proceedings of the 8th National Conference on AI (AAAI–90)*, pages 126–131, Boston, MA, 1990.

[25] P. P. Shenoy. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning*, 17:1–25, 1997.

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

[27] S. Srinivas. A generalization of the noisy-OR model. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence (UAI'93)*, pages 208–215, Washington D.C., 1993. Morgan Kaufmann, San Mateo, CA.

[28] M. Takikawa and B. D'Ambriosio. Multiplicative factorization of the noisy-MAX. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 622–630, Stockholm, Sweden, 1999. Morgan Kaufmann, San Francisco, CA.

[29] J. Vomlel. Exploiting functional dependence in Bayesian network inference with a computerized adaptive test as an example. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pages 528–535, Edmonton, Canada, 2002. Morgan Kaufmann, San Francisco, CA.

[30] N. L. Zhang. Inference with causal independence in the CPCS network. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, pages 582–589, Montreal, Canada, 1995. Morgan Kaufmann, San Francisco, CA.

[31] N. L. Zhang. Computational properties of two exact algorithms for Bayesian networks. *Applied Intelligence*, 9:173–183, 1998.

[32] N. L. Zhang and D. Poole. Intercausal independence and heterogeneous factorization. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 606–14, Seattle, WA, 1994. Morgan Kaufmann, San Francisco, CA.

[33] N. L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

[34] N. L. Zhang and L. Yan. Independence of causal influence and clique tree propagation. *International Journal of Approximate Reasoning*, 19:335–349, 1997.