

Computing Probabilities of Events in Bayesian Networks

R. Haenni

Institute of Informatics
University of Fribourg
Switzerland
rolf.haenni@unifr.ch

J. Kohlas

Institute of Informatics
University of Fribourg
Switzerland
juerg.kohlas@unifr.ch

N. Lehmann

Institute of Informatics
University of Fribourg
Switzerland
norbert.lehmann@unifr.ch

Abstract

This paper proposes a new approach for computing probabilities of events in Bayesian networks. The idea is to replace the outward phase of the propagation algorithm by a second (partial) inward propagation phase. The benefit of this idea is that the attention can be focussed on optimizing the inward phase.¹

1 Introduction

Several architectures have been developed for computing marginal distributions of variables in Bayesian networks. The following three architectures are the most popular ones:

- the HUGIN architecture [2],
- the Lauritzen-Spiegelhalter architecture (LS-architecture for short) [3],
- the Shenoy-Shafer architecture (SS-architecture for short) [7].

All these techniques are based on a message-passing scheme related to certain graphical structures (join or junction trees), where the two main operations of **multiplication** and **marginalization** (summation) of potentials are always performed locally on relatively small families of variables. A common feature of the three architectures is that they use a two-phase computation, consisting of a so-called **inward propagation**, in which information is collected, and an **outward propagation** phase, in which information is distributed. At the end of the outward phase, the marginal distributions of certain families of variables

are available. This permits (by summation) to compute the probabilities of events related to such a family of variables.

The main difference between these architectures is the way in which the outward phase is performed. The HUGIN and the LS-architecture need a further operation of **division** to prepare or to perform the outward phase. These are relatively costly operations, which are avoided in the SS-architecture. It has been shown that the SS-architecture is at least as efficient as the HUGIN architecture considered so far to be the most efficient one (see [5]).

This paper proposes an alternative approach for computing probabilities of events. The new feature is that outward propagation is replaced by a second **partial inward propagation**. In this new approach, the question of selecting an appropriate architecture is therefore only of minor importance. Furthermore, eliminating the outward phase allows an optimization of the inward phase, not possible otherwise.

The underlying idea of the method presented in this paper has first been developed for the framework of **probabilistic argumentation systems** [1]. The same technique has then been applied to the closely related domain of belief function networks [4]. Clearly, since Bayesian networks can be considered as special cases of belief function networks, it is not surprising, that the method can also be adapted to Bayesian networks. Possibly, it is an approach which can be formulated in the general theory of valuation networks [7], although this has still to be verified.

The basic theorem on which the method is based is formulated in Section 2. In Section 3, the theoretical result is used to design a two-phase inward procedure. Finally, the optimization of the inward propagation is discussed in Section 4.

¹Research supported by grant No. 21-53500.98 of the Swiss National Foundation for Research.

2 Probability of Events and Normalization Constants

Consider a set of **variables** $V = \{X_1, \dots, X_n\}$ where every variable X_i has a corresponding finite **frame** Θ_i . Furthermore, if I is a subset of indices in $N = \{1, \dots, n\}$, then \mathbf{X}_I denotes a vector of variables X_i with $i \in I$. The cartesian product $\Theta_I = \prod_{i \in I} \Theta_i$ represents the set of all such vectors relative to I . A **potential** on Θ_I (or a potential on I for short) is a mapping $\psi : \Theta_I \rightarrow \mathbb{R}^+$. It corresponds to an $|I|$ -dimensional table which is written as $\psi(\mathbf{X}_I)$. I is called the **domain** of $\psi(\mathbf{X}_I)$. In the following, a potential on I is either a probability distribution over Θ_I , a conditional probability distribution over Θ_J given Θ_K (where $J \cup K = I$ and $J \cap K = \emptyset$), or an evidence $\mathbf{X}_I \in E_I$, where E_I is a subset of Θ_I . In this latter case, the potential is a table with $\psi(\mathbf{X}_I) = 1$ if $\mathbf{X}_I \in E_I$ and $\psi(\mathbf{X}_I) = 0$ otherwise.

Let $p(\mathbf{X}_N)$ be a probability distribution over all n variables and suppose that there exists a corresponding factorization

$$p(\mathbf{X}_N) = \prod_{I \in \mathcal{H}} \psi(\mathbf{X}_I), \quad (2.1)$$

where \mathcal{H} is a family of domains $I \subseteq N$ (a hypergraph). Furthermore, suppose that a number of evidences, represented by potentials $\psi_E(\mathbf{X}_I)$ with $I \in \mathcal{H}^* \subseteq \mathcal{H}$, are added. These potentials describe the observations that $\mathbf{X}_I \in E_I$. The conditional distribution of the n variables given these evidences (abbreviated by E) is then given by

$$p(\mathbf{X}_N|E) = \frac{1}{c} \cdot \prod_{I \in \mathcal{H}} \psi(\mathbf{X}_I) \cdot \prod_{I \in \mathcal{H}^*} \psi_E(\mathbf{X}_I), \quad (2.2)$$

where

$$c = \sum_{\mathbf{X}_N \in \Theta_N} \left[\prod_{I \in \mathcal{H}} \psi(\mathbf{X}_I) \cdot \prod_{I \in \mathcal{H}^*} \psi_E(\mathbf{X}_I) \right] \quad (2.3)$$

is the **normalization constant** of the factorization. Finally, suppose that $H \subseteq \Theta_N$ represents an **event** (also called a hypothesis), then

$$p(\mathbf{X}_N \in H|E) = \sum_{\mathbf{X}_N \in H} p(\mathbf{X}_N|E) \quad (2.4)$$

is the conditional probability of the event H given the evidence E . Note that H is often given as a set $H_I \subseteq \Theta_I$. In such a case, H can simply be considered as the cylindrical extension of H_I to Θ_N .

The expression in (2.4) represents the classical way of computing conditional probabilities of events. However, the following theorem shows, that there is an interesting alternative.

Theorem 1 Suppose that $\psi_H(\mathbf{X}_N)$ represents the potential obtained by considering the event H as an additional evidence. If

$$c' = \sum_{\mathbf{X}_N \in \Theta_N} [\psi_H(\mathbf{X}_N) \cdot \prod_{I \in \mathcal{H}} \psi(\mathbf{X}_I) \cdot \prod_{I \in \mathcal{H}^*} \psi_E(\mathbf{X}_I)] \quad (2.5)$$

denotes the new factorization constant, then

$$p(\mathbf{X}_N \in H|E) = \frac{c'}{c}. \quad (2.6)$$

Proof. The proof of this theorem is based on the fact that $\psi_H(\mathbf{X}_N)$ can be seen as a filter for the sum in (2.5). It selects all the values of \mathbf{X}_N which are in H . The second normalization factor can therefore be written as

$$c' = \sum_{\mathbf{X}_N \in H} \left[\prod_{I \in \mathcal{H}} \psi(\mathbf{X}_I) \cdot \prod_{I \in \mathcal{H}^*} \psi_E(\mathbf{X}_I) \right].$$

Finally, the equivalence between (2.4) and (2.6) can be demonstrated as follows:

$$\begin{aligned} \frac{c'}{c} &= \frac{1}{c} \cdot \sum_{\mathbf{X}_N \in H} \left[\prod_{I \in \mathcal{H}} \psi(\mathbf{X}_I) \cdot \prod_{I \in \mathcal{H}^*} \psi_E(\mathbf{X}_I) \right] \\ &= \sum_{\mathbf{X}_N \in H} \left[\frac{1}{c} \cdot \prod_{I \in \mathcal{H}} \psi(\mathbf{X}_I) \cdot \prod_{I \in \mathcal{H}^*} \psi_E(\mathbf{X}_I) \right] \\ &= \sum_{\mathbf{X}_N \in H} p(\mathbf{X}_N|E) = p(\mathbf{X}_N \in H|E). \end{aligned}$$

□

This theorem shows that the probability of an event is determined by the normalization constants of two slightly different factorizations. This will be exploited in the next section for an efficient computation of this probability.

3 Computing Normalization Constants

As shown in the previous section, the problem of computing probabilities of events can be reduced to the problem of computing two related normalization constants c and c' . The first normalization constant c remains the same for all possible events. In contrast, the second normalization constant c' depends on H and must therefore be re-computed for every event of interest.

This section shows how normalization constants can be computed efficiently. The general idea is the same as in the architectures mentioned at the beginning. The point is that a factorization on a hypergraph (as required in (2.2), for example) can be transformed into an equivalent factorization on a join tree [7]. The computation can then be arranged as a message-passing

scheme between the nodes of the join tree. A message between two nodes I and J is a potential on $I \cap J$ that contains the information required by the receiving node. In this way, the information is propagated through the entire join tree. The benefit of this is that the potentials are always multiplied locally on relatively small domains.

The propagation algorithm of all three architectures includes two phases. In a first phase (called the inward phase), messages are sent from the leaves of the join tree towards an arbitrarily chosen root. If R denotes the domain of the root, then, after the inward phase, it is possible to derive probabilities of events $H_R \subseteq \Theta_R$. During the second phase (called the outward phase), messages are sent from the root to the leaves of the join tree. At the end, it is possible to compute probabilities of events relative to all the domains appearing in the join tree. The figures 3.1 and 3.2 illustrate the inward and outward phases of the propagation algorithm in a join tree.

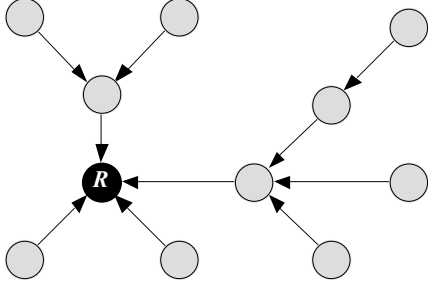


Figure 3.1: Inward phase.

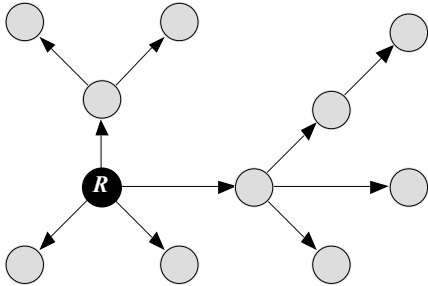


Figure 3.2: Outward phase.

At the beginning of the propagation process, a corresponding potential $\psi(\mathbf{X}_I)$ is stored for every node I of the network. As soon as all incoming messages $M_1(\mathbf{X}_{I \cap I_1})$ to $M_k(\mathbf{X}_{I \cap I_k})$ are received at the node I , a new potential

$$\tilde{\psi}(\mathbf{X}_I) = \psi(\mathbf{X}_I) \cdot \prod_{i=1, \dots, k} M_i(\mathbf{X}_{I \cap I_i}) \quad (3.7)$$

can be computed and stored. This situation of a node with $k + 1$ neighbors is depicted in Figure 3.3.

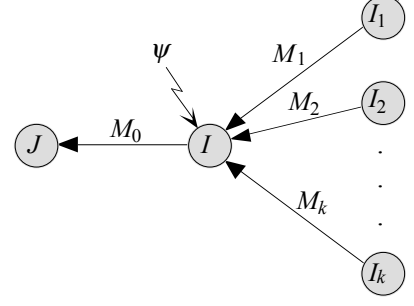


Figure 3.3: Receiving and sending messages.

If $L = I - J$ denotes the set of variables contained in I but not in its neighbor J , then the message $M_0(\mathbf{X}_{I \cap J})$ to be sent from I to J is determined by

$$M_0(\mathbf{X}_{I \cap J}) = \sum_{\mathbf{X}_L \in \Theta_L} \tilde{\psi}(\mathbf{X}_I). \quad (3.8)$$

Note that at the beginning, only the leaves of the join tree are able to send their messages. In a second step, all the neighbors of the leaves can send their messages, and so on. Finally, the potential $\tilde{\psi}(\mathbf{X}_R)$ is obtained at the root. It determines the first normalization constant by

$$c = \sum_{\mathbf{X}_R \in \Theta_R} \tilde{\psi}(\mathbf{X}_R). \quad (3.9)$$

Thus, the first normalization constant c is already known at the end of the inward phase. This value remains the same for all further computations.

Let H_I be an event relative to a node I appearing in the join tree, and suppose that $\psi_H(\mathbf{X}_I)$ is the corresponding potential on I representing the event H_I . This new potential can then be introduced at the corresponding node I . The product $\psi_H(\mathbf{X}_I) \cdot \tilde{\psi}(\mathbf{X}_I)$ determines a new potential $\tilde{\psi}'(\mathbf{X}_I)$ for the node I . By re-computing the messages along the unique path from I towards the root R , a new potential $\tilde{\psi}'(\mathbf{X}_R)$ is obtained at the root, from which the second normalization constant can be derived:

$$c' = \sum_{\mathbf{X}_R \in \Theta_R} \tilde{\psi}'(\mathbf{X}_R). \quad (3.10)$$

Finally, the probability of the event H_I is obtained by (2.6). Therefore, it is possible to compute probabilities of events in join trees without the outward phase. However, re-computing the messages along the unique path towards the root is necessary for every event of interest. Two such partial inward phases are shown in Figure 3.4 and Figure 3.5 for two events H_{I_1} and H_{I_2} .

Sometimes, the event of interest concerns a domain I which does not correspond to a node of the join tree.

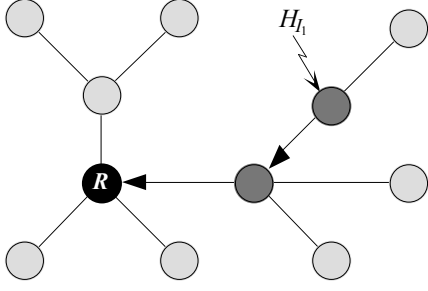


Figure 3.4: Partial inward phase for an event H_{I_1} .

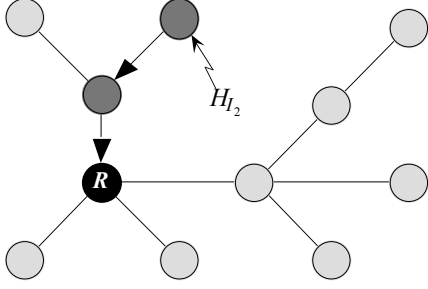


Figure 3.5: Partial inward phase for an event H_{I_2} .

However, if I is a subset of one or several nodes of the join tree, then it is possible to implant the corresponding potential $\psi_H(\mathbf{X}_I)$ on any of these nodes. Clearly, it is important to select the node with the smallest distance to the root (if possible, on the root itself of course). This guarantees that the number of messages to be re-computed is minimal. If the domain I is not a subset of at least one node the join tree, then it is necessary to re-compute the join tree such that I is included.

4 Optimizations

As the method proposed in this paper uses only inward but not outward propagation, it is important to organize the inward phase as efficient as possible. Several aspects of the procedure sketched in the previous section can be further optimized. First of all, consider the combination of the incoming messages during the first inward phase (see Figure 3.3). The result is a product of potentials as described by (3.7). Clearly, because multiplication of potentials is commutative and associative, there are many different ways to compute this product.

A strategy for finding a good sequence of multiplications is based on the fact that multiplying potentials is less expensive for small domains. Observe that the k messages received at node I are potentials on different domains $I \cap I_1$ to $I \cap I_k$. Therefore, it is advantageous to start with a pair of messages M_i and M_j , such that

the union of their domains $I \cap (I_i \cup I_j)$ is as small as possible. The same strategy is then applied for selecting a pair of messages in the second step, and so on. Note that each step of the procedure, resulting potentials from previous steps may be selected.

For example, consider a node I with four incoming messages M_1 to M_4 . The product of the four messages can then be computed in 15 different ways. Two of them are depicted in Figure 4.6 and Figure 4.7. If $L_1 = \{1, 2\}$, $L_2 = \{1\}$, $L_3 = \{2, 3, 4\}$, $L_i = I_i \cap I$ and $L_4 = \{3, 4\}$ are the domains of the incoming messages, then

$$[M_1(\mathbf{X}_{L_1}) \cdot M_2(\mathbf{X}_{L_2})] \cdot [M_3(\mathbf{X}_{L_3}) \cdot M_4(\mathbf{X}_{L_4})]$$

represents the optimal way of computing the corresponding product (it includes three multiplications on the domains $\{1, 2\}$, $\{2, 3, 4\}$, and $\{1, 2, 3, 4\}$, respectively). It can easily be verified that the optimal solution is obtained from the strategy described above. It corresponds to Figure 4.6.

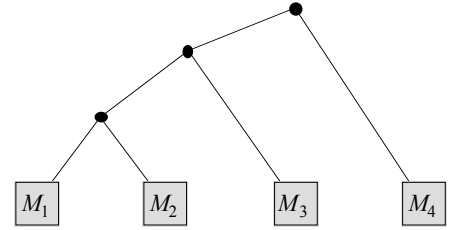


Figure 4.6: One possibility of computing the product of incoming messages.

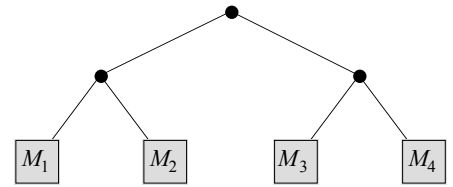


Figure 4.7: Another possibility of computing the product of incoming messages.

A second important problem is the optimization of the second (partial) inward phase. As described in Section 3, the partial inward phase consists in re-computing the messages along the unique path between a particular node and the root of the join tree.

A trivial solution for this is to re-compute entirely the product of the incoming messages at each node on the path. Clearly, this presupposes that all incoming messages have been stored during the first inward phase. More formally, let $M_1(\mathbf{X}_{I \cap I_1})$ to $M_k(\mathbf{X}_{I \cap I_k})$ be the messages received and stored at node I during the first inward phase. If $M_j'(\mathbf{X}_{I \cap I_j})$, $1 \leq j \leq k$, is the

new message received during the partial inward phase, replacing the old message $M_j(\mathbf{X}_{I \cap I_j})$, then the new potential obtained for I can be computed as follows:

$$\tilde{\psi}'(\mathbf{X}_I) = M_j'(\mathbf{X}_{I \cap I_j}) \cdot \psi(\mathbf{X}_I) \cdot \prod_{\substack{i=1, \dots, k \\ i \neq j}} M_i(\mathbf{X}_{I \cap I_i}) \quad (4.11)$$

The problem with this approach is that the number k of multiplications required at node I remains the same for the first and the second inward phase. Depending on the sequence of multiplications selected during the first inward phase, certain computations are repeated unnecessarily during the second inward phase.

A better solution is possible, if not only the incoming messages are stored during the first inward phase, but also the potential $\tilde{\psi}(\mathbf{X}_I)$. The new potential $\tilde{\psi}'(\mathbf{X}_I)$ can then be computed with constantly two operations: one division on $I \cap I_j$ and one multiplication on I :

$$\tilde{\psi}'(\mathbf{X}_I) = M_j'(\mathbf{X}_{I \cap I_j}) \cdot \frac{\tilde{\psi}(\mathbf{X}_I)}{M_j(\mathbf{X}_{I \cap I_j})} \quad (4.12)$$

$$= \frac{M_j'(\mathbf{X}_{I \cap I_j})}{M_j(\mathbf{X}_{I \cap I_j})} \cdot \tilde{\psi}(\mathbf{X}_I). \quad (4.13)$$

Note that the division-by-zero problem is automatically handled, since every 0-value in $M_j(\mathbf{X}_{I \cap I_j})$ is also a 0-value in $M_j'(\mathbf{X}_{I \cap I_j})$ and in $\tilde{\psi}(\mathbf{X}_I)$. This solution performs better than the trivial approach as soon as $k \geq 3$. In contrast, the trivial approach is preferred for $k = 2$, because multiplication is computationally less expensive than division, and also for $k = 1$, because only one multiplication is required.

Another approach uses the concept of binary join trees [6]. A binary join tree is a join tree such that no node has more than three neighbors. Therefore, if I is an arbitrary node in a binary join tree, then $k = 2$ is the maximal number of messages received at node I . A binary join tree can be constructed during the first inward phase by inserting initially empty nodes. For example, consider the case where I receives four incoming messages M_1 to M_4 , and suppose that their product is computed as depicted in Figure 4.7. A corresponding binary join tree can then be constructed according to Figure 4.8.

Let $L_{12} = L_1 \cup L_2$ be the domain of a newly inserted empty node. If $M_1(\mathbf{X}_{L_1})$ and $M_2(\mathbf{X}_{L_2})$ are the corresponding incoming messages, then $\psi(\mathbf{X}_{L_{12}}) = M_1(\mathbf{X}_{L_1}) \cdot M_2(\mathbf{X}_{L_2})$ is the potential to be computed during the first inward phase. Note that no initial potential $\psi(\mathbf{X}_{L_{12}})$ is needed. During the second inward phase, the new potential to be computed is either $\tilde{\psi}'(\mathbf{X}_{L_{12}}) = M_1'(\mathbf{X}_{L_1}) \cdot M_2(\mathbf{X}_{L_2})$ or $\tilde{\psi}'(\mathbf{X}_{L_{12}}) = M_1(\mathbf{X}_{L_1}) \cdot M_2'(\mathbf{X}_{L_2})$, depending on whether M_1 or M_2 has changed.

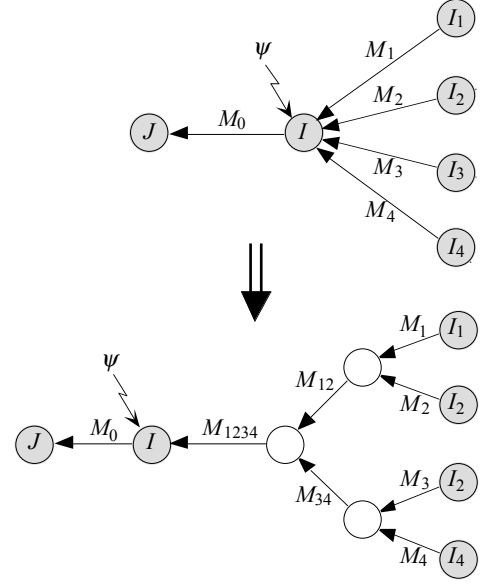


Figure 4.8: Constructing a binary join tree.

The strength of this approach is that during the first and the second inward phase only one multiplication is required at each node of the join tree. However, because new nodes are inserted during the first inward phase, the size of the join tree has increased. Therefore, the average length of the path to be re-computed during the second inward phase has increased as well.

5 Conclusion

The approach to compute the probability of certain events on a Bayesian network replaces the usual method, which consists of computing the marginal of all variables in an outward phase, by a second inward phase. In the first method, the outward propagation, depending on the architecture used, contains a lot of multiplications and even divisions of potentials. But once the outward propagation is terminated, the computation of the probability of an event reduces to summing up. In contrast, in the method presented here, the second inward phase, contains multiplications of potentials along the path to the root.

In both cases the first inward phase constitutes a necessary preparation. In the method proposed here, only this necessary phase is executed. The other computations needed to process queries are executed only on demand. In the classical method, by the outward propagation one precomputes a lot data, in expectation of query demands. It all depends on how many queries are finally to be processed to decide which one of the two methods are more efficient.

References

- [1] R. Haenni, J. Kohlas, and N. Lehmann. Probabilistic argumentation systems. Technical Report 99-9, Institute of Informatics, University of Fribourg, 1999.
- [2] F. Jensen, S. Lauritzen, and K. Olesen. Bayesian updating in causal probabilistic networks by local computations. *SIAM Journal on Computing*, 4:269–282, 1990.
- [3] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistical Society*, 50(2):157–224, 1988.
- [4] N. Lehmann and R. Haenni. An alternative to outward propagation for Dempster-Shafer belief functions. In A. Hunter and S. Parsons, editors, *ECSQARU'99, Qualitative and Quantitative Approaches to Reasoning with Uncertainty*. Springer, 1999.
- [5] V. Lepar and P. Shenoy. A comparison of Lauritzen-Spiegelhalter, Hugin and Shenoy-Shafer architectures for computing marginals of probability distributions. In G. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 328–337. Morgan Kaufmann, 1998.
- [6] P. Shenoy. Binary join trees. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 492–499. Morgan Kaufmann, 1996.
- [7] P. Shenoy and G. Shafer. Axioms for probability and belief functions propagation. In R. Shachter and al., editors, *Uncertainty in Artificial Intelligence 4*. North Holland, 1990.