A Computational Architecture for N-way Sensitivity Analysis of Bayesian Networks

Veerle M.H. Coupé [1], *Finn V. Jensen* [2], *Uffe Kjærulff* [2], *Linda C. van der Gaag* [3]

[1] Center for Clinical Decision Sciences,
Department of Public Health, Erasmus University Rotterdam,
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
e-mail: `coupe@mgz.fgg.eur.nl`

[2] Department of Computer Science, Aalborg University,
Frederik Bajers Vej 7, DK-9220 Aalborg, Denmark

[3] Department of Computer Science, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

The relation between a probability computed from a Bayesian network and the parameters in the network is a simple mathematical function. Any prior probability can be expressed as a multilinear function in the network's parameters; any posterior probability is a quotient of such functions. These functions serve to yield insight into the robustness of a Bayesian network. To this end, it suffices to establish the coefficients in the functions. In general, the investigation into the relation between a probability computed from the network and the network's parameters is called a *sensitivity analysis*. In the past, various methods for sensitivity analysis have been suggested, most of them being computationally very expensive. In this paper, a new method is presented, that is both elegant and efficient. The method builds upon a junction tree representation of a Bayesian network for its computational architecture. Basically, the method amounts to the propagation of vectors of coefficients through the junction tree. These vectors are accumulated to give the coefficients required for

the functional relation under study.

1 Introduction

A Bayesian network is a concise representation of a joint probability distribution on a set of statistical variables. It can be used for computing the prior or posterior probability of any variable in the network in view of the currently available evidence. To that end, various methods for propagating evidence in a Bayesian network have been developed [Pearl, 1988, Lauritzen & Spiegelhalter, 1988].

A Bayesian network contains a considerable number of parameters. These parameters are conditional probabilities that generally have to be estimated by experts. Such estimates inevitably are inaccurate. To study the possible effects of inaccuracies in a network's parameters on a probability of interest, a sensitivity analysis can be performed.

Basically, sensitivity analysis of a Bayesian network amounts to varying the estimates for one or more parameters in the network simultaneously and investigating the effects on a probability of interest. Varying one parameter estimate at a time is called a *one-way sensitivity analysis*. It serves to reveal the independent effect of the parameter under study on a probability of interest. The term *n-way sensitivity analysis* is used to indicate that n, $n \geq 1$, network parameters are varied simultaneously. It reveals how the n parameters interact in their effect on a probability of interest. In clinical decision analyses [Habbema *et al.*, 1990, Dippel *et al.*, 1992], it is customary to perform a two-way or three-way sensitivity analysis as well as a one-way sensitivity analysis, since it yields additional insight in a model's robustness. The results of such an analysis are relatively easy to interpret and can be represented graphically. Higher order sensitivity analyses are uncommon since their results are harder to interpret. However, for generality, we consider in this paper the situation in which an arbitrary number of n parameters is varied.

The most straightforward way of performing a sensitivity analysis of a Bayesian network is to vary the parameters under study stepwise and in a systematic way. The effect of these stepwise variations on the probability of interest is evaluated by using any standard propagation algorithm. It is clear that this approach is computationally expensive.

In [Laskey, 1995], K. Blackmond Laskey focuses on one-way sensitivity analysis of a Bayesian network. She presents an efficient method for analytically computing *sensitivity* *values*, that is, first-order approximations of one-way sensitivity analyses. A sensitivity value captures the approximate effect of small variations in the parameter's estimate on the probability of interest. Compared to straightforward variation, Laskey's method requires considerably less computational effort. The method, however, provides insight in the effect of small variations only: as Laskey indicates, when larger variations are considered, the quality of the approximation may break down rapidly. As the estimates for a Bayesian network's parameters often are highly inaccurate, we prefer exact sensitivity analysis.

In [Chang & Fung, 1995] and [**?**], a different approach is taken to sensitivity analysis of Bayesian networks. The idea of symbolic propagation is exploited. Instead of yielding a single number as the standard propagation algorithms do, a symbolic propagation algorithm yields an algebraic expression for a network's probability of interest in terms of all parameters in the network. From this expression, the sensitivity of the probability of interest to a parameter under study is readily computed, basically by filling in the specified estimates for all other parameters. This method can be used for one-way sensitivity analysis as well as for higher order sensitivity analyses. A disadvantage of building upon symbolic propagation is that it is time-consuming. Methods for sensitivity analysis that build upon the faster standard propagation algorithms are therefore preferable.

The use of symbolic propagation for sensitivity analysis, however, led E. Castillo et al. to recognize the mathematical form of the function relating a probability of interest to the network's parameters [Castillo *et al.*, 1995]; in [Coupé and Van der Gaag, 1998], this function was analytically derived for a one-way sensitivity analysis of a Bayesian network. The function expressing a probability of interest in terms of n parameters, called an *n-way sensitivity function*, is a quotient of two multilinear functions. Exploiting this knowledge, standard propagation algorithms [Pearl, 1988, Jensen *et al.*, 1990, Shafer & Shenoy, 1988, Madsen & Jensen, 1998]) can be used to obtain the coefficients in this function.

In [Castillo *et al.*, 1997], E. Castillo et al. propose an algorithm for computing the coefficients in an n-way sensitivity function. Different combinations of values are assumed for the parameters under study; one different combination of values for each coefficient is required. For each combination of values, the probability of interest is computed from network using a standard propagation method. As a result, a system of linear equations is obtained which is solved to give the required coefficients. The method thus exploits knowledge of the mathematical form of the function. As such, the method is far more efficient than straighforward sensitivity analysis. However, the number of network computations required increases exponentially with the number of parameters considered in the analysis.

For a one-way sensitivity analysis, L.C. van der Gaag and U. Kjærulff present a substantial computational improvement of the methods discussed above [Van der Gaag & Kjærulff, 1999]. They suggest using the computational architecture of a junction tree to obtain the coefficients in the functions expressing a probability of interest in terms of every single parameter in the network. Just one inward and two outward propagations in the junction tree are needed to obtain the one-way sensitivity analyses with respect to all parameters in the network. Basically, the idea is to compute the coefficients for each parameter under study within the clique in the junction tree in which the parameter resides. Currently, this is the most efficient method for performing a one-way sensitivity analysis of a Bayesian network. Unfortunately, it is not easily extended to higher order sensitivity analyses.

In this paper, a method for n-way sensitivity analysis is presented that computationally improves the method in [Castillo *et al.*, 1997]. As in [Van der Gaag & Kjærulff, 1999], we build upon the computational architecture of a junction tree derived from a Bayesian network. Basically, the algorithm presented is an adaptation of a standard propagation algorithm. The messages sent between the cliques in the junction tree are vectors of coefficients. These coefficients are processed locally per clique and are accumulated to yield the coefficients in the required n-way sensitivity function, describing the probability of interest in terms of the n parameters under study.

In its simplest form, our method performs equally well as the optimal version of the method given in [Castillo *et al.*, 1997]. The substantial advantage of our method, however, is that it is elegantly integrated in an existing propagation scheme, thereby providing a framework that can be easily extended and further optimized.

The paper is structured as follows. In Section 2, preliminaries of Bayesian networks, junction trees, and propagation in a junction tree are presented. The properties of sensitivity analysis of a Bayesian network are reviewed in Section 3. Section 4, then, describes our algorithm for computing the coefficients in an n-way sensitivity function. Section 5 deals with the determination of the optimal clique in the junction tree to start the computation of the required coefficients. Some optimizations of our method are briefly discussed in Section 6. The paper ends with a discussion and conclusions in Section 7.

2 Bayesian networks and junction trees

A *Bayesian network* basically is a concise representation of a joint probability distribution on a set of statistical variables [Pearl, 1988]. Information about the independences holding among the variables is explicitly separated from the numerical quantities involved in the distribution. To this end, the network comprises a qualitative part and an associated quantitative part.

The qualitative part of a Bayesian network BN is a graphical representation of the independences holding among the variables in the probability distribution that is being represented. It takes the form of an *acyclic directed graph* G with nodes $V(G)$. In this digraph, each node V_i, $i = 1, \ldots, r$, $r \geq 1$, represents a statistical variable that can take one of a finite set of values. In the sequel, the set of possible values of variable V_i, called the *universe* of V_i, will be indicated by Ω_{V_i}. Any subset of variables $W \subseteq V(G)$ has associated a universe Ω_W that is defined as the Cartesian product of the universes of all variables from W, that is, $\Omega_W = \times_{V_i \in W} \Omega_{V_i}$. An element from a universe will be termed a *configuration*. In the remainder of the paper, formula will generally be stated as schemata involving variables. From such schemata multiple instantiations can be obtained by filling in configurations for the variables involved.

The set of arcs in the digraph of a Bayesian network models the independences among the represented variables. Informally speaking, we take an arc $V_i \rightarrow V_j$ to represent a direct influential or causal relationship between the variables V_i and V_j; the arc's direction designates V_j as the effect or consequence of the cause V_i. Absence of an arc between two nodes means that the corresponding variables do not influence each other directly and, hence, are (conditionally) independent.

Associated with the qualitative part of a Bayesian network are numbers that describe the strengths of the influential relationships among the represented variables. With each node V_i of the network's digraph is associated a set of conditional probability distributions describing the joint influence of the various values for the node's (immediate) predecessors $\pi(V_i)$ on the probabilities of the values of this node itself, that is, with each node V_i are associated conditional probability distributions $p(V_i \mid \pi(V_i))$. A conditional probability $p(v_i \mid \pi')$, for a specific value $v_i \in \Omega_{V_i}$ and a configuration $\pi' \in \Omega_{\pi(V_i)}$, is termed a *parameter* of the network. The conditional probability distributions jointly constitute the quantitative part of the network.

We illustrate the concept of Bayesian network by means of an example that will be used as a running example throughout the paper.

Example 2.1 Consider the Bayesian network shown in Figure 1a. The network pertains to the statistical variables V_i, $i = 1, \ldots, 8$. From the network's digraph various independences are read. For example, variable V_5 is dependent on variable V_1; when evidence on the value of variable V_3 becomes available, however, V_1 and V_5 become independent. If the value of variable V_8 is not observed, the variables V_6 and V_7 are independent of each other. Observing V_8, however, induces a dependence between V_6 and V_7. Associated with the variables in the network are the conditional probability distributions $p(V_i \mid \pi(V_i))$. We assume that the variables in our example network are binary. That is, they either take the value *true*, denoted by v_i, or *false*, denoted by $\neg v_i$. In Figure 1a, for each variable the estimates for the required conditional probabilities are shown. From the conditional probabilities for the variable V_6 we see, for example, that the probability that V_6 takes the value *true* is high, unless both parents V_3 and V_4 are *false*. $\square$

The conditional probability distributions of a Bayesian network provide all information necessary for uniquely defining a joint probability distribution on the variables discerned that respects the independences portrayed by the network's qualitative part. Hence, from the network, any (prior or posterior) probability of interest can be computed. For this purpose various algorithms are available [Pearl, 1988, Lauritzen & Spiegelhalter, 1988, Shachter, 1986, Shafer & Shenoy, 1988, Jensen *et al.*, 1990].

The algorithm for evidence propagation developed by S.L. Lauritzen and D.J. Spiegelhalter forms the basis of the method for sensitivity analysis that is presented in this paper. This algorithm and and its underlying computational architecture, the *junction tree* [Jensen *et al.*, 1990], are therefore reviewed briefly. The algorithm transforms a Bayesian network into an equivalent undirected representation. To this end, the network's digraph is transformed into a *triangulated moral graph.* A triangulated graph is an undirected graph in which no cycle of length four or more exists without a shortcut. The transformation of a Bayesian network's graph G into a triangulated moral graph H involves three steps. First, arcs are added to G such that the (original) predecessors of each node in $V(G)$ are connected. As a result, all pairs of variables that may influence each other directly are

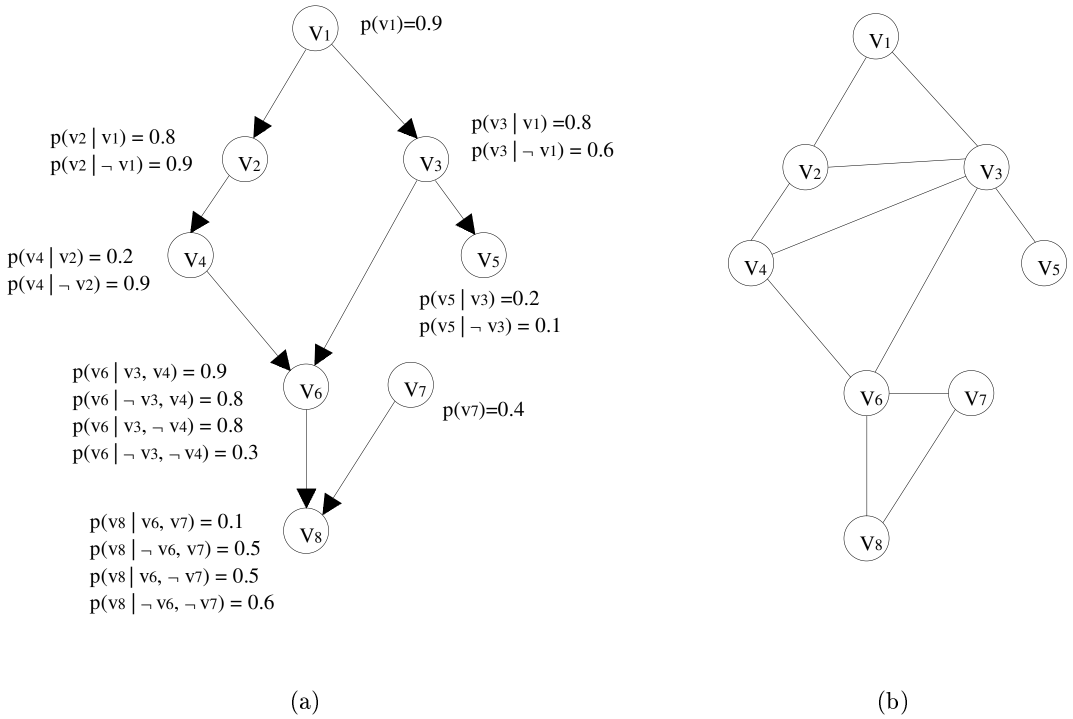

Figure 1: (a) An example Bayesian network and (b) the triangulated moral graph obtained from this network.

connected. In the second step, the direction of the arcs are dropped. Now, an undirected representation of the independences in the probability distribution is obtained. The third, and last, step consists of cutting short each cycle of length four or more by adding an edge. Note that, in general, an acyclic digraph allows several different triangulated moral graphs. In the following, an illustration of this transformation is given.

Example 2.2 Consider once more the digraph G depicted in Figure 1a. The undirected graph in Figure 1b is a triangulated moral graph for G. $\square$

A triangulated moral graph H has associated local *potential functions* on small sets of variables to arrive at a representation of the joint probability distribution on the problem domain. These potential functions will be detailed shortly. The triangulated moral graph together with the potential functions allows for an efficient propagation algorithm, in which the computations to be performed are local to these small sets of variables. For that purpose, the computational architecture of the *junction tree* is used [Jensen *et al.*, 1990].

A junction tree T for a triangulated moral graph H includes for its nodes the *maximal cliques* U_i , $i = 1, \ldots k$. A *clique* of a graph H is a subgraph I in H, such that any two variables in I are connected by an edge. Clique I in H is a maximal clique if there is no clique in H larger than I that properly contains I. In the sequel, for ease of exposition, we will write U_i to denote both the clique itself and the set of variables in U_i. The intersections between the cliques U_i, $i = 1, \ldots k$, in T give rise to the tree's edges. These edges satisfy the following property: for any two cliques $U_i, U_j \in T$ and each clique U_k on the (unique) path from U_i to U_j in T, we have that $U_i \cap U_j \subseteq U_k$. Associated with each edge (U_i, U_j) is the clique intersection of U_i and U_j. These clique intersections are called *separators* and will be denoted by S_l, $l < k$. Usually, a triangulated graph allows various different junction trees.

With each clique U_i in the junction tree, a potential function ϕ_{U_i} is associated. These potential functions capture probabilistic information about the variables involved and are obtained from the conditional probability distributions from the original Bayesian network BN. The potential function ϕ_{U_i} of U_i is a product of probability distributions. To compose this product, the probability distributions of each variable in the original network BN are assigned to a clique under the following conditions; a probability distribution can only be assigned to a clique that includes both the variable to which this distribition pertains as well as the predecessors of this variable and every probability distribution can be assigned to one clique only. Consider a clique U_i to which the conditional probability distributions $p(V_j \mid \pi(V_j))$ of the nodes V_j from a set $W \subseteq U_i$ are assigned. The potential function ϕ_{U_i} then equals

$$\phi_{U_i}(U_i) = \prod_{V_j \in W} p(V_j \mid \pi(V_j))$$

We illustrate this with our running example.

Example 2.3 Consider once more the digraph from Figure 1a. Figure 2 shows a junction tree for this network, constructed from the triangulated graph in Figure 1b. The ovals represent the cliques and the boxes indicate the separators. The conditional probability distributions for the variables in the Bayesian network are assigned to the cliques in the junction tree. The conditional probability distributions for node V_6, for example, are

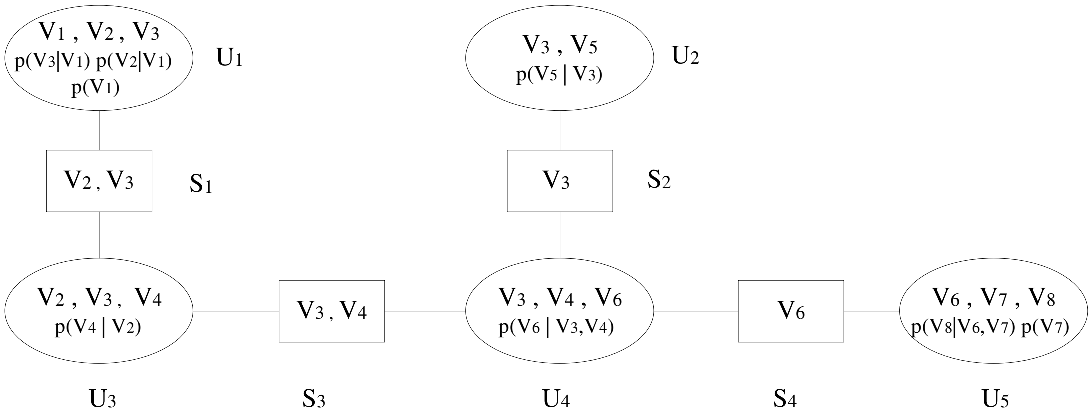

Figure 2: The junction tree obtained from the network in Figure 1a.

attached to clique U_4. Therefore, all the individual parameters from these conditional probability distributions are located in clique U_4 only. □

The junction tree is taken as a computational architecture for processing evidence and for computing an updated probability distribution from a Bayesian network. The cliques in the tree are viewed as autonomous objects and the tree's edges are looked upon as bi-directional communication channels. Through the communication channels, the cliques send each other messages providing information about the represented joint probability distribution and about the evidence entered in the tree. Each clique is able to compute the (updated) marginal probability distribution on its variables from its local marginal distribution and the information it receives from its neighbours [Jensen, 1996]. A clique is allowed to send a message over the edge to a neighbouring clique only if it has received a message from all its other neighbours; it then is said to be triggered. Initially, all leaf cliques are triggered and there is always a triggered clique until a message has been passed in both directions over all edges. The computations can be organised as *rooted propagation*: a clique is chosen as the root of the junction tree and the message passing is directed to and from the root. First, there is an *inward propagation* in which all messages are directed at the root; subsequently, messages are sent from the root towards the leaves, called *outward propagation*. Before propagating messages through the tree, first the evidence available is entered in the tree. To that end, the values of the observed variables are kept fixed in the potential functions associated with each clique.

To further detail the propagation of information in the junction tree, we focus on

Shafer-Shenoy propagation [Shafer & Shenoy, 1988]. All separators in the junction tree are looked upon as having two mail boxes, one for each direction. Consider a clique U_i with neighbouring separators $S_{i_1}, \cdots, S_{i_{t+1}}$, $t \geq 0$, and a clique U_j that is connected with U_i through the separator S_{i_t}. Clique U_i computes its message to U_j via $S_{i_{t+1}}$, written $\hat{\phi}_{U_i \to S_{i_{t+1}}}$, by multiplying, for each configuration $u \in \Omega_{U_i}$, its marginal potential $\phi_{U_i}(u)$ with the message $\hat{\phi}_{S_{i_j} \to U_i}(s_j)$, where $s_j \in \Omega_{S_{i_j}}$ and $s_j \wedge u = u$ and $j = 1, \ldots, t$:

$$\phi'_{U_i}(u) = \phi_{U_i}(u) \cdot \prod_{j=1,\ldots,t} \hat{\phi}_{S_{i_j} \to U_i}(s_j)$$

The resulting function ϕ'_{U_i} is *projected* onto $S_{i_{t+1}}$, that is, ϕ'_{U_i} is marginalized over all variables that are not in $S_{i_{t+1}}$. The result is the message $\hat{\phi}_{U_i \to S_{i_{t+1}}}$ to U_j via $S_{i_{t+1}}$;

$$\hat{\phi}_{U_i \to S_{i_{t+1}}}(s_{t+1}) = \sum_{u: s_{t+1} \wedge u = u} \phi'_{U_i}(u)$$

The message, a function over the variables in $S_{i_{t+1}}$ is placed in the appropriate mail box. When message passing has stopped, the two mail boxes of each separator contain two such functions. The marginal distribution over the variables in any clique U_i can now be computed by multiplying, for each configuration $u \in \Omega_{U_i}$, the potential $\phi_{U_i}(u)$ with all incoming messages. To obtain the probability distribution over one specific variable V_k, the marginal distribution of a clique containing V_k is projected onto V_k. To obtain the probability of the evidence, the marginal probability distribution of any clique U_i is marginalised over all variables in that clique. In the following, inward propagation towards a chosen root R is described in pseudocode. At root R, the probability of the evidence $\Pr(e)$ is computed by marginalising the marginal distribution of R over all variables in R.

the propagation algorithm amounts to the following:

procedure propagate (T, e, R)

begin

step 1 enter evidence e into the clique tree T;

  until root R in T is triggered do

    for each triggered clique U_i do

step 2   for each configuration $u \in \Omega_{U_i}$ do

        for each configuration $s_j \in \Omega_{S_{i_j}}$ with $s_j \wedge u = u$ do

          compute $\phi'_{U_i}(u) = \phi_{U_i}(u) \cdot \prod_{j=1,\ldots,t} \hat{\phi}_{S_{i_j} \to U_i}(s_j)$

        od;

      od;

step 3   project $\phi'_{U_i}(U_i)$ onto $S_{i_{t+1}}$, giving $\hat{\phi}_{U_i \to S_{i_{t+1}}}(S_{i_{t+1}})$;

step 4   send $\hat{\phi}_{U_i \to S_{i_{t+1}}}$ to $S_{i_{t+1}}$

    od

  od;

step 5 for each configuration $r \in \Omega_R$ do

    for each configuration $s_j \in \Omega_{S_{R_j}}$ with $s_j \wedge r = r$ do

      compute $\Pr(r, e) = \phi_R(r) \cdot \prod_{j=1,\ldots,k} \hat{\phi}_{S_{R_j} \to R}(s_j)$

    od

  od;

step 6 compute $\Pr(e) = \sum_{r \in \Omega_R} \Pr(r, e)$;

end

In the following, we illustrate evidence propagation in a junction tree with our running example.

Example 2.4 Consider again the junction tree in Figure 2. Suppose the variable V_8 is observed to take the value v_8. Furthermore, assume that we are interested in the posterior probability that the variable V_6 takes the value v_6, that is, we want to compute $\Pr(v_6 \mid v_8)$. To that end, we perform an inward propagation towards clique U_4 in Figure 2. The outward propagation from U_4 towards the leaves is omitted. We only need the incoming messages in U_4 to compute $\Pr(v_6 \mid v_8)$ because V_6 is located in U_4. For the computation of several posterior probabilities simultaneously, an outward propagation should be performed too.

First, the evidence $V_8 = v_8$ is entered in the junction tree. This accords with `step 1` of procedure `propagate`. Then `step 2` to `step 4` of `propagate` are carried out for all triggered cliques until no triggered cliques remain. Initially, only leaf cliques are triggered. We start by computing the message $\hat{\phi}_{U_1 \to S_1}$ that leaf U_1 sends to separator S_1. `Step 2` is omitted since U_1 is a leaf node for which there are no incoming messages. In `step 3`, the local potential function ϕ_{U_1} in clique U_1 is projected onto separator S_1, that is, we sum over the values of the variable V_1:

$$\hat{\phi}_{U_1 \to S_1}(V_2, V_3) = p(V_3 \mid v_1) \cdot p(V_2 \mid v_1) \cdot p(v_1) + p(V_3 \mid \neg v_1) \cdot p(V_2 \mid \neg v_1) \cdot p(v_1)$$

For the four combinations of values for V_2 and V_3, the function values $\hat{\phi}_{U_1 \to S_1}(v_2, v_3)$, $\hat{\phi}_{U_1 \to S_1}(\neg v_2, v_3)$, $\hat{\phi}_{U_1 \to S_1}(v_2, \neg v_3)$, and $\hat{\phi}_{U_1 \to S_1}(\neg v_2, \neg v_3)$ are sent to clique U_3 via S_1 (`step 4`). Represented as a table, separator S_1 receives

	v_3	$\neg v_3$
v_2	0.63	0.18
$\neg v_2$	0.15	0.04

Note that, since $\hat{\phi}_{U_1 \to S_1}$ is sent on to clique U_3, $\hat{\phi}_{U_1 \to S_1} = \hat{\phi}_{S_1 \to U_3}$. For the message from clique U_2 to S_2 we find,

$$\hat{\phi}_{U_2 \to S_2}(V_3) = p(v_5 \mid V_3) + p(\neg v_5 \mid V_3) = 1$$

Represented as a table, $\hat{\phi}_{U_2 \to S_2}$ equals

v_3	1
$\neg v_3$	1

The message $\hat{\phi}_{U_5 \to S_4}$ from U_5 to S_4 is computed in the same way as $\hat{\phi}_{U_1 \to S_1}$ and $\hat{\phi}_{U_1 \to S_2}$. Recall that the variable V_8 is observed to take the value v_8. $\hat{\phi}_{U_5 \to S_4}(V_6)$ then equals

v_6	0.34
$\neg v_6$	0.56

To compute the message from U_3 to S_3, the marginal potential function $\phi_{U_3}(V_2, V_3, V_4) = p(V_4 \mid V_2)$ of clique U_3 is multiplied with the incoming messages$\hat{\phi}_{U_1 \to S_1}(V_2, V_3)$ from S_1 (`step 2`) and subsequently projected onto S_3 (`step 3`). That is,

$$\hat{\phi}_{U_3 \to S_3}(V_3, V_4) = p(V_4 \mid v_2) \cdot \hat{\phi}_{U_1 \to S_1}(v_2, V_3) + p(V_4 \mid \neg v_2) \cdot \hat{\phi}_{U_1 \to S_1}(\neg v_2, V_3)$$

Represented as a table, the message is

	v_3	$\neg v_3$
v_4	0.261	0.072
$\neg v_4$	0.519	0.148

At clique U_4 the messages from the three separators S_2, S_3, and S_4 are available. The probability distribution over the variables V_3, V_4 and V_6 from U_4 with the evidence $V_8 = v_8$ is now computed by carrying out `step 5` of procedure `propagate`;

$$\begin{aligned}\Pr(V_3 \wedge V_4 \wedge V_6 \wedge v_8) &= \phi_{U_4}(V_3, V_4, V_6) \cdot \hat{\phi}_{U_2 \to S_2}(V_3) \cdot \hat{\phi}_{U_3 \to S_3}(V_3, V_4) \cdot \hat{\phi}_{U_5 \to S_4}(V_6) \\ &= p(V_6 \mid V_3, V_4) \cdot \hat{\phi}_{U_2 \to S_2}(V_3) \cdot \hat{\phi}_{U_3 \to S_3}(V_3, V_4) \cdot \hat{\phi}_{U_5 \to S_4}(V_6)\end{aligned}$$

Represented as a table $\Pr(V_3 \wedge V_4 \wedge V_6 \wedge v_8)$ equals

	v_3		$\neg v_3$	
	v_4	$\neg v_4$	v_4	$\neg v_4$
v_6	0.0799	0.1412	0.0196	0.0151
$\neg v_6$	0.0146	0.0581	0.0081	0.0580

As we are interested in $\Pr(v_6 \mid v_8)$, we marginalize over the variables V_3 and V_4, to give

$$\Pr(v_6, v_8) = 0.2557$$

and

$$\Pr(\neg v_6, v_8) = 0.1388$$

The posterior probability of interest $\Pr(v_6 \mid v_8)$, now, is the quotient of $\Pr(v_6 \wedge v_8)$ and the probability of the evidence $\Pr(v_8)$. This latter probability is obtained by marginalization of $\Pr(V_6 \wedge v_8)$ over V_6. So,

$$\Pr(v_6 \mid v_8) = \frac{\Pr(v_6, v_8)}{\Pr(v_8)} = \frac{\Pr(v_6, v_8)}{\Pr(v_6, v_8) + \Pr(\neg v_6, v_8)} = \frac{0.2557}{0.3945} = 0.6482$$

□

3 Sensitivity analysis of a Bayesian network

Sensitivity analysis is a technique to systematically study the effects of variations in the parameters of a mathematical model on this model's outcome. The technique is widely used in the fields of decision theory and mathematical modelling to investigate the possible consequences of inaccuracies in a model's parameters [Habbema *et al.*, 1990, Morgan & Henrion, 1990, Von Winterfeldt & Edwards, 1986]. For a Bayesian network, sensitivity analysis provides for studying the effects of variations in the estimates for an arbitrary subset of parameters in the network on a prior or posterior probability computed from the network. As such, sensitivity analysis of a Bayesian network allows for identifying network parameters that independently or jointly have a large effect on a probability of interest. The results of such an analysis can be used to measure the robustness of the network to parameter variation and to guide refinement of the network by pointing out highly influential parameters. In this paper, we present a method for performing sensitivity analysis of a Bayesian network in an efficient way. To that end, we review here relevant work on sensitivity analysis.

In its most general form, in a sensitivity analysis the estimates for n network parameters are varied stepwise in a systematic way, while the estimates for the remaining network parameters are kept fixed. This is termed an n-way sensitivity analysis. It reveals the joint effect of variation of the n parameters under study.

When varying the estimate for a parameter under study, the estimates for the network's parameters that pertain to the same conditional probability distribution have to be adjusted such that the sum of all parameter estimates in this distribution again equals one. The parameters that have to be adjusted are called *co-varying* parameters for the parameter under study. Assume, that we have a parameter under study $x = p(b_i \mid \pi')$ that pertains to the value b_i of the variable B and the configuration π' for the parents of B. The possible values for the variable B are $b_1, \ldots, b_k$, $k \geq 1$. The co-varying parameters for x then are the conditional probabilities $p(b_j \mid \pi')$, $j \neq i$. The sum of the estimates for the co-varying parameters, here called the *residual probability*, equals $1 - x$. Now, in varying the estimate for x, the estimates for these co-varying parameters are adjusted by the ratio of the new residual probability $1 - x$ and the original residual probability $1 - p(b_i \mid \pi')$. Considering $p(b_j \mid \pi')$ as a function of x, denoted $p(b_j \mid \pi')(x)$, we have that

$$p(b_j \mid \pi')(x) = p(b_j \mid \pi') \cdot \frac{1 - x}{1 - p(b_i \mid \pi')}$$

If more than one parameter from the same probability distribution is taken as a parameter under study, the residual probability equals one minus the sum of the estimates for these parameters.

Under the assumption that systematic variation of network parameters is carried out as described above, any prior probability can be expressed as a multilinear function in the n network parameters under study; any posterior probability is a quotient of such multilinear functions [Castillo *et al.*, 1995, Coupé and Van der Gaag, 1998]. The functional relation describing a probability of interest in terms of n parameters under study is termed an n-way sensitivity function.

Suppose, we are interested in the posterior probability $\Pr(a \mid e)$ for some specific value a of variable A and evidence e. Assume, furthermore, that we take for the parameters under study the n network parameters $X = \{x_0, \ldots, x_{n-1}\}$, $n \geq 1$. Recall that, by definition, $\Pr(a \mid e) = \Pr(a \wedge e)/\Pr(e)$. Both the numerator, $\Pr(a \wedge e)$, and the denominator, $\Pr(e)$, in this equation can be expressed as a multilinear relation in the parameters $x_0, \ldots, x_{n-1}$. For each multilinear relation 2^{n-1} coefficients are required. In studying the robustness of $\Pr(a \mid e)$ with respect to variations in the n parameters under study, it suffices to establish for $\Pr(a \wedge e)$ and $\Pr(e)$ the values of these 2^{n-1} coefficients. In the following, we illustrate determining these coefficients with our running example.

Example 3.1 Consider the junction tree of Figure 2. As in Example 2.4, we assume that for the variable V_8 the value v_8 is observed. The probability of interest again is the probability that the variable V_6 takes the value v_6. In the following, the sensitivity of $\Pr(v_6 \mid v_8)$ to variations in the estimates for the network parameters $x_0 = p(v_8 \mid v_6, v_7)$, $x_1 = p(v_7)$, and $x_2 = p(v_4 \mid \neg v_2)$ is investigated. In the remainder of the paper, the notation $\Pr(v_6 \mid v_8)(x_0, x_1, x_2)$, refering to the expression for $\Pr(v_6 \mid v_8)$ in terms of x_0, x_1, and x_2, will be omitted if there is no doubt that the functional form rather than the function value s meant. The general form of the relationship between $\Pr(v_6 \mid v_8)$ and x_0, x_1 and x_2 then is,

$$\Pr(v_6 \mid v_8) = \frac{\Pr(v_6, v_8)}{\Pr(v_8)} =$$

$$= \frac{c_0 + c_1 \cdot x_0 + c_2 \cdot x_1 + c_3 \cdot x_0 x_1 + c_4 \cdot x_2 + c_5 \cdot x_0 x_2 + c_6 \cdot x_1 x_2 + c_7 \cdot x_0 x_1 x_2}{d_0 + d_1 \cdot x_0 + d_2 \cdot x_1 + d_3 \cdot x_0 x_1 + d_4 \cdot x_2 + d_5 \cdot x_0 x_2 + d_6 \cdot x_1 x_2 + d_7 \cdot x_0 x_1 x_2}$$

In studying the robustness of $\Pr(v_6 \mid v_8)$ to variations in the estimates for x_0, x_1 and x_2, it suffices to establish the value of all coefficients in the functional relation of $\Pr(v_6 \mid v_8)$. We start with the computation of the coefficients d_j, $j = 0, \ldots, 7$. To that end, we compute the probability $\Pr(v_8)$ from the junction tree for eight different combinations of values for x_0, x_1 and x_2; for the remaining network parameters, the estimates specified in Figure 1a are taken. Computing for each selected combination of values for $\{x_0, x_1, x_2\}$ the probability $\Pr(v_8)_{\{x_0,x_1,x_2\}}$, we find for example

$$\begin{aligned}
\Pr(v_8)_{\{0.4,0.4,0.4\}} &= 0.4865 \\
\Pr(v_8)_{\{0.6,0.4,0.4\}} &= 0.5453 \\
\Pr(v_8)_{\{0.4,0.6,0.4\}} &= 0.4665 \\
\Pr(v_8)_{\{0.4,0.4,0.6\}} &= 0.4858 \\
\Pr(v_8)_{\{0.6,0.6,0.4\}} &= 0.5547 \\
\Pr(v_8)_{\{0.6,0.4,0.6\}} &= 0.5452 \\
\Pr(v_8)_{\{0.4,0.6,0.6\}} &= 0.4658 \\
\Pr(v_8)_{\{0.6,0.6,0.6\}} &= 0.5548
\end{aligned}$$

Substituting the left hand side in each of the equations above with $d_0 + d_1 \cdot x_0 + d_2 \cdot x_1 + d_3 \cdot x_0 \cdot x_1 + d_4 \cdot x_2 + d_5 \cdot x_0 \cdot x_2 + d_6 \cdot x_1 \cdot x_2 + d_7 \cdot x_0 \cdot x_1 \cdot x_2$ and filling in the appropriate values for x_0, x_1, and x_2 gives a system of multi-linear equations. Solving this system, yields the coefficients

$$\begin{aligned}
d_0 &= 0.5279 \\
d_1 &= 0 \\
d_2 &= -0.3882 \\
d_3 &= 0.7206 \\
d_4 &= -0.0035 \\
d_5 &= 0 \\
d_6 &= -0.0140 \\
d_7 &= 0.0350
\end{aligned}$$

The coefficients c_i, $i = 0, \ldots, 7$ are determined analogously. We find for c_0, c_1, c_2, c_3, c_4, c_5, c_6, and c_7 the values 0.3603, 0, −0.3603, 0.7206, 0.0175, 0, −0.0175, and 0.0350, respectively. The relationship of $\Pr(v_6 \mid v_8)$ with x_0, x_1 and x_2 thus equals

$$\Pr(v_6 \mid v_8) =$$

$$= \frac{0.3603 - 0.3603 \cdot x_1 + 0.7206 \cdot x_0 x_1 + 0.0175 \cdot x_2 - 0.0175 \cdot x_1 x_2 + 0.0350 \cdot x_0 x_1 x_2}{0.5279 - 0.3882 \cdot x_1 + 0.7206 \cdot x_0 x_1 - 0.0035 \cdot x_2 - 0.0140 \cdot x_1 x_2 + 0.0350 \cdot x_0 x_1 x_2}$$

□

From the example, it is readily seen that a multilinear function in $X = \{x_0, \dots, x_{n-1}\}$, in general, contains a term for every possible subset $X' \subseteq X$ of these parameters. This term is the product of the parameters in X' and a coefficient c. In the following, we define a coding for the coefficients in an n-way sensitivity function that serves to uniquely identify the subset X' of parameters to which it pertains. In the method for sensitivity analysis, presented in Section 4, this coding is required to trace the parameters under study that are taken into account at any time during propagation in a junction tree. Now, the product of the parameters from the subset X' can be written as a product of *all* parameters in X, by raising the parameters $x_i \in X'$ to the power one and the parameters $x_j \in X \setminus X'$ to the power zero, that is,

$$\prod_{x_i \in X'} x_i = \Big(\prod_{x_i \in X'} x_i^1\Big) \cdot \Big(\prod_{x_j \in X \setminus X'} x_j^0\Big)$$

With every possible subset $X' \subseteq X$ thus is associated a set of exponents, one for each parameter in X. The exponent for a parameter x_i is indicated by ϵ_i. To identify that the coefficient c pertains to subset X', a subscript is given to c. To that end, the exponents ϵ_i, $i = 0, \dots, n-1$, encoding X' in terms of X are placed one after the other, starting from ϵ_{n-1} upto ϵ_0. A binary number, $bin(\epsilon_{n-1}, \dots, \epsilon_0)$, results. This binary number provides a unique code for the subset X'. The same holds for the numerical number $\mathcal{N}(\epsilon_{n-1}, \dots, \epsilon_0)$ into which $bin(\epsilon_{n-1}, \dots, \epsilon_0)$ can be translated. Consider, for example, the subscripts of the coefficients c_i and d_j, $i, j = 0, \dots, 7$, in Example 3.1. For the product of parameters $x_1 x_2$, we can write $x_0^0 x_1^1 x_2^1$. Placing the exponents ϵ_2, $epsilon_1$, and ϵ_0 for x_2, x_1 and x_0, respectively, one after the other, we obtain the binary number 110, which stands for the numerical number 6. The coefficients corresponding with $x_1 \cdot x_2$, thus are indicated by c_6 and d_6.

Using the notations introduced above, the relationship between a posterior probability of interest $\Pr(a \mid e)$ and the parameters $x_0, \dots, x_{n-1}$ can be written as

$$\Pr(a \mid e) = \frac{\displaystyle\sum_{k:k=\mathcal{N}(\epsilon_{n-1},\dots,\epsilon_0)} \Big(c_k \cdot \prod_{i=0,\dots,n-1} x_i^{\epsilon_i}\Big)}{\displaystyle\sum_{l:l=\mathcal{N}(\epsilon_{n-1},\dots,\epsilon_0)} \Big(c_l \cdot \prod_{j=0,\dots,n-1} x_j^{\epsilon_j}\Big)}$$

4 An efficient method for computing n-way sensitivity

In this section, we outline our method for obtaining the coefficients in a sensitivity function expressing the prior probability of the evidence in a Bayesian network in terms of an arbitrary subset of network parameters. As was seen in Example 3.1, the computation of the coefficients in the n-way sensitivity function of a posterior probability proceeds analogously; it is built from the n-way sensitivity functions of two prior probabilities. We start, in Section 4.1, by giving an example of the method, using the junction tree introduced in Section 2. In Section 4.2, then, a formal description of the method is given.

4.1 An illustration

The method that we present in this section is closely related to standard propagation methods for evidence propagation in a junction tree. Our method can in fact be seen as a variant of these methods. In a junction tree, the probability of the available evidence is computed by performing one inward propagation to some arbitrary clique and subsequent marginalization over all variables from that clique (see Section 2). For this inward propagation, standard algorithms send messages taking the form of potential functions. These messages can be represented as tables, containing in each entry a single numerical value. The algorithm we propose in this paper closely follows the basic idea of these propagation algorithms. In our method, instead of numerical numbers, expressions are being propagated through the junction tree. The idea resembles symbolic propagation; in our method, however, the use of a specific encoding renders the propagation of symbols unnecessary.

With each potential function in standard evidence propagation, in our method a vector table is associated. For each value of a potential function, the corresponding vector table has an entry that contains a vector of coefficients which expresses this function value in terms of the parameters under study in the junction tree. The messages in our method are the *vector tables* corresponding with the messages in standard evidence propagation. Following the notation introduced in Section 2, a local potential function $\phi_{U_i}(U_i)$ for clique U_i has an associated vector table $\psi_{U_i}(U_i)$. A message $\hat{\phi}_{U_i \to S_{i_j}}(S_{i_j})$ from clique U_i to separator S_{i_j} in standard evidence propagation corresponds with a message $\hat{\psi}_{U_i \to S_{i_j}}(S_{i_j})$ in our method.

Example 4.1 We illustrate the idea of propagating vector tables with the example junction tree in Figure 2. We again assume that for the variable V_8 the value v_8 is observed. The prior probability of interest is the probability of the evidence, $\Pr(v_8)$. We are interested in the sensitivity of $\Pr(v_8)$ to variations in the estimates for the three network parameters $x_0 = p(v_8 \mid v_6, v_7)$, $x_1 = p(v_7)$, and $x_2 = p(v_4 \mid \neg v_2)$. To that end, the coefficients in the multilinear function expressing $\Pr(v_8)$ in terms of x_0, x_1 and x_2 are determined. The form of this multilinear function is

$$\Pr(e) = c_0 + c_1 \cdot x_0 + c_2 \cdot x_1 + c_3 \cdot x_0 x_1 + c_4 \cdot x_2 + c_5 \cdot x_0 x_2 + c_6 \cdot x_1 x_2 + c_7 \cdot x_0 x_1 x_2 \tag{1}$$

Our method starts with an initialization step; for each clique U_i in the junction tree, a local vector table $\psi_{U_i}(U_i)$ is derived from the potential function $\phi_{U_i}(U_i)$. For cliques U_i that do not contain any of the parameters under study, $\psi_{U_i}(U_i)$, in essence, equals $\phi_{U_i}(U_i)$; each function value in $\phi_{U_i}(U_i)$, however, is taken as a vector with one element in $\psi_{U_i}(U_i)$. Since cliques U_1, U_2 and U_4 do not contain any of the three parameters under study, we thus basically have that $\psi_{U_1}(U_1) = \phi_{U_1}(U_1)$, $\psi_{U_2}(U_2) = \phi_{U_2}(U_2)$, and $\psi_{U_4}(U_4) = \phi_{U_4}(U_4)$. Clique U_5 specifies the parameters x_0 and x_1. The potential function $\phi_{U_5}(U_5)$ is now expressed in terms of x_0 and x_1:

v_8	v_7	$\neg v_7$
v_6	$x_0 \cdot x_1$	$0.5 \cdot (1 - x_1)$
$\neg v_6$	$0.5 \cdot x_1$	$0.6 \cdot (1 - x_1)$

Each entry in this table is a multilinear function in x_0 and x_1. The general shape of the various different entries is $c_0(U_5) + c_1(U_5) \cdot x_0 + c_2(U_5) \cdot x_1 + c_3(U_5) \cdot x_0 x_1$ or, alternatively, $\sum_{k:k=\mathcal{N}(\epsilon_1,\epsilon_0)} \left(c_k(U_5) \cdot \prod_{i=0,1} x_i^{\epsilon_i} \right)$. The notation $c_k(U_5)$, $k = 0, \ldots, 3$, is used to indicate that the coefficient c_k, which is a constant with respect to the parameters x_0 and x_1, may differ for the clique U_5 from configuration to configuration. Each entry in the table can be encoded by a vector of coefficients, $(c_0(U_5), c_1(U_5), c_2(U_5), c_3(U_5))$, where each coefficient pertains to a specific product of parameters from $\{x_0, x_1\}$. The vector table $\psi_{U_5}(U_5)$ corresponding with $\phi_{U_5}(U_5)$ then becomes

v_8	v_7	$\neg v_7$
v_6	$(0,0,0,1)$	$(0.5,0,-0.5,0)$
$\neg v_6$	$(0,0,0.5,0)$	$(0.6,0,-0.6,0)$

Clique U_3, finally, contains the parameter x_2 under study. Writing the potential function $\phi_{U_3}(U_3)$ in terms of x_2 gives:

	v_2	$\neg v_2$
v_4	0.2	x_2
$\neg v_4$	0.8	$1 - x_2$

Each value of $\phi_{U_3}(U_3)$ is a linear function in x_2, with the general shape $c_0(U_3) + c_1(U_3) \cdot x_2$ or $\sum_{k:k=\mathcal{N}(\epsilon_2)} \left(c_k(U_3) \cdot x_2^{\epsilon_2} \right)$. Encoding each value of $\phi_{U_3}(U_3)$ as a vector of coefficients $(c_0(U_3), c_1(U_3))$ gives the following vector table $\psi_{U_3}(U_3)$:

	v_2	$\neg v_2$
v_4	$(0.2,0)$	$(0,1)$
$\neg v_4$	$(0.8,0)$	$(1,-1)$

Note, that for clique U_3, the local numbering of the coefficients does not coincide with the global numbering of coefficients in Equation 1. The subscript 1 of $c_1(U_3)$, for example, would refer to the product $x_0^1 x_1^0 x_2^0 = x_0$ in the global numbering. During propagation of vector tables in the junction tree, therefore, the order number of the highest ordered parameter to which the vectors pertain should be sent along with the table. It serves to identify, at each time during propagation, which parameters of the set of all parameters under study are considered.

Initialization of the junction tree has now taken place. Propagation of vector tables follows. We assume that, from the leaves of the junction tree, an inward propagation towards clique U_4 is performed, that is, clique U_4 is taken as the root of the tree. Each leaf computes from its local vector table the message it should send to its neighbouring separator(s). If this vector table equals the clique's potential function, because it does not contain any parameter under study, this message will basically be the same as in standard evidence propagation. Both leaf U_1 and U_2 do not contain any of the parameters x_0, x_1 or x_2. The messages $\hat{\psi}_{U_1 \to S_1}$ from U_1 to S_1 and $\hat{\psi}_{U_2 \to S_2}$ from U_2 to S_2, therefore, equal

$\hat{\phi}_{U_1 \to S_1}$ and $\hat{\phi}_{U_2 \to S_2}$, respectively. So $\hat{\psi}_{U_1 \to S_1}$ equals

	v_3	$\neg v_3$
v_2	(0.63)	(0.18)
$\neg v_2$	(0.15)	(0.04)

and $\hat{\psi}_{U_2 \to S_2}$ equals

v_3	(1)
$\neg v_3$	(1)

Leaf U_5 contains the parameters x_0 and x_1. To compute the message from clique U_5 to separator S_4 in standard evidence propagation, the potential function of U_5 is projected onto S_4 (conform `step 3` of the procedure `propagate`). From a conceptual point of view, our method is equivalent to standard propagation in a junction tree. Therefore, to compute the message $\hat{\psi}_{U_5 \to S_4}$ from U_5 to S_4, the vector table ψ_{U_5} is projected onto S_4. Projection of a vector table amounts to vector summation of the appropriate vectors of coefficients. From the table ψ_{U_5}, the vectors in the entries that specify for V_6 the value v_6 and the value $\neg v_6$, respectively, are added. The result is

v_6	$(0.5, 0, -0.5, 1)$
$\neg v_6$	$(0.6, 0, -0.1, 0)$

The coefficients in the vector table $\hat{\psi}_{U_5 \to S_4}(S_4)$ are $c_0(S_4)$, $c_1(S_4)$, $c_2(S_4)$, and $c_3(S_4)$ refering to the multilinear relation $c_0(S_4) + c_1(S_4) \cdot x_0 + c_2(S_4) \cdot x_1 + c_3(S_4) \cdot x_0 x_1$.

For cliques in the junction tree that are not leaves, the local vector table is multiplied with the incoming messages (conform `step 2` of the procedure `propagate`) before projection onto the next separator. Multiplication of a local vector table $\psi_{U_i}(U_i)$ with an incoming message $\hat{\psi}_{S_{i_j} \to U_i}(S_{i_j})$ proceeds as follows. Assume that we have that $\psi_{U_i}(U_i) = (c_0(U_i), \ldots, c_k(U_i))$ and $\hat{\psi}_{S_{i_j} \to U_i}(S_{i_j}) = (c_0(S_{i_j}), \ldots, c_m(S_{i_j}))$. For each configuration $u \in \Omega_{U_i}$ and $s \in \Omega_{S_{i_j}}$ with $s \wedge u = u$, the vector product of $\psi_{U_i}(u)$ and $\hat{\psi}_{S_{i_j} \to U_i}(s)$, indicated by $\psi_{U_i}(u) * \hat{\psi}_{S_{i_j} \to U_i}(s)$, equals

$$\begin{aligned} \psi_{U_i}(u) * \hat{\psi}_{S_{i_j} \to U_i}(s) &= (c_0(u), \ldots, c_k(u)) * (c_0(s), \ldots, c_m(s)) = \\ &= (c_0(u) \cdot c_0(s), \ldots, c_0(u) \cdot c_m(s), c_1(u) \cdot c_0(s), \ldots c_k(u) \cdot c_m(s)) \end{aligned}$$

Note that the multiplication is not commutative, that is, $\psi_{U_i}(u) * \hat{\psi}_{S_{i_j} \to U_i}(s)$ is not necessarily equal to $\hat{\psi}_{S_{i_j} \to U_i}(s) * \psi_{U_i}(u)$. To obtain the global ordering on coefficients at the root of the junction tree, the following order of multiplying a vector table $\psi_{U_i}(U_i)$ with incoming messages $\hat{\psi}_{S_{i_j} \to U_i}$, $j = 1, \ldots, t$, is taken. The messages $\hat{\psi}_{S_{i_j} \to U_i}$ are ordered from 1 to t, according to the descending number of their highest ordered parameter. For each configuration $u \in \Omega_{U_i}$, and $s_j \in \Omega_{S_{i_j}}$, $j = 1, \ldots, t$, with $s_j \wedge u = u$, the multiplication is defined as follows:

$$\psi_{U_i}(u) * \prod^{*}_{j=1,\ldots,t} \hat{\psi}_{S_{i_j} \to U_i}(s_j) =$$

$$= \left(\cdots\left(\left(\psi_{U_i}(u) * \hat{\psi}_{S_{i_1} \to U_i}(s_1)\right) * \hat{\psi}_{S_{i_2} \to U_i}(s_2)\right) * \cdots * \hat{\psi}_{S_{i_{t-1}} \to U_i}(s_t)\right)$$

Consider now clique U_3 in our running example. The local vector table ψ_{U_3} is multiplied with the message $\hat{\psi}_{S_1 \to U_3}$ coming from S_1. Since the message $\hat{\psi}_{S_1 \to U_3}$ comes from a part of the junction tree where no parameters under study reside, this vector table contains in each entry a single coefficient $c(S_1)$. For each configuration $u \in \Omega_{U_3}$ and $s \in \Omega_{S_1}$ with $u \wedge s = u$ equals:

$$\begin{aligned}
\psi_{U_3}(u) * \hat{\psi}_{S_1 \to U_3}(s) &= \psi_{U_3}(u) * \hat{\psi}_{U_1 \to S_1}(s) \\
&= (c_0(u), c_1(u)) * (c(s)) = \\
&= (c_0(u) \cdot c(s), c_1(u) \cdot c(s)) = \\
&= (c'_0(u), c'_1(u))
\end{aligned}$$

For example, the vector $(0.15, -0.15)$ in the product for the configuration $\neg v_2, v_3, \neg v_4$ is obtained after multiplication of the vector $(1, -1)$ in $\psi_{U_3}(\neg v_2, v_3, \neg v_4)$ with the coefficient 0.15 in $\hat{\psi}_{U_1 \to S_1}(\neg v_2, v_3)$; note that this is equivalent to multiplying the linear function $1 - x_2$ with the constant 0.15. The vector table $\psi_{U_3} * \hat{\psi}_{U_1 \to S_1}$ equals:

	v_2		$\neg v_2$	
	v_3	$\neg v_3$	v_3	$\neg v_3$
v_4	$(0.126, 0)$	$(0.036, 0)$	$(0, 0.15)$	$(0, 0.04)$
$\neg v_4$	$(0.504, 0)$	$(0.144, 0)$	$(0.15, -0.15)$	$(0.04, -0.04)$

To obtain the message $\hat{\psi}_{U_3 \to S_3}$, finally, this vector table is projected onto S_3, giving the following table with in each entry a vector of coefficients $(c_0(S_3), c_1(S_3))$:

	v_3	$\neg v_3$
v_4	$(0.126, 0.15)$	$(0.036, 0.04)$
$\neg v_4$	$(0.654, -0.15)$	$(0.184, -0.04)$

At clique U_4, now, all messages from neighbouring separators are available. This clique can therefore compute the coefficients in the n-way sensitivity function expressing $\Pr(e)$ in terms of the parameters x_0, x_1, and x_2 under study. To that end, the local vector table ψ_{U_4} is multiplied, subsequently, with the messages $\hat{\psi}_{S_2 \to U_4}$, $\hat{\psi}_{S_3 \to U_4}$, and $\hat{\psi}_{S_4 \to U_4}$. As the message $\hat{\psi}_{U_2 \to S_2}$ equals unity, it has no effect on ψ_{U_4}. Multiplication with the messages $\hat{\psi}_{S_3 \to U_4}$ and $\hat{\psi}_{S_4 \to U_4}$ is carried out in the order of decreasing number of the highest ordered parameter. Thus, ψ_{U_4} is first multiplied by $\hat{\psi}_{S_3 \to U_4}$. For each configuration $u \in \Omega_{U_4}$ and $s \in \Omega_{S_3}$ with $s \wedge u = u$, $\psi_{U_4}(u) = (c(u))$ is multiplied by $\hat{\psi}_{S_3 \to U_4}(s) = (c_0(s), c_1(s))$ giving $\psi'_{U_4}(u) = (c'_0(u), c'_1(u)) = (c(u) \cdot c_0(s), c(u) \cdot c_1(s))$

	v_4		$\neg v_4$	
	v_3	$\neg v_3$	v_3	$\neg v_3$
v_6	$(0.1134, 0.135)$	$(0.0288, 0.032)$	$(0.5232, -0.12)$	$(0.0552, -0.012)$
$\neg v_6$	$(0.0126, 0.015)$	$(0.0072, 0.008)$	$(0.1308, -0.03)$	$(0.1288, -0.028)$

Note that the coefficients in $\psi'_{U_4}(U_4)$ refer to a linear relation in x_2.

Multiplying the vector table ψ'_{U_4} with $\hat{\psi}_{S_4 \to U_4}$, finally, proceeds as follows. For each configuration $u \in \Omega_{U_4}$ and $s \in \Omega_{S_4}$ with $s \wedge u = u$, we compute

$$\begin{aligned}
\psi''_{U_4}(u) &= \psi'_{U_4}(u) * \hat{\psi}_{S_4 \to U_4}(s) = \\
&= (c'_0(u) \cdot c_0(s)\ ,\ c'_0(u) \cdot c_1(s)\ ,\ c'_0(u) \cdot c_2(s)\ ,\ c'_0(u) \cdot c_3(s)\ , \\
&\qquad c'_1(u) \cdot c_0(s)\ ,\ c'_1(u) \cdot c'_1(s)\ ,\ c'_1(u) \cdot c_2(s)\ ,\ c'_1(u) \cdot c_3(s)) \\
&= (c''_0(u)\ ,\ c''_1(u)\ ,\ c''_2(u)\ ,\ c''_3(u)\ ,\ c''_4(u)\ ,\ c''_5(u)\ ,\ c''_6(u)\ ,\ c''_7(u))
\end{aligned}$$

This vector refers to the coefficients yielded by multiplication of the two multilinear functions

$$\sum_{k:k=\mathcal{N}(\epsilon_2)} \left(c'_k(u) \cdot x_i^{\epsilon_2}\right) \cdot \sum_{l:l=\mathcal{N}(\epsilon_1,\epsilon_0)} \left(c_l(s) \cdot \prod_{i=0,1} x_i^{\epsilon_i}\right) =$$

$$= \sum_{\substack{k\,:\,k=\mathcal{N}(\epsilon_2)\\ l\,:\,l=\mathcal{N}(\epsilon_1,\epsilon_0)}} \left(c'_k(u) \cdot c_l(s) \cdot \prod_{i=0,1,2} x_i^{\epsilon_i}\right) =$$

$$= \sum_{k:k=\mathcal{N}(\epsilon_2,\epsilon_1,\epsilon_0)} \left(c''_k(u) \cdot \prod_{i=0,1,2} x_i^{\epsilon_i}\right)$$

Note that each subscript for the new coefficients $c''_k(u)$, $k = 0,\ldots 7$, refers to the product of a specific subset of parameters in $\{x_0, x_1, x_2\}$. This is achieved by placing the binary number of a coefficient in ψ'_{U_4} to the right of the binary number of the coefficient in $\hat{\psi}_{U_5,\to S_4}$ with which it is multiplied. The vector table $\psi''_{U_4}(U_4)$ resulting from this multiplication now equals

	v_4		$\neg v_4$	
	v_3	$\neg v_3$	v_3	$\neg v_3$
v_6	$(0.057, 0, -0.057, 0.113, 0.068, 0, -0.068, 0.135)$	$(0.014, 0, -0.014, 0.029, 0.016, 0, -0.016, 0.032)$	$(0.262, 0, -0.262, 0.523, -0.06, 0, 0.06, -0.12)$	$(0.028, 0, -0.028, 0.055, -0.006, 0, 0.006, -0.012)$
$\neg v_6$	$(0.0076, 0, -0.0013, 0, 0.009, 0, -0.0015, 0)$	$(0.0043, 0, -0.0007, 0, 0.0048, 0, -0.0008, 0)$	$(0.078, 0, -0.013, 0, -0.018, 0, 0.003, 0)$	$(0.077, 0, -0.0013, 0, -0.017, 0, 0.0028, 0)$

The vector table $\psi''_{U_4}(U_4)$ now contains the coefficients in the multilinear function expressing $\Pr(V_3, V_4, V_6, v_8)$ in terms of the parameters x_0, x_1, and x_2. To obtain from $\psi''_{U_4}(U_4)$ the function expressing $\Pr(v_8)$ in terms of x_0, x_1, and x_2, all entries of $\psi''_{U_4}(U_4)$ are added. This gives the vector $(0.5279, 0, -0.3882, 0.7206,\ -0.0035, 0, -0.0140, 0.0350)$ which indicates that

$$\Pr(v_8) = 0.5279 - 0.3882 \cdot x_1 + 0.7206 \cdot x_0 x_1 - 0.0035 \cdot x_2 - 0.0140 \cdot x_1 x_2 + 0.0350 \cdot x_0 x_1 x_2$$

□

4.2 Formal description of the method

In this section, we formally detail our method for computing the coefficients in the n-way sensitivity function expressing a prior probability of evidence in terms of the parameters under study $x_0, \ldots, x_{n-1}$. We begin by introducing some concepts and notational conventions. Then, we sketch the method in pseudocode. From this sketch, it will be evident that the method closely resembles standard evidence propagation. We therefore focus attention on the steps in our method that differ from standard evidence propagation. We show that this difference basically lies in the representation of information. At any time during propagation, a translation of potential functions into vector tables and vice versa can be made.

The following notation is used. As in Section 2, for a clique U_i in a junction tree T, the neighbouring separators are indicated by S_{i_j}, $j = 1, \ldots, t+1$; we assume, without loss of generality that $S_{i_{t+1}}$ is the separator to which U_i is to send a message and that the incoming messages $\hat{\psi}_{S_{i_j} \to U_i}$, $j = 1, \ldots, t$, are ordered according to the descending number of their highest ordered parameter. For root R of junction tree T, we indicate the neighbouring separators by S_{R_j}, $j = 1, \ldots, k$. The coefficients in the n-way sensitivity function to be computed are indicated by $(c_0, \ldots, c_{2^n-1})$. R is used to indicate the root of T. The notation T_{U_i} indicates the subtree of T rooted at U_i; that is, U_i lies on the (unique) path from R to any clique $U_j \in T_{U_i}$.

The parameters under study in T are ordered according to a *parameter ordering.*

Definition 4.2 *Let T be a junction tree with the cliques $U(T)$ and let $R \in U(T)$ be the root of T. Let $X = \{x_0, \ldots, x_{n-1}\}$ be the set of parameters under study. Let $o : X \longleftrightarrow \{0, \ldots, n-1\}$ be a total ordering on X, such that*

- *for any two parameters $x_i, x_j \in X$ with x_i in U and x_j in U', for which it holds that U' lies on the (unique) path from R to U, we have $o(x_i) < o(x_j)$;*
- *for any clique U, for the subset of parameters $X' \subseteq X$ located in T_U, we have that $\max\{o(x_i) \mid x_i \in X'\} - \min\{o(x_i) \mid x_i \in X'\} = |X'|$;*

Then, o is a parameter ordering *of X with respect to R.*

Informally speaking, the parameters are ordered in such a way that their order numbers increase from the leaves towards the root and that at any clique the total set of parameters in the subtree rooted at this clique is consecutively numbered. A parameter ordering of the parameters under study in a junction tree is not unique; within a clique, the parameters under study can be ordered arbitrarily and between cliques in general also various different possibilites exist. The purpose of a parameter ordering is to allow for identifying to which subset of parameters the coefficients in a vector table pertain at each point during propagation. When successively multiplying a local vector table with incoming vector tables, the binary subscripts of the incoming coefficients are placed to the right of the binary subscript of each coefficient resulting from the multiplication so far. For every coefficient at the root then, automatically, a subscript is obtained whose binary number indicates to which subset of parameters under study it pertains. This idea was also presented and applied in Example 4.1.

Now, in pseudocode our method amounts to the following.

procedure vector-propagation (T, e, R, X)

begin

step 1 enter evidence e into junction tree T;

step 2 order the set X of n parameters under study from 0 to $n-1$;

step 3 for each node U_i in T do (initialization)

derive ψ_{U_i} from ϕ_{U_i}

od;

until root R in T is triggered do

for each triggered clique U_i do

step 4 for each configuration $u \in \Omega_{U_i}$ do

for each configuration $s_j \in \Omega_{S_{i_j}}$ with $s_j \wedge u = u$, $j = 1, \ldots t$, do

compute $\psi'_{U_i}(u) = \psi_{U_i}(u_i) * \displaystyle\prod^{*}_{j=1,\ldots,t} \hat{\psi}_{S_{i_j} \to U_i}(s_j)$

od

od;

step 5 project $\psi'_{U_i}(U_i)$ onto $S_{i_{t+1}}$, giving $\hat{\psi}_{U_i \to S_{i_{t+1}}}(S_{i_{t+1}})$;

step 6 send $\hat{\psi}_{U_i \to S_{i_{t+1}}}$ to $S_{i_{t+1}}$,

along with the number of the highest ordered parameter

od

od;

step 7 for each configuration $r \in \Omega_R$ do

for each configuration $s_j \in \Omega_{S_{R_j}}$ with $s_j \wedge r = r$, $j = 1, \ldots k$, do

compute $\psi'_R(r) = \psi_R(r) * \displaystyle\prod^{*}_{j=1,\ldots,k} \hat{\psi}_{S_{R_j} \to R}(s_j)$

od

od;

step 8 compute $(c_0, \ldots, c_{2^n-1}) = \displaystyle\sum_{r \in \Omega_R} \psi'_R(r)$

end

Using the parameter ordering as introduced in Definition 4.2, we first focus on the initialization phase (conform `step` 3 of the procedure `vector-propagation`). In the initialization phase, the vector table ψ_{U_i} for clique U_i is derived from the local potential function ϕ_{U_i}. We formally define ψ_{U_i} and then show that ϕ_{U_i} and ψ_{U_i} basically contain the same information; filling in the values for the parameters under study in U_i in the multilinear relation represented by the coefficients in ψ_{U_i} gives ϕ_{U_i}.

Definition 4.3 *Let T be a junction tree and let U_i be a clique in T. Let ϕ_{U_i} be U_i's local potential function. Let $X = \{x_0, \ldots, x_{m-1}\}$ be the set of parameters under study that are located in U_i, $m \geq 1$; for each parameter x_j, $j = 0, \ldots, m-1$, let p_j be the value for x_j as specified in T. Let $\{\epsilon_0, \ldots, \epsilon_{m-1}\}$ be the set of exponents that is associated with $\{x_0, \ldots, x_{m-1}\}$, as introduced in Section 3. For each configuration $u \in \Omega_{U_i}$, the set X is partitioned into three subsets of parameters X^u_{var}, X^u_{co-var}, and X^u_{other}, where for each $x_j = p(v \mid \pi)$, $j = 0, \ldots, m-1$, we have that*

$$\begin{array}{ll} x_j \in X^u_{var} & \textit{if} \quad v \wedge \pi \wedge u = u \\ x_j \in X^u_{co-var} & \textit{if} \quad v \wedge u = \text{false} \ \textit{and} \ \pi \wedge u = u \\ X^u_{other} = X \setminus (X^u_{var} \cup X^u_{co-var}) & \end{array}$$

Then, the vector table ψ_{U_i} of U_i is defined, for each configuration $u \in \Omega_{U_i}$, by

$$\psi_{U_i}(u) = (c_0(u), \ldots, c_{2^m-1}(u))$$

where for each k with $k = \mathcal{N}(\epsilon_{m-1}, \ldots, \epsilon_0)$ we have that

$$c_k(u) = \begin{cases} 0 & \text{, if there is an } x_i \in X^u_{var} \text{ with } \epsilon_i = 0 \\ & \quad \text{or an } x_i \in X^u_{other} \text{ with } \epsilon_i = 1 \\ (-1)^{D(k)} \cdot \dfrac{\phi_{U_i}(u)}{(\prod\limits_{x_j \in X^u_{var}} p_j) \cdot (\prod\limits_{x_l \in X^u_{co-var}} (1 - p_l))} & \text{, otherwise} \end{cases}$$

where $D(k) = \sum\limits_{j : x_j \in X^u_{co-var}} \epsilon_j$.

We recall that the coefficients in a vector $\psi_{Ui}(u)$ specify the multilinear relation of the potential function value $\phi_{U_i}(u)$ in terms of the parameters $x_0, \ldots, x_{m-1}$. Informally speaking Definition 4.3 now states that a coefficient $c_k(u)$ in $\psi_{U_i}(u)$ equals zero if it pertains to a subset of parameters that cannot co-occur in the configuration u of U_i. As the configuration u matches all the parameters from X^u_{var}, any coefficient indicating a subset of parameters that does not contain all the parameters from X^u_{var} equals zero. Similarly, as configuration u does *not* match any parameter from X^u_{other}, any coefficient for a subset of parameters that does include a parameter from X^u_{other} equals zero. The remaining coefficients are obtained from the potential function value $\phi_{U_i}(u)$ by factorizing out the parameter values of the parameters under study that concord with u. Since the values of all parameters from X^u_{var} are contained in $\phi_{U_i}(u)$, these parameter values are factorized out by simply dividing $\phi_{U_i}(u)$ by $\prod_{x_j \in X^u_{var}} p_j$. For each parameter from X^u_{co-var}, the parameter value of a co-varying parameter is contained in $\phi_{U_i}(u)$. Recall from Section 3 that for a parameter x_l taking value p_l, the value p'_l of a specific co-varying parameter equals $p'_l(x) = p'_l \cdot \frac{1-x}{1-p} = f_l \cdot (1-x)$. To factorize out of $\phi_{U_i}(u)$ the parameter values for parameters x_l from X^u_{co-var}, we thus divide by $\prod_{x_l \in X^u_{co-var}} (1-p_l)$. Note that in doing this, the fraction f_l for the co-varying parameter of $x_l \in X^u_{co-var}$ matching u is contained in $c_k(u)$. The sign of a coefficient c_k depends on the number of parameters from X^u_{co-var} in the subset of parameters to which it pertains.

In the following proposition, we will show that ψ_{U_i} in essence is equivalent to ϕ_{U_i}. Filling in, for a specific configuration $u \in \Omega_{U_i}$, the *values* of the parameters under study residing in U_i in the multilinear relation represented by $\psi_{U_i}(u)$ yields the potential function value $\phi_{U_i}(u)$. Therefore, ϕ_{U_i} and $\psi_{U_i}(u)$ basically give the same information.

Proposition 4.4 *Let T be a junction tree and let U_i be a clique in T. Let ϕ_{U_i} be U_i's local potential function. Let $X = \{x_0, \ldots, x_{m-1}\}$ be the set of parameters under study that are located in U_i, $m \geq 1$; for each parameter x_j, $j = 0, \ldots, m-1$, let p_j be the value for x_j as specified in T. Let $\psi_{U_i}(U_i) = (c_0(U_i), \ldots, c_{2^m-1}(U_i))$ be U_i's vector table as defined above. Then, for each configuration $u \in \Omega_{U_i}$, we have that*

$$\phi_{U_i}(u) = \sum_{k:k=\mathcal{N}(\epsilon_{m-1},\ldots,\epsilon_0)} \left(c_k(u) \cdot \prod_{j=0,\ldots,m-1} p_j^{\epsilon_j} \right)$$

Proof. Without loss of generality, we assume that the subsets X^u_{var}, X^u_{co-var}, and X^u_{other} of X equal $\{x_0, \ldots x_{f-1}\}$, $\{x_f, \ldots x_{h-1}\}$, and $\{x_h, \ldots x_{m-1}\}$, for some $f, h \geq 1$, respectively. For each configuration $u \in \Omega_{U_i}$, we use the notation $f_{U_i}(u)(x_0, \ldots, x_{m-1})$ to indicate the multilinear function in of $x_0, \ldots, x_{m-1}$ specified by $\psi_{U_i}(u)$; that is,

$$f_{U_i}(u)(x_0, \ldots, x_{m-1}) = \sum_{k:k=\mathcal{N}(\epsilon_{m-1},\ldots,\epsilon_0)} \left(c_k(u) \cdot \prod_{j=0,\ldots,m-1} x_j^{\epsilon_j} \right)$$

We will now show that $f_{U_i}(u)(p_0, \dots, p_{m-1}) = \phi_{U_i}(u)$. By substituting in the function $f_{U_i}(u)$ every coefficient $c_k(u)$ with the expression for $c_k(u)$ as defined in Definition 4.3 and subsequently filling in the values p_j for the parameters x_j, $j = 0, \dots, m-1$, we find

$$f_{U_i}(u)(p_0, \dots, p_{m-1}) =$$

$$= \sum_{\substack{k \,:\, k = \mathcal{N}(\epsilon_{m-1}, \dots, \epsilon_0), \\ \epsilon_0, \dots, \epsilon_{f-1} = 1, \\ \epsilon_h, \dots, \epsilon_{m-1} = 0}} \left((-1)^{D(k)} \cdot \frac{\phi_{U_i}(u)}{(\prod_{x_r \in X^u_{var}} p_r) \cdot (\prod_{x_s \in X^u_{co-var}} (1 - p_s))} \cdot \prod_{j=0,\dots,m-1} p_j^{\epsilon_j} \right)$$

where $D(k) = \sum_{j : x_j \in X^u_{co-var}} \epsilon_j = \epsilon_f + \cdots + \epsilon_{h-1}$.

Now, since the exponents $\epsilon_h, \dots, \epsilon_{m-1}$ equal zero, the product $\prod_{j=h,\dots,m-1} p_j^{\epsilon_j}$ equals one. Multiplication with one has no effect, so this product can be left out from the expression for $f_{U_i}(u)(p_0, \dots, p_{m-1})$. Substituting, furthermore, X^u_{var} and X^u_{co-var} with $\{x_0, \dots x_{f-1}\}$, and $\{x_f, \dots x_{h-1}\}$, respectively, gives

$$f_{U_i}(u)(p_0, \dots, p_{m-1}) =$$

$$= \sum_{\substack{k \,:\, k = \mathcal{N}(\epsilon_{h-1}, \dots, \epsilon_0), \\ \epsilon_0, \dots, \epsilon_{f-1} = 1}} \left((-1)^{(\epsilon_f + \cdots + \epsilon_{h-1})} \cdot \frac{\phi_{U_i}(u)}{(\prod_{r=0,\dots,f-1} p_r) \cdot (\prod_{s=f,\dots,h-1} (1 - p_s))} \cdot \prod_{j=0,\dots,h-1} p_j^{\epsilon_j} \right)$$

Since the exponents $\epsilon_0, \dots, \epsilon_{f-1}$ equal one, we have that the product $\prod_{j=0,\dots,f-1} p_j^{\epsilon_j}$ equals the product $\prod_{j=0,\dots,f-1} p_j$ from the denominator of the equation above. So,

$$f_{U_i}(u)(p_0, \dots, p_{m-1}) =$$

$$= \left(\sum_{k : k = \mathcal{N}(\epsilon_{h-1}, \dots, \epsilon_f)} \left((-1)^{(\epsilon_f + \cdots + \epsilon_{h-1})} \cdot \frac{\phi_{U_i}(u)}{(\prod_{s=f,\dots,h-1} (1 - p_s))} \cdot \prod_{j=f,\dots,h-1} p_j^{\epsilon_j} \right) \right) =$$

$$= \frac{\phi_{U_i}(u)}{(\prod_{s=f,\dots,h-1} (1 - p_s))} \cdot \left(\sum_{k : k = \mathcal{N}(\epsilon_{h-1}, \dots, \epsilon_f)} \left((-1)^{(\epsilon_f + \cdots + \epsilon_{h-1})} \cdot \prod_{j=f,\dots,h-1} p_j^{\epsilon_j} \right) \right)$$

We focus attention on the term $\sum_{k : k = \mathcal{N}(\epsilon_{h-1}, \dots, \epsilon_f)} \left((-1)^{(\epsilon_f + \cdots + \epsilon_{h-1})} \cdot \prod_{j=f,\dots,h-1} p_j^{\epsilon_j} \right)$ and show that it equals $\prod_{s=f,\dots,h-1} (1 - p_s)$. The product $\prod_{j=f,\dots,h-1} (1 - p_s)$ can be written as a sum of products. Writing P for the set $\{p_f, \ldots, p_{h-1}\}$, each product in this sum is composed from parameter values from P and a constant which equals either one or minus one. The constant pertaining to a specific subset $P' \subseteq P$ equals one if the cardinality of P', denoted $\|P'\|$, is even; it equals minus one if $\|P'\|$ is odd. So, the constant equals $(-1)^{\|P'\|}$. As was introduced in Section 3, the product of a subset of parameter values P' can be written as a product of all parameter values in P where each $p_l \in P'$ is raised to the power one, $\epsilon_l = 1$, and each $p_l \notin P'$ is raised to the power zero, $\epsilon_l = 0$. Writing $\|P'\|$ in terms of the exponents, that is, $\|P'\| = \epsilon_f + \cdots + \epsilon_{h-1}$, we find

$$\prod_{l=f,\ldots,h-1} (1 - p_l) = \sum_{k:k=\mathcal{N}(\epsilon_{h-1},\ldots,\epsilon_f)} \left((-1)^{(\epsilon_f + \cdots + \epsilon_{h-1})} \cdot \prod_{j=f,\ldots,h-1} p_j^{\epsilon_j} \right)$$

We conclude that

$$f_{U_i}(u)(p_0, \ldots, p_{m-1}) = \phi_{U_i}(u)$$

□

In the proof of Proposition 4.4, we have shown that, for each configuration $u \in \Omega_{U_i}$, the vector table $\psi_{U_i}(u)$ for clique U_i contains the coefficients describing $\phi_{U_i}(u)$ in terms of the parameters under study in U_i. During propagation, the vector table ψ_{U_i} is multiplied by various other vector tables. Before proceeding, we formally define the multiplication of vector tables, called the *vector product*; the vector product was informally introduced in Example 4.1.

Definition 4.5 *Let W_i, $i = 1, \ldots, t$, $t \geq 1$, be a set of variables and let $\psi_i(W_i)$, $i = 1, \ldots, t$ be a vector table over the variables in W_i. The vector product $\prod^{*}_{i=1,\ldots,t} \psi_i(W_i)$ of these vector tables equals*

$$\prod^{*}_{i=1,\ldots,t} \psi_i(W_i) = \big(\cdots \big(\big(\psi_1(W_1) * \psi_2(W_2) \big) * \psi_3(W_3) \big) * \cdots * \psi_t(W_t) \big)$$

where for every two vector tables $\psi_i(W_i)$ and $\psi_j(W_j)$ with $\psi_i(W_i) = (c_{i_0}(W_i), \ldots, c_{i_k}(W_i))$ and $\psi_j(W_j) = (c_{j_0}(W_j), \ldots, c_{j_l}(W_j))$, for every configuration $w_i \in \Omega_{W_i}$ and $w_j \in \Omega_{W_j}$ the *vector product* $\psi_i(w_i) * \psi_j(w_j)$ *equals*

$$\begin{aligned}
&\psi_i(w_i) * \psi_j(w_j) = \\
&= (c_{i_0}(w_i), \ldots, c_{i_k}(w_i)) * (c_{j_0}(w_j), \ldots, c_{j_l}(w_j)) \\
&= (c_{i_0}(w_i) \cdot c_{j_0}(w_j), \ldots, c_{i_0}(w_i) \cdot c_{j_l}(w_j), c_{i_1}(w_i) \cdot c_{j_0}(w_j), \ldots, c_{i_k}(w_i) \cdot c_{j_l}(w_j))
\end{aligned}$$

This vector product is used in multiplying vector table ψ_{U_i} for clique U_i with the incoming messages $\hat{\psi}_{S_{i_j} \to U_i}$ from cliques S_{i_j}, $j = 1, \ldots, t$ (conform `step` 4 of procedure `vector-propagate`). The resulting vector table ψ'_{U_i} basically contains the same information as the potential function ϕ'_{U_i} resulting from standard propagation; for each configuration $u \in \Omega_{U_i}$, $\psi'_{U_i}(u)$ contains precisely the coefficients in the multilinear function describing the potential function value $\phi'_{U_i}(u)$ in terms of the parameters under study in subtree T_{U_i}. Filling in the specified values for the parameters in T_{U_i} in the multilinear function represented by $\psi'_{U_i}(u)$ gives $\phi'_{U_i}(u)$. Below, this is detailed for a clique U_i receiving one incoming message.

Proposition 4.6 *Let T be a junction tree and let U_i be a clique in T. Let ϕ_{U_i} be U_i's local potential function. Let $X = \{x_0, \ldots, x_{m-1}\}$, $m \geq 1$, be the set of parameters under study that are located in T_{U_i}, ordered as in Definition 4.2; for each parameter x_j, $j = 0, \ldots, m-1$, let p_j be the value for x_j as specified in T. Let ψ_{U_i} be the vector table associated with ϕ_{U_i}. Let S be a neighbouring separator for U_i, let $\hat{\phi}_{S \to U_i}(S)$ be the message from U_i to S in standard propagation and let $\hat{\psi}_{S \to U_i}(S)$ be the vector table associated with $\hat{\phi}_{S \to U_i}(S)$. For each configuration $u \in \Omega_{U_i}$ and $s \in \Omega_S$ with $s \wedge u = u$, let $\phi'_{U_i}(u) = \phi_{U_i}(u) \cdot \hat{\phi}_{S \to U_i}(s)$ and let $\psi'_{U_i}(u) = \psi_{U_i}(u) * \hat{\psi}_{S \to U_i}(s)$ Then,*

$$\phi'_{U_i}(u) = \sum_{k : k = \mathcal{N}(\epsilon_{m-1}, \ldots, \epsilon_0)} \left(c'_k(u) \cdot \prod_{j=0,\ldots,m-1} p_j^{\epsilon_j} \right)$$

where

$$(c'_0(u), \ldots, c'_{2^m-1}(u)) = \psi'_{U_i}(u)$$

Proof. Without loss of generality, we assume the parameters under study in clique U_i to be $x_f, \ldots, x_{m-1}$ and the parameters in $T_{U_i} \setminus U_i$ to be $x_0, \ldots, x_{f-1}$. We assume that, for each configuration $u \in \Omega_{U_i}$, $\psi_{U_i}(u) = (c_0(u), \ldots, c_{2^{m-f}-1}(u))$ and, for each configuration $s \in \Omega_S$, $\hat{\psi}_{S \to U_i}(s) = (c_0(s), \ldots, c_{2^f-1}(s))$. Furthermore, for each configuration $u \in \Omega_{U_i}$, we use $f_{U_i}(u)(x_0, \dots, x_{m-1})$ to indicate the multilinear function in $x_0, \dots, x_{m-1}$ specified by $\psi'_{U_i}(u)$, that is,

$$f_{U_i}(u)(x_0, \dots, x_{m-1}) = \sum_{l:'l=\mathcal{N}(\epsilon_{m-1},\dots,\epsilon_0)} \left(c'_l(u) \cdot \prod_{j=0,\dots,m-1} x_j^{\epsilon_j} \right)$$

where $(c'_0(u), \dots, c'_{2^m-1}(u)) = \psi'_{U_i}(u)$. We will now show that $f_{U_i}(u)(p_0, \dots, p_{m-1}) = \phi'_{U_i}(u)$. From $\psi'_{U_i}(u) = \psi_{U_i}(u) * \hat{\psi}_{S \to U_i}(s)$ we find that

$$\begin{aligned} \psi'_{U_i}(u) &= \left(c'_0(u), \dots, c'_{2^m-1}(u)\right) = \\ &= \left(c_0(u), \dots, c_{2^{m-f}-1}(u)\right) * \left(c_0(s), \dots, c_{2^f-1}(s)\right) = \\ &= \left(c_0(u) \cdot c_0(s), \dots, c_0(u) \cdot c_{2^f-1}(s), c_1(u) \cdot c_0(s), \dots, c_{2^{m-f}-1}(u) \cdot c_{2^f-1}(s)\right) \end{aligned}$$

or alternatively,

$$c'_l(u) = c_k(u) \cdot c_{k'}(s)$$

where for each $k = \mathcal{N}(\epsilon_{m-1}, \dots, \epsilon_f)$ and $k' = \mathcal{N}(\epsilon_{f-1}, \dots, \epsilon_0)$, l equals $\mathcal{N}(\epsilon_{m-1}, \dots, \epsilon_0)$. The expression $f_{U_i}(u)(x_0, \dots, x_{m-1})$ can therefore be written as

$$f_{U_i}(u)(x_0, \dots, x_{m-1}) =$$

$$= \sum_{k:k=\mathcal{N}(\epsilon_{m-1},\dots,\epsilon_f)} \sum_{k':k'=\mathcal{N}(\epsilon_{f-1},\dots,\epsilon_0)} \left(c_k(u) \cdot c_{k'}(s) \cdot \Big(\prod_{j=f,\dots,m-1} x_j^{\epsilon_j} \Big) \cdot \Big(\prod_{l=0,\dots,f-1} x_l^{\epsilon_l} \Big) \right)$$

$$= \left(\sum_{k:k=\mathcal{N}(\epsilon_{m-1},\dots,\epsilon_f)} \left(c_k(u) \cdot \prod_{j=f,\dots,m-1} x_j^{\epsilon_j} \right) \right) \cdot \left(\sum_{k':k'=\mathcal{N}(\epsilon_{f-1},\dots,\epsilon_0)} \left(c_{k'}(s) \cdot \prod_{l=0,\dots,f-1} x_l^{\epsilon_l} \right) \right)$$

Filling in the value p_j for parameter x_j, $j = 0, \dots m-1$, we find for $f_{U_i}(u)(p_0, \dots, p_{m-1})$ that

$$f_{U_i}(u)(p_0, \dots, p_{m-1}) =$$

$$\left(\sum_{k:k=\mathcal{N}(\epsilon_{m-1},\dots,\epsilon_f)} \left(c_k(u) \cdot \prod_{j=f,\dots,m-1} p_j^{\epsilon_j} \right) \right) \cdot \left(\sum_{k':k'=\mathcal{N}(\epsilon_{f-1},\dots,\epsilon_0)} \left(c_{k'}(s) \cdot \prod_{l=0,\dots,f-1} p_l^{\epsilon_l} \right) \right)$$

From proposition 4.4, we know that the two factors of this multiplication are the two mulilinear relations expressing $\phi_{U_i}(u)$ and $\hat{\phi}_{S \to U_i}(s)$ in terms of the parameters $x_f, \dots, x_{m-1}$

and $x_0, \dots, x_{f-1}$, respectively. So,

$$\begin{aligned} f_{U_i}(u)(p_0, \dots, p_{m-1}) &= \phi_{U_i}(u) \cdot \hat{\phi}_{S \to U_i}(s) \\ &= \phi'_{U_i}(u) \end{aligned}$$

□

In Proposition 4.6, we have shown that for a clique U_i receiving an incoming message $\hat{\psi}_{S \to U_i}$, the vector product of U_i's vector table ψ_{U_i} with $\hat{\psi}_{S \to U_i}$, that is ψ'_{U_i}, gives for each configuration $u \in \Omega_{U_i}$ the coefficients describing ϕ_{U_i} in terms of the parameters under study in subtree T_{U_i}. Now, proposition 4.6 can be extended to apply to multiple incoming messages. For a clique U_i receiving the messages $\hat{\psi}_{S_{i_j} \to U_i}$, $j = 1, \dots t$, for each configuration $u \in \Omega_{U_i}$ and $s_j \in \Omega_{S_{i_j}}$, with $s_j \wedge u = u$ and $j = 1, \dots, t$, the vector product $\psi_{U_i}(u_i) * \boxast_{j=1,\dots,t} \hat{\psi}_{S_{i_j} \to U_i}(s_j)$ gives ψ'_{U_i} describing ϕ'_{U_i} in terms of the parameters in T_{U_i}. Since this extension is straightforward, we will not detail this any further.

In step 5 of the procedure `vector-propagation`, the vector table ψ'_{U_i} is projected onto the separator $S_{i_{t+1}}$ to which a message is to be sent. This step corresponds with step 3 of the standard propagation procedure. In standard propagation, the potential function ψ'_{U_i} is projected onto separator $S_{i_{t+1}}$ by marginalizing over all variables that are not contained in $S_{i_{t+1}}$. Similarly, ψ'_{U_i} is projected onto $S_{i_{t+1}}$ by marginalizing over all variables that are not in $S_{i_{t+1}}$. For ψ'_{U_i}, this amounts to adding the vectors of coefficients in the entries specifying the same configuration for $S_{i_{t+1}}$. Adding vectors of coefficients is done by straightforward vector summation. The result of projection onto $S_{i_{t+1}}$ of ϕ'_{U_i} and ψ'_{U_i} is the message $\hat{\phi}_{U_i \to S_{i_{t+1}}}(s)$ and $\hat{\psi}_{U_i \to S_{i_{t+1}}}(s)$, respectively. Below, we show that, for a specific configuration $s \in \Omega_{S_{i_{t+1}}}$, $\hat{\psi}_{U_i \to S_{i_{t+1}}}(s)$ contains the coefficients that describe $\hat{\phi}_{U_i \to S_{i_{t+1}}}(s)$ in terms of the parameters under study in T_{U_i}; filling in the specified values for the parameters in T_{U_i} in the multilinear function represented by $\hat{\psi}_{U_i \to S_{i_{t+1}}}(s)$ gives $\hat{\phi}_{U_i \to S_{i_{t+1}}}(s)$.

Proposition 4.7 *Let T be a junction tree and let U_i be a clique in T. Let $X = \{x_0, \dots, x_{m-1}\}$, $m \geq 1$, be the parameters under study in subtree T_{U_i} ordered as in Definition 4.2; for each parameter x_j, $j = 0, \dots, m-1$, let p_j be the value for x_j as specified in T. Let $\phi'_{U_i}(U_i)$ be a potential function for U_i and let $\psi'_{U_i}(U_i)$ be the vector table associated with $\phi'_{U_i}(U_i)$. Let*

S be a neighbouring separator of U_i and let $\hat{\phi}_{U_i \to S}(S) = \sum_{U_i \setminus S} \phi'_{U_i}(U_i)$ be the messages from U_i to S. Let $\hat{\psi}_{U_i \to S}(S)$ be $\sum_{U_i \setminus S} \psi'_{U_i}(U_i)$. Then, for each configuration $s \in \Omega_S$, we have that

$$\hat{\phi}_{U_i \to S}(s) = \sum_{k:k=\mathcal{N}(\epsilon_{m-1},\ldots,\epsilon_0)} \left(c_k(s) \cdot \prod_{j=0,\ldots,m-1} p_j^{\epsilon_j} \right)$$

where

$$\big(c_0(s), \ldots, c_{2^m-1}(s)\big) = \hat{\psi}_{U_i \to S}(s)$$

Proof. For each configuration $s \in \Omega_S$, we use $f_S(s)(x_0, \ldots, x_{m-1})$ to indicate the mult-linear function in terms of $x_0, \ldots, x_{m-1}$ specified by the coefficients in $\hat{\psi}_{U_i \to S}(s)$, that is

$$f_S(s)(x_0, \ldots, x_{m-1}) = \sum_{k:k=\mathcal{N}(\epsilon_{m-1},\ldots,\epsilon_0)} \left(c_k(s) \cdot \prod_{j=0,\ldots,m-1} x_j^{\epsilon_j} \right) \tag{2}$$

where $(c_0(s), \ldots, c_{2^m-1}(s)) = \hat{\psi}_{U_i \to S}(s)$. We will now show that $f_S(s)(p_0, \ldots, p_{m-1}) = \hat{\phi}_{U_i \to S}(s)$. By definition, the vector table $\hat{\psi}_{U_i \to S}(s)$ equals $\sum_{u' \in \Omega_{U_i \setminus S}} \psi'_{U_i}(u' \wedge s)$ Without loss of generality, we assume that $\psi'_{U_i}(u) = (c_0(u), \ldots, c_{2^m-1}(u))$. The coefficients $c_k(s)$, $k = 0, \ldots 2^m - 1$ can be written as

$$\begin{aligned} \big(c_0(s), \ldots, c_{2^m-1}(s)\big) &= \sum_{u' \in \Omega_{U_i \setminus S}} \big(c_0(u' \wedge s), \ldots, c_{2^m-1}(u' \wedge s)\big) \\ &= \left(\sum_{u' \in \Omega_{U_i \setminus S}} c_0(u' \wedge s), \ldots, \sum_{u' \in \Omega_{U_i \setminus S}} c_{2^m-1}(u' \wedge s) \right) \end{aligned}$$

that is

$$c_k(s) = \sum_{u' \in \Omega_{U_i \setminus S}} c_k(u' \wedge s)$$

for $k = 0, \ldots 2^m - 1$. Substituing this expressiion for $c_k(s)$ in Equation 2, we find for $f_S(s)(x_0, \ldots, x_{m-1})$ that

$$
\begin{aligned}
f_S(s)(x_0,\dots,x_{m-1}) &= \sum_{k:k=\mathcal{N}(\epsilon_{m-1},\dots,\epsilon_0)} \left(\left(\sum_{u'\in\Omega_{U_i\setminus S}} c_k(u'\wedge s) \right) \cdot \left(\prod_{j=0,\dots,m-1} x_j^{\epsilon_j} \right) \right) \\
&= \sum_{k:k=\mathcal{N}(\epsilon_{m-1},\dots,\epsilon_0)} \left(\sum_{u'\in\Omega_{U_i\setminus S}} \left(c_k(u'\wedge s) \cdot \prod_{j=0,\dots,m-1} x_j^{\epsilon_j} \right) \right) \\
&= \sum_{u'\in\Omega_{U_i\setminus S}} \left(\sum_{k:k=\mathcal{N}(\epsilon_{m-1},\dots,\epsilon_0)} \left(c_k(u'\wedge s) \cdot \prod_{j=0,\dots,m-1} x_j^{\epsilon_j} \right) \right)
\end{aligned}
$$

By definition, the multilinear relation between brackets is the expression of $\phi'_{U_i}(u)$ in terms of the parameters $x_0, \dots x_{m-1}$. Filling in the value p_j for parameter x_j, $j = 0, \dots m-1$, we find for $f_S(s)(p_0, \dots, p_{m-1})$

$$
\begin{aligned}
f_S(s)(p_0,\dots,p_{m-1}) &= \sum_{u'\in\Omega_{U_i\setminus S}} \phi'_{U_i}(u'\wedge s) \\
&= \hat{\phi}_{U_i\to S}(s)
\end{aligned}
$$

□

After one inward propagation of vector tables from the leaves to the root R of a junction tree T, at root R all incoming messages are available. Now, in step 7 of procedure `vector-propagation`, the vector table ψ_R of R is multiplied with these messages. This corresponds with step 5 in procedure `propagate`. For a specific configuration $r \in \Omega_R$, the resulting vector $\psi'_R(r)$ contains the coefficients that describe $\Pr(r, e)$ in terms of all parameters under study in T. Since this step is very similar to step 4 in procedure `vector-propagation`, we will omit a formal description of this multiplication.

Finally, in the last step, step 8, of procedure `vector-propagation`, all vectors in ψ'_R are added. This corresponds, in standard evidence propagation, with the computation of the probability $\Pr(e)$ from the marginal probability distribution $\Pr(R, e)$ over the variables in R. The result of the summation $\sum_{r\in\Omega_R} \psi'_R(r)$ in procedure `vector-propagation` thus is the vector of coefficients $(c_0, \dots, c_{2^n-1})$ describing $\Pr(e)$ in terms of all parameters $x_0, \dots, x_{n-1}$ under study in the junction tree. Technically, this last step is the same as step 5 in `vector-propagation`; it will not be further detailed.

5 Finding the optimal root

In the previous section, it is shown that the coefficients in an n-way sensitivity function expressing the probability of the evidence in terms of n of the network's parameters can be established by performing one inward propagation of vector tables towards any root. The location in the junction tree of the clique that is chosen as the root of procedure `vector-propagate` determines the number of computations that is required to obtain these coefficients. We call that clique the *optimal root*, for which the number of computations to be performed is minimized. The optimal root is determined with respect to a specific computation scheme for establishing the required coefficients. In Section 4, we have presented the scheme `vector-propagate`. More efficient computation schemes exist; in particular with respect to the order of multiplying incoming messages and marginalizing over variables, optimizations of the scheme `vector propagate` can be achieved. In Section 6, possibilities for optimization are discussed briefly. The root which is optimal for our computation scheme `vector-propagate` is not necessarily optimal for other computation schemes. In this section, we discuss the identification of the optimal root for procedure `vector-propagate`. We first introduce the basic idea; subsequently, the idea is illustrated using our running example.

For procedure `vector-propagate`, the number of computations that is required to obtain the coefficients in a sensitivity function from a root R in junction tree T equals the sum of the number of computations required for each step in this procedure. In establishing the total number of computations for a root R, we assume that initialization of the junction tree has already taken place; the number of computations required for the initialization is independent of the chosen root and is therefore not relevant in determining the optimal root. The total number of computations $N_R(T)$ required if R is the root of `vector-propagate` then is the sum of the number of computations performed at each clique U_i in `steps` 4 and 5 of procedure `vector-propagate` and the computations at root R in `steps` 7 and 8 of `vector-propagate`.

The number of computations $N(U_i)$ for a clique U_i consist of the number of multplications $M(U_i)$ required for multiplying ψ_{U_i} with the messages $\hat{\psi}_{S_{i_j} \to U_i}$, $j = 1, \ldots, t$, (`step` 4) and the number of additions $A(U_i)$ required for projecting ψ'_{U_i} onto $S_{i_{t+1}}$ (`step` 5). Both concern floating point operations, in the sequel called *flops*. From Example 4.1, it will be clear that the number of flops $N(U_i)$ depends on the size of clique U_i, denoted $sp(U_i)$, the number of parameters under study in U_i, denoted $n(U_i)$, and the number of parameters under study in the incoming vector tables $\hat{\psi}_{S_{i_j} \to U_i}$, denoted $n(S_{i_j}, U_i)$, $j = 1, \ldots, t$. This latter information is not locally available at clique U_i; separators S_{i_j}, $j = 1, \ldots, t$, provide this information. For every clique U_i in T, where $U_i \neq R$, $N(U_i)$ is determined. In a similar way, for root R the number of flops $N(R)$ is determined. $N(R)$ again depends on $sp(R)$, $n(R)$ and $n(S_{R_j}, R)$, $j = 1, \ldots, k$. The total number of flops $N_R(T)$ to obtain the required coefficients from R then equals $N(R) + \sum_{U_i \in T \setminus R} N(U_i)$.

To know the number of parameters under study $n(S_{R_j}, R)$, $j = 1, \ldots, k$. in the incoming vector tables of R, each neighbouring clique U_j of R should send the number of parameters under study in subtree T_{U_j} to R. This can be formulated recursively; each clique U_i in T sends the number of parameters under study in subtree T_{U_i}, denoted $n(U_i, S_{i_{t+1}})$, to $S_{i_{t+1}}$. As such, the computation of $N_R(T)$ itself can be formulated recursively, that is, the idea of propagation in a junction tree is again exploited. Starting from the leaves of T, each triggered clique U_i sends to $S_{i_{t+1}}$ the number of parameters under study in subtree T_{U_i} and the number of flops done so far. The number of parameters $n(U_i, S_{i_{t+1}})$ is obtained by adding $n(U_i)$ to the incoming numbers of parameters $n(S_{i_j}, U_i)$, $j = 1, \ldots, t$. The number of flops done so far, denoted $N(U_i, S_{i_{t+1}})$, is the sum of $N(U_i)$ and the incoming numbers of flops $N(S_{i_j}, U_i)$, $j = 1, \ldots, t$. In going from the leaves of T to R, now, the total number of flops $N_R(T)$ is accumulated. In pseudocode, the computation of $N_R(T)$ amount to the following.

```
procedure compute-flops (T, e, R, X)
begin
   until root R in T is triggered do
      for each triggered clique U_i do
step 1      compute N(U_i) = M(U_i) + A(U_i);
step 2      compute N(U_i, S_{i_{t+1}}) = N(U_i) + Σ_{j=1,...t} N(S_{i_j}, U_i);
step 3      compute n(U_i, S_{i_{t+1}}) = n(U_i) + Σ_{j=1,...t} n(S_{i_j}, U_i);
step 4      send N(U_i, S_{i_{t+1}}) and n(U_i, S_{i_{t+1}}) to S_{i_{t+1}}
      od
   od;
step 5   compute N(R) = M(R) + A(R);
step 6   compute N_R(T) = N(R) + Σ_{j=1,...k} N(S_{R_j}, R);
end.
```

By taking every clique in T as the root of procedure `compute-flops`, the clique that minimizes the number of computations to establish the coefficients in the required sensitivity function is found. This clique is the optimal root for procedure `vector propagate`. Note that, instead of repeating `compute-flops` for every clique in T, it is more efficient to extend the procedure such that $N_{U_i}(T)$ is computed for all cliques U_i simultaneously. To this end, the propagation of numbers of flops and numbers of parameters is not stopped when the root of `compute-flops` is reached, but continues until a full inward and outward propagation has been performed. Then, at each clique U_i in T the incoming numbers of flops and parameters are given and this clique can compute the total number of flops $N_{U_i}(T)$ required to establish all coefficients from U_i. Since the idea is relatively straightforward, it will not be detailed further. Below, we illustrate the computation of the optimal root for our running example.

Example 5.1 Consider again our running example in Figure 2. As in Example 4.1, we focus on the probability $\Pr(v_8)$ of the evidence $V_8 = v_8$ and we are interested in the sensitivity of $\Pr(v_8)$ to variations in the estimates for the three network parameters $x_0 = p(v_8 \mid v_6, v_7)$, $x_1 = p(v_7)$, and $x_2 = p(v_4 \mid \neg v_2)$. To that end, the coefficients in the multilinear function expressing $\Pr(v_8)$ in terms of x_0, x_1 and x_2 are determined from procedure `vector-propagate`. Before applying procedure `vector-propagate`, however, the optimal root to start the procedure is established from procedure `compute-flops`. First, we compute the number of flops if the required coefficients are established using clique U_4 as root of T; we perform `compute-flops` $(T, v_8, U_4, \{x_0, x_1, x_2\})$.

From the leaves of the junction tree, an inward propagation towards clique U_4 is performed. Each leaf U_i computes from its size and its number of parameters under study the number of flops required to compute the vector table it should send to its neighbouring separator. Leaf U_1 does not contain any parameter under study; $n(U_1)$ equals zero. The size of U_1, $sp(U_1)$, is the product of the number of values for the variables in U_1; $sp(U_1) = 2 \cdot 2 \cdot 2 = 8$. From $n(U_1)$ and $sp(U_1)$, it is known that vector table ψ_{U_1} consists of eight entries each containing one coefficient $(2^{n(U_1)})$. Projecting ψ_{U_1} onto S_1 amounts to marginalizing over variable V_1 with $sp(V_1) = 2$, giving the vector table $\hat{\psi}_{U_1 \to S_1}$ with $sp(S_1) = 4$. The number of additions $A(U_1)$ required for this operation equals four. Leaf U_1 thus send to separator S_1 the messages;

$$N(U_1, S_1) = A(U_1) = 4$$

and

$$n(U_1, S_1) = n(U_1) = 0$$

Similarly, clique U_2 sends to S_2 the information that

$$N(U_2, S_2) = A(U_1) = 2$$

and

$$n(U_2, S_2) = n(U_1) = 0$$

Leaf U_5 contains the two parameters x_0 and x_1. Since V_8 is observed, the size of clique U_5 is four. The vector table ψ_{U_5} thus consists of four entries each containing four coefficients. Projecting ψ_{U_5} onto separator S_4, with $sp(S_4) = 2$, then requires eight additions; $A(U_5) = 8$. We thus find

$$N(U_5, S_4) = A(U_5) = 8$$

and

$$n(U_5, S_4) = n(U_5) = 2$$

For a clique U_i that is not a leaf, the vector table ψ_{U_i} should be multiplied with the incoming vector tables before projection onto the separator to which a message is to be sent. The number of flops performed by U_i then is the sum of the number of multiplications $M(U_i)$ required for this first step and the number of additions $A(U_i)$ for the latter. Consider clique U_3 containing the parameter x_2; $n(U_3) = 1$. The size of clique U_3 is eight. The vector table ψ_{U_3} thus consist of eight entries each containing two $(2^{n(U_3)})$ coefficients. ψ_{U_3} is to be multiplied with the incoming vector table $\hat{\psi}_{S_1 \to U_3}$, which equals $\hat{\psi}_{U_1 \to S_1}$ and contains $n(U_1, S_1) = 0$ parameters under study. The two coefficients in each of the eight entries in ψ_{U_3} have to be multiplied with the one coefficient in the appropriate entry in $\hat{\psi}_{S_1 \to U_3}$. As such, $M(U_3)$ equals $8 \cdot 2 \cdot 1 = 16$. The resulting table ψ'_{U_3}, again consisting of eight entries with each two coefficients, is projected onto S_3, with $sp(S_3) = 4$. This requires eight additions; $A(U_3) = 8$. For the total number of local computations at clique U_3 we thus find

$$N(U_3) = M(U_3) + A(U_3) = 16 + 8 = 24$$

Adding this to the incoming number of flops $N(S_1, U_3) = N(U_1, S_1) = 4$ gives

$$N(U_3, S_3) = N(U_3) + N(S_1, U_3) = 28$$

The total number of parameters under study accumulated so far is

$$n(U_3, S_3) = n(U_3) + n(S_1, U_3) = 1$$

At clique U_4, the root of procedure `compute-flops`, now, for each neighbouring separator S_2, S_3, and S_4, the number of flops required to compute the incoming messages $\hat{\psi}_{S_2 \to U_4}$, $\hat{\psi}_{S_3 \to U_4}$, and $\hat{\psi}_{S_4 \to U_4}$ in procedure `vector-propagate` is available. Furthermore, U_4 receives the number of parameters $n(S_2, U_4)$, $n(S_3, U_4)$, and $n(S_4, U_4)$ in each of these messages. At root U_4 then the number of multiplications $M(U_4)$ required to multiply the vector table ψ_{U_4} with these incoming vector tables can be determined. We assume that ψ_{U_4} is multiplied with the incoming vector tables in order of increasing total number of parameters in each table. That is, we consider first multiplication with $\hat{\psi}_{S_2 \to U_4}$, then with $\hat{\psi}_{S_3 \to U_4}$, and finally with $\hat{\psi}_{S_4 \to U_4}$. Recall that $sp(U_4) = 8$ and $n(U_4) = 0$. The message $\hat{\psi}_{S_2 \to U_4}$ contains no parameters under study. As such, for each of the eight entries in ψ_{U_4}, each single coefficient is multiplied with the one coefficient in the appropriate entry in

$\hat{\psi}_{S_2 \to U_4}$. This requires $8 \cdot 1 \cdot 1 = 8$ multiplications. Message $\hat{\psi}_{S_3 \to U_4}$ contains one parameter under study, corresponding with two coefficients in each entry. Subsequent multiplication with this message thus requires $8 \cdot 1 \cdot 2 = 16$ multiplications. The vector table resulting from the multiplication so far has eight entries each containing two coefficients; it is multiplied with $\hat{\psi}_{S_4 \to U_4}$. This requires $8 \cdot 2 \cdot 4 = 64$ multiplications. In total thus $M(U_4) = 88$. After the multiplication of ψ_{U_4} with all incoming vector tables, again a vector table with eight entries results. Each entry now contains $2^{n(U_4)} \cdot 2^{n(S_2,U_4)} \cdot 2^{n(S_3,U_4)} \cdot 2^{n(S_4,U_4)} = 8$ coefficients. Marginalizing over all variables in U_4 thus required $(sp(U_4) - 1) \cdot 8 = 56$ additions; $A(U_4) = 56$. For root U_4 thus a total of $N(U_4) = M(U_4) + A(U_4) = 88 + 56 = 144$ flops are required. Adding this to the incoming number of flops gives

$$N_{U_4}(T) = N(U_4) + N(S_2, U_4) + N(S_3, U_4) + N(S_4, U_4) = 144 + 2 + 28 + 8 = 182$$

The total number of parameters under study in T evidently equals

$$n(T) = n(U_4) + +n(S_2, U_4) + n(S_3, U_4) + n(S_4, U_4) = 0 + 0 + 1 + 2 = 3$$

Now, procedure `compute-flops` has been carried out with clique U_4 as root.

By repeating the procedure for every clique in the junction tree T in Figure 2, for each clique U_i, $i = 1, \ldots, 5$ the number of flops $N_{U_i}(T)$ required to compute the coefficients in the three-way sensivity function of $\Pr(v_8)$ in terms of the parameters $x_0 = p(v_8 \mid v_6, v_7)$, $x_1 = p(v_7)$, and $x_2 = p(v_4 \mid \neg v_2)$ is obtained. As was noted earlier, instead of repeating `compute-flops` for every clique U_i in T, the number of flops can also be computed for all cliques simultaneously. This is done by performing a full (inward and outward) propagation of numbers of flops and numbers of parameters. In Figure 3, the result of such a full propagation is shown. At each clique, now, the number of flops required to compute all incoming me mssages is available. With this information each clique U_i computes the total number of flops $N_{U_i}(T)$ required to compute all coefficients in the three-way sensitivity function. In Table 1, the numbers $N_{U_i}(T)$, $i = 1, \ldots 5$, are shown .

Table 1 shows that clique U_5 requires the minimum number of flops for computing all coefficients in the three-way sensitivity function expressing $Pr(v_8)$ in terms of x_0, x_1, and x_2, when the computation scheme of procedure `vector-propagate` is used.

Now, we show that by optimizing procedure `vector-propagate` slightly, an other optimal root results. In procedure `vector-propagate`, the vector table ψ_{U_i} of U_i is first

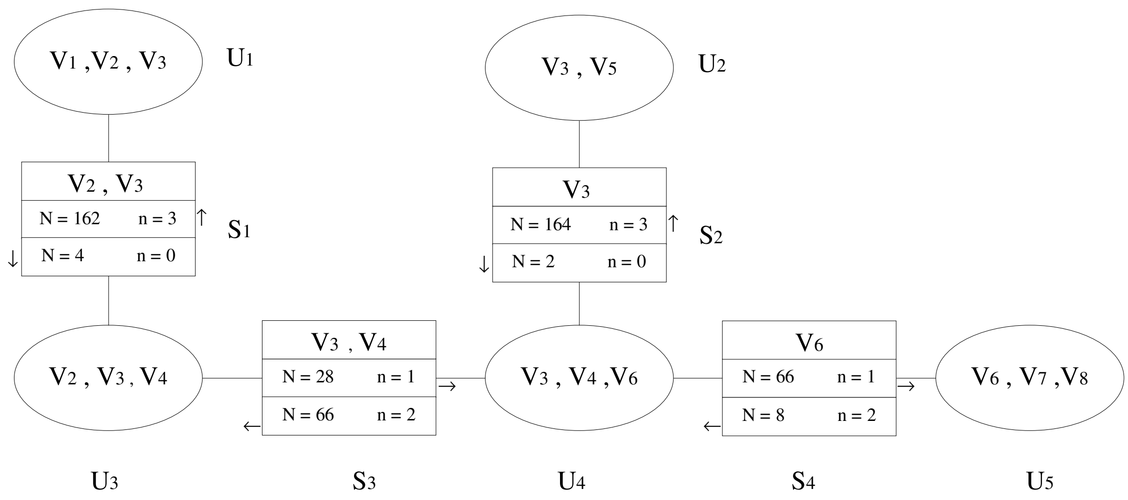

Figure 3: The example junction tree from Figure 2, in which one full propagation of numbers of floating point operations (flops) and numbers of parameters has been performed. At each clique, then, the number of flops required to compute each incoming vector table is available.

multiplied with all incoming messages before marginalization takes place. It is more efficient, however, to alternate the multiplication by incoming messages with marginalization over variables as soon as this is possible. For clique U_4, for example, it is possible to marginalize over variables V_3 and V_4 after ψ_{U_4} is multiplied with $\hat{\psi}_{S_2 \to U_4}$ and $\hat{\psi}_{S_3 \to U_4}$. Marginalization over these two variables can be carried out before multiplication with $\hat{\psi}_{S_4 \to U_4}$ since this message is not a function of these variables. Multiplication of ψ_{U_4} with $\hat{\psi}_{S_2 \to U_4}$ and $\hat{\psi}_{S_3 \to U_4}$ requires $M_1(U_4) = 24$ multiplications. Subsequent marginalization over V_3 and V_4 requires $A_1(U_4) = 12$ additions. Now, the resulting vector table with two entries each containing two coeffcients is multiplied with $\hat{\psi}_{S_4 \to U_4}$ and marginalized over V_6. This requires $M_2(U_4) = 16$ multiplications and $A_1(U_4) = 8$ additions. In alternating multiplications and additions at U_4 thus a total of $N'(U_4) = 60$ flops are required. In that case, $N'_{U_4}(T) = 98$ compared to $N'_{U_4}(T) = 182$ in Table 1. Assuming that multiplication by incoming messages and marginalization over variables is carried out alternatingly, we find for each clique U_i, $i = 1, \dots, 5$, the number of flops $N'_{U_i}(T)$ as given in Table 2. The minimum number of flops is now found for clique U_4. □

clique U_i	$N_{U_i}(T)$
U_1	282
U_2	220
U_3	206
U_4	182
U_5	122

Table 1: The number of floating point operations (flops) required to compute the coefficients in the three-way sensitivity function of $\Pr(v_8)$ in terms of the parameters $x_0 = p(v_8 \mid v_6, v_7)$, $x_1 = p(v_7)$, and $x_2 = p(v_4 \mid \neg v_2)$. The computation scheme used is procedure `vector-propagate`.

clique U_i	$N'_{U_i}(T)$
U_1	282
U_2	164
U_3	150
U_4	98
U_5	122

Table 2: The number of floating point operations (flops) required to compute the coefficients in the three-way sensitivity function of $\Pr(v_8)$ in terms of the parameters $x_0 = p(v_8 \mid v_6, v_7)$, $x_1 = p(v_7)$, and $x_2 = p(v_4 \mid \neg v_2)$. The computation scheme used is procedure `vector-propagate` in which order of multiplying incoming messages and marginalizing over variables is optimized.

6 Optimizations

In Section 4, we have presented the basic idea of computing the coefficients in a sensitivity function by propagating vector tables through the junction tree. In pseudocode, procedure `vector-propagate` presents a computation scheme for obtaining these coefficients. Although it reflects the basic idea, this scheme is not optimized with respect to computation time. In this section, we will discuss several possibilities for increasing the efficiency of the basic computation scheme. First, we will focus on efficient computation of the messages that are propagated through the junction tree. It concerns optimising step 4 and 5 in procedure `vector-propagate`. This was also briefly touched upon in Section 5. Subsequently, we discuss the acquirement of a factorized form of the sensitivity function under study. A factorized form of the sensitivity function can, for example, be obtained by omitting the computations at the root of the junction tree (step 7 and 8 in procedure `vector-propagate`). The number of computations required for a factorized sensitivity function is smaller then for the unfactorized form and the information contained in the sensivity function is equivalent. Finally, we focus on exploiting independence relations induced by evidence to reduce the number of computations. Firstly, if there is evidence that causes the junction tree to fall apart in two or more independent subgraphs, the sensitivity function is a product of the functions obtained from each subgraph separately. Secondly, evidence for separator variables may allow for postponing the multiplication with incoming vector tables. In the following, the above mentioned possibilities for improvement will be discussed briefly without going into technical details.

In procedure `vector-propagate`, the message of a clique U_i to separator $S_{i_{t+1}}$ is computed by first multiplying ψ_{U_i} with each incoming message $\hat{\psi}_{S_{i_j} \rightarrow U_i}$ for separator S_{i_j}, $j = 1, \ldots, t$, and subsequently projecting the resulting vector table onto $S_{i_{t+1}}$. In Section 5, it was already noted that first carrying out all multiplications and after this all marginalizations is not efficient. It is far more efficient to marginalize over a variable as soon as this is possible. We can marginalize over a variable when ψ_{U_i} has been multiplied with all incoming messages that are a function of this variable. In doing this, the total number of entries in a vector table that is stepwise multiplied with incoming messages is kept at a minimum, thereby reducing the number of multiplications required for subsequent incoming messages. Note that the idea of alternating multiplications and marginalizations is not limited to its use in propagating vector tables. It also increases the efficiency of standard propagation. This has been shown by [Shenoy, 1996]. To alternate multiplications and marginalizations in an efficient way, P. Shenoy proposes to construct from a junction tree a binary junction tree in which each clique receives at most two incoming messages.

A second way to increase the efficiency of computing message $\hat{\psi}_{U_i \to S_{i_{t+1}}}$ from U_i to $S_{i_{t+1}}$ is to order the incoming messages according to increasing total amount of their parameters; ψ_{U_i} is first multiplied with the incoming vector table that contains the smallest total number of parameters and for subsequent incoming vector tables this number increases. Using this constraint on the ordering of incoming vector tables decreases the number of multiplications in step 4 of procedure `vector-propagate`. Basically, incorporating this constraint means that the ordering of incoming messages according to descending number of their highest ordered parameter, as assumed in Section 4, corresponds with increasing total number of their parameters. In the parameter ordering o defined in Section 4, this additional constaint can be incorporated by enforcing that for each two neighbouring cliques of U, denoted U' and U'', for which hold that U lies on the path from U' and U'' to R and for which hold that X' and X'' denote the subsets of parameters contained in $T_{U'}$ and $T_{U''}$, respectively, we have that $\max\{o(x_i) \mid x_i \in X'\} > \max\{o(x_i) \mid x_i \in X''\}$ if $|X'| < |X''|$. Now, using this latter constraint in combination with the use of binary junction trees allows for computing $\hat{\psi}_{U_i \to S_{i_{t+1}}}$ efficiently.

Until now, we have focussed on computing the 2^n coefficients in an unfactorized n-way sensitivity function. Instead of computing these 2^n coefficients, it is also possible to obtain the coefficients for a factorized form of the sensitivity function. First, we discuss the possibility to establish a factorized n-way sensitivity function from the root of procedure `vector-propagate`. Basically, a factorized n-way sensitivity function at root R is obtained by omitting any computations at R; as such, the vector table ψ_R and the incoming messages $\hat{\psi}_{S_{R_j} \to U_i}$, $j = 1.\ldots, k$, are kept as they are. The factors in the factorized n-way sensitivity function are the multilinear relations established by the coefficient in each entry of these vector tables. The sensitivity function is a sum of products of the appropriate factors. We will illustrate this using Example 4.1 in Section 4.

Example 6.1 Consider again the three-way sensitivity analysis discussed in Example 4.1. We are interested in the sensitivity of the probability of the evidence $\Pr(v_8)$ to variations in the estimates for the three network parameters $x_0 = p(v_8 \mid v_6, v_7)$, $x_1 = p(v_7)$, and

$x_2 = p(v_4 \mid \neg v_2)$. To that end, in Example 4.1, the coefficients in the unfactorized multilinear function expressing $\Pr(v_8)$ in terms of x_0, x_1 and x_2 are determined from root U_4. Here, we focus on the factorized form of the sensitivity function that is built from the multilinear functions established by the coefficients in vector table ψ_{U_4} and the incoming vector tables $\hat{\psi}_{S_2\to U_4}$, $\hat{\psi}_{S_3\to U_4}$, and $\hat{\psi}_{S_4\to U_4}$. For each configuration $u \in \Omega_{U_4}$, the multilinear function expressing the probability $\Pr(u, v_8)$ in terms of x_0, x_1, and x_2 is the product of the multilinear functions established by the coefficients in $\psi_{U_4}(u)$, $\hat{\psi}_{S_2\to U_4}(s)$, $\hat{\psi}_{S_3\to U_4}(s')$ and $\hat{\psi}_{S_3\to U_4}(s'')$ where $u \wedge s \wedge s' \wedge s'' = u$. Using the vector tables given in Example 4.1, we find for $u = v_3, v_4, v_6$

$$\Pr(v_3, v_4, v_6, v_8) = 0.9 \cdot (0.126 + 0.15x_2) \cdot (0.5 - 0.5x_1 + x_0x_1)$$

Adding the factorized expressions of $\Pr(u, v_8)$ for all $u \in \Omega_{U_4}$, we find the factorized multilinear function expressing $\Pr(v_8)$ in terms of x_0, x_1, and x_2;

$$\begin{aligned}
\Pr(v_8) \quad = \quad & \Pr(v_3, v_4, v_6, v_8) + \Pr(\neg v_3, v_4, v_6, v_8) + \Pr(v_3, \neg v_4, v_6, v_8) + \\
& + \Pr(\neg v_3, \neg v_4, v_6, v_8) + \Pr(v_3, v_4, \neg v_6, v_8) + \Pr(\neg v_3, v_4, \neg v_6, v_8) + \\
& + \Pr(v_3, \neg v_4, \neg v_6, v_8) + \Pr(\neg v_3, \neg v_4, \neg v_6, v_8) \\
= \quad & 0.9 \cdot (0.126 + 0.15x_2) \cdot (0.5 - 0.5x_1 + x_0x_1) + \\
& +0.8 \cdot (0.036 + 0.04x_2) \cdot (0.5 - 0.5x_1 + x_0x_1) + \\
& +0.8 \cdot (0.654 - 0.15x_2) \cdot (0.5 - 0.5x_1 + x_0x_1) + \\
& +0.3 \cdot (0.184 - 0.04x_2) \cdot (0.5 - 0.5x_1 + x_0x_1) + \\
& +0.1 \cdot (0.126 + 0.15x_2) \cdot (0.6 - 0.1x_1) + \\
& +0.2 \cdot (0.036 + 0.04x_2) \cdot (0.6 - 0.1x_1) + \\
& +0.2 \cdot (0.654 - 0.15x_2) \cdot (0.6 - 0.1x_1) + \\
& +0.7 \cdot (0.184 - 0.04x_2) \cdot (0.6 - 0.1x_1)
\end{aligned}$$

□

From the example, we see that a factorized form of our three-way sensitivity function contains more coefficients than the unfactorized three-way sensitivity function. However, for a larger number of parameters under study, in general, the factorized form of the n-way sensitivity function will contain less coeffcients. Moreover, the number of computations required to obtain the factorized form is smaller without any loss of information.

In general, for a factorized sensitivity function to be useful for interpretation it should not contain too many different factors. This can be established by taking as a root a clique with a limited number of incoming messages and with a limited size. Another possibility is to take a sufficiently small separator as root. In that case, we perform an inward propagation to this separator and compose the factorized sensitivity function from the messages coming from both directions. We again illustrate this with the sensitivity analysis presented in Example 4.1.

Example 6.2 Consider again the three-way sensitivity analysis of $\Pr(v_8)$ with respect to variations in the estimates for the three network parameters $x_0 = p(v_8 \mid v_6, v_7)$, $x_1 = p(v_7)$, , and $x_2 = p(v_4 \mid \neg v_2)$, as discussed in Example 4.1. In Example 4.1 an inward propagation to U_4 is performed. With the incoming messages $\hat{\psi}_{S_2 \to U_4}$, and $\hat{\psi}_{S_3 \to U_4}$ at U_4, this clique can compute its message $\hat{\psi}_{U_4 \to S_4}$ to separator S_4. This is done by marginalizing the product $\psi'_{U_4} = (\psi_{U_4} * \hat{\psi}_{S_2 \to U_4}) * \hat{\psi}_{S_3 \to U_4}$ over the variables V_3 and V_4. From the vector table ψ'_{U_4} given in Example 4.1, it can be seen that $\hat{\psi}_{U_4 \to S_4}$ equals

$V_6 = v_6$	$(0.7206, 0.035)$
$V_6 = \neg v_6$	$(0.2794, -0.035)$

Now, from this message and the message $\hat{\psi}_{U_5 \to S_4}$, at S_4 a factorized form of the three-way sensitivity function expressing $\Pr(v_8)$ in terms of x_0, x_1 and x_2 can be composed;

$$\begin{aligned}
\Pr(v_8) &= \Pr(v_6, v_8) + \Pr(\neg v_6, v_8) \\
&= (0.7206 + 0.035x_2) \cdot (0.5 - 0.5x_1 + x_0x_1) + \\
&\quad +(0.2794 - 0.035x_2) \cdot (0.6 - 0.1x_1)
\end{aligned}$$

□

Obtaining a factorized form of the sensitivity function by omitting multiplication with incoming messages, as was done above at the root of a junction tree, can be used recursively. That is, recursively the junction tree is split in subgraphs each containing their own root at which a factorized function of the parameters in that subgraph is computed. Further research is required to establish the benefits of this approach. We expect that this approach saves a considerable number of computations.

Finally, we discuss the use of independence relations induced by evidence to improve the efficiency of computing an n-way sensitivity function. By entering evidence in the junction tree, it may occur that several subgraphs of the junction tree become independent.

By independence of two subgraphs H_1 and H_2, we mean that the probability distribution over variables in H_1 is independent of any of the variable values or parameter values in H_2, and the other way around. A junction tree falls apart in several independent subgraphs when all variables in one or more separators are observed. Now, the n-way sensitivity function is a product of the sensitivity functions in these independent subgraphs. Since propagation of vector tables can now be restricted to a set of smaller subgraphs, the total number of computations needed to obtain all coefficients is much smaller than by applying procedure `vector-propagate` on the whole junction tree.

Evidence in the junction tree may also be useful to postpone or avoid multiplications. Carrying out the multiplication of a vector table associated with clique U by an incoming message is only needed if this message is a function of a variable over which we have to marginalize in order to compute the message of clique U to the next separator. Otherwise, the multiplication can be postponed. In that case, the message from U to the next separator is not a single vector table but a set of vector tables. This idea is known as *lazy evaluation* and is presented in detail in [Madsen & Jensen, 1998, Madsen, 1999]. Although postponing the multiplication of vector tables leads to the propagation of a set of vector tables, it saves computations, mainly by keeping the lengths of the vectors of coefficients in each vector table at a minimum.

7 Discussion and conclusions

In building Bayesian networks, sensitivity analysis plays an important role, as it allows for investigating the robustness of the output of the network for one or more network parameters. Until now, however, techniques for performing sensitivity analysis of a Bayesian network are computationally expensive. This fact is the basic motivation for the research presented here.

This paper describes a method for elegantly and efficiently computing the coefficients in an n-way sensitivity function expressing a prior probability in the network in terms of n network parameters under study. Our method exploits the computational architecture of a junction tree derived from the Bayesian network under consideration. The idea behind our method is closely related to standard propagation algorithms. In standard propagation, messages taking the form of potential functions are propagated through the junction tree. A potential function value is a single numerical number. In our method, vector tables are propagated through the junction tree. With each potential function, in standard propagation a vector table in our method is associated. For a specific value of this potential function, the corersponding, the corresponding vector table contains the coefficients expressing this potential function value in terms of the parameters under study in the junction tree. By performing a single inward propagation of vector tables towards a chosen root in the junction tree, all coefficients required for the sensitivity function of interest are accumulated.

Note, that our method actually closely resembles symbolic propagation in a junction tree. However, due to a specific encoding of the symbols, that is, the parameters under study, we do not need to propagate the symbols themselves. Our method provides for performing symbolic propagation using numerical propagation algorithms only. In the past, symbolic propagation has also been used to perform sensitivity analysis of a belief network [Chang & Fung, 1995, Castillo *et al.*, 1997]. Those approaches, however, required the use of software packages for symbolic manipulations, which are much slower then software packages for performing numerical manipulations. Therefore, our method is much more efficient than these approaches.

In [Castillo *et al.*, 1997], E. Castillo et al. show that the coefficients in an n-way sensitivity function can be obtained by evaluating a Bayesian network once for every coefficient to be determined, each time assuming a different combination of values for the set of parameters under study. A total of 2^n network evaluations are required. The result of these evaluations is a system of multilinear equations which is to be solved to give the required coefficients. The order in which the parameter under study are varied determines the number of flops for this method. The optimal order of varying the values of the parameters under study is the one that minimizes the total number of flops. The optimal order depends on the location of the root of the junction tree, the location of the parameters under study compared to the root, and the size of the cliques and separators on the path from each parameter to the root. When a parameter value is varied, this change needs to be propagated only through the relevant subgraph of the junction tree. Therefore, the order in which parameter values are changed is essential. Now, if the optimal order is chosen for changing the values of the parameters under study, the method proposed by Castillo et al. performs equally well as our method using the optimal root of the junction tree (see Section 5). The reason for this is that our method automatically takes into account the effect of the parameters under study in the optimal order. The substantial advantage of the method presented in this paper is that it provides a framework which easily allows for incorporating further optimizations in the computation of the coefficients. In Section 6, several possibilities for optimizations are presented. For example, the independencies induced by evidence can be exploited to postpone or omit the multiplication of incoming messages. Furthermore, efficiency is increased by computing a factorized form of the sensitivity function.

In this paper, we have focussed on the multilinear relation of a prior probability in a Bayesian network in terms of a number of network parameters under study. This can be easily extended to obtain the function describing a posterior probability in terms of the network parameters under study. From the definition of a conditional probability, we know that any posterior probability can be written as a quotient of two prior probabilities. Therefore, basically, our method is carried out twice. Note, that the second run of the method will generally be more efficient than the first run, since the propagation has to be carried out only in those parts of the junction tree in which new evidence is entered or evidence is removed.

Until now, our method has not been extensively tested. Therefore, we do not yet have a clear picture of the strengths and weaknesses of the method. For this purpose, experimental investigation is required which will be undertaken in the near future. We have shown, however, that our method has the potential to be a valuable tool in performing sensitivity analysis of a Bayesian network.

References

[Andreassen *et al.*, 1987] S. Andreassen, M. Woldbye, B. Falck, and S.K. Andersen (1987). MUNIN – A causal probabilistic network for interpretation of electromyographic findings. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 366 – 372.

[Castillo *et al.*, 1995] E. Castillo, J.M. Gutiérrez, and A.S. Hadi (1997). Parametric Structure of Probabilities in Bayesian Networks. *Lecture Notes in Artificial Intelligence:*

Symbolic and Quantitative Approaches to Reasoning and Uncertainty, C. Froidevaux and J. Kohlas, Eds. New York: Springer-Verlag, vol. 946, pp. 89 – 98.

[Castillo *et al.*, 1997] E. Castillo, J.M. Gutiérrez, and A.S. Hadi (1997). Sensitivity analysis in Discrete Bayesian Networks. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, pp. 412 – 423.

[Chang & Fung, 1995] K. Chang and R. Fung (1995). Symbolic Probabilistic Inference with Both Discrete and Continuous Variables. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, pp. 910 – 916.

[Coupé and Van der Gaag, 1998] V.M.H. Coupé and L.C. van der Gaag (1998). Properties of sensitivity analysis of Bayesian belief networks, submitted for publication.

[Dippel *et al.*, 1992] D.W.J. Dippel, J.W.M. ter Berg, and J.D.F. Habbema (1992). Screening for unruptured familial intracranial aneurysms. A decision analysis. *Acta Neurologica Scandinavica*, vol. 86, pp. 381 – 389.

[Golumbic, 1980] M.C. Golumbic (1980). *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York.

[Habbema *et al.*, 1990] J.D.F. Habbema, P.M.M. Bossuyt, and D.W.J. Dippel, Analysing clinical decision analyses, *Statistics in Medicine 9* (1990) 1229–1242.

[Henrion *et al.*, 1996] M. Henrion, M. Pradhan, B. Del Favero, K. Huang, G. Provan, and P. O'Rorke (1996). Why is diagnosis using belief networks insensitive to imprecision in probabilities ? *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 307 – 314.

[Jensen *et al.*, 1990] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, vol. 4, pp. 269 – 282.

[Jensen, 1996] F.V. Jensen (1996). *An Introduction to Bayesian Networks*, UCL Press, London.

[Laskey, 1995] K.B. Laskey (1995). Sensitivity analysis for probability assessments in Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, pp. 901 – 909.

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

[Madsen & Jensen, 1998] A.L. Madsen and F.V. Jensen (1998). Lazy propagation in junction trees *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 362 – 369.

[Madsen, 1999] A.L. Madsen (1999). All good things come to those who are lazy; efficient inference in Bayesian networks and influence diagrams based on lazy evaluation. Ph. D. thesis, Department of Computer Science, Aalborg University.

[Morgan & Henrion, 1990] M.G. Morgan and M. Henrion (1990). *Uncertainty, a Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis.* Cambridge University Press, Cambridge.

[Pearl, 1988] J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference.* Morgan Kaufmann, Palo Alto.

[Pradhan *et al.*, 1996] M. Pradhan, M. Henrion, G. Provan, B. Del Favero, and K. Huang (1996). The sensitivity of belief networks to imprecise probabilities: an experimental investigation. *Artificial Intelligence*, vol. 85, pp. 363 – 397.

[Shachter, 1986] R.D. Shachter (1986). Evaluating influence diagrams, *Operations Research*, vol. 34, pp. 79 – 90.

[Shafer & Shenoy, 1988] G. Shafer, and P.P. Shenoy (1989). Probability propagation. *Annals of Mathematics and Artificial Intelligence*,

[Shenoy, 1996] P.P. Shenoy (1996). Binary join tress *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 492 – 499.

[Tessem, 1992] B. Tessem (1992). Interval Probability Propagation. *Int. J. Approximate Reasoning*, vol. 7, pp. 95 – 120.

[Van der Gaag & Kjærulff, 1999] L.C. van der Gaag, and U. Kjærulff (1999). Making sensitivity analysis computationally efficient, in preparation.

[Von Winterfeldt & Edwards, 1986] D. von Winterfeldt and W. Edwards, *Decision Analysis and Behavioral Research* (Cambridge University Press, Cambridge, 1986).

[Wolfram, 1996] S. Wolfram, (1996). *The Mathematica Book.* 3rd edition Wolfram Media, Cambridge University Press, Cambridge.