Control and Cybernetics

vol. **29** (2000) No. 4

An algorithm for finding most likely explanations in valuation based systems

by

Sławomir T. Wierzchoń[1,2] and Mieczysław A. Kłopotek[1,3,4]

[1]Institute of Computer Science, Polish Academy of Sciences
[2]Dept of Computer Science, Białystok University of Technology
[3]Dept. of Computer Science, University of Podlasie
[4]Institute of Mathematics, Warsaw University of Technology
e-mail: {stw,klopotek}@ipipan.waw.pl

Abstract: A method for finding a number of best explanations in so-called valuation based system is presented. Roughly speaking, the method allows to sort (decreasingly or increasingly) a function of many variables without explicit computation of values of this function. The only condition is that the function be decomposable, i.e. can be expressed as a combination of a number of low-dimensional functions called components. Two cases are considered: the combination operator has an inverse and a more elaborated case when the combination operator has no inverse.

Keywords: probabilistic reasoning, bayesian networks, valuation based systems, most probable configurations.

1. Introduction

The idea of graphical expert systems, called also bayesian networks (BN) was initiated by Pearl (1986), and worked-out by Lauritzen and Spiegelhalter (1988). Its generalization to different uncertainty formalisms was proposed by Shenoy (1989) under the name of valuation based systems (VBS in short).

In the probabilistic context knowledge about interrelationships among variables from a fixed set X is represented on the qualitative level by a directed acyclic graph G and on the quantitative level by a set of conditional probabilities (consult, e.g., Pearl, 1986, for a rigorous definition).

Finding $i \geq 1$ probable explanations in a BN is equivalent to an optimization problem of identifying a set of configurations $x^1, \ldots, x^i$ such that $P(x^1) = \max_{x \in X} \geq P(x^2) = \max_{x \in X - \{x^1\}} \cdots \geq P(x^i) = \max_{x \in X - \{x^1, x^2, \ldots, n^{i-1}\}}$, where P is the joint probability distribution represented by the BN. We call x^1 the most probable explanation (MPE), and $x^2, \ldots, x^i$—probable explanations or probable configurations. Partial ordering of the probable configurations is im-

assessment of certain design methodologies, Sy (1989), Keramaris and Sy (1990). Its importance in such formalism like dynamical programming (a special case of VBS) is obvious—cf. Wierzchoń (1994).

The problem of finding the total ordering of configurations is NP-hard, Yen (1990), Pal et al. (1992). Also the problem of finding several most probable configurations is generally NP-hard, Cooper (1990). Previously, two general approaches were proposed to handle the complexity of the task. One is to restrict the type of the network to be dealt with to the so-called singly connected network, Sy (1993), Henrion (1990), or bipartite graph, Wu (1990). Another approach is to shift the complexity to spatial domain in order to keep the computational complexity in a linear order. For example, Shimony and Charniak (1990) convert a Bayesian belief network into a Weight Boolean Function Directed Acyclic Graph (WBFDAG) which permits the use of best-first search strategy. This method maintains a linear time with respect to the size of the graph, but the number of nodes in a WBFDAG could be exponential as compared to the original network. Several algorithms have been also elaborated for the special task of finding only MPE, based on integer programming, and also on some modifications of the message-passing algorithm of Shenoy and Shafer (1996).

For singly connected networks, when no compromise is made between spatial and time complexity, an efficient algorithm for finding the first two probable configurations has been developed by Pearl (1988). The basic idea of Pearl's algorithm is that each node in a network is associated with a causal and diagnostic function, through which the largest values of these functions will propagate to each other for obtaining the information needed to compute $P(x^1)$. Sy (1993), proposed an efficient computational method for obtaining w probable configurations in a singly connected network. The algorithm involves essentially a kind of message passing process.

However, methods developed for singly connected networks fail for multiply connected networks. One possible way out of this problem would be to create compound variables (combining several variables into one) in order to obtain a singly connected graph, Pearl (1988). However, the computational load of processing the compound variable is exponentially increased. A similar approach would be to generalize Sy's approach to joint tree representation of a Bayesian network. (Consult Jensen, 1996, for appropriate definitions). To overcome the problems occurring when a junction tree must be computed, Wierzchoń, Kłopotek and Michalewicz (1997) proposed another, genetic, approach to solve this problem within the general framework of VBS's. Some successful solution to the problem of finding probable configurations in junction trees was proposed recently by Seroussi and Golmard (1994) and Nilsson (1996). Particularly the last solution seems to be especially attractive.

In this paper we study the problem of finding the first $i \geq 1$ probable explanations (configurations) in a VBS. Section 2 is a rigorous problem formulation.

the nonserial dynamic programming introduced by Bertele and Brioschi (1972), and the explanation process is just the optimization, in the rest of the paper we will use the terms "function" (instead of "valuation") and we will speak about optimization. Section 3 is devoted to the mechanics of finding the first explanation—it introduces Shenoy's methodology. To better understand this mechanics we use a simple optimization example with combination realized in terms of standard addition. It is simpler to add integers than, for instance, multiply real numbers. Knowing the best explanation we are ready to compute further explanations. This problem is divided into two parts. First, in Section 4 we discuss the problem of finding the second best explanation, and we generalize it in Section 5 to finding the next explanation. Furthermore, in Section 4 we show a solution to the problem when combination has no inverse. Finally, in Section 6 we summarize the algorithm.

2. Problem formulation

Let $\mathtt{X} = \{X_1, \ldots, X_n\}$ be a set of variables with discrete domains $D_1, \ldots, D_n$, respectively. Let D stand for the Cartesian product of these domains, $D = D_1 \times \ldots \times D_n$; it represents the space of total configurations. Suppose $\mathtt{H} = \{H_1, \ldots, H_k\}$ is a family of subsets of $\mathtt{X}$, and $D(H_i)$ is the Cartesian product of the domains of variables in H_i. If $x \in D$ then $x.H_i$ stands for the projection of the configuration x onto the set of variables H_i. In the sequel we will write alternatively $f_i(H_i)$ to show that f_i is simply a function of variables specified in the set H_i, or $f_i(x.H_i)$ to refer to a particular value of the function in $x \in D$. With each H_i we associate a real valued function $f_i : D(H_i) \to \mathsf{R}$, $i = 1, \ldots, k$, and we define the function $f : D \to \mathsf{R}$ being the "combination" of the functions f_i, i.e.

$$f(x) = \bigotimes_{i=1}^{k} f_i(x.H_i), \ x \in D. \tag{1}$$

Here $\otimes : \mathsf{R} \times \mathsf{R} \to \mathsf{R}$ is a binary operation such that $(\mathsf{R}, \otimes)$ is an abelian group. That is: (a) $\otimes$ is commutative and distributive, (b) $\otimes$ admits an inverse operation $\circledR$, i.e. for any $a, b, c \in \mathsf{R}$ if $a \otimes b = c$ then $a = c \circledR b$, and (c) there exists a neutral element $0 \in \mathsf{R}$ such that $0 \otimes a = a$ for all $a \in \mathsf{R}$. Later (Section 4, Lemma 5) we relax this assumption; in fact it suffices to assume that $(\mathsf{R}, \otimes)$ is an abelian monoid.

Having in mind this distinction, we assume that $\otimes$ is such an operation that if $f(H_1), g(H_2)$, are two real-valued functions, where $H_1, H_2 \subseteq \mathtt{X}$, and H is a subset of H_2 such that $H \cap H_1 = \emptyset$, then

$$\max\{f(H_1) \otimes g(H_2) | X \in H\} = f(H_1) \otimes \max\{g(H_2) | X \in H\}.$$

We are interested in sorting decreasingly configurations in D with respect to a function f given in the factored form (1), i.e. we want to find a configuration

such that $f(x^1) \geq f(x^2) \geq \ldots \geq f(x^m)$, where $m \leq |D|$, the cardinality of D. To solve this problem we briefly review the method of finding the first maximal configuration, x^1, next we show how to find the second configuration, x^2, knowing x^1, and later we generalize this problem to finding x^m, $m > 2$, knowing all previous solutions.

3. Finding the first maximal configuration

A commonly used approach to finding the optimal value of the factored function f is so-called *nonserial dynamic programming*, Bertele and Brioschi (1972), consisting of two parts. In the *forward part* the variables in X are ordered in an appropriate way and the problem dimensionality is gradually reduced by maximization over subsequent (according to the assumed order) variables. This part returns the maximal value of f. In the *backward part* a configuration for which the maximum has been achieved is determined.

EXAMPLE 1 *Suppose f is a sum of three functions f_i defined over the sets of variables $H_1 = \{X_1, X_3, X_5\}$, $H_2 = \{X_1, X_2\}$, and $H_3 = \{X_2, X_4, X_5\}$. Hence $X = \{X_1, X_2, X_3, X_4, X_5\}$ and $\otimes = +$. Suppose the maximization is done in the order $\{X_3, X_4, X_5, X_1, X_2\}$, that is—first we maximize over (the domain of) the variable X_3, next over X_4, and so on. The first step (maximization over X_3) of the forward part looks like*

$$\begin{aligned}
&\max_{\{X_1,\ldots,X_5\}} f(\mathtt{X}) = \max_{\{X_1,X_2,X_4,X_5\}} \max_{\{X_3\}} (f_1(H_1) + f_2(H_2) + f_3(H_3)) \\
&= \max_{\{X_1,X_2,X_4,X_5\}} (\max_{\{X_3\}} f_1(H_1) + f_2(H_2) + f_3(H_3)) \\
&= \max_{\{X_1,X_2,X_4,X_5\}} (m(\{X_1, X_5\}) + f_2(H_2) + f_3(H_3)).
\end{aligned}$$

By maximizing over X_3 we have reduced problem dimensionality: in the next step the maximization is done over four variables only, $\{X_1, X_2, X_4, X_5\}$. We say that the variable X_3 has been *deleted* from the original problem. The problem remaining after removal of X_3 is of the same form as the original problem. The only difference resides in a minor modification of the objective function: it consists of the original functions f_2, f_3 and a new function m defined over the set $\{X_1, X_5\}$.

3.1. Hypergraphs and junction trees

The complexity of computations described above hardly depends on the order of deletions. If we choose, say, X_5, as the first variable to delete, then to compute the auxiliary function m, we must first create a function of five variables, $m(\{X_1, X_2, X_3, X_4, X_5\}) = f_1(\{X_1, X_3, X_5\}) + f_3(\{X_2, X_4, X_5\})$, that is—nothing has been gained! To find a good deletion ordering Shenoy (1989)

Bertele and Brioschi (1972). Observe namely that from a mathematical standpoint, the family $\mathtt{H}$ is nothing but a hypergraph $(\mathtt{X}, \mathtt{H})$[1].

Among hypergraphs of special importance are the so-called acyclic hypergraphs, denoted $(\mathtt{X}, \mathtt{N})$, or simply $\mathtt{N}$. They have two attractive properties. First, their hypernodes, denoted by N_j, $j = 1, \ldots, q$ (q is the cardinality of $\mathtt{N}$), can always be ordered in such a way that the *running intersection property*[2], or RIP for short, holds:

$$(\forall k \geq 2)(\exists j < k) : N_j \supseteq N_k \cap (N_1 \cup \ldots \cup N_{k-1}).$$

This property means that the variables in node N_k, also contained in previous nodes $(N_1, \ldots, N_{k-1})$ are all members of one previous node, N_j. Second, the hypernodes of $\mathtt{N}$ can be organized into a tree, called *junction tree.* To introduce this tree we need further definitions. The set

$$S_k = N_k \cap (N_1 \cup \ldots \cup N_{k-1})$$

is said to be *separator*: it separates the *residual*

$$R_k = N_k \setminus S_k$$

from $(N_1 \cup \ldots \cup N_{k-1}) \setminus S_k$. Particularly, $S_1 = \emptyset$ and $R_1 = N_1$. Any node N_j containing S_k with $j < k$ is called a *parent* of N_k in the tree; similarly the node N_k is called a *child* of N_j. Hence N_1 is the root of the junction tree and N_q is a leaf node in this tree.

Lemma 1 below characterizes some useful properties of the set of residuals.

LEMMA 1 *Let* $\mathtt{N}$ *be an acyclic hypergraph with nodes ordered such that the RIP holds, and let* $\mathtt{T}$ *be its junction tree. Denote by* $Ch(N_i)$ *the set of indices of the children of the node* N_i, $T(N_i)$ *the set of indices of the nodes belonging to the subtree rooted at the node* N_i. *Let* $De(N_i)$ *be the set of indices of the descendants of the node* N_i, *i.e.* $De(N_i) = T(N_i) \setminus \{i\}$. *Then*

a) *if* $\bigcup De(N_i)$ *stands for the set theoretical union of all the subsets being descendants of the node* N_i, *then* $\bigcup\{R_j | j \in De(N_i)\} = \bigcup De(N_i) \setminus N_i$,
b) *the residuals* $R_2, \ldots, R_q$ *form a partition of* $\mathtt{X}/N_1$.

Proof. (a) We prove this identity by induction. It trivially holds if N_i is a leaf in the junction tree. Suppose it holds for any $i > 1$, and let N_p be the parent of N_i. Then

$$\bigcup\{R_w | w \in De(N_p)\} = \bigcup_{k \in Ch(n_p)} \left[R_k \cup \left(\bigcup De(N_k) \setminus N_k\right)\right].$$

[1] Usually a hypergraph is identified with the set of its hypernodes. Hence we will write simply $\mathtt{H}$ instead of $(\mathtt{X}, \mathtt{H})$.

[2] Precise description of all notions concerning hypergraphs, used here, can be found e.g. in

Consider a single child, say N_k, of N_p. Since $R_k \subseteq N_k$ we have $R_k \cup (\bigcup De(N_k) \setminus N_k) = \bigcup De(N_k) \setminus S_k = \bigcup De(N_k) \setminus (N_k \cap N_p) = \bigcup De(N_k) \setminus N_p$. By the identity $A_1 \setminus B \cup \ldots \cup A_n \setminus B = (A_1 \cup \ldots \cup A_n) \setminus B$ we obtain the assertion.

(b) To prove this statement we must show that: (i) the residuals are mutually exclusive, and (ii) their set theoretical union equals $\mathtt{X}/N_1$. Part (ii) is just the case (a) when $i = 1$, i.e. when N_i is the root of the junction tree. Hence we must prove only part (i). Let A^c be the complement of A in $\mathtt{X}$. Then $R_i = N_i \setminus S_i = N_i \cap (N_i \cap (N_1 \cup \ldots \cup N_{i-1}))^c = N_i \cap ({N_1}^c \cap \ldots \cap {N_{i-1}}^c)$. Now, let R_j, $j > i$, be another separator. Then $R_i \cap R_j = ({N_1}^c \cap \ldots \cap {N_{i-1}}^c \cap N_i) \cap ({N_1}^c \cap \ldots \cap {N_i}^c \cap \ldots \cap {N_{j-1}}^c \cap N_j) = \emptyset$. ■

Graham test allows verifying if a hypergraph $\mathtt{H}$ is acyclic. It consists of two, performed in any order, rules: (i) if a variable X belongs to exactly one hypernode $\mathtt{H}$, remove it from $\mathtt{X}$, i.e. $H \leftarrow H - \{X\}$, and (ii) if H_1, H_2 are two hypernodes such that $H_1 \subset H_2$, and $H_1 \neq H_2$ then delete H_1 from $\mathtt{H}$. If any of two rules cannot be applied further, and $\mathtt{H} = \emptyset$, it means that $\mathtt{H}$ is acyclic.

EXAMPLE 2 *Consider the hypergraph* $\mathtt{H}$ *from Example* 1. *Variable* X_3 *belongs to* H_1 *only and the variable* X_4 *belongs to* H_3 *only. Hence, by applying twice the rule* (i) *we obtain* $\mathtt{H} = \{\{X_1, X_5\}, \{X_1, X_2\}, \{X_2, X_5\}\}$ *and we can do nothing more. Observe, however, that by adding the hypernode* $H' = \{X_1, X_2, X_5\}$ *to* $\mathtt{H}$ *we make it acyclic. Indeed, both the modified, by rule* (i), *hypernodes* H_1 *and* H_3, *and the hypernode* H_2 *are subsets of* H', *and—by rule* (ii)*—they can be deleted from* $\mathtt{H}$ *which now takes the form* $\mathtt{H} = \{\{X_1, X_2, X_5\}\}$. *Adding some new hypernodes to* $\mathtt{H}$ *is a general method of transforming a non-acyclic hypergraph into acyclic one.*

There is another way of finding acyclic coverage of a given hypergraph (i.e. we search for $\mathtt{N} \supset \mathtt{H}$). Define namely 2-section of a hypergraph $\mathtt{H}$ as an undirected graph $\mathtt{G(H)} = (\mathtt{X}, \mathtt{E})$ such that $\{X_i, X_j\} \in \mathtt{E}$ only if X_i and X_j belong to a common hypernode $H \in \mathtt{H}$. If $\mathtt{H}$ is acyclic then $\mathtt{G(H)}$ is triangulated and hypernodes in $\mathtt{H}$ are precisely cliques of $\mathtt{G(H)}$. Hence, to find an acyclic coverage of the original hypergraph, we construct its 2-section, triangulate it, and the cliques of such a graph are hypernodes in the acyclic hypergraph $(\mathtt{X}, \mathtt{N})$. This explains also another name of the junction tree: *tree of cliques.*

EXAMPLE 3 *Let us illustrate the notions of separator, residual, and tree of cliques. Suppose the modified set of hyperedges* $\mathtt{N}$ *consists of the hyperedges* $N_1 = \{X_1, X_2\}$, $N_2 = \{X_1, X_2, X_5\}$, $N_3 = \{X_2, X_4, X_5\}$, $N_4 = \{X_1, X_3, X_5\}$. *It is easy to check that such ordered sets admit the RIP. Separators, residuals and potential parents are given in the table below.*

It is instructive to compare this table with the computations described in the forward part. Observe first that the residuals written from the last to the first

Node	Separator	Residual	Potential parent
$N_1 = \{X_1, X_2\}$	$S_1 = \emptyset$	$R_1 = \{X_1, X_2\}$	—
$N_2 = \{X_1, X_2, X_5\}$	$S_2 = \{X_1, X_2\}$	$R_2 = \{X_5\}$	N_1
$N_3 = \{X_2, X_4, X_5\}$	$S_3 = \{X_2, X_5\}$	$R_3 = \{X_4\}$	N_2
$N_4 = \{X_1, X_3, X_5\}$	$S_4 = \{X_1, X_5\}$	$R_4 = \{X_3\}$	N_2

already proposed. Separator $S_4 = \{X_1, X_5\}$ agrees with the domain variables of the function m computed in the first step of the forward part. Similarly, after removal of variable X_4 we obtain another auxiliary function of two variables X_2, X_5—and this is just the separator S_3. Now, to remove variable X_5 we must first add the two auxiliary m functions obtaining new function, f', with domain variables X_1, X_2, and X_5—this is just the added hypernode N_2. Since this is a new function we define it as $f'(\xi) = 0$, $\xi \in D(\{X_1, X_2, X_5\})$. After maximization over X_5 we obtain the next auxiliary function with the domain variables $S_2 = \{X_1, X_2\}$. Joining this function with the function f_1 we can maximize it over the variables in $R_1 = \{X_1, X_2\}$. After this last maximization we obtain a single value, hence it can be treated as a "function" with no arguments ($S_1 = \emptyset$).

3.2. A message passing algorithm

Now we are able to reformulate the forward part in this new framework. Suppose $\mathtt{H} = \{H_1, \ldots, H_k\}$ is a family of subsets of the set of variables $\mathtt{X}$, and with each H_j there is associated a function f_j. Since the domains of all variables in $\mathtt{X}$ are discrete, the functions f_j are actually arrays of real numbers, i.e. tables. The f_j's are factors of the function f given in equation (1). Now let $\mathtt{N} \supseteq \mathtt{H}$ be an acyclic hypergraph covering $\mathtt{H}$. If $N_i \in \mathtt{N}$ is a hypernode such that there exists $H_j \in \mathtt{H}$ that $N_i = H_j$, we associate with N_i the table f_j; otherwise with N_i we associate the "empty" table f' such that $f'(\xi) = 0$ for all $\xi \in D(N_i)$. In other words we add a number of empty tables (defined over appropriate domains) to the original set $\{f_1, \ldots, f_k\}$ and we renumber this new set of factors according to the RIP. Because of the properties of $\otimes$ this rearrangement does not affect resulting function, that is, $f_1 \otimes \ldots \otimes f_q$ equals the original function f given in equation (1).

Next, assuming that the hyperedges in the acyclic hypergraph $\mathtt{N}$ are ordered so that they satisfy the RIP, let $\mathtt{T}$ be a junction tree representing $\mathtt{N}$. Its root is N_1 and it has at least one leaf N_q. The forward part consists of a number of send-absorb steps described below.

1. Mark all the nodes in the junction tree as *non-blocked* and, additionally, mark all leaves of the tree as *current leaves*. Particularly N_q is a current leaf.
2. *Send-step*: If N_l is a current, non-blocked leaf and N_p is the parent of N_l in the tree, compute the message $m_{l \to p}$ which N_l sends to its parent:
$$m_{l \to p}(x.S_l) = \max\{f_l(y.N_l) | y \in D : y.S_l = x.S_l\}$$

3. *Absorb-step*: The message is absorbed by the parent, i.e. the table $f_p(x.N_p)$ changes to the table $f_p(x.N_p) \otimes m_{l \to p}(x.S_l)$.
4. Repeat steps (2) and (3) for all non-blocked children of the parent N_p. When N_p has obtained messages from all its children, mark it as current leaf. When parent of the node N_p has all its children marked as *current leaf* and *non-blocked*, go to step (2).

Generally, the sequence of send-absorb steps bears a resemblance to evaluation of a parse tree. Hence, a node can send its message only if it is labeled as *current leaf* and *non-blocked*.

To characterize some properties of the algorithm we need further notations. Let $A(r) = \{1, \ldots, r\}$, $r \leq q$, be the set of indices of non-blocked nodes in the junction tree, and $B(r) = \{r+1, \ldots, q\}$ be the set of indices of blocked nodes. The nodes with indices in $A(r)$ form a subtree of T referred to as the *current tree*. In the course of node deletion from the junction tree the table f_i changes according to the rule described in the send-step. To distinguish between the original and modified table, the later will be denoted as t_i. Initially, $t_i = f_i$. So, if the current tree consists of r (non-blocked) nodes, the table t_i assigned to a non-blocked node N_i equals:

$$t_i = \begin{cases} f_i & \text{if } Ch(N_i) \cap B(r) = \emptyset \\ f_i \otimes (\bigotimes_{j \in Ch(N_i) \cap B(r)} m_{j \to i}) & \text{if } Ch(N_i) \cap B(r) \neq \emptyset. \end{cases}$$

LEMMA 2 *If all the children of a node N_i are blocked then its table t_i equals*

$$t_i = f_i \otimes \max_{X \in \bigcup De(N_i) \setminus N_i} \Big(\bigotimes_{j \in De(N_i)} f_j \Big)$$

Proof by induction. If $i = q$ then $De(N_q) = \emptyset$ and Lemma trivially holds. Suppose that N_p is the parent of N_i and that Lemma is true for all the children of the node N_i. Then $t_p = f_p \otimes \{\bigotimes m_{j \to p} | j \in Ch(N_p)\} = f_p \otimes \{\bigotimes \max_{X \in R_j} t_j | j \in Ch(N_p)\} = \max_{X \in \{\bigcup R_j | j \in Ch(N_p)\}} (f_p \otimes \{\bigotimes t_j | j \in Ch(N_p)\})$. Expanding all terms t_j and using Lemma 1(a) we obtain the assertion. ∎

The reader can verify that the values of table t_i can be expressed as

$$t_i(x.N_i) = f_i(x.N_i) \otimes \max \Big\{ \bigotimes_{j \in De(N_i)} f_j(y.N_j) | y \in D : N_i = x.N_i \Big\}.$$

COROLLARY 1 *Let the current tree consist of the nodes with indices in the set $A(r)$. Then $t_1 \otimes \ldots \otimes t_r = \max\{f_1 \otimes \ldots \otimes f_q | X \in R_{r+1} \cup \ldots \cup R_q\}$.*

COROLLARY 2 *After the forward part the root, N_1, of the junction tree stores the table $t_1 = \max\{f_1 \otimes \ldots \otimes f_q | X \in \mathbb{X} \setminus N_1\}$, i.e. $t_1(x.N_1) = \max\{f(y) | y \in D :$*

EXAMPLE 4 *Let* N *be the acyclic hypergraph from example 3. Let* $D_i = \{0,1\}$, $i = 1,\ldots,5$ *and* $\otimes$ *be the* $+$ *operator. Suppose that* f_2 *is the empty table, i.e.* $f_2(\xi) = 0$, $\xi \in D(\{X_1, X_2, X_5\})$, *and the remaining tables are given below.*

x_1	x_3	x_5	f_4
0	0	0	1
0	0	1	3
0	1	0	5
0	1	1	8
1	0	0	2
1	0	1	6
1	1	0	2
1	1	1	4

Table 1

x_1	x_2	f_1
0	0	4
0	1	8
1	0	0
1	1	5

Table 2

x_2	x_4	x_5	f_3
0	0	0	0
0	0	1	5
0	1	0	6
0	1	1	3
1	0	0	5
1	0	1	1
1	1	0	4
1	1	1	3

Table 3

Recall that according to the RIP the nodes N_4 *and* N_3 *are children of* N_2, *which is a child of the root* N_1. *According to the procedure, the nodes* N_4 *and* N_3 *send their messages* $m_{4\to 2}, m_{3\to 2}$ *to the parent* N_2. *The messages are defined on the domains of the variables belonging to the appropriate separators, and the maximization is done over the variables belonging to the residuals, respectively,* $R_4 = \{X_3\}$ *and* $R_3 = \{X_4\}$.

x_1	x_5	$m_{4\to 2}(x_1, x_5)$
0	0	$\max(1,5) = 5$
0	1	$\max(3,8) = 8$
1	0	$\max(2,2) = 2$
1	1	$\max(6,4) = 6$

Table 4

x_2	x_5	$m_{3\to 2}(x_2, x_5)$
0	0	$\max(0,6) = 6$
0	1	$\max(5,3) = 5$
1	0	$\max(5,4) = 5$
1	1	$\max(1,3) = 3$

Table 5

Both messages modify the original table f_2 *which now, according to our convention, equals* $t_2(x_1, x_2, x_5) = f_2(x_1, x_2, x_5) + m_{4\to 2}(x_1, x_5) + m_{3\to 2}(x_2, x_5)$, *see Table 6. Now, the node* N_2 *can send the message* $m_{2\to 1}$ *(Table 7) which adds to* f_1. *The resulting table* t_1 *takes the form given in Table 8.*

x_1	x_2	x_5	$t_2(x_1, x_2, x_5)$
0	0	0	$0+5+6 = 11$
0	0	1	$0+8+5 = 13$
0	1	0	$0+5+5 = 10$
0	1	1	$0+8+3 = 11$
1	0	0	$0+2+6 = 8$
1	0	1	$0+6+5 = 11$
1	1	0	$0+2+5 = 7$
1	1	1	$0+6+3 = 9$

x_1	x_2	$m_{2\to 1}(x_1, x_2)$
0	0	$\max(11, 13) = 13$
0	1	$\max(10, 11) = 11$
1	0	$\max(8, 11) = 11$
1	1	$\max(7, 9) = 9$

Table 7

x_1	x_2	$t_1(x_1, x_2) = f_1(x_1, x_2) + m_{2\to 1}(x_1, x_2)$
0	0	$4 + 13 = 17$
0	1	$8 + 11 = 19$
1	0	$0 + 11 = 11$
1	1	$5 + 9 = 14$

Table 8

3.3. Identification of the configuration

To characterize backward part, suppose we have found the first optimal configuration $x^1.(N_1 \cup \ldots \cup N_{j-1})$. To extend it to the configuration $x^1.(N_1 \cup \ldots \cup N_j)$ we search for such an assignment ξ of variables in N_j that $\xi.S_j = x^1.S_j$ and ξ maximizes the table t_j.

EXAMPLE 4 (continuation) *We find the maximal value in Table 8, namely 19, achieved for the configuration* $x_1 = 0$, $x_2 = 1$. *Now, according to the recipe, we move back to the table* t_2 *and we search for the configuration* ($x_1 = 0$, $x_2 = 1$, $x_5 = ?$) *such that* $t_2(x_1 = 0, x_2 = 1, x_5 = ?)$ *takes its maximum; this is achieved for* $x_5 = 1$. *Proceeding similarly with the tables* $t_3 = f_3$ *and* $t_4 = f_4$ *we state that* $x_3 = 1$, $x_4 = 1$, *i.e. the optimal configuration* $x^1 = (x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 1, x_5 = 1)$, *and indeed,* $f(x^1) = f_1(x_1 = 0, x_2 = 1) + f_3(x_2 = 1, x_4 = 1, x_5 = 1) + f_4(x_1 = 0, x_3 = 1, x_5 = 1) = 8 + 5 + 8 = 19$.

LEMMA 3 *Let* x^1 *be the first optimal configuration.*

a) $m_{q\to p}(x^1.S_q) = t_q(x^1.N_q) = f_q(x^1.N_q)$

b) $t_i(x^1.N_i) = \max\{\bigotimes_{j\in T(N_i)} f_j(y.N_j) | y \in D : y.N_i = x^1.N_i\}$
$= f_i(x^1.N_i) \otimes (\bigotimes_{N_j \in De(N_i)} t_j(x^1.N_j))$.

Proof. (a) $m_{q\to p}(x^1.S_q) = \max\{f_q(y.N_q) | y \in D : y.S_q = x^1.S_q\}$, hence the table is maximized over the variables in residual R_q. But according to the construction of x_1 this maximum is achieved for the values $x^1.R_q$. Hence the result.

(b) is left as a simple exercise. ∎

4. Finding the second maximum

Let x^1 be the optimum configuration and let x^2 stand for the "second-best" configuration, i.e. $f(x^2) = \max\{f(x) | x \in D \setminus \{x^1\}\}$. Suppose that x^1 and x^2 differ on at least one position corresponding to a variable, say, X_k. Let N_j be a node such that $X_k \in N_j$ and X_k does not belong to any node outside the subtree $T(N_j)$. Obviously, if $X_k \in N_j = S_j \cup R_j$ then X_k must be a member of the residual R_j. Indeed, if $X_k \in S_j$ then X_k must belong to the parent

in $T \setminus T(N_j)$. From Corollary 1 we know that the combination of functions f_j belonging to the current tree, i.e. with $j \in A(r)$, $r \in [1, q]$ is just the maximum of f over $D(R_{r+1} \cup \ldots \cup R_q)$. To find $f(x_2)$ we must review the maximal values of f over the sets

$$D^j = \{y \in D : y.(N_1 \cup \ldots \cup N_{j-1}) = x^1.(N_1 \cup \ldots \cup N_{j-1}),$$
$$y.R_j \neq x^1.R_j\},\ j = 2, \ldots, q$$
$$D^1 = \{y \in D : y.N_1 \neq x^1.R_1\},\ \text{since } S_1 = \emptyset \text{ and } R_1 = N_1 \setminus S_1 = N_1$$

each time obtaining maximum value of f^j over the set $D(R_{j+1} \cup \ldots \cup R_q)$. Hence $\max\{f(x)|x \in D \setminus \{x^1\}\} = \max\{f^j|j = 1, \ldots, q\}$. Applying again Corollary 1, we state that f_j can be computed as

$$\max\{f|x \in D^j\} = [t_1(x^1.N_1) \otimes \ldots \otimes t_{j-1}(x^1.N_{j-1})]$$
$$\otimes \max\{t_j(y.N_j)|y \in D(N_j) : y.S_j = x^1.S_j,\ y.R_j \neq x^1.R_j\}.$$

To find the first term in this equation we use Lemma 3

$$\max\{t_j(y.N_j)|y \in D(N_j) : y.S_j = x^1.S_j,\ y.R_j = x^1.R_j\} = t_j(x^1.N_j)$$

Hence

$$t_1(x^1.N_1) \otimes \ldots \otimes t_{j-1}(x^1.N_{j-1}) = f(x^1) \circledR t_j(x^1.N_j).$$

and finally

$$\max\{f|x \in D^j\} = [f(x^1) \circledR t_j(x^1.N_j)] \otimes \max\{t_j(y.N_j)|y \in D(N_j) :$$
$$y.S_j = x^1.S_j,\ y.R_j \neq x^1.R_j\}.$$

This way we have proved the very important lemma allowing the use of local search for determining the maximum of f over the sets D^j:

LEMMA 4 *Let $\mathbb{N}$ be an acyclic hypergraph representing factored function f with hypernodes ordered such that the RIP is fulfilled. Let $(\otimes, \mathsf{R})$ be an abelian group. Then the maximal value of the function f over the set D^j equals*

$$\max\{f|x \in Dj\} = [f(x^1) \circledR t_j(x^1.N_j)] \otimes \max\{t_j(y.N_j)|y \in D(N_j) :$$
$$y.S_j = x^1.S_j,\ y.R_j \neq x^1.R_j\}$$

This lemma has been first published by Nilsson (1996) but in the context of the HUGIN architecture. Our proof uses a much simpler propagation scheme, and it allows to solve the problem of finding $\max\{f|x \in D^j\}$ in the cases when $\otimes$ does not have the inverse operator, $\circledR$. This is shown in Lemma 5.

LEMMA 5 *Let $\mathbb{N}$ be an acyclic hypergraph representing factored function f with hypernodes ordered such that the RIP is fulfilled. Let $(\otimes, \mathsf{R})$ be a commutative*

the root to the node N_j in the junction tree, let N_p be the parent of N_j. Denote by $F = \bigcup\{Ch(N_i)|i \in \Pi(N_j)\} \setminus \Pi(N_j)$ the set of children of the nodes belonging to the path excluding the nodes from this path. Then the maximal value of the function f over the set D^j equals

$$\max\{f|x \in D^j\} = \Big(\bigotimes_{k\in F} t_k(x^1.N_k)\Big) \otimes \Big(\bigotimes_{l\in\Pi(N_j)\setminus\{j\}} f_l(x^1.N_l)\Big)$$

$$\otimes \max_{y\in D(N_j): y.S_j = x^1.S_j,\, y.R_j \neq x^1.R_j} t_j(y.N_j)$$

Proof. After the message passing algorithm has been finished the set of blocked nodes equals $B(0)$, i.e. it contains all the nodes from the junction tree. Let us remove from $B(0)$ all the nodes belonging to the path from N_1 to N_j. If $N_{p(j)}$ is the parent of N_j then, by Lemma 3b, the table $t_{p(j)}$ is recomputed as follows: $t'_{p(j)}(x^1.N_{p(j)}) = f_{p(j)} \otimes \{\bigotimes t_c(x^1.N_c)|c \in Ch(N_{p(j)}) \setminus \{j\}\} \otimes t_j(z)$ where z is a configuration from $D(N_i)$ such that $t_j(z) = \max\{t_j(y)|y \in D : y.S_j = x^1.S_j,\, y.R_j \neq x^1.R_j\}$. Certainly this new value $t'_{p(j)}(x^1.N_{p(j)})$ is different from the value $t_{p(j)}(x^1.N_{p(j)})$ computed during the original absorption stage. Hence, to find new value $t'_{p(p(j))}(x^1.S_{p(p(j))})$ for the parent $p(p(j))$ of the node $p(j)$ we must use again the procedure similar to that described above, i.e. $t'_{p(p(j))}(x^1.N_{p(p(j))}) = f_{p(p(j))} \otimes \{\bigotimes t_c(x^1.N_c)|c \in Ch(N_{p(p(j))}) \setminus \{p(j)\}\} \otimes t'_{p(j)}(x^1.N_{p(j)})$. ■

EXAMPLE 5 (illustration of Lemma 5) *Suppose that* $\mathtt{N} = \{N_1, \ldots, N_7\}$ *and* $N_1 = \{A, B, C\}$, $N_2 = \{A, B, D\}$, $N_3 = \{B, C, E\}$, $N_4 = \{B, D, F\}$, $N_5 = \{A, D, G\}$, $N_6 = \{B, C, H\}$, $N_7 = \{C, E, I\}$. *The nodes are ordered such that the RIP holds. Suppose that all the variables are binary, and the optimal solution is* $x^1 = (a = 0,\, b = 1,\, c = 0,\, d = 0,\, e = 0,\, f = 1,\, g = 1,\, h = 1,\, i = 0)$. *Let us find the maximum value of a factored function over the set* D^6. *According to our convention* $\Pi(N_6) = \{1, 3, 6\}$ *and* $F = \{2, 7\}$. *Hence* $\max\{f|x \in D^j\} = [t_2(0,1,1) \otimes t_7(0,0,0)] \otimes [f_1(0,1,0) \otimes f_3(1,0,0)] \otimes t_6(z)$ *where* $z = (1,0,0)$.

The next example shows how to use Lemma 4 to find x^2 knowing the configuration x^1 computed in Example 4.

EXAMPLE 6 *Consider the tables* $t_1, \ldots, t_4$ *from Example 4. Here the sets* $D^1, \ldots, D^4$ *are:* $D^1 = \{(y_1, y_2, y_3, y_4, y_5) \in D : (y_1, y_2) \neq (0,1)\}$, $D^2 = \{(y_1, y_2, y_3, y_4, y_5) \in D : (y_1, y_2) = (0,1),\, y_5 \neq 1\}$, $D^3 = \{(y_1, y_2, y_3, y_4, y_5) \in D : (y_1, y_2, y_5) = (0,1,1),\, y_4 \neq 1\}$, $D^4 = \{(y_1, y_2, y_3, y_4, y_5) \in D : (y_1, y_2, y_5, y_4) = (0,1,1,1),\, y_3 \neq 1\}$. *To find the maximum of* f *over* D^1 *we simply review the table* t_1 *over all configurations* (y_1, y_2) *different from* $(0,1)$. *We state that* $\max\{f|y \in D^1\} = \max\{t_1|(y_1, y_2) : (y_1, y_2) \neq (0,1)\} = t_1(y_1 = 0,\, y_2 = 0) = 17$.

Lemma 4 to find $\max\{f|y \in D^2\}$ *we must compute first* $[f(x^1) \circledR t_2(x^1.N_2)]$. *But* $f(x^1) = 19$, $x^1.N_2 = (x_1^1 = 0, x_2^1 = 1, x_5^1 = 1)$, *and* $[f(x^1) \circledR t_2(x^1.N_2)] = 19 - t_2(0,1,0) = 19 - 11 = 8$. *Because all the variables are binary then* $D_5 \setminus \{y_5 = 1\} = \{y_5 = 0\}$ *and* $\max\{t_2(y.N_2)|y \in D(N_2) : y.S_2 = x^1.S_j,\ y.R_2 \neq x^1.R_2\} = t_2(0,1,0) = 10$ *from which it follows that* $\max\{f|y \in D^2\} = 8 + 10 = 18$. *Similarly* $\max\{f|y \in D^3\} = [19 - t_3(1,1,1)] + t_3(1,0,1) = (19-3)+1 = 17$, *and* $\max\{f|y \in D^4\} = [19 - t_4(0,1,1)] + t_4(0,0,1) = (19-8)+3 = 14$. *The greatest value among these maxima is 18 and this value has been achieved on the set* D^2.

It is obvious that when the maximum is achieved on the set, say D^j, we know the partial optimal configuration x^2 such that $x^2.(N_1 \cup \ldots \cup N_{j-1}) = x^1.(N_1 \cup \ldots \cup N_{j-1})$, $x^2.R_j$ equals the value that maximizes the table tj over the set $D(S_j) \cup (D(R_j) \setminus \{x^1.R_j\})$. In our case the maximum is achieved on D^2, that is, $x^2.N_2 = (x_1^2 = 0, x_2^2 = 1, x_5^2 = 0)$. To find the values of (x_3^2, x_4^2) we use the backward part of the algorithm from Section 3 taking the subtree rooted in N_2. In our case we find that $(x_3^2, x_4^2) = (1,0)$.

Observe that the residuals of the children of the node N_j are disjoint from the variables in this node. This means that the variables specified in these residuals cannot occur in any node from the set $\mathrm{T} \setminus T(N_j)$, which follows from the RIP. Hence the variables belonging to the nodes in $\mathrm{T} \setminus T(N_j)$ are already instantiated and, in fact, we only need to instantiate the variables belonging to the descendants of N_j. This is substantial saving of computations in comparison with the algorithm proposed by Nilsson (1996).

5. Finding i-th ($i > 2$) optimal configuration

Suppose we have determined the configurations x^1 and x^2. To find the third configuration, x^3, we must search the space $D \setminus \{x^1, x^2\}$. To prepare further refinement, $D^1(x^2), \ldots, D^q(x^2)$, of this set one should observe, first, that if $f(x_2)$ has been identified on the set D^j it means that $x^2.(N_1 \cup \ldots \cup N_j) = x^1.(N_1 \cup \ldots \cup N_j)$ and there is no need to search maxima over the sets $D^1(x^2), \ldots, D^{j-1}(x^2)$. The set $D^j(x^2)$ has now the form

$$D^j(x^2) = \{y \in D : y.(N_1 \cup \ldots \cup N_{k-1}) = x^1.(N_1 \cup \ldots \cup N_{k-1}),\ y.R_k \neq x^1.R_k,\ y.R_k \neq x^2.R_k\}$$

and the sets $D^k(x^2)$, $k = j+1, \ldots, q$ are of the form

$$D^k(x^2) = \{y \in D : y.(N_1 \cup \ldots \cup N_{k-1}) = x^2.(N_1 \cup \ldots \cup N_{k-1}),\ y.R_k \neq x^2.R_k\}$$

because $x^2.(N_{j+1} \cup \ldots \cup N_q)$ is independent of $x^1.(N_{j+1} \cup \ldots \cup N_q)$.

EXAMPLE 7 *We know that* $x^1 = (x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 1, x_5 = 1)$,

has been found on the set D_2. It means that we must search the maximum values over the sets $D^2(x^2)$, $D^3(x^2)$ and $D^4(x^2)$. It is easy to verify that $D^2/\{x^2\} = \emptyset$. Continuing the computations we obtain

$$\max\{f|y \in D^3\} = [f(x^2) - t_3(1,0,0)] + t_3(1,1,0) = 18 - 5 + 4 = 17$$
$$\max\{f|y \in D^4\} = [f(x^2) - t_4(0,1,0)] + t_4(0,0,0) = 18 - 5 + 1 = 14.$$

Now we choose among the next values (× means already exploited value):

solution	node N_1	node N_2	node N_3	node N_4
x^1	17	×	17	14
x^2			17	14

Suppose we choose the value 17 from the first column. From Example 6 we know that this value fixes the variables X_1 and X_2 at the values $(0,0)$. Using backward part we identify the third configuration: $x^3 = (x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1)$. Completing all necessary computations we find the table

solution	node N_1	node N_2	Node N_3	node N_4
x^1	×	×	17	14
x^2			17	14
x^3	11	15	15	12

from which we state that $f(x^4) = 17$.

6. Summary

Let us summarize the algorithm. Suppose all the solutions $x^1, \ldots, x^k$ are stored in the table *solutions*. Suppose also that the information stored in the tables is represented in a linked list. Each item of the list should contain the next data: the number of a node, *node_no*, the number of a solution, *sol_no*, from which the maximal value of the table t_{node_no} on the set $D^{node\_no}(x^{sol\_no})$ is computed, and the number of configuration on which this maximum was achieved, *conf_no*. For instance the information from the table above can be represented as follows (observe that the solution x^1 has no predecessor, hence $sol\_no = 0$; similarly the solution x^3 has been obtained from x^1, hence $sol\_no = 1$):

sol_no	*node_no*	*max*	*conf_no*
0	3	17	6
0	4	14	2
1	3	17	7
1	4	17	1
1	1	11	3
1	2	15	1
1	3	14	3

It is better to sort this list decreasingly, hence we always choose the first element of the list. Suppose we have identified first k solutions and we are interested in finding x^{k+1}. If we know only first solution, x^1 (it must be identified using the method described in Example 4), we set $node\_no = 1$. The algorithm can be formulated as follows:

1. Choose the first element from the list. Block the configuration $conf\_no$ in the table t_{node_no} and tracing back (using the field sol_no)—block all previous configurations on this node.
2. Compute maxima $f_j = D_j(x^k)$, $j = node\_no, \ldots, q$, and add them to the list.
3. Find $max = \{f_j | j = node\_no, \ldots, q\}$.
4. Identify the solution, that is using the information from step 1, set $x^{k+1}.(N_1 \cup \ldots \cup N_{node\_no-1}) = x^{sol\_no}.(N_1 \cup \ldots \cup N_{node\_no-1})$ and fill in the remaining values of the variables by the backward step starting from the node $node\_no$).

Of course to use it, we must first compute the configuration x^1 by using the message-passing algorithm.

It is important to notice also that when the function achieves its maximum on the set D^j, then in order to identify appropriate configuration we can restrict the backward part to the nodes belonging to $T(N_j)$ only, what follows from the RIP. Particularly, when N_j is a leaf in the junction tree, the backward part is not necessary.

References

BEERI, C. et al. (1983) On the desirability of acyclic database schemes. *J. ACM*, **30**, 479–513.

BERTELE, U. and BRIOSCHI, F. (1972) *Nonserial Dynamic Programming.* Academic Press, New York.

COOPER, G. (1990) The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.*, **42**, 395–405.

DECHTER, R. (1996) Bucket elimination: A unifying framework for probabilistic inference algorithms. In *Uncertainty in AI (UAI-96)*, 211–219.

HENRION, M. (1990) An introduction to algorithm for inference in belief nets, *Uncertainty in Artificial Intelligence*, **5**.

JENSEN, F.V. (1996) *An Introduction to Bayesian Networks.* University College Press, London.

KERAMARIS, V. and SY, B.K. (1990) A syllable based approach towards the intelligibility study of impaired speech. *Proc. 12th Annual International Conference of IEEE-EMBS*, Philadelphia, Penn..

LAURITZEN, S.L. and SPIEGELHALTER, D.J. (1988) Local computations with probabilities on graphical structures and their application to expert sys-

NILSSON, D. (1996) An efficient algorithm for finding the M most probable configurations in Bayesian networks. Institute for Electronic Systems, Aalborg University, Denmark, Technical Report R-96-2020. A revised version of this report will appear in *Statistics and Computing.*

PAL, N., BEZDEK, J. and HEMANISHA, R. (1992) Uncertainty measures for evidential reasoning I: a review. *Int. J. Approx. Reas.*, **7** 162–183.

PEARL, J. (1986) Fusion, propagation and structuring in Bayesian networks. *Artif. Intel.*, **28**, 241–288.

PEARL, J. (1988) *Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers.

SEROUSSI, B. and GOLMARD, J.L. (1994) An algorithm for finding the K most probable configurations in Bayesian networks. *Int. J. Approx. Reasoning*, **8** 17–50.

SHAFER, G. (1996) Probabilistic Expert Systems, *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*, Philadelphia, 67.

SHENOY, P.P. (1989) A valuation-based language for expert systems. *Int. J. Approx. Reas.*, **3** 383–411.

SHENOY, P.P. (1993) Conditional independence in valuation-based systems. *Int. J. Approx. Reas.*, **10** 203–234.

SHIMONY, S.E. and CHARNIAK, E. (1990) A new algorithm for finding MAP assignments to belief networks. *Proc. Conference on Uncertainty in AI*, Cambridge, MA, 98–103.

SY, B.K. (1989) An AI based CAD/CAM approach to assess design methodology of a user-specified nonvocal communication device. *Proc. 15th Northeast Bioengine-ering Conference*, Boston, Mass., 185–192.

SY, B.K. (1993) A recurrence local computation approach towards ordering composite beliefs in Bayesian belief networks. *Int. J. Approx. Reasoning*, **11** 205–233.

WIERZCHOŃ, S.T. (1994) Constraint propagation over restricted space of configurations, and its use in optimization. In: R.R. Yager, J. Kacprzyk and M. Fedrizzi (eds.), *Advances in the Dempster-Shafer Theory of Evidence*, J. Wiley, New York, 375–394.

WIERZCHOŃ, S.T., KŁOPOTEK, M.A. and MICHALEWICZ, M. (1997) Reasoning and facts explanation in valuation based systems, *Fundamenta Informaticae*, **30** 359–371.

WU, T. (1990) A problem decomposition method for efficient diagnosis and interpretation of multiple disorders, *Proc. 114th. Symp. Computer Appl. in Medical Care*, 86–92.

YEN, J. (1990) Generalizing Dempster-Shafer theory to fuzzy sets. *IEEE Trans. Syst. Man. Cybern.*, **20**, 3, 559–570.