Uncertainty in Artificial Intelligence 6
P.P. Bonissone, M. Henrion, L.N. Kanal and J.F. Lemmer (Editors)
© 1991 Elsevier Science Publishers B.V. All rights reserved

VALUATION-BASED SYSTEMS FOR DISCRETE OPTIMIZATION

by

Prakash P. Shenoy
School of Business, University of Kansas, Lawrence, KS 66045-2003, USA
Tel: (913)-864-7551, Fax: (913)-864-5328, Bitnet: PSHENOY@UKANVM

ABSTRACT

This paper describes valuation-based systems for representing and solving discrete optimization problems. In valuation-based systems, we represent information in an optimization problem using variables, sample spaces of variables, a set of values, and functions that map sample spaces of sets of variables to the set of values. The functions, called valuations, represent the factors of an objective function. Solving the optimization problem involves using two operations called combination and marginalization. Combination tells us how to combine the factors of the joint objective function. Marginalization is either maximization or minimization. Solving an optimization problem can be simply described as finding the marginal of the joint objective function for the empty set. We state some simple axioms that combination and marginalization need to satisfy to enable us to solve an optimization problem using local computation. For optimization problems, the solution method of valuation-based systems reduces to non-serial dynamic programming. Thus our solution method for VBS can be regarded as an abstract description of dynamic programming. And our axioms can be viewed as conditions that permit the use of dynamic programming.

Subject classification: Dynamic programming: axioms, theory, algorithm.

1. INTRODUCTION

The main objective of this paper is to describe a valuation-based system (VBS) for representing and solving discrete optimization problems. There are several reasons why this is useful.

First, I initially proposed VBSs for managing uncertainty in expert systems [Shenoy, 1989, 1991]. Here I show that these systems also have the expressive power to represent and solve optimization problems.

Second, problems in decision analysis involve managing uncertainty and optimization. That these problems can be solved in a common framework suggests that decision problems also can be represented and solved in the framework of VBS. Indeed, Shenoy [1990a,b] shows that this is true. In fact, the solution procedure of VBSs when applied to decision problems results in a method that is computationally more efficient than decision trees and influence diagrams.

Third, the solution procedure of VBS when applied to optimization problems results in a method called non-serial dynamic programming [Bertele and Brioschi, 1972]. Thus in an abstract sense, the local computation algorithms that have been described by Pearl [1986], Shenoy and Shafer [1986], Dempster and Kong [1988], and Lauritzen and Spiegelhalter [1988] are just dynamic programming.

Fourth, we provide an answer to the question: What is dynamic programming? Dynamic programming is commonly viewed as an optimization technique. This is how Bellman [1957] described it. However, it is also recognized that dynamic programming is more than an optimization technique. For example, Aho, Hopcroft and Ullman [1974] refer to dynamic programming as a "divide-and-conquer" methodology. In this paper, we give a formal definition of a problem and a formal method solving the problem. The formal method for solving the problem can be thought of as an abstract definition of dynamic programming.

Fifth, we provide an answer to the question: When does dynamic programming work? We describe some simple axioms for combination and marginalization that enable the use of dynamic programming for solving optimization problems. We believe these axioms are new. They are weaker than those proposed by Mitten [1964].

Sixth, the VBS described here can be easily adapted to represent propositional logic [Shenoy 1990c,d] and constraint satisfaction problems [Shenoy and Shafer, 1988].

An outline of this paper is as follows. In Section 2, we show how to represent an optimization problem as a VBS. In Section 3, we state some simple axioms that justify the use of local computation in solving VBSs. In Section 4, we show how to solve a VBS. Throughout the paper, we use one example to illustrate all definitions and the solution method. In section 5, we compare our axioms to those proposed by Mitten [1964] for serial dynamic programming. In section 6, we make some concluding remarks. Finally, in section 7 we provide proofs for the main results in the paper.

2. REPRESENTATION OF OPTIMIZATION PROBLEMS

A valuation-based system representation of an optimization problem uses variables, frames, and valuations. We will discuss each of these in detail. We will illustrate all definitions using an optimization problem from Bertele and Brioschi [1972].

An Optimization Problem. There are five variables labeled as A, B, C, D, and E. Each variable has two possible values. Let a and ~a denote the possible values of A, etc. The joint objective function F for variables A, B, C, D, and E factors additively as follows:

$F(v,w,x,y,z) = F_1(v,x,z) + F_2(v,w) + F_3(w,y,z)$, where F_1, F_2, and F_3, are as shown below in Figure 1. The problem is to find the minimum value of F and a configuration (v,w,x,y,z) that minimizes F.

Figure 1. The factors of the objective function, F_1, F_2, and F_3.

$w \in \mathcal{W}_{\{A,C,E\}}$	$F_1(w)$
a c e	1
a c ~e	3
a ~c e	5
a ~c ~e	8
~a c e	2
~a c ~e	6
~a ~c e	2
~a ~c ~e	4

$w \in \mathcal{W}_{\{A,B\}}$	$F_2(w)$
a b	4
a ~b	8
~a b	0
~a ~b	5

$w \in \mathcal{W}_{\{B,D,E\}}$	$F_3(w)$
b d e	0
b d ~e	5
b ~d e	6
b ~d ~e	3
~b d e	5
~b d ~e	1
~b ~d e	4
~b ~d ~e	3

Variables, Frames, and Configurations. We use the symbol $\mathcal{W}_X$ for the set of possible values of a variable X, and we call $\mathcal{W}_X$ the *frame* for X. We are concerned with a finite set $\mathcal{X}$ of variables, and we assume that all the variables in $\mathcal{X}$ have finite frames.

Given a finite nonempty set h of variables, we let $\mathcal{W}_h$ denote the Cartesian product of $\mathcal{W}_X$ for X in h, i.e., $\mathcal{W}_h = \times\{\mathcal{W}_X \mid X \in h\}$. We call $\mathcal{W}_h$ the *frame* for h. We call elements of $\mathcal{W}_h$ *configurations of h*. Lower-case bold-faced letters, such as **x**, **y**, etc., denote configurations. If **x** is a configuration of g, **y** is a configuration of h, and $g \cap h = \varnothing$, then (**x**,**y**) denotes the configuration of $g \cup h$ obtained by concatenating **x** and **y**.

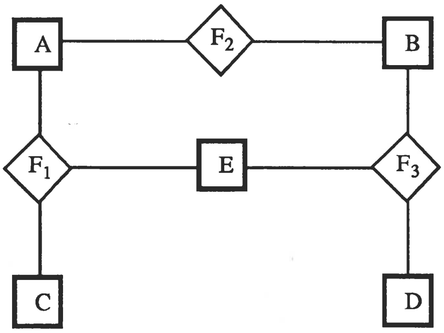
It is convenient to allow the set of variables h to be empty. We adopt the convention that the frame for the empty set $\varnothing$ consists of a single element, and we use the symbol $\blacklozenge$ to name that element; $\mathcal{W}_\varnothing = \{\blacklozenge\}$. If **x** is a configuration of g, then $(\mathbf{x}, \blacklozenge)$ is simply **x**.

Values and Valuations. We are concerned with a set $\mathbb{V}$ whose elements are called *values*. $\mathbb{V}$ may be finite or infinite. Given a set h of variables, we call any function $H: \mathcal{W}_h \rightarrow \mathbb{V}$, a *valuation for h*. Note that to specify a valuation for $\varnothing$, we need to specify only a single value, $H(\blacklozenge)$. If H is a valuation for h and $X \in h$, then we say H *bears on* X.

In our problem, the set $\mathbb{V}$ corresponds to the set of real numbers, and we have three valuations F_1, F_2 and F_3. F_1 is a valuation for {A,C,E}, F_2 is a valuation for {A,B} and F_3 is a valuation for {B,D,E}. Figure 2 shows a graphical depiction of the optimization problem. We call such a graph a *valuation network*. In a valuation network, square nodes represent variables, and diamond-shaped nodes represent valuations. Each valuation is linked to the variables it bears on.

Let $\mathcal{V}_h$ denote the set of valuations for h, and let $\mathcal{V}$ denote the set of valuations, i.e., $\mathcal{V} = \cup\{\mathcal{V}_h \mid h \subseteq \mathcal{X}\}$.

Figure 2. The valuation network for the optimization problem.



Projection of Configurations. *Projection* of configurations simply means dropping extra coordinates; if (~a,b,~c,d,e) is a configuration of {A,B,C,D,E}, for example, then the projection of (~a,b,~c,d,e) to {A,C,E} is simply (~a,~c,e), which is a configuration of {A,C,E}.

If g and h are sets of variables, $h \subseteq g$, and **x** is a configuration of g, then let $\mathbf{x}^{\downarrow h}$ denote the projection of **x** to h. The projection $\mathbf{x}^{\downarrow h}$ is always a configuration of h. If h=g and **x** is a configuration of g, then $\mathbf{x}^{\downarrow h} = \mathbf{x}$. If $h = \varnothing$, then of course $\mathbf{x}^{\downarrow h} = \blacklozenge$.

Combination. We assume there is a mapping $\copyright: \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{V}$ called *combination* so that if $u, v \in \mathbb{V}$, then $u \copyright v$ is the value representing the combination of u and v. We define a mapping $\oplus: \mathcal{V}\times\mathcal{V} \to \mathcal{V}$ in terms of ©, also called *combination*, such that if G and H are valuations for g and h, respectively, then G⊕H is the valuation for g∪h given by

$$(G\oplus H)(\mathbf{x}) = G(\mathbf{x}^{\downarrow g})©H(\mathbf{x}^{\downarrow h}) \qquad (2.1)$$

for all $\mathbf{x}\in \mathcal{W}_g$. We call G⊕H the *combination of G and H.*

In our optimization problem, © is simply addition, i.e.

$$(G\oplus H)(\mathbf{x}) = G(\mathbf{x}^{\downarrow g}) + H(\mathbf{x}^{\downarrow h}) \qquad (2.2)$$

Using (2.2), we can express the joint objective function F as follows $F = F_1\oplus F_2\oplus F_3$.

Marginalization. We assume that for each $h\subseteq\mathcal{X}$, there is a mapping $\downarrow h: \cup\{\mathcal{V}_g \mid g\supseteq h\} \to \mathcal{V}_h$, called *marginalization to h,* such that if G is a valuation for g and $g\supseteq h$, then $G^{\downarrow h}$ is a valuation for h. We call $G^{\downarrow h}$ the *marginal of G for h.*

For our optimization problem, we define marginalization as follows:

$$G^{\downarrow h}(\mathbf{x}) = \text{MIN}\{G(\mathbf{x},\mathbf{y}) \mid \mathbf{y}\in \mathcal{W}_{g-h}\} \qquad (2.3)$$

for all $\mathbf{x}\in \mathcal{W}_h$. Thus, if F is an objective function, then $F^{\downarrow\varnothing}(\blacklozenge)$ represents the minimum value of F.

In an optimization problem, besides the minimum value, we are usually also interested in finding a configuration where the minimum of the joint valuation is achieved. This motivates the following definition.

Solution for a Valuation. Suppose H is a valuation for h. We call $\mathbf{x}\in \mathcal{W}_h$ a *solution for H* if $H(\mathbf{x}) = H^{\downarrow\varnothing}(\blacklozenge)$.

Solution for a Variable. As we will see, once we have computed the minimum value of a valuation, computing a solution for the valuation is a matter of bookkeeping. Each time we eliminate a variable from a valuation using minimization, we store a table of configurations of the eliminated variable where the minimums are achieved. We can think of this table as a function. We call this function "a solution for the variable." Formally, we define a solution for a variable as follows. Suppose X is a variable, suppose g is a subset of variables containing X, and suppose G is a valuation for g. We call a function $\Psi_X: \mathcal{W}_{g-\{X\}} \to \mathcal{W}_X$ a *solution for X (with respect to G)* if

$$G^{\downarrow(g-\{X\})}(\mathbf{c}) = G(\mathbf{c},\Psi_X(\mathbf{c})) \qquad (2.4)$$

for all $\mathbf{c}\in \mathcal{W}_{g-\{X\}}$.

If $\mathcal{X}$ is a large set of variables, then a brute force computation of F and an exhaustive search of the set of all configurations of $\mathcal{X}$ to determine a solution for F is computationally infeasible. In the next section we will state axioms for combination and marginalization that make it possible to use local computation to compute the minimum value of F and a solution for F.

3. THE AXIOMS

We state three axioms. Axiom A1 is for combination. Axiom A2 is for marginalization. And Axiom A3 is for combination and marginalization.

A1. (*Commutativity and associativity of combination*). Suppose u, v, and w are values. Then u©w = v©u and u©(v©w) = (u©v)©w.

A2. (*Consonance of marginalization*). Suppose G is a valuation for g, and $k\subseteq h\subseteq g$. Then $(G^{\downarrow h})^{\downarrow k} = G^{\downarrow k}$.

A3. (*Distributivity of marginalization over combination*). Suppose G and H are valuations for g and h, respectively. Then $(G \oplus H)^{\downarrow g} = G \oplus (H^{\downarrow g \cap h})$.

It follows from axiom A1 that $\oplus$ is commutative and associative. Therefore, the combination of several valuations can be written without using parentheses. For example, $(...((F_1 \oplus F_2) \oplus F_3) \oplus ... \oplus F_k)$ can be simply written as $F_1 \oplus ... \oplus F_k$ without specifying the order in which to do the combination.

If we regard marginalization as a reduction of a valuation by deleting variables, then axiom A2 can be interpreted as saying that the order in which we delete the variables does not matter.

Axiom A3 is the crucial axiom that makes local computation of marginals and solution possible. Axiom A3 states that computation of $(G \oplus H)^{\downarrow g}$ can be done without having to compute $G \oplus H$.

For our optimization problem, it is easy to see that the definitions of combination in (2.2) and marginalization in (2.3) satisfy the three axioms.

4. SOLVING A VBS USING LOCAL COMPUTATION

Suppose we are given a collection of valuations $\{F_1, ..., F_k\}$ where each valuation F_i is for subset h_i of $\mathcal{X}$. The problem is (i) to find the minimum value of $F = F_1 \oplus ... \oplus F_k$ and (ii) to find a solution for F. We assume that combination and marginalization satisfy the three axioms.

We call the collection of subsets $\{h_1, ..., h_k\}$ for which we have valuations a *hypergraph* and denote it by $\mathcal{H}$.

Solving a VBS proceeds in three phases. In phase one, we arrange the subsets of variables in $\mathcal{H}$ in a "rooted Markov tree." In the phase two, we "propagate" the valuations $\{F_1, ..., F_k\}$ in the rooted Markov tree using a local message-passing scheme resulting in the computation of the marginal $F^{\downarrow\emptyset}$. In the phase three, we construct a solution for F again using a local message-passing scheme.

4.1. PHASE ONE: FINDING A ROOTED MARKOV TREE ARRANGEMENT

A *Markov tree* is a topological tree, whose vertices are subsets of variables, with the property that when a variable belongs to two distinct vertices, then every vertex lying on the path between these two vertices contains the variable.

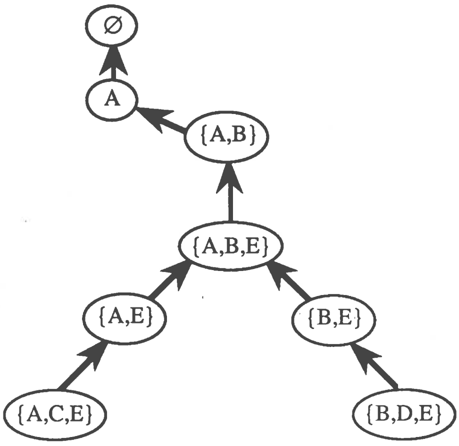
A *rooted Markov tree* is a Markov tree with the empty subset $\emptyset$ as the root and such that all edges in the tree are directed toward the root.

First, note that the only information we need in phase one is the set $\mathcal{H}$. Second, in arranging a set of subsets in a rooted Markov tree, we may have to add some subsets to the hypergraph $\mathcal{H}$. Figure 3 shows a rooted Markov tree arrangement of the subsets {A,C,E}, {A,B}, and {B,D,E}. Subsets {A,E}, {B,E}, {A,B,E}, {A}, and $\emptyset$ have been added during the arrangement process. Third, in general, there may be many rooted Markov tree arrangements of a hypergraph.

The computational efficiency of phase two depends on the sizes of the frames of the vertices of the Markov tree constructed in the phase one. Finding an optimal rooted Markov tree (a rooted Markov tree whose largest frame is as small as possible) has been shown to be a NP-complete problem [Arnborg *et al.*, 1987]. Thus we have to balance the computational efforts in the two phases. We should emphasize, however, that this is strictly a computational effort question. If computational effort is not an issue, then it does not matter which rooted Markov tree is used for propagating the valuations. All rooted Markov trees give the same final answer, i.e., the marginal of the joint valuation for the empty set. We will describe a heuristic called "one-step-look-ahead" due to Kong [1986] to find a good rooted Markov tree.

The method described below for arranging a hypergraph in a rooted Markov tree is due to Kong [1986] and Mellouli [1987].

Figure 3. A rooted Markov tree for the optimization problem.



Suppose $\mathcal{H}$ is a hypergraph on $\mathcal{X}$. To arrange the subsets in $\mathcal{H}$ in a Markov tree, we first pick a sequence of variables in $\mathcal{X}$. As we will see, each sequence of the variables gives rise to a Markov tree arrangement. Mellouli [1987] has shown that an optimal Markov tree arrangement can be found by picking some sequence. Of course, since there are an exponential number of sequences, finding an optimal sequence is, in general, a difficult problem.

Suppose we have a sequence of variables. Consider the first variable, say X_1, in the sequence. We add two subsets $g_1 = \cup\{h \mid X_1 \in h\}$ and $f_1 = g_1 - \{X_1\}$ to $\mathcal{H}$. We form the rooted Markov tree $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{h \in \mathcal{H} \mid X_i \in h\} \cup \{f_i\} \cup \{g_i\}$ and $\mathcal{E} = \{(h,g_i) \mid h \in (\mathcal{H} - \{g_i,f_i\}), X_i \in h\} \cup \{(g_i,f_i)\}$. We now consider X_1 as *marked* and subsets that contain X_1 as *arranged*. We repeat this process for the unarranged subsets until all variables are marked.

Consider the following set of instructions in pseudo-Pascal:

```
$u := \mathcal{X}$          {Initialization}
$\mathcal{H}_0 := \mathcal{H}$    {Initialization}
$\mathcal{V} := \varnothing$        {Initialization}
$\mathcal{E} := \varnothing$        {Initialization}
for i = 1 to n do
    begin
        Pick a variable from set u and call it $X_i$
        $u := u - \{X_i\}$
        $g_i := \cup\{h \in \mathcal{H}_{i-1} \mid X_i \in h\}$.
        $f_i := g_i - \{X_i\}$.
        $\mathcal{V} := \mathcal{V} \cup \{h \in \mathcal{H}_{i-1} \mid X_i \in h\} \cup \{f_i\} \cup \{g_i\}$
        $\mathcal{E} := \mathcal{E} \cup \{(h,g_i) \mid h \in (\mathcal{H}_{i-1} - \{g_i,f_i\}), X_i \in h\} \cup \{(g_i,f_i)\}$
        $\mathcal{H}_i := \{h \in \mathcal{H}_{i-1} \mid X_i \notin h\} \cup \{f_i\}$
    end {for}
```

After the execution of the above set of instructions, it is easily seen that the pair ($\mathcal{V}$, $\mathcal{E}$) is a rooted Markov tree arrangement of $\mathcal{H}$ where $\mathcal{V}$ denotes the set of vertices of the rooted Markov tree and $\mathcal{E}$ denotes the set of edges directed toward the root.

Kong [1986] has suggested a heuristic called *one-step-look-ahead* for finding a good Markov tree. This heuristic tells us which variable to mark next. As the name of the heuristic suggests, the variable that should be marked next is an unmarked variable X_i such that the cardinality of $\mathcal{W}_{f_i}$ is the smallest. Thus, the heuristic attempts to keep the sizes of the frames of the added vertices as small as possible by focussing only on the next subset added. In the optimization problem, a marking sequence selected by the one-step-look-ahead heuristic is CDEBA. Figure 4 illustrates the construction of a rooted Markov tree using this marking sequence. The resulting Markov tree is the same as that shown in Figure 3. See Zhang [1988] for other heuristics for good Markov tree construction.

4.2. Phase Two: Finding the Marginal of the Joint Valuation

Suppose we have arranged the hypergraph $\mathcal{H}$ in a rooted Markov tree. Let $\mathcal{H}'$ denote the set of subsets in the Markov tree. Clearly $\mathcal{H}' \supseteq \mathcal{H}$. To simplify the exposition, we assume there is exactly one valuation for each nonempty subset $h \in \mathcal{H}'$. If h is a subset that was added during the rooted Markov tree construction process, then we can associate the vacuous valuation (the valuation whose values are all 0) with it. On the other hand, if subset h had more than one valuation defined for it, then we can combine these valuations to obtain one valuation.

If we assume that the directed edges of a rooted Markov tree point from a child to its parent, then the rooted Markov tree defines a parent-child relation between adjacent vertices. If there is an edge (h_i,h_j) in the rooted Markov tree, we refer to h_j as h_i's *parent* and refer to h_i as h_j's *child*. Let $h_0 = \varnothing$ denote the *root* of the Markov tree. Let Pa(h) denote h's parent and let Ch(h) denote the set of h's children. Every non-root vertex has exactly one parent. Some vertices have no children and we call such vertices *leaves*. Note that the root has exactly one child.

In describing the process of finding the marginal of the joint valuation for the empty set, we will pretend that there is a processor at each vertex of the rooted Markov tree. Also, we assume these processors are connected using the same architecture as the Markov tree. In other words, each processor can directly communicate only with its parent and its children.

In the propagation process, each subset (except the root h_0) transmits a valuation to its parent. We call the valuation transmitted by subset h_i to its parent $Pa(h_i)$ a *valuation message* and denote it by $V^{h_i \to Pa(h_i)}$. Suppose $\mathcal{H}' = \{h_0, h_1, ..., h_k\}$ and let F_i denote the valuation associated with nonempty subset h_i. Then, the valuation message transmitted by a subset h_i to its parent $Pa(h_i)$ is given by

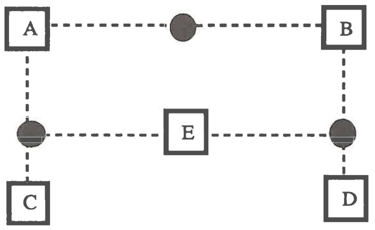
$$V^{h_i \to Pa(h_i)} = (\oplus\{V^{h \to h_i} \mid h \in Ch(h_i)\} \oplus F_i)^{\downarrow(h_i \cap Pa(h_i))} \qquad (4.1)$$

In words, the valuation message transmitted by a subset to its parent consists of the combination of the valuation messages it receives from its children plus its own valuation suitably marginalized. Note that the combination operation in (4.1) is on the frame $\mathcal{W}_{h_i}$.

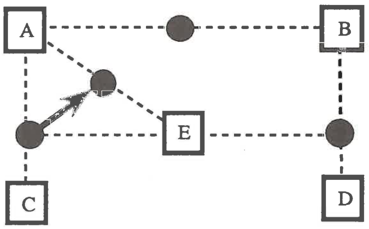
Expression (4.1) is a recursive formula. We need to start the recursion somewhere. Note that if subset h_i has no children, then $Ch(h_i) = \varnothing$ and the expression in (4.1) reduces to

$$V^{h_i \to Pa(h_i)} = (F_i)^{\downarrow(h_i \cap Pa(h_i))} \qquad (4.2)$$

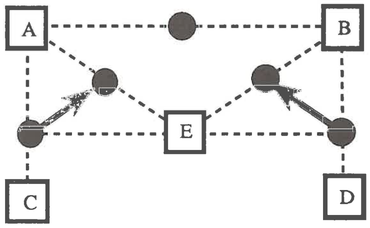
Figure 4. The construction of the rooted Markov tree for the optimization problem.



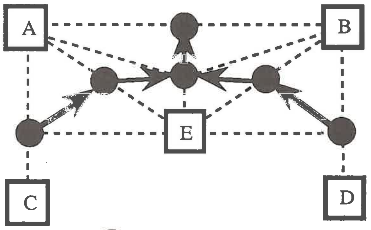
1. The initial hypergraph. Variables are shown as squares and subsets are shown as black disks. The elements of each subset are indicated by dotted lines.



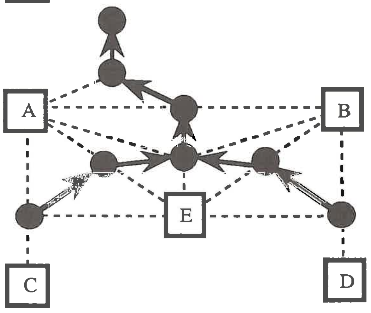
2. The Markov tree fragment after C is marked. Subset {A,E} is added to the hypergraph. Subset {A,C,E} is now arranged.



3. The Markov tree fragment after D is marked. Subset {B,E} is added to the hypergraph. Subset {B,D,E} is now arranged.



4. The Markov tree fragment after E is marked. Subset {A,B,E} is added to the hypergraph. Subsets {A,E}, {B.E} and {A,B,E} are now arranged.



5. The Markov tree fragment after B and then A are marked. Subsets {A} and ∅ are added to the hypergraph. All subsets are now arranged.

Thus the leaves of the Markov tree (the subsets that have no children) can send valuation messages to their parents right away. The others wait until they have heard from all their children before they send a valuation message to their parent.

The following theorem states that the valuation message from h_0's child to h_0 is indeed the desired marginal.

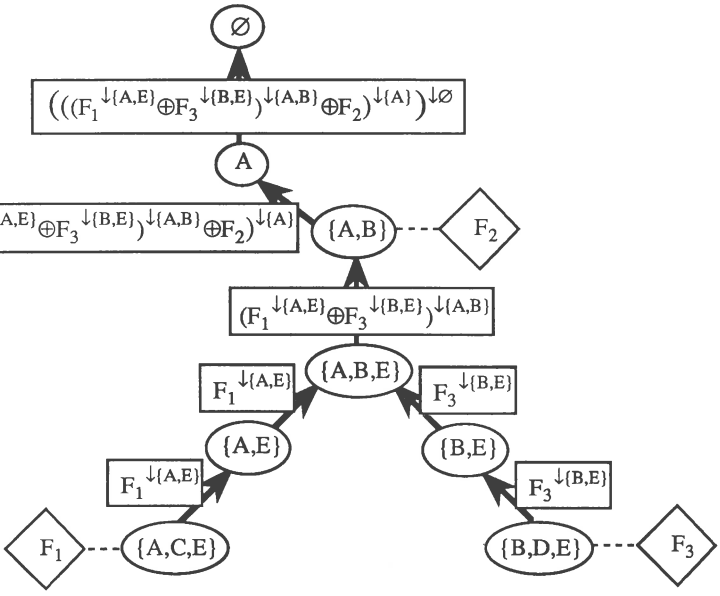
> ***Theorem 1.*** The marginal of the joint valuation for the empty set is equal to the message received by the root, i.e., $(F_1 \oplus \ldots \oplus F_k)^{\downarrow\varnothing} = V^{h \rightarrow h_0}$.

Theorem 1 is valid not only for optimization problems but for any VBS where axioms A1, A2, A3 hold. We give a simple proof of Theorem 1 in section 6.

The essence of the propagation method described above is to combine valuations on smaller frames instead of combining all valuations on the global frame associated with $\mathcal{X}$. To ensure that this method gives us the correct answers, the smaller frames have to be arranged in a rooted Markov tree.

Figure 5 shows the propagation of valuations in the optimization problem. Figure 6 shows the details of the valuation messages. As is clear from Figure 6, the minimum value of the joint objective function F is 2.

Figure 5. The propagation of valuations in the optimization problem. The valuation messages are shown as rectangles overlapping the corresponding edges. The valuations associated with the vertices are shown as diamonds linked to the corresponding vertices by dotted lines.



$\varnothing$
$(((F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}})^{\downarrow\{A,B\}} \oplus F_2)^{\downarrow\{A\}})^{\downarrow\varnothing}$
A
$((F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}})^{\downarrow\{A,B\}} \oplus F_2)^{\downarrow\{A\}}$
{A,B}
F_2
$(F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}})^{\downarrow\{A,B\}}$
{A,B,E}
$F_1^{\downarrow\{A,E\}}$
$F_3^{\downarrow\{B,E\}}$
{A,E}
{B,E}
$F_1^{\downarrow\{A,E\}}$
$F_3^{\downarrow\{B,E\}}$
F_1
{A,C,E}
{B,D,E}
F_3

Figure 6. The details of the valuation messages for the optimization problem.

$\mathcal{W}_{\{A,C,E\}}$	F_1
a c e	1
a c ~e	3
a ~c e	5
a ~c ~e	8
~a c e	2
~a c ~e	6
~a ~c e	2
~a ~c ~e	4

$\mathcal{W}_{\{A,E\}}$	$F_1^{\downarrow\{A,E\}}$	Ψ_C
a e	1	c
a ~e	3	c
~a e	2	c or ~c
~a ~e	4	~c

$\mathcal{W}_{\{B,D,E\}}$	F_3
b d e	0
b d ~e	5
b ~d e	6
b ~d ~e	3
~b d e	5
~b d ~e	1
~b ~d e	4
~b ~d ~e	3

$\mathcal{W}_{\{B,E\}}$	$F_3^{\downarrow\{B,E\}}$	Ψ_D
b e	0	d
b ~e	3	~d
~b e	4	~d
~b ~e	1	d

$\mathcal{W}_{\{A,B,E\}}$	$F_1^{\downarrow\{A,E\}}$	$F_3^{\downarrow\{B,E\}}$	$F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}}$
a b e	1	0	1
a b ~e	3	3	6
a ~b e	1	4	5
a ~b ~e	3	1	4
~a b e	2	0	2
~a b ~e	4	3	7
~a ~b e	2	4	6
~a ~b ~e	4	1	5

$\mathcal{W}_{\{A,B\}}$	$(F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}})^{\downarrow\{A,B\}}$	Ψ_E
a b	1	e
a ~b	4	~e
~a b	2	e
~a ~b	5	~e

$\mathcal{W}_{\{A,B\}}$	$(F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}})^{\downarrow\{A,B\}}$	F_2	$(F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}})^{\downarrow\{A,B\}} \oplus F_2$
a b	1	4	5
a ~b	4	8	12
~a b	2	0	2
~a ~b	5	5	10

$\mathcal{W}_{\{A\}}$	$((F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}})^{\downarrow\{A,B\}} \oplus F_2)^{\downarrow\{A\}}$	Ψ_B
a	5	b
~a	2	b

$\mathcal{W}_{\varnothing}$	$(((F_1^{\downarrow\{A,E\}} \oplus F_3^{\downarrow\{B,E\}})^{\downarrow\{A,B\}} \oplus F_2)^{\downarrow\{A\}})^{\downarrow\varnothing}$	Ψ_A
◆	2	~a

4.3. PHASE THREE: FINDING A SOLUTION

In phase two, each time we marginalize a variable, assume that we store the corresponding solution for that variable at the vertex where we do the marginalization. For example, in the optimization problem, we store a solution for C at vertex {A,C,E}, we store a solution for D at vertex {B,D,E}, we store a solution for E at vertex {A,B,E}, we store a solution for B at vertex {A,B}, and we store a solution for A at vertex {A} (see Figures 4, 5, and 6).

In this phase, each vertex of the rooted Markov tree sends a configuration to each of its children. We call the configuration transmitted by vertex h_i to its child $h_j \in Ch(h_i)$ as a *configuration message* and denote it by $c^{h_i \to h_j}$. $c^{h_i \to h_j}$ will always be an element of $\mathcal{W}_{h_i \cap h_j}$. As in phase two, we give a recursive definition of configuration messages.

The messages start at the root and travel toward the leaves. The configuration message from vertex $\varnothing$ to its child, say h_1, is given by

$$c^{\varnothing \to h_1} = \blacklozenge. \tag{4.3}$$

In general, consider vertex h_i. It receives a configuration message $c^{Pa(h_i) \to h_i}$ from its parent $Pa(h_i)$. Let h_j be a child of h_i. The configuration message from h_i to h_j depends on whether h_i has a solution for a variable stored at its location. (Remember that vertex h_i has a solution for X stored with it if $h_i - Pa(h_i) = \{X\}$).

If h_i has a solution for a variable stored at its location, then

$$c^{h_i \to h_j} = (c^{Pa(h_i) \to h_i}, \Psi_X(c^{Pa(h_i) \to h_i}))^{\downarrow (h_i \cap h_j)} \tag{4.4}$$

where X is such that $\{X\} = h_i - Pa(h_i)$.

If h_i has no solution for a variable stored at its location, then

$$c^{h_i \to h_j} = (c^{Pa(h_i) \to h_i})^{\downarrow (h_i \cap h_j)}. \tag{4.5}$$

We stop the message passing process when each vertex that has a solution stored at its location has received a configuration message.

Theorem 2. Suppose h_X denotes the vertex that has the solution for X stored at its location. Then $z \in \mathcal{W}_{\mathfrak{X}}$ given by

$$z^{\downarrow \{X\}} = \Psi_X(c^{Pa(h_X) \to h_X}), \quad \text{for every } X \in \mathfrak{X} \tag{4.6}$$

is a solution for $F_1 \oplus \ldots \oplus F_k$.

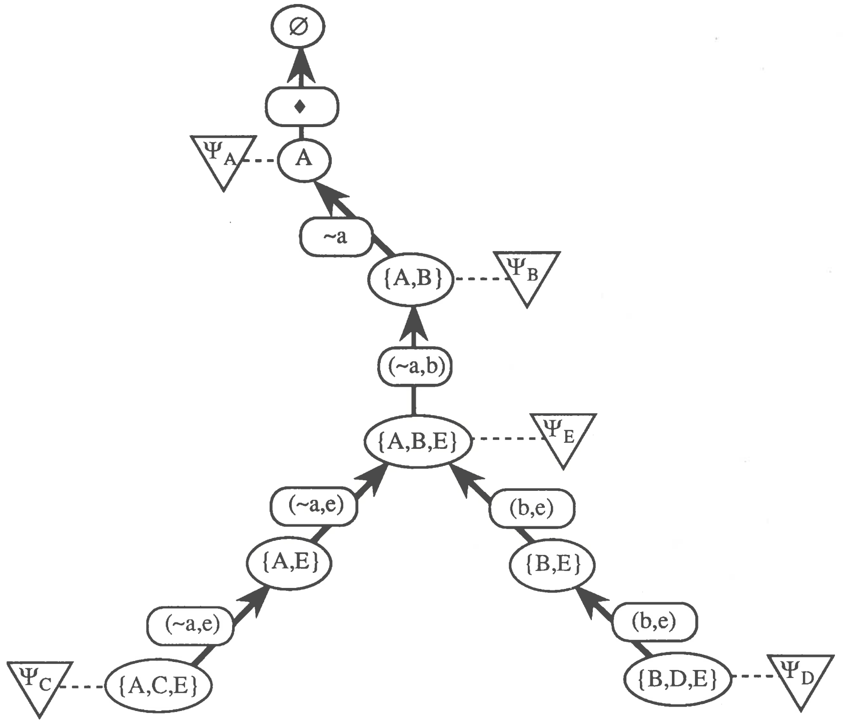
Figure 7 illustrates the message passing scheme for the optimization problem. As per Theorem 2, a solution for F is given by $(\Psi_A(c^{\varnothing \to \{A\}}), \Psi_B(c^{\{A\} \to \{A,B\}}), \Psi_C(c^{\{A,E\} \to \{A,C,E\}}), \Psi_D(c^{\{B,E\} \to \{B,D,E\}}), \Psi_E(c^{\{A,B\} \to \{A,B,E\}}))$. From Figures 6 and 7, we see that configurations (~a,b,c,d,e) and (~a,b,~c,d,e) are both solutions for F.

5. MITTEN'S AXIOMS FOR DYNAMIC PROGRAMMING

In optimization problems, the computational scheme described in section 4 is essentially the same as the method of non-serial dynamic programming (Nemhauser, 1966; Bertele and Brioschi, 1972). Bellman's dynamic programming methodology appealed to a principle of optimality that translates into axiom A3 with combination interpreted as addition and marginalization interpreted as maximization over the deleted variables (Bellman 1957). Mitten (1964) has described a more general framework for discrete dynamic programming. In this section, we describe Mitten's framework in terms of our notation.

Values and Valuations. The *value space* is $\mathbb{R}$, the set of real numbers. A *valuation for* h is a real-valued function on $\mathcal{W}_h$.

Figure 7. The propagation of configuration messages in the optimization problem. The configuration messages are shown as rectangles with rounded corners overlapping the corresponding edges. Note that the direction of messages is opposite to the direction of the edges. The solutions for the five variables are shown as inverted triangles attached to the vertices (where they are stored) by dotted lines.



Combination. There is a mapping ©: $\mathbb{R}\times\mathbb{R} \to \mathbb{R}$ that is commutative and associative. Define a mapping $\oplus:\mathcal{V}\times\mathcal{V} \to \mathcal{V}$ such that whenever G and H are valuations for g and h respectively, G⊕H is a valuation for $g\cup h$ given by

$$(G\oplus H)(\mathbf{x}) = G(\mathbf{x}^{\downarrow g}) \copyright H(\mathbf{x}^{\downarrow h})$$

for all $\mathbf{x}\in \mathcal{W}_{g\cup h}$.

Monotonicity of Combination. We say that © is *monotonic* if $u©v_1 \geq u©v_2$ whenever $v_1 \geq v_2$. Suppose H_1 and H_2 are valuations for h. We say that $H_1 \geq H_2$ if $H_1(\mathbf{x}) \geq H_2(\mathbf{x})$ for all $\mathbf{x}\in \mathcal{W}_h$. Note that if © is monotonic, then $G\oplus H_1 \geq G\oplus H_2$ whenever $H_1 \geq H_2$.

Marginalization. Define a mapping $\downarrow h$: $\cup\{\mathcal{V}_g \mid g\supseteq h\} \to \mathcal{V}_h$ such that whenever G is a valuation for g, $G^{\downarrow h}$ is a valuation for h given by

$$G^{\downarrow h}(\mathbf{x}) = \text{MAX}\{G(\mathbf{x},\mathbf{y}) \mid \mathbf{y}\in \mathcal{W}_{g-h}\} \qquad (5.1)$$

for all $\mathbf{x}\in \mathcal{W}_h$.

Theorem 3. Suppose the value space is $\mathbb{R}$ and suppose marginalization is defined as in (5.1). If © is monotonic, and G and H are valuations for g and h, respectively, then $(G\oplus H)^{\downarrow g} = G\oplus(H^{\downarrow g\cap h})$.

Thus monotonicity of © implies axiom A3. The other condition that Mitten requires is called *separability* and it amounts to a serial factorization of the joint objective function. In our framework, we do not require any particular structure for the factorization of the joint valuation.

6. CONCLUSIONS

In the introduction, we raised two questions: What is dynamic programming? And, when does dynamic programming work? The main contribution of this paper is the abstract framework of valuation-based systems consisting of variables, frames of variables, values, valuations, and two operations—combination and marginalization. Assuming that combination and marginalization satisfy three simple axioms, we have described a method for computing a solution for the joint valuation using only local computation. We can think of the framework and its solution method as the answer to the first question. The three axioms constitute one answer to the second question.

7. PROOFS

In this section, we provide proofs for the Theorems 1 and 2 stated in section 5 and Theorem 3 stated in section 6. We prove Theorems 1 and 2 only using axioms A1, A2 and A3. In other words, we do not assume that combination is addition and marginalization is minimization.

Lemma 7.1. Suppose $h_1, ..., h_k$ are the vertices of a rooted Markov tree. Suppose for $i = 1, ..., k$, vertex h_i has the valuation F_i associated with it, where F_i is a valuation for h_i. Suppose h_k is a leaf in the rooted Markov tree with parent h_{k-1}. Suppose $\mathfrak{X}$ denotes $h_1\cup...\cup h_k$ and $\mathfrak{X}'$ denotes $h_1\cup...\cup h_{k-1}$. Then

$$(F_1\otimes...\otimes F_k)^{\downarrow\mathfrak{X}'} = F_1\otimes...\otimes F_{k-2}\otimes(F_{k-1}\otimes F_k{}^{\downarrow(h_k\cap h_{k-1})}) \tag{7.1}$$

Proof of Lemma 7.1. Note that axiom A1 allows us to write the LHS of (7.1) as is written above. The result in (7.1) follows directly from axiom A3 by substituting $\mathfrak{X}'$ for g, h_k for h, $F_1\otimes...\otimes F_{k-1}$ for G, and F_k for H. Since h_k is a leaf in the rooted Markov tree with parent h_{k-1}, $h_k\cap h_{k-1} \subseteq h_{k-1}$. Thus $F_{k-1}\otimes F_k{}^{\downarrow(h_k\cap h_{k-1})}$ is a valuation for h_{k-1}. ■

Proof of Theorem 1. By axiom A2, $(F_1\otimes...\otimes F_k)^{\downarrow\varnothing}$ is obtained by sequentially marginalizing all variables in any sequence. A proof of this theorem is obtained by repeatedly applying the result of Lemma 7.1. At each step, a leaf of the rooted Markov tree sends a message to its parent, the parent combines this message with its own valuation, and the leaf is deleted from the tree. When the tree is reduced to only one vertex, the root, we have the result. ■

Next, we state a lemma that is needed to prove Theorem 2.

Lemma 7.2. Suppose $h_1, ..., h_k$ are the vertices of a rooted Markov tree. Suppose for $i = 1, ..., k$, vertex h_i has the valuation F_i associated with it, where F_i is a valuation for h_i. Suppose h_k is a leaf in the rooted Markov tree with parent h_{k-1} and suppose $h_k-(h_k\cap h_{k-1}) = \{X_j\}$ If Ψ_{X_j} is a solution for X_j (with respect to F_k), and **c** is a solution for $F_1\otimes...\otimes F_{k-2}\otimes F_{k-1}\otimes F_k{}^{\downarrow(h_k\cap h_{k-1})}$, then $(\mathbf{c}, \Psi_{X_j}(\mathbf{c}^{\downarrow(h_k\cap h_{k-1})}))$ is a solution for $F_1\otimes...\otimes F_k$.

Proof of Lemma 7.2. We need to prove that $(F_1\otimes...\otimes F_k)(\mathbf{c}, \Psi_{X_j}(\mathbf{c}^{\downarrow(h_k\cap h_{k-1})})) = (F_1\otimes...\otimes F_k)^{\downarrow\varnothing}(\blacklozenge)$. We have $(F_1\otimes...\otimes F_k)(\mathbf{c}, \Psi_{X_j}(\mathbf{c}^{\downarrow(h_k\cap h_{k-1})}))$

$= (F_1\otimes...\otimes F_{k-1})(\mathbf{c}) \copyright F_k(\mathbf{c}^{\downarrow(h_k\cap h_{k-1})}, \Psi_{X_j}(\mathbf{c}^{\downarrow(h_k\cap h_{k-1})}))$ (by definition of combination)

$= (F_1\otimes...\otimes F_{k-1})(\mathbf{c}) \copyright F_k{}^{\downarrow(h_k\cap h_{k-1})}(\mathbf{c}^{\downarrow(h_k\cap h_{k-1})})$

(since Ψ_{X_j} is a solution for X_j with respect to F_k)

$= (F_1\otimes...\otimes F_{k-2}\otimes F_{k-1}\otimes F_k{}^{\downarrow(h_k\cap h_{k-1})})(\mathbf{c})$ (by definition of combination)

$= (F_1\otimes...\otimes F_{k-2}\otimes F_{k-1}\otimes F_k{}^{\downarrow(h_k\cap h_{k-1})})^{\downarrow\varnothing}(\blacklozenge)$

(since **c** is a solution for $(F_1\otimes...\otimes F_{k-2}\otimes F_{k-1}\otimes F_k{}^{\downarrow(h_k\cap h_{k-1})})$)

$= ((F_1\otimes...\otimes F_k)^{\downarrow(h_1\cup...\cup h_k)-\{X_j\}})^{\downarrow\varnothing}(\blacklozenge)$ (using Lemma 7.1)

$= (F_1\otimes...\otimes F_k)^{\downarrow\varnothing}(\blacklozenge)$ (using axiom A2)

■

Proof of Theorem 2. A proof of this theorem is obtained by repeated application of Lemma 7.2. First we apply Lemma 7.2 for the entire rooted Markov tree. In our rooted Markov tree construction algorithm, if h_{k-1} is a parent of h_k, then either $h_k-(h_k\cap h_{k-1}) = \{X_j\}$ for some $j\in \mathcal{X}$, or $h_k\subseteq h_{k-1}$. The first case corresponds to the statement of Lemma 7.2. In the second case, when h_k sends a valuation message to h_{k-1}, there is no marginalization. Hence, there is no solution function stored at h_k. But in this case, $F_1\otimes...\otimes F_{k-2}\otimes F_{k-1}\otimes F_k{}^{\downarrow(h_k\cap h_{k-1})} = F_1\oplus...\oplus F_k$. Next, we apply Lemma 7.2 to the rooted Markov tree with vertex h_k and edge (h_k,h_{k-1}) deleted. And so on, until the only vertex left is $\varnothing$. But $\blacklozenge$ is the solution for $(F_1\otimes...\otimes F_k)^{\downarrow\varnothing}$. Thus the configuration messages as defined in (4.3), (4.4) and (4.5) give us the solution for $F_1\otimes...\otimes F_k$ as stated in Theorem 2. ■

Proof of Theorem 3. Suppose $\mathbf{x}\in \mathcal{W}_{g-h}$ and $\mathbf{y}\in \mathcal{W}_{g\cap h}$. Then

$$\begin{aligned}(G\oplus H)^{\downarrow g}(\mathbf{x},\mathbf{y}) &= \mathrm{MAX}\{(G\oplus H)(\mathbf{x},\mathbf{y},\mathbf{z}) \mid \mathbf{z}\in \mathcal{W}_{h-g}\}\\ &= \mathrm{MAX}\{G(\mathbf{x},\mathbf{y})\copyright H(\mathbf{y},\mathbf{z}) \mid \mathbf{z}\in \mathcal{W}_{h-g}\}\\ &\geq G(\mathbf{x},\mathbf{y})\copyright(\mathrm{MAX}\{H(\mathbf{y},\mathbf{z}) \mid \mathbf{z}\in \mathcal{W}_{h-g}\})\end{aligned}$$

In other words, $(G\oplus H)^{\downarrow g} \geq G\oplus(H^{\downarrow g\cap h})$. But since © is monotonic and $\mathrm{MAX}\{H(\mathbf{y},\mathbf{z}) \mid \mathbf{z}\in \mathcal{W}_{h-g}\} \geq H(\mathbf{y},\mathbf{z})$ for all $\mathbf{z}\in \mathcal{W}_{h-g}$, we have

$$G(\mathbf{x},\mathbf{y})\copyright(\mathrm{MAX}\{H(\mathbf{y},\mathbf{z}) \mid \mathbf{z}\in \mathcal{W}_{h-g}\}) \geq G(\mathbf{x},\mathbf{y})\copyright H(\mathbf{y},\mathbf{z})$$

for all $\mathbf{z}\in \mathcal{W}_{h-g}$. In particular, this inequality must hold for the maximum of the RHS with respect to **z**, i.e., $G(\mathbf{x},\mathbf{y})\copyright(\mathrm{MAX}\{H(\mathbf{y},\mathbf{z}) \mid \mathbf{z}\in \mathcal{W}_{h-g}\}) \geq \mathrm{MAX}\{G(\mathbf{x},\mathbf{y})\copyright H(\mathbf{y},\mathbf{z}) \mid \mathbf{z}\in \mathcal{W}_{h-g}\}$, i.e., $G\oplus(H^{\downarrow g\cap h}) \geq (G\oplus H)^{\downarrow g}$. Since we have already shown that $(G\oplus H)^{\downarrow g} \geq G\oplus(H^{\downarrow g\cap h})$, we have the result. ■

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under grant IRI-8902444. I am grateful for comments from Judea Pearl, Glenn Shafer and Po-Lung Yu on previous drafts of this paper.

REFERENCES

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1974), *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA.

Arnborg, S., Corneil, D. G. and Proskurowski, A. (1987), "Complexity of finding embeddings in a k-tree," *SIAM Journal of Algebraic and Discrete Methods*, **8**, 277–284.

Bellman, R. E. (1957), *Dynamic Programming*, Princeton University Press, Princeton, NJ.

Bertele, U. and Brioschi, F. (1972), *Nonserial Dynamic Programming*, Academic Press, New York, NY.

Dempster, A. P. and Kong, A. (1988), "Uncertain evidence and artificial analysis," *Journal of Statistical Planning and Inference*, **20**, 355–368.

Kong, A. (1986), "Multivariate belief functions and graphical models," Ph.D. thesis, Department of Statistics, Harvard University, Cambridge, MA.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988), "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *Journal of the Royal Statistical Society*, series B, **50**(2), 157–224.

Mellouli, K. (1987), "On the propagation of beliefs in networks using the Dempster-Shafer theory of evidence," Ph.D. thesis, School of Business, University of Kansas, Lawrence, KS.

Mitten, L. G. (1964), "Composition principles for synthesis of optimal multistage processes," *Operations Research*, **12**, 610–619.

Nemhauser, G. L. (1966), *Introduction to Dynamic Programming*, John Wiley and Sons, New York, NY.

Pearl, J. (1986), "Fusion, propagation and structuring in belief networks," *Artificial Intelligence*, **29**, 241–288.

Shenoy, P. P. (1989), "A valuation-based language for expert systems," *International Journal for Approximate Reasoning*, **3**(5), 383–411, 1989.

Shenoy, P. P. (1990a), "Valuation-based systems for Bayesian decision analysis," Working Paper No. 220, School of Business, University of Kansas, Lawrence, KS.

Shenoy, P. P. (1990b), "A new method for representing and solving Bayesian decision problems," Working Paper No. 223, School of Business, University of Kansas, Lawrence, KS.

Shenoy, P. P. (1990c), "Consistency in valuation-based systems," Working Paper No. 216, School of Business, University of Kansas, Lawrence, KS.

Shenoy, P. P. (1990d), "Valuation-based systems for propositional logic," in Ras, Z. W., Zemankova, M., and Emrich, M. L., eds., *Methodologies for Intelligent Systems*, **5**, 305–312, North-Holland, Amsterdam.

Shenoy, P. P. (1991), "Valuation-based systems: A framework for managing uncertainty in expert systems," Working Paper No. 226, School of Business, University of Kansas, Lawrence, KS.

Shenoy, P. P. and Shafer, G. (1986), "Propagating belief functions using local computations," *IEEE Expert*, **1**(3), 43–52.

Shenoy, P. P. and Shafer, G. (1988), "Constraint propagation," Working Paper No. 208, School of Business, University of Kansas, Lawrence, KS.

Zhang, L. (1988), "Studies on finding hypertree covers of hypergraphs," Working Paper No. 198, School of Business, University of Kansas, Lawrence, KS.