Appeared in: Gammerman, A. (ed.), Computational Learning and Probabilistic Reasoning, 1996, 259–275, JohnWiley & Sons, Ltd.

AXIOMS FOR DYNAMIC PROGRAMMING

Prakash P. Shenoy

School of Business, University of Kansas, Summerfield Hall, Lawrence, KS 66045-7585, USA
pshenoy@ku.edu

ABSTRACT

This paper describes an abstract framework, called valuation network (VN), for representing and solving discrete optimization problems. In VNs, we represent information in an optimization problem using functions called valuations. Valuations represent factors of an objective function. Solving a VN involves using two operators called combination and marginalization. The combination operator tells us how to combine the factors of the objective function to form the global objective function (also called joint valuation). Marginalization is either maximization or minimization. Solving a VN can be described simply as finding the marginal of the joint valuation for the empty set. We state some simple axioms that combination and marginalization need to satisfy to enable us to solve a VN using local computation. We describe a fusion algorithm for solving a VN using local computation. For optimization problems, the fusion algorithm reduces to non-serial dynamic programming. Thus the fusion algorithm can be regarded as an abstract description of the dynamic programming method, and the axioms can be viewed as conditions that permit the use of dynamic programming.

Subject classification: Dynamic programming: theory, algorithm.

1 INTRODUCTION

The main objective of this paper is to describe axioms that permit the use of dynamic programming methodology. We describe an abstract framework for representing and solving discrete optimization problems. The abstract framework is called a valuation network (VN). In VNs, information is represented by a collection of functions called valuations. The system includes two operators called combination and marginalization that operate on valuations. Combination tells us how to combine the valuations. Marginalization tells us how to reduce valuations by deleting variables. Solving a VN can be described simply as finding the marginal of the joint valuation for the empty set. The joint valuation is the valuation obtained by combining all valuations. We describe a fusion algorithm for solving a VN using local computation. We describe three axioms for combination and marginalization that make local computation possible. We compare these axioms with the axioms proposed by Mitten [1964]. There are several reasons why this is useful.

First, I initially proposed VNs for managing uncertainty in expert systems [Shenoy 1989, Shenoy and Shafer 1990, Shenoy 1992a]. Here I show that these systems also have the expressive power to represent and solve discrete optimization problems. Two of the three axioms described here are exactly the same as the axioms described in [Shenoy 1992a]. One axiom is slightly stronger than the corresponding axiom in [Shenoy 1992a].

Second, problems in Bayesian decision analysis involve managing probability and optimization. That these problems can be solved in a common framework suggests that Bayesian decision problems also can be represented and solved in the VN framework. Indeed, [Shenoy 1992b, 1993] show that this is true. In fact, the fusion algorithm is always computationally more efficient than the arc-reversal method of influence diagrams. And for symmetric decision problems, the fusion algorithm is more efficient than the backward recursion method of decision trees [Shenoy 1994a].

Third, the fusion algorithm when applied to optimization problems results in a method called non-serial dynamic programming [Nemhauser 1966, Bertele and Brioschi 1972]. Thus, in an abstract sense, the local computational algorithms that have been described by Pearl [1986], Shenoy and Shafer [1986], Dempster and Kong [1988], Lauritzen and Spiegelhalter [1988], and Shafer and Shenoy [1990] are just instances of 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 an abstract definition of a problem and an abstract method solving the problem. The abstract method for solving the problem, the fusion algorithm, can be thought of as a general definition of the dynamic programming method.

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. These axioms are new. They are weaker than those proposed by Mitten [1964].

Sixth, the VN described here can be easily adapted to represent propositional logic [Shenoy 1990, 1994b] and constraint satisfaction problems [Shenoy and Shafer 1988].

An outline of this paper is as follows. In Section 2, we describe the VN framework and show how a discrete optimization problem fits in this framework. In Section 3, we state some simple axioms that justify the use of local computation in solving VNs. In Section 4, we describe a fusion algorithm for solving a VN using local computation. Throughout the paper, we use one example to illustrate all definitions and to illustrate the fusion algorithm. 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 all theorems in the paper.

2 VALUATION NETWORKS AND OPTIMIZATION

A valuation network consists of valuations, combination, and marginalization. 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 A, B, C, D, and E. Each variable has two possible values. Let a and ~a denote the possible values of A, etc. The global objective function ϕ for variables A, B, C, D, and E factors additively as follows: $\phi = \phi_1 + \phi_2 + \phi_3$, where ϕ_1 is a function for E, A, and C, ϕ_2 is a function for B and A, and ϕ_3 is a function for E, B, and D. Table I shows the details of these three functions. The problem is to find the minimum value of ϕ and to find a configuration of all variables where the minimum value is achieved.

Variables, State Spaces, and Configurations. We use the symbol Ω_X for the set of possible values of a variable X, and we call Ω_X the *state space* for X. We are concerned with a finite set Θ of variables, and we assume that all the variables in Θ have finite state spaces.

Given a finite non-empty set h of variables, let Ω_h denote the Cartesian product of Ω_X for X in h, i.e., $\Omega_h = \times\{\Omega_X \mid X \in h\}$. We call Ω_h the *state space* for h. We call elements of Ω_h *configurations* of h. Lower-case bold-faced letters, **x**, **y**, etc., will denote configurations. If **x** is a configuration of g, **y** is a configuration of h, and $g \cap h = \varnothing$, then (**x**, **y**) will denote 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 state space for the empty set $\varnothing$ consists of a single element, and we use the symbol ♦ to name that element; $\Omega_\varnothing = \{♦\}$. If **x** is a configuration of g, then (**x**, ♦) is simply **x.**

Values and Valuations. We are concerned with a set Δ whose elements are called *values*. Δ may be finite or infinite. Given a set h of variables, we call any function $\sigma: \Omega_h \rightarrow \Delta$, *a valuation for h*, and we call h the *domain* of σ. Note that to specify a valuation σ for $\varnothing$, we need to specify only a single value, $\sigma(♦)$. We will use lower-case Greek letters to denote valuations.

In our optimization problem, the set Δ is the set of real numbers, and we have three valuations ϕ_1, ϕ_2 and ϕ_3. ϕ_1 is a valuation for {E, A, C}, ϕ_2 is a valuation for {B, A}, and ϕ_3 is a valuation

Table I.

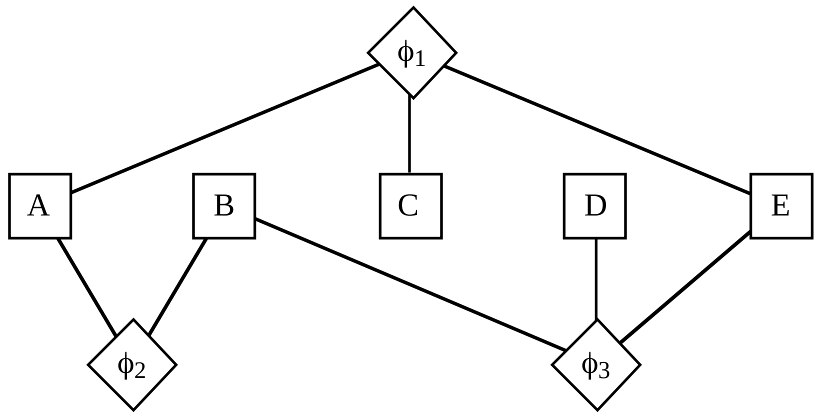
The Factors of the Objective Function, ϕ_1, ϕ_2, and ϕ_3

$\Omega_{\{E, A, C\}}$			ϕ_1
e	a	c	1
e	a	~c	5
e	~a	c	2
e	~a	~c	2
~e	a	c	3
~e	a	~c	8
~e	~a	c	6
~e	~a	~c	4

$\Omega_{\{B, A\}}$		ϕ_2
b	a	4
b	~a	0
~b	a	8
~b	~a	5

$\Omega_{\{E, B, D\}}$			ϕ_3
e	b	d	0
e	b	~d	6
e	~b	d	5
e	~b	~d	4
~e	b	d	5
~e	b	~d	3
~e	~b	d	1
~e	~b	~d	3

Figure 1. A valuation network for the optimization problem.



for {E, B, D}. Figure 1 shows a graphical depiction of the qualitative features of the optimization problem. We call such a graph a *valuation network*. In a valuation network, rectangular nodes represent variables, and diamond-shaped nodes represent valuations. Each valuation is connected to the variables in its domain by undirected edges.

Let ϑ_h denote the set of valuations for *h*, and let ϑ denote the set of valuations, i.e., $\vartheta = \cup\{\vartheta_h | h \subseteq \Theta\}$.

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 ©: $\Delta \times \Delta \rightarrow \Delta$ called *combination* so that if u, v $\in \Delta$, then u©v is the value representing the combination of u and v. We define a mapping $\oplus: \vartheta \times \vartheta \rightarrow \vartheta$ in terms of ©, also called *combination*, such that if γ and η are valuations for *g* and *h*, respectively, then $\gamma \oplus \eta$ is the valuation for $g \cup h$ given by

$$(\gamma \oplus \eta)(\mathbf{x}) = \gamma(\mathbf{x}^{\downarrow g}) © \eta(\mathbf{x}^{\downarrow h}) \tag{2.1}$$

for all $\mathbf{x} \in \Omega_{g \cup h}$. We call $\gamma \oplus \eta$ the *combination* of γ and η.

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

$$(\gamma \oplus \eta)(\mathbf{x}) = \gamma(\mathbf{x}^{\downarrow g}) + \eta(\mathbf{x}^{\downarrow h}) \tag{2.2}$$

Using (2.2), we can express the global objective function ϕ as follows: $\phi = \phi_1 \oplus \phi_2 \oplus \phi_3$.

Marginalization. We assume that for each $h \subseteq \Theta$, and for each $X \in h$, there is a mapping $\downarrow(h - \{X\}): \vartheta_h \rightarrow \vartheta_{h - \{X\}}$, called *marginalization to h – {X}*, such that if η is a valuation for *h* and $X \in h$, then $\eta^{\downarrow(h - \{X\})}$ is a valuation for *h* – {X}. We call $\eta^{\downarrow(h - \{X\})}$ the *marginal* of η for h – {X}.

For our optimization problem, we define marginalization as follows:

$$\eta^{\downarrow(h - \{X\})}(\mathbf{y}) = \text{MIN}\{\eta(\mathbf{y}, \mathbf{x}) \mid \mathbf{x} \in \Omega_X\} \quad (2.3)$$

for all $\mathbf{y} \in \Omega_{h - \{X\}}$.

If γ is a valuation for g, and $h \subseteq g$, then $\gamma^{\downarrow h}$ will denote the marginal of γ for h obtained by sequentially marginalizing all variables in $g - h$ out of γ in some sequence. In the next section, we will state an axiom that says that the sequence in which variables are marginalized out of a valuation does not affect the final answer. This axiom allows us to use this notation. Note that if ϕ is a global objective function for Θ, then, using this notation, $\phi^{\downarrow\emptyset}(\blacklozenge)$ represents the minimum value of ϕ.

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

Solution for a Valuation. Suppose η is a valuation for h. We call $\mathbf{x} \in \Omega_h$ a *solution* for η if $\eta(\mathbf{x}) = \eta^{\downarrow\emptyset}(\blacklozenge)$.

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 h is a subset of variables containing X, and suppose η is a valuation for h. We call a function $\Psi_X: \Omega_{h - \{X\}} \to \Omega_X$ a *solution* for X (with respect to η) if

$$\eta^{\downarrow(h - \{X\})}(\mathbf{c}) = \eta(\mathbf{c}, \Psi_X(\mathbf{c})) \quad (2.4)$$

for all $\mathbf{c} \in \Omega_{h - \{X\}}$.

In summary, a VN consists of a set of variables Θ, a state space for each variable $\{\Omega_X\}_{X \in \Theta}$, a set of values Δ, a collection of valuations $\{\sigma_1, ..., \sigma_k\}$, a definition of combination ©, and a definition of marginalization $\downarrow$. Given a VN, there are two problems of interest. First, we would like to compute $(\sigma_1 \oplus ... \oplus \sigma_k)^{\downarrow\emptyset}(\blacklozenge)$. (In an optimization problem, this represents the minimum value of the joint objective function.) Second, we would like to compute a solution for $\sigma_1 \oplus ... \oplus \sigma_k$. (In an optimization problem, this represents an optimal solution.)

If Θ is a large set of variables, and $\sigma = \sigma_1 \oplus ... \oplus \sigma_k$ is a valuation for Θ, then a brute force computation of σ and an exhaustive search of the set of all configurations of Θ to determine a solution for σ are not computationally tractable. In the next section we will state three axioms for combination and marginalization that allow the use of local computation to compute the minimum value of σ and to compute a solution for σ.

3 THE AXIOMS

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

Axiom A1 (*Commutativity and associativity of combination for values*): Suppose u, v, w $\in \Delta$. Then

$$u©v = v©u, \text{ and } u©(v©w) = (u©v)©w.$$

Axiom A2 (*Order of deletion does not matter*): Suppose γ is a valuation for g, and suppose $X_1, X_2 \in g$. Then

$$(\gamma^{\downarrow(g - \{X_1\})})^{\downarrow(g - \{X_1, X_2\})} = (\gamma^{\downarrow(g - \{X_2\})})^{\downarrow(g - \{X_1, X_2\})}.$$

Axiom A3 (*Distributivity of marginalization over combination*): Suppose γ and η are valuations for g and h, respectively, suppose $X \in h$, and suppose $X \notin g$. Then

$$(\gamma\oplus\eta)^{\downarrow((g\cup h) - \{X\})} = \gamma\oplus(\eta^{\downarrow(h - \{X\})}).$$

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, $(...((\sigma_1\oplus\sigma_2)\oplus\sigma_3)\oplus...\oplus\sigma_k)$ can be simply written as $\oplus\{\sigma_1, ..., \sigma_k\}$ or as $\sigma_1\oplus...\oplus\sigma_k$ without specifying the order in which the combination is done.

If we regard marginalization as a coarsening of a valuation by deleting variables, then Axiom A2 says that the order in which the variables are deleted does not matter. One implication of this axiom is that $(\gamma^{\downarrow(g - \{X_1\})})^{\downarrow(g - \{X_1, X_2\})}$ can be written simply as $\gamma^{\downarrow(g - \{X_1, X_2\})}$, i.e., we need not indicate the order in which the variables are deleted.

Axiom A3 is the crucial axiom that makes local computation possible. Axiom A3 states that computation of $(\gamma\oplus\eta)^{\downarrow((g\cup h) - \{X\})}$ can be accomplished without having to compute $\gamma\oplus\eta$. The combination operation in $\gamma\oplus\eta$ is on the state space for $g\cup h$, whereas the combination operation in $\gamma\oplus(\eta^{\downarrow(h - \{X\})})$ is on the state space for $(g\cup h) - \{X\}$.

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

4 A FUSION ALGORITHM

In this section, we describe a fusion algorithm for solving a VN using local computation, i.e., for computing exactly the marginal of the joint valuation for the empty set without explicitly computing the joint valuation.

Suppose $\rho = \{\Theta, \{\Omega_X\}_{X\in\Theta}, \Delta, \{\sigma_1,..., \sigma_k\}, ©, \downarrow\}$ is a VN, and suppose © and $\downarrow$ satisfy Axioms A1–A3. We will describe a fusion algorithm for computing the marginal $(\sigma_1\oplus\ldots\oplus\sigma_k)^{\downarrow\emptyset}$ and for computing a solution for $\sigma_1\oplus\ldots\oplus\sigma_k$ using local computation

The basic idea of the method is to successively delete all variables from the VN. Any sequence may be used. Axiom A2 tells us that all deletion sequences will lead to the same answers. But different deletion sequences may involve different computational costs. We will comment on good deletion sequences at the end of this section.

When we delete a variable, we have to do a “fusion” operation on the valuations. Consider a set of m valuations $\{\alpha_1, ..., \alpha_m\}$. Suppose α_i is a valuation for a_i for i = 1, ..., m. Let $Fus_{X_j}\{\alpha_1, ..., \alpha_m\}$ denote the collection of valuations after fusing the valuations in the set $\{\alpha_1, ..., \alpha_k\}$ with respect to variable X_j. Then

$$Fus_{X_j}\{\alpha_1, ..., \alpha_m\} = \{\alpha^{\downarrow(g_j - \{X_j\})}\} \cup \{\alpha_i \mid X_j \notin a_i\}, \tag{4.1}$$

where $\alpha = \oplus\{\alpha_i \mid X_j \in a_i\}$, and $g_j = \cup\{a_i \mid X_j \in a_i\}$. After fusion, the set of valuations is changed as follows. All valuations that have X_j in their domain are combined, and the resulting valuation is marginalized such that X_j is eliminated from its domain. The valuations that do not have X_j in their domains remain unchanged.

When we compute the marginal $\alpha^{\downarrow(g_j - \{X_j\})}$ in (4.1), assume that we store a solution for X_j with respect to α, $\Psi_{X_j}: \Omega_{g_j - \{X_j\}} \rightarrow \Omega_{X_j}$. We will describe a method for constructing a solution for the joint valuation using these solutions.

We are ready to state the main theorem which describes the fusion algorithm.

Theorem 1 (*Fusion Algorithm*). Suppose $\rho = \left\{\Theta, \{\Omega_X\}_{X \in \Theta}, \Delta, \{\sigma_1, ..., \sigma_k\}, ©, \downarrow\right\}$ is a VN satisfying Axioms A1–A3. Suppose $X_1X_2...X_n$ is a sequence of variables in Θ. Then

$$(\sigma_1 \oplus \ldots \oplus \sigma_k)^{\downarrow\varnothing} = \oplus\ Fus_{X_n}\left\{...\ Fus_{X_2}\{Fus_{X_1}\{\sigma_1, ..., \sigma_k\}\}\right\}.$$

We give a simple proof of Theorem 1 in section 7. The essence of the fusion algorithm is to combine valuations on smaller state spaces instead of combining all valuations on the global state space associated with Θ.

Figure 2 shows a graphical depiction of the fusion algorithm for the optimization problem using deletion sequence CDABE. In this figure, valuation network labeled 0 is the original network. Valuation network 1 is the result after fusion with respect to C. Since there is only one valuation with C in its domain, there is no combination here, only marginalization of ϕ_1 to {A, E}. Valuation network 2 is the result after fusion with respect to D. Again, since there is only one valuation with D in its domain, there is no combination here, only marginalization of ϕ_3 to {B, E}. Valuation network 3 is the result after fusion with respect to A. Since A is in the domain of $\phi_1^{\downarrow\{A,E\}}$ and ϕ_2, we first combine these two valuations and then marginalize A out of the resulting valuation. Valuation network 4 is the result after fusion with respect to B. And finally, valuation network 5 is the result after fusion with respect to E. Theorem 1 tells us that $[(\phi_1^{\downarrow\{A,E\}} \oplus \phi_2)^{\downarrow\{B,E\}} \oplus \phi_3^{\downarrow\{B,E\}}]^{\downarrow\varnothing} = (\phi_1 \oplus \phi_2 \oplus \phi_3)^{\downarrow\varnothing}$. Table II shows the details of the computations in the fusion algorithm. The minimum value of the objective function is 2.

Figure 2. The fusion algorithm for the optimization problem using deletion sequence CDABE.

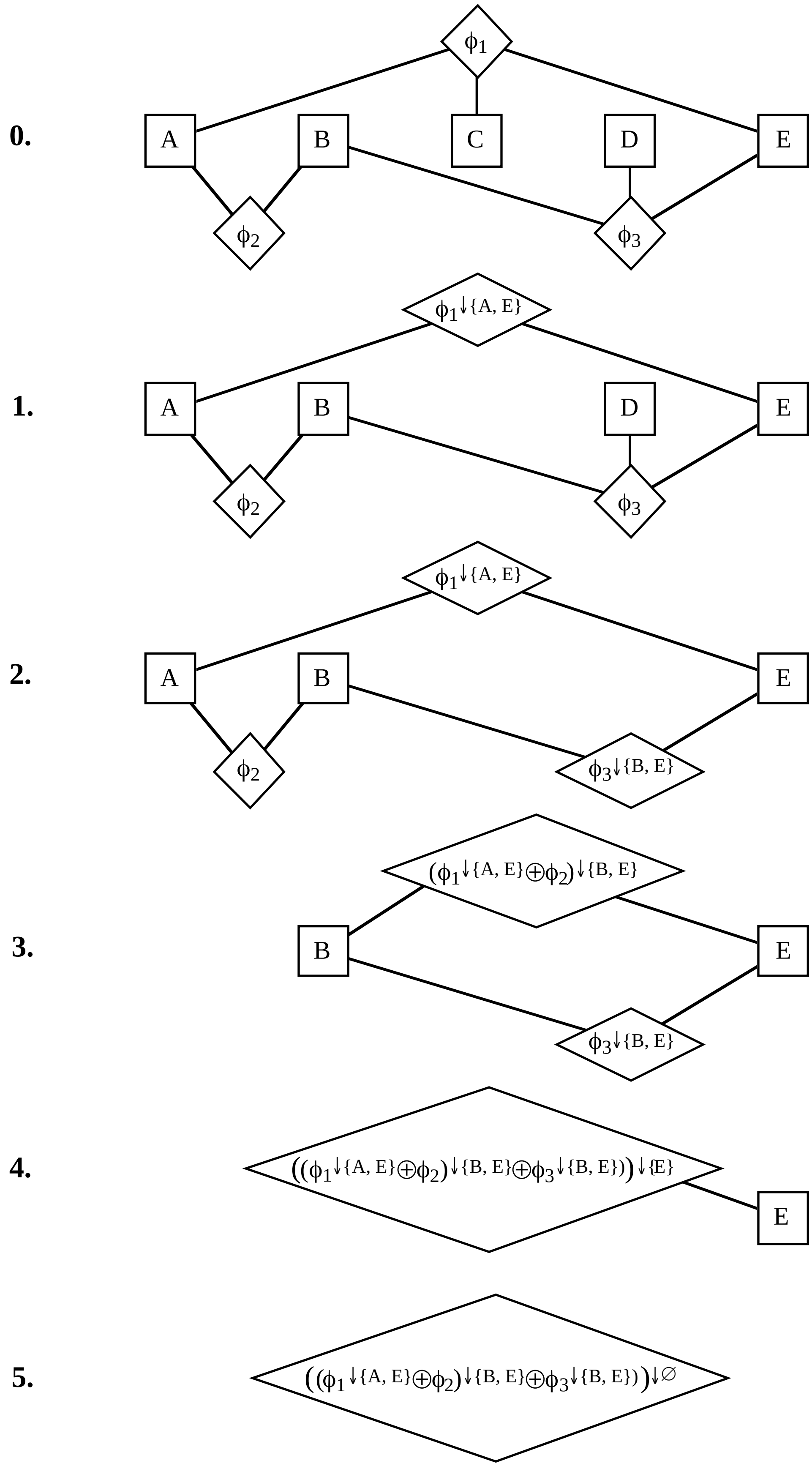


Table II.

The Numerical Computations in the Fusion Algorithm for the Optimization Problem

$\Omega_{\{E, A, C\}}$	ϕ_1	$\phi_1^{\downarrow\{E, A\}}$	Ψ_C
e a c	1	1	c
e a ~c	5		
e ~a c	2	2	c or ~c
e ~a ~c	2		
~e a c	3	3	c
~e a ~c	8		
~e ~a c	6	4	~c
~e ~a ~c	4		

$\Omega_{\{E, B, D\}}$	ϕ_3	$\phi_3^{\downarrow\{E, B\}}$	Ψ_D
e b d	0	0	d
e b ~d	6		
e ~b d	5	4	~d
e ~b ~d	4		
~e b d	5	3	~d
~e b ~d	3		
~e ~b d	1	1	d
~e ~b ~d	3		

$\Omega_{\{E, B, A\}}$	$\phi_1^{\downarrow\{E, A\}}$	ϕ_2	$\phi_1^{\downarrow\{E, A\}}\oplus\phi_2$	$(\phi_1^{\downarrow\{E, A\}}\oplus\phi_2)^{\downarrow\{E, B\}}$	Ψ_A
e b a	1	4	5	2	~a
e b ~a	2	0	2		
e ~b a	1	8	9	7	~a
e ~b ~a	2	5	7		
~e b a	3	4	7	4	~a
~e ~d ~a	4	0	4		
~e ~b a	3	8	11	9	~a
~e ~b ~a	4	5	9		

$\Omega_{\{E, B\}}$	$(\phi_1^{\downarrow\{E, A\}}\oplus\phi_2)^{\downarrow\{E, B\}}$	$\phi_3^{\downarrow\{E, B\}}$	$(\phi_1^{\downarrow\{E,A\}}\oplus\phi_2)^{\downarrow\{E,B\}}\oplus\phi_3^{\downarrow\{E, B\}}$	$[(\phi_1^{\downarrow\{E,A\}}\oplus\phi_2)^{\downarrow\{E,B\}}\oplus\phi_3^{\downarrow\{E, B\}}]^{\downarrow\{E\}}$	Ψ_B
e b	2	0	2	2	b
e ~b	7	4	11		
~e b	4	3	7	7	b
~e ~b	9	1	10		

Ω_E	$[(\phi_1^{\downarrow\{E,A\}}\oplus\phi_2)^{\downarrow\{E,B\}}\oplus\phi_3^{\downarrow\{E, B\}}]^{\downarrow\{E\}}$	$[(\phi_1^{\downarrow\{E,A\}}\oplus\phi_2)^{\downarrow\{E,B\}}\oplus\phi_3^{\downarrow\{E, B\}}]^{\downarrow\varnothing}(\blacklozenge)$	$\Psi_E(\blacklozenge)$
e	2	2	e
~e	7		

When we implement the fusion algorithm, each time we marginalize a variable, assume that we store a solution for that variable. If we use deletion sequence $X_1X_2...X_n$, then at the end of the fusion algorithm, we have for each variable X_j, a solution $\Psi_{X_j}: \Omega_{g_j - \{X_j\}} \to \Omega_{X_j}$, where g_j is as defined in (4.1). Note that $g_1 = \cup\{h_i \mid X_1 \in h_i\}$. The precise definition of g_2 will depend on the valuations in the set $Fus_{X_1}\{\sigma_1,..., \sigma_k\}$. However, since X_1 has been deleted, $g_2 \subseteq \{X_2, ..., X_n\}$ and $X_2 \in g_2$. In general, $g_i \subseteq \{X_i, ..., X_n\}$, and $X_i \in g_i$ for i = 1, ..., n. Note that $g_n = \{X_n\}$.

Theorem 2 describes a recursive method for constructing a solution for the joint valuation. The solution is constructed piecemeal starting with the component in Ω_{X_n} and working sequentially opposite to the deletion sequence.

Theorem 2 (*Constructing a Solution*). Suppose $\rho = \{\Theta, \{\Omega_X\}_{X \in \Theta}, \Delta, \{\sigma_1,..., \sigma_k\}, ©, \downarrow\}$ a VN satisfying Axioms A1–A3. Suppose $X_1X_2...X_n$ is a sequence of variables in Θ. Suppose $\Psi_{X_j}: \Omega_{g_j - \{X_j\}} \to \Omega_{X_j}$ is a solution for X_j computed during fusion of $Fus_{X_{j-1}}\{...Fus_{X_2}\{Fus_{X_1}\{\sigma_1,...,\sigma_k\}\}\}$ with respect to X_j, for j = 1, ..., n. Then $\mathbf{z} \in \Omega_\Theta$ given by

$$\mathbf{z}^{\downarrow\{X_j\}} = \Psi_{X_j}(\mathbf{z}^{\downarrow(g_j - \{X_j\})}) \text{ for } j = n, n-1, ..., 1,$$

is a solution for $\sigma_1 \oplus ... \oplus \sigma_k$.

To illustrate Theorem 2, consider the optimization problem. We computed the minimum value of $\phi_1 \oplus \phi_2 \oplus \phi_3$ using deletion sequence CDABE. In the process, we have a solution for C, Ψ_C: $\Omega_{\{A, E\}} \to \Omega_C$, a solution for D, Ψ_D: $\Omega_{\{E, B\}} \to \Omega_D$, a solution for A, Ψ_A: $\Omega_{\{E, B\}} \to \Omega_A$, a solution for B, Ψ_B: $\Omega_E \to \Omega_B$, and a solution for E, Ψ_E: $\Omega_\varnothing \to \Omega_E$. Theorem 2 tells us we can construct a solution as follows. Working opposite to the deletion sequence, first, $\Psi_E(\blacklozenge)$ = e. Next, Ψ_B(e) = b. Next, Ψ_A(e, b) = ~a. Next, Ψ_D(e, b) = d. And finally, Ψ_C(e, ~a) = c or ~c. Thus, configurations (~a, b, c, d, e) and (~a, b, ~c, d, e) are both solutions for ϕ.

Deletion Sequences. The sequence in which we delete variables in the fusion algorithm is called the *deletion sequence*. Which deletion sequence should one use? First, note that all deletion sequences lead to the same final result. This is implied in the statement of Theorem 1. Second, different deletion sequences may involve different computational efforts. For example, consider the VN representation of the optimization problem shown in Figure 1. In this example, all deletion sequences starting with variable E involve more computational effort than sequences that do not start with E, as the former involves combination on the state space of all five variables. Finding an optimal deletion sequence is a secondary optimization problem that has shown to be NP-complete [Arnborg *et al*. 1987]. But, there are several heuristics for finding good deletion sequences [Olmsted 1983, Kong 1986, Mellouli 1987].

One such heuristic is called one-step-look-ahead [Olmsted 1983, Kong 1986]. This heuristic tells us which variable to delete next. As per this heuristic, the variable that should be deleted next is one that leads to combination over the smallest state space with ties resolved arbitrarily. For example, in the VN of Figure 2, for the first deletion, this heuristic would pick either C or D since no combination is involved with these deletions. After deletion of C and D, any remaining variables can be used for successive deletions as they all lead to combinations over state spaces of equal sizes.

5 MITTEN'S AXIOMS FOR DYNAMIC PROGRAMMING

For optimization problems, the fusion algorithm described in Section 4 reduces to 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 will describe Mitten's framework using our notation, and compare his axiom with ours.

Values and Valuations. The *value space* is **R**, the set of real numbers. A *valuation for h* is a real-valued function on Ω_h.

Combination. There is a mapping ©: **R**×**R** → **R** that is commutative and associative. Define a mapping $\oplus: \vartheta \times \vartheta \to \vartheta$ such that whenever γ and η are valuations for g and h, respectively, $\gamma \oplus \eta$ is a valuation for $g \cup h$ given by

$$(\gamma \oplus \eta)(\mathbf{x}) = \gamma(\mathbf{x}^{\downarrow g}) © \eta(\mathbf{x}^{\downarrow h})$$

for all $\mathbf{x} \in \Omega_{g \cup h}$.

Monotonicity of Combination. We shall say that © is *monotonic* if for any u, v_1, $v_2 \in$ **R**, u©$v_1 \geq$ u©v_2 whenever $v_1 \geq v_2$. Suppose η_1 and η_2 are valuations for h. We shall say that $\eta_1 \geq \eta_2$ if $\eta_1(\mathbf{x}) \geq \eta_2(\mathbf{x})$ for all $\mathbf{x} \in \Omega_h$. Note that if © is monotonic, then for all valuations γ, $\gamma \oplus \eta_1 \geq \gamma \oplus \eta_2$ whenever $\eta_1 \geq \eta_2$.

Marginalization. Define a mapping $\downarrow(h - \{X\}): \vartheta_h \to \vartheta_{h - \{X\}}$ such that whenever η is a valuation for h, $\eta^{\downarrow(h - \{X\})}$ is a valuation for $h - \{X\}$ given by

$$\eta^{\downarrow(h - \{X\})}(\mathbf{y}) = \mathrm{MAX}\{\eta(\mathbf{y}, \mathbf{x}) \mid \mathbf{x} \in \Omega_X\} \tag{5.1}$$

for all $\mathbf{y} \in \Omega_{h - \{X\}}$.

Theorem 3. Suppose the value space is **R**, suppose marginalization is defined as in (5.1), and suppose © is monotonic. If γ is a valuations for g, η is a valuation for h, $X \in h$, and $X \notin g$, then $(\gamma \oplus \eta)^{\downarrow((g \cup h) - \{X\})} = \gamma \oplus (\eta^{\downarrow(h - \{X\})})$.

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

6 CONCLUSION

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 VN framework and three axioms that permit the use of local computation in solving a VN. We can think of the framework and the fusion algorithm as the answer to the first question. The three axioms constitute one answer to the second question.

7 PROOFS

In this section, we will provide proofs for Theorems 1 and 2 stated in Section 4 and Theorem 3 stated in Section 5. We start with a lemma we need to prove Theorem 1.

Lemma 1. Suppose $\{\Theta, \{\Omega_X\}_{X\in\Theta}, \Delta, \{\sigma_1, ..., \sigma_k\}, ©, \downarrow\}$ is a VN where σ_i is a valuation for h_i for i = 1, ..., k, $\Theta = h_1\cup...\cup h_k$, and © and ↓ satisfy Axioms A1–A3. Suppose $X\in\Theta$. Then

$$(\sigma_1\oplus...\oplus\sigma_k)^{\downarrow(\Theta - \{X\})} = \oplus\ \mathrm{Fus}_X\{\sigma_1, ..., \sigma_k\}.$$

Proof of Lemma 1: Let $g = \cup\{h_i \mid X\notin h_i\}$, and let $h = \cup\{h_i \mid X\in h_i\}$. Let $\gamma = \oplus\{\sigma_i \mid X\notin h_i\}$, and $\eta = \oplus\{\sigma_i \mid X\in h_i\}$. Note that $X\in h$, and $X\notin g$. Then

$$\begin{aligned}(\sigma_1\oplus...\oplus\sigma_k)^{\downarrow(\Theta - \{X\})} &= (\gamma\oplus\eta)^{\downarrow((g\cup h) - \{X\})} \\ &= \gamma\oplus(\eta^{\downarrow(h - \{X\})}) && \text{(using Axiom A3)} \\ &= (\oplus\{\sigma_i \mid X\notin h_i\}) \oplus (\oplus\{\sigma_i \mid X\in h_i\})^{\downarrow(h - \{X\})} \\ &= \oplus\mathrm{Fus}_X\{\sigma_1, ..., \sigma_k\}. && \text{(by definition of } \mathrm{Fus}_X\{\sigma_1, ..., \sigma_k\})\end{aligned}$$

■

Proof of Theorem 1: By Axiom A2, $(\sigma_1\oplus...\oplus\sigma_k)^{\downarrow\emptyset}$ is obtained by sequentially marginalizing all variables out of the joint valuation. A proof of this theorem is obtained by repeatedly applying the result of Lemma 1. At each step, we delete a variable and fuse the set of all valuations with respect to this variable. Using Lemma 1, after fusion with respect to X_1, the combination of all valuations in the resulting VN is equal to $(\sigma_1\oplus...\oplus\sigma_k)^{\downarrow(\Theta - \{X_1\})}$. Again, using Lemma 1, after fusion with respect to X_2, the combination of all valuations in the resulting VN is equal to $(\sigma_1\oplus...\oplus\sigma_k)^{\downarrow(\Theta-\{X_1, X_2\})}$. And so on. When all variables have been deleted, we have the result.

■

Next, we state and prove a lemma we need to prove Theorem 2.

Lemma 2. Suppose $\{\Theta, \{\Omega_X\}_{X\in\Theta}, \Delta, \{\sigma_1, ..., \sigma_k\}, ©, \downarrow\}$ is a VN where σ_i is a valuation for h_i for i = 1, ..., k, $\Theta = h_1\cup...\cup h_k$, and © and $\downarrow$ satisfy Axioms A1–A3. Suppose $\Psi_X: \Omega_{g-\{X\}} \to \Omega_X$ is a solution for X computed during the marginalization operation involved in the fusion of $\{\sigma_1, ..., \sigma_k\}$ with respect to X, and suppose $\mathbf{c}\in\Omega_{\Theta-\{X\}}$ is a solution for $(\sigma_1\oplus...\oplus\sigma_k)^{\downarrow(\Theta-\{X\})}$. Then $(\mathbf{c}, \Psi_X(\mathbf{c}^{\downarrow(g-\{X\})}))$ is a solution for $\sigma_1\oplus...\oplus\sigma_k$.

Proof of Lemma 2: Without loss of generality, suppose that $\sigma_1, ..., \sigma_m$ are the only valuations that have X in their domain. Let $\sigma = \sigma_1\oplus...\oplus\sigma_m$, let $g = g_1\cup...\cup g_m$, and let $\mathbf{c}\in\Omega_{\Theta-\{X\}}$. We need to prove that $(\sigma_1\oplus...\oplus\sigma_k)(\mathbf{c}, \Psi_X(\mathbf{c}^{\downarrow(g-\{X\})})) = (\sigma_1\oplus...\oplus\sigma_k)^{\downarrow\varnothing}(\blacklozenge)$. We have

$(\sigma_1\oplus...\oplus\sigma_k)(\mathbf{c}, \Psi_X(\mathbf{c}^{\downarrow(g-\{X\})}))$

$= \sigma(\mathbf{c}^{\downarrow(g-\{X\})}, \Psi_X(\mathbf{c}^{\downarrow(g-\{X\})}))©\sigma_{m+1}(\mathbf{c}^{\downarrow g_{m+1}})©...©\sigma_k(\mathbf{c}^{\downarrow g_k})$ (by definition of combination)

$= \sigma^{\downarrow(g-\{X\})}(\mathbf{c}^{\downarrow(g-\{X\})})©\sigma_{m+1}(\mathbf{c}^{\downarrow g_{m+1}})©...©\sigma_k(\mathbf{c}^{\downarrow g_k})$ (since Ψ_X is a solution for X)

$= \oplus Fus_X\{\sigma_1, ..., \sigma_k\}(\mathbf{c})$ (by definition of $Fus_X\{\sigma_1, ..., \sigma_k\}$)

$= (\sigma_1\oplus...\oplus\sigma_k)^{\downarrow(\Theta-\{X\})}(\mathbf{c})$ (by Lemma 1)

$= (\sigma_1\oplus...\oplus\sigma_k)^{\downarrow\varnothing}(\blacklozenge)$ (since **c** is a solution for $(\sigma_1\oplus...\oplus\sigma_k)^{\downarrow(\Theta-\{X\})}$)

■

Proof of Theorem 2: A proof of this theorem is obtained by repeatedly applying Lemma 2. Consider the VN $\oplus Fus_{X_n}\{... Fus_{X_2}\{Fus_{X_1}\{\sigma_1,..., \sigma_k\}\}\}$. There is only one valuation in this VN and it is for the empty set. From Theorem 1, $(\oplus Fus_{X_n}\{...Fus_{X_2}\{Fus_{X_1}\{\sigma_1,..., \sigma_k\}\}\})(\blacklozenge) = (\sigma_1\oplus\ldots\oplus\sigma_k)^{\downarrow\varnothing}(\blacklozenge)$. Since $\blacklozenge$ is a solution for $(\sigma_1\oplus\ldots\oplus\sigma_k)^{\downarrow\varnothing}$, by Lemma 2, $(\blacklozenge, \Psi_{X_n}(\blacklozenge)) = \Psi_{X_n}(\blacklozenge) = \mathbf{z}^{\downarrow\{X_n\}}$ is a solution for $(\sigma_1\oplus\ldots\oplus\sigma_k)^{\downarrow\{X_n\}}$.

Since $\mathbf{z}^{\downarrow\{X_n\}}$ is a solution for $(\sigma_1\oplus\ldots\oplus\sigma_k)^{\downarrow\{X_n\}}$, and $\Psi_{X_{n-1}}: \Omega_{g_{n-1}-\{X_{n-1}\}} \to \Omega_{X_{n-1}}$ is a solution for X_{n-1}, by Lemma 2, $(\mathbf{z}^{\downarrow\{X_n\}}, \Psi_{X_{n-1}}(\mathbf{z}^{\downarrow(g_{n-1}-\{X_{n-1}\})})) = (\mathbf{z}^{\downarrow\{X_n\}}, \mathbf{z}^{\downarrow\{X_{n-1}\}}) = \mathbf{z}^{\downarrow\{X_n, X_{n-1}\}}$ is a solution for $(\sigma_1\oplus\ldots\oplus\sigma_k)^{\downarrow\{X_n, X_{n-1}\}}$.

Continuing in this fashion, we get the result that **z** is a solution for $\sigma_1\oplus\ldots\oplus\sigma_k$. ■

Proof of Theorem 3: Suppose $\mathbf{w}\in\Omega_{g-h}$, $\mathbf{y}\in\Omega_{g\cap h}$, and $\mathbf{z}\in\Omega_{h-g-\{X\}}$.

$(\gamma\oplus\eta)^{\downarrow((g\cup h)-\{X\})}(\mathbf{w}, \mathbf{y}, \mathbf{z}) = MAX\{(\gamma\oplus\eta)(\mathbf{w}, \mathbf{y}, \mathbf{z}, \mathbf{x}) \mid \mathbf{x}\in\Omega_X\}$

$= MAX\{\gamma(\mathbf{w}, \mathbf{y})©\eta(\mathbf{y}, \mathbf{z}, \mathbf{x}) \mid \mathbf{x}\in\Omega_X\}$

$\geq \gamma(\mathbf{w}, \mathbf{y})©(MAX\{\eta(\mathbf{y}, \mathbf{z}, \mathbf{x}) \mid \mathbf{x}\in\Omega_X\})$

$= \gamma(\mathbf{w}, \mathbf{y})©(\eta^{\downarrow(h-\{X\})}(\mathbf{y}, \mathbf{z}))$

$= (\gamma\oplus(\eta^{\downarrow(h-\{X\})}))(\mathbf{w}, \mathbf{y}, \mathbf{z})$

In other words, $(\gamma\oplus\eta)^{\downarrow((g\cup h)-\{X\})} \geq \gamma\oplus(\eta^{\downarrow(h-\{X\})})$.

Since © is monotonic and $MAX\{\eta(\mathbf{y}, \mathbf{z}, \mathbf{x})|\mathbf{x}\in\Omega_X\} \geq \eta(\mathbf{y}, \mathbf{z}, \mathbf{x})$ for all $\mathbf{x}\in\Omega_X$, we have $\gamma(\mathbf{w}, \mathbf{y})©(MAX\{\eta(\mathbf{y}, \mathbf{z}, \mathbf{x}) \mid \mathbf{x}\in\Omega_X\}) \geq \gamma(\mathbf{w}, \mathbf{y})©\eta(\mathbf{y}, \mathbf{z}, \mathbf{x})$

for all $\mathbf{x}\in\Omega_X$. In particular, this inequality must hold for the maximum of the RHS with respect to $\mathbf{x}$, i.e.,

$$\gamma(\mathbf{w}, \mathbf{y})\copyright\big(\mathrm{MAX}\{\eta(\mathbf{y}, \mathbf{z}, \mathbf{x}) \mid \mathbf{x}\in\Omega_X\}\big) \geq \mathrm{MAX}\{\gamma(\mathbf{w}, \mathbf{y})\copyright\eta(\mathbf{y}, \mathbf{z}, \mathbf{x}) \mid \mathbf{x}\in\Omega_X\},$$

i.e., $\gamma\oplus(\eta^{\downarrow(h - \{X\})}) \geq (\gamma\oplus\eta)^{\downarrow((g\cup h) - \{X\})}$. Earlier we showed that $(\gamma\oplus\eta)^{\downarrow((g\cup h) - \{X\})} \geq \gamma\oplus(\eta^{\downarrow(h - \{X\})})$. Therefore, $(\gamma\oplus\eta)^{\downarrow((g\cup h) - \{X\})} = \gamma\oplus(\eta^{\downarrow(h - \{X\})})$. ■

ACKNOWLEDGEMENTS

This work was supported in part by a grant from Hughes Research Laboratories.

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.

Olmsted, S. M. (1983), “On representing and solving decision problems,” Ph.D. thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

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

Shafer, G. and P. P. Shenoy (1990), “Probability propagation,” *Annals of Mathematics and Artificial Intelligence*, **2**(1–4), 327–352.

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

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

Shenoy, P. P. (1992a), “Valuation-based systems: A framework for managing uncertainty in expert systems,” in L. A. Zadeh and J. Kacprzyk (eds.), *Fuzzy Logic for the Management of Uncertainty*, 83–104, John Wiley & Sons, New York, NY.

Shenoy, P. P. (1992b), “Valuation-based systems for Bayesian decision analysis,” *Operations Research*, **40**(3), 463–484.

Shenoy, P. P. (1993), “A new method for representing and solving Bayesian decision problems,” in D. J. Hand (eds.), *Artificial Intelligence Frontiers in Statistics: AI and Statistics III*, 119–138, Chapman & Hall, London.

Shenoy, P. P. (1994a), “A comparison of graphical techniques for decision analysis,” *European Journal of Operational Research*, **78**(1), 1–21.

Shenoy, P. P. (1994b), “Consistency in valuation-based systems,” *ORSA Journal on Computing*, **6**(3), 281–291.

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

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

Shenoy, P. P. and G. Shafer (1990), “Axioms for probability and belief-function propagation,” in Shachter, R. D., T. S. Levitt, J. F. Lemmer, and L. N. Kanal (eds.), *Uncertainty in Artificial Intelligence*, **4**, 169–198, North-Holland, Amsterdam. Reprinted in Shafer, G. and J. Pearl, eds. (1990), *Readings in Uncertain Reasoning*, 575–610, Morgan Kaufmann, San Mateo, CA.