A Comparison of Graphical Techniques for Asymmetric Decision Problems

Article *in* Management Science · November 1999

DOI: 10.1287/mnsc.45.11.1552 · Source: CiteSeer

2 authors:

Concha Bielza
Universidad Politécnica de Madrid
367 PUBLICATIONS **6,360** CITATIONS

SEE PROFILE

Prakash P. Shenoy
University of Kansas
173 PUBLICATIONS **6,764** CITATIONS

SEE PROFILE

A Comparison of Graphical Techniques for Asymmetric Decision Problems

Concha Bielza • Prakash P. Shenoy
Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain
School of Business, University of Kansas, Lawrence, Kansas 66045
mcbielza@fi.upm.es • pshenoy@ukans.edu

We compare four graphical techniques for representation and solution of asymmetric decision problems—decision trees, influence diagrams, valuation networks, and sequential decision diagrams. We solve a modified version of Covaliu and Oliver's Reactor problem using each of the four techniques. For each technique, we highlight the strengths, weaknesses, and some open issues that perhaps can be resolved with further research.
(*Asymmetric Decision Problems*; *Decision Trees*; *Influence Diagrams*; *Valuation Networks*; *Sequential Decision Diagrams*)

1. Introduction

This paper compares four graphical techniques for representing and solving asymmetric decision problems—traditional decision trees (DTs), Smith, Holtzman, and Matheson's (SHM) (1993) influence diagrams (IDs), Shenoy's (1993b, 1996) valuation networks (VNs), and Covaliu and Oliver's (1995) sequential decision diagrams (SDDs).

We focus our attention on techniques designed for asymmetric decision problems. In a decision tree, a path from the root to a leaf node is called a *scenario*. We say a decision problem is *asymmetric* if the number of scenarios in a decision tree representation is less than the cardinality of the Cartesian product of the state spaces of all chance and decision variables.

Each technique has a distinct way of encoding asymmetry. DTs encode asymmetry through the use of scenarios. IDs encode asymmetry using graphical structures called "distribution trees." VNs encode asymmetry using functions called "indicator valuations." Finally, SDDs encode asymmetry by showing all scenarios in a compact fashion using graphs called "sequential decision diagrams."

The main contribution of this paper is to highlight the strengths, weaknesses, and some open issues that perhaps can be resolved with further study of the four techniques. By strengths and weaknesses, we mean intrinsic features we find desirable and undesirable, respectively.

An outline of the remainder of the paper is as follows. In §2, we give a complete statement of a modified version of the Reactor problem (Covaliu and Oliver 1995), describe a DT representation and solution of it, and discuss strengths, weaknesses, and open issues associated with DTs. In §3, we represent and solve the same problem using Smith, Holtzman, and Matheson's IDs, and discuss strengths, weaknesses, and open issues associated with IDs. In §4, we do the same using Shenoy's VNs. In §5, we focus on Covaliu and Oliver's SDDs. Finally, in §6, we summarize our conclusions.

2. Decision Trees

In this section, we describe a DT representation and solution of a small asymmetric decision problem called the Reactor problem, and we discuss strengths, weaknesses, and some open issues associated with DTs. The Reactor problem is a modified version of the problem described by Covaliu and Oliver (1995). In our version, Bayesian revision of probabilities is re-

Management Science © 1999 INFORMS
Vol. 45, No. 11, November 1999 pp. 1552–1569

0025-1909/99/4511/1552$05.00
1526-5501 electronic ISSN

quired during the solution process, and the joint utility function decomposes into only three factors.

2.1. A Statement of the Reactor Problem

An electric utility firm must decide whether to build (D_2) a reactor of advanced design (a), a reactor of conventional design (c), or neither (n). If successful, an advanced reactor is more profitable, but is riskier. Based on past experience, a conventional reactor (C) has probability 0.980 of no failure (cs), and a probability 0.020 of a failure (cf). On the other hand, an advanced reactor (A) has probability 0.660 of no failure (as), probability 0.244 of a limited accident (al), and probability 0.096 of a major accident (am). The profits for the case the firm builds a conventional reactor are \$8B if there is no failure, and −\$4B if there is a failure. The profits for the case the firm builds an advanced reactor are \$12B if there is no failure, −\$6B if there is a limited accident, and −\$10B if there is a major accident. The firm's utility function is a linear function of the profits.

Before making this decision, the firm can conduct an expensive test of the components of the advanced reactor. The test results (T) can be classified as bad (b), good (g), excellent (e), or no result (nr). The cost of this test is \$1B. If the test is done, its results are correlated with the success or failure of the advanced reactor. The likelihoods for the test results are as follows: $P(g|as) = 0.182$, $P(e|as) = 0.818$, $P(b|al) = 0.288$, $P(g|al) = 0.565$, $P(e|al) = 0.147$, $P(b|am) = 0.313$, $P(g|am) = 0.437$, and $P(e|am) = 0.250$. If the test results are bad, the Nuclear Regulatory Commission will not permit an advanced reactor. The firm needs to decide (D_1) whether to conduct the test (t), or not (nt). If the decision is nt, the test outcome is nr.

2.2. DT Representation and Solution

Figure 1 shows a decision tree representation and solution of this problem. The order in which the nodes are traversed from left to right is the chronological order in which decisions are made and/or outcomes of chance events are revealed to the decision-maker, and every available branch at every node is explicitly shown. Thus, the decision tree gives a chronological and fully detailed view of the structure of the decision problem.

Notice that even before the decision tree can be completely specified, the conditional probabilities required by the decision tree representation have to be computed from those specified in the problem using the standard procedure called *preprocessing*. In this procedure, given the priors and the likelihoods, first we compute the joints, then the preposteriors, and finally the posteriors. Details of the results of the procedure for the Reactor problem can be found in Bielza and Shenoy (1998).

Figure 1 **A Decision Tree Representation and Solution of the Reactor Problem**

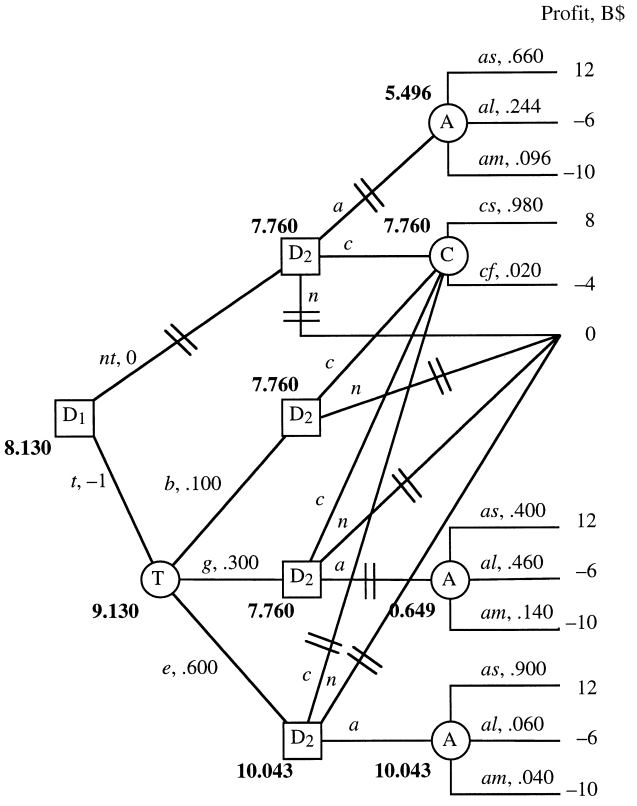


Figure 1 also shows the solution of the Reactor problem using rollback. The optimal strategy is to do the test; build a conventional reactor if the test results are bad or good, and build an advanced reactor if the test results are excellent. The expected profit associated with this strategy is \$8.130B.

2.3. Strengths of DTs

DTs are easy to understand and easy to solve. DTs encode asymmetries through use of scenarios without

introducing dummy states for variables. If a variable is not relevant in a scenario, a DT simply does not include it. As we will see shortly, IDs and VNs introduce dummy states for chance and decision variables in the process of encoding asymmetry. This decreases their computational efficiency. Like DTs, SDDs do not introduce dummy states for variables.

2.4. Weaknesses of DTs

DTs capture asymmetries globally in the form of scenarios. This contributes to the exponential growth of the decision tree representation and limits the use of DTs to small problems. In comparison, IDs, VNs, and SDDs capture asymmetries locally.

Although we have shown the decision tree representation using coalescence (Olmsted 1983), it should be noted that automating coalescence in decision trees is not easy because it involves constructing the complete uncoalesced tree and then recognizing repeated subtrees.[1] This is a major drawback of DTs (as compared to IDs, VNs, and SDDs), and it limits the use of DT representation to small decision problems.

2.5. Some Open Issues

To complete a DT representation of a problem, the probability model may need preprocessing, and this makes the automation of DTs difficult. One method for avoiding preprocessing is to use von Neumann-Morgenstern (1944) information sets to encode information constraints—see Shenoy (1995, 1998) for details. However, adding information sets makes the resulting representation more complex.

Since conditional independence is not explicitly encoded in probability trees, doing the preprocessing by computing the joint probability distribution for all chance variables is computationally intractable in problems with many chance variables. This issue can be resolved by using a Bayesian network representation of the probability model, and Olmsted's (1983) and Shachter's (1986) arc-reversal method can then be used to compute the probability model demanded by the DT representation. However, this raises the issue of determining a sequence of arc reversals so as to achieve the desired probability model with minimum computation.

[1] When a DT has repeating subtrees, they are shown just once and are pointed to by all scenarios in which they occur. This is referred to as *coalescence*.

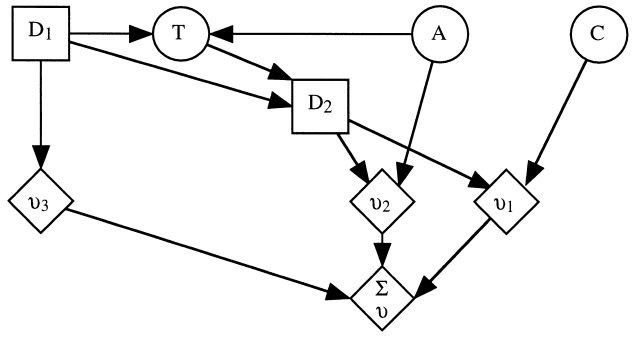
3. Asymmetric Influence Diagrams

In this section, we will represent and solve the Reactor problem using Smith, Holtzman, and Matheson's (1993) (henceforth, SHM) asymmetric influence diagram technique. Although SHM describe their technique for a single undecomposed utility function, we use Tatman and Shachter's (1990) extension of the ID technique that allows for a decomposition of the joint utility function. The symmetric ID technique was initially developed by Howard and Matheson (1981), Olmsted (1983), and Shachter (1986). Modifications of the symmetric ID solution technique have been proposed by Smith (1989), Shachter and Peot (1992), Ndilikilikesha (1994), Jensen et al. (1994), Cowell (1994), Zhang et al. (1994), Goutis (1995), and others. Besides SHM, asymmetric extensions of the influence diagram technique have been proposed by, e.g., Call and Miller (1990), Fung and Shachter (1990), and Qi et al. (1994).

3.1. ID Representation

An influence diagram representation of a problem is specified at three levels—graphical, functional, and numerical. At the graphical level, we have a directed acyclic graph, called an influence diagram, that displays decision variables, chance variables, factorization of the joint probability distribution into conditionals, factorization of the joint utility function, and information constraints. Figure 2 shows an influence diagram for the Reactor problem at the graphical level.

Figure 2 An ID for the Reactor Problem at the Graphical Level



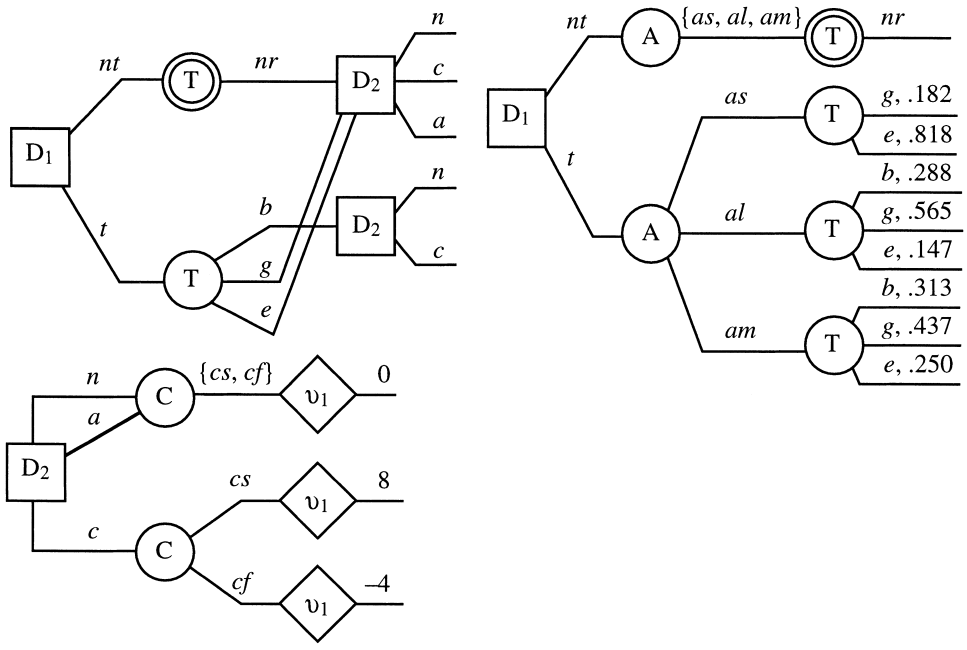
Note that the arcs pointing to chance nodes reflect the way in which their joint probability distribution is currently factored, which is not necessarily the chronological order in which their outcomes will be revealed to the decision maker. Arcs between pairs of chance nodes may be reversed by changing the way in which the joint distribution is factored, as in applications of Bayes' theorem. Also, note that the ID avoids the combinatorial explosion of the decision tree (the so-called "bushy mess" phenomenon) by suppressing details of the number of branches available at each decision or chance node. The latter information is encoded deeper down at the functional level instead.

At the functional level, we specify the structure of the conditional distribution (or simply, conditional) for each node (except super value nodes) in the ID, and at the numerical level, we specify the numerical details of the probability distributions and the utilities. The key idea of the SHM technique is a new tree representation for describing the conditionals. These are called *distribution trees* with paths showing the *conditioning scenarios* that lead to *atomic distributions* that describe either probability distributions, set of alternatives, or (expected) utilities, assigned in each conditioning scenario. A conditional for a chance node represents a factor of the joint probability distribution. A conditional for a decision node can be thought of as describing the alternatives available to the decision-maker in each conditioning scenario. A conditional for a value node represents a factor of the joint utility function. For the Reactor problem, some of the conditionals are shown in Figure 3 (the complete set is found in Bielza and Shenoy 1998).

The distribution tree for D_2 has two atomic distributions. The firm will choose among three alternatives (conventional or advanced reactor or neither) only if it decides to not do the test ($D_1 = nt$) or if it conducts the test and its result is good or excellent. The conditional for D_2 is *coalesced*, i.e., the atomic distribution with three alternatives is shared by three distinct scenarios, and is *clipped*, i.e., many branches in conditioning scenarios are omitted because the corresponding conditioning scenarios are impossible. For example, if the firm chooses to not do the test, then it is impossible to observe any test results.

The distribution tree for T shows that if the firm decides to not perform the test ($D_1 = nt$), then $T = nr$ with probability 1 regardless of the advanced reactor

Figure 3 **Distribution Trees for Decision Node D_2, Chance Node T, and Utility Node v_1**



state. Thus, the conditional for T can be *collapsed* across A given $D_1 = nt$. Collapsed scenarios are shown by indicating the set of possible states on a single edge emanating from the node. They allow the representation of conditional independence between variables that holds only given particular outcomes of some other variables. Deterministic atomic distributions for chance and decision variables are shown by double-bordered nodes.

Although not all are shown in Figure 3, the conditionals for the three utility nodes provide other examples of coalesced, clipped, and collapsed distributions. They are deterministic nodes because we assign a single utility for each conditioning scenario. Since utility functions are always deterministic, and we use diamond-shaped nodes to indicate utility functions, we do not draw these nodes with a double border.

Another feature of distribution trees not illustrated in the Reactor problem is *unspecified distributions* where certain atomic distributions of a chance node are left unspecified since they are not required during the solution phase. If only the probabilities are unspecified, then we have a *partially* unspecified distribution. All of these features—coalesced, clipped, collapsed, and unspecified distributions—provide a more compact and expressive representation than the usual table in the symmetric ID literature.

3.2. ID Solution

The algorithm for solving an asymmetric ID is conceptually the same as that for conventional ID. However, SHM describe methods for exploiting different features of a distribution tree (such as clipped scenarios, coalescence, collapsed scenarios, etc.) to simplify the computations.

We solve an ID by reducing variables in a sequence that respects the information constraints. If the true state of a chance variable C is not known at the time the decision maker must choose an alternative from the atomic distribution of decision variable D, then C must be reduced before D, and vice versa. In the Reactor problem, there are two possible reduction sequences, CAD_2TD_1 and ACD_2TD_1. Both of these reduction sequences require the same computational effort. In the following, we use the first reduction sequence CAD_2TD_1. We use this sequence also when we solve this problem with the VN and the SDD techniques.

We start by reducing node C. Essentially, we absorb the conditional for C into utility function ν_1 using the expectation operation (following Theorem 5 in Tatman and Shachter (1990)). The expectation operation is carried out by considering each conditioning scenario separately. Figure 4 shows the ID after reducing C. Next, we reduce A. To do so, we first reverse the arc (A, T), and then absorb the posterior for A into utility function ν_2 using the expectation operation. Figure 4 shows the ID after reduction of A.

Next, we need to reduce D_2. Since D_2 has two value node successors, before we reduce D_2, we introduce a new super-value node ω, and then we merge ν_1 and ν_2 into ω (as per Theorem 5 in Tatman and Shachter 1990). We reduce D_2 by maximizing ω over the states of D_2 permitted by the distribution tree for D_2. Notice that this distribution tree (shown in Figure 3) has asymmetry in the atomic alternative sets, but this is not exploited either during the reduction of A or during the processing prior to reduction of D_2. We will comment further on this aspect of SHM technique in §3.5. Figure 4 shows the ID after reduction of D_2.

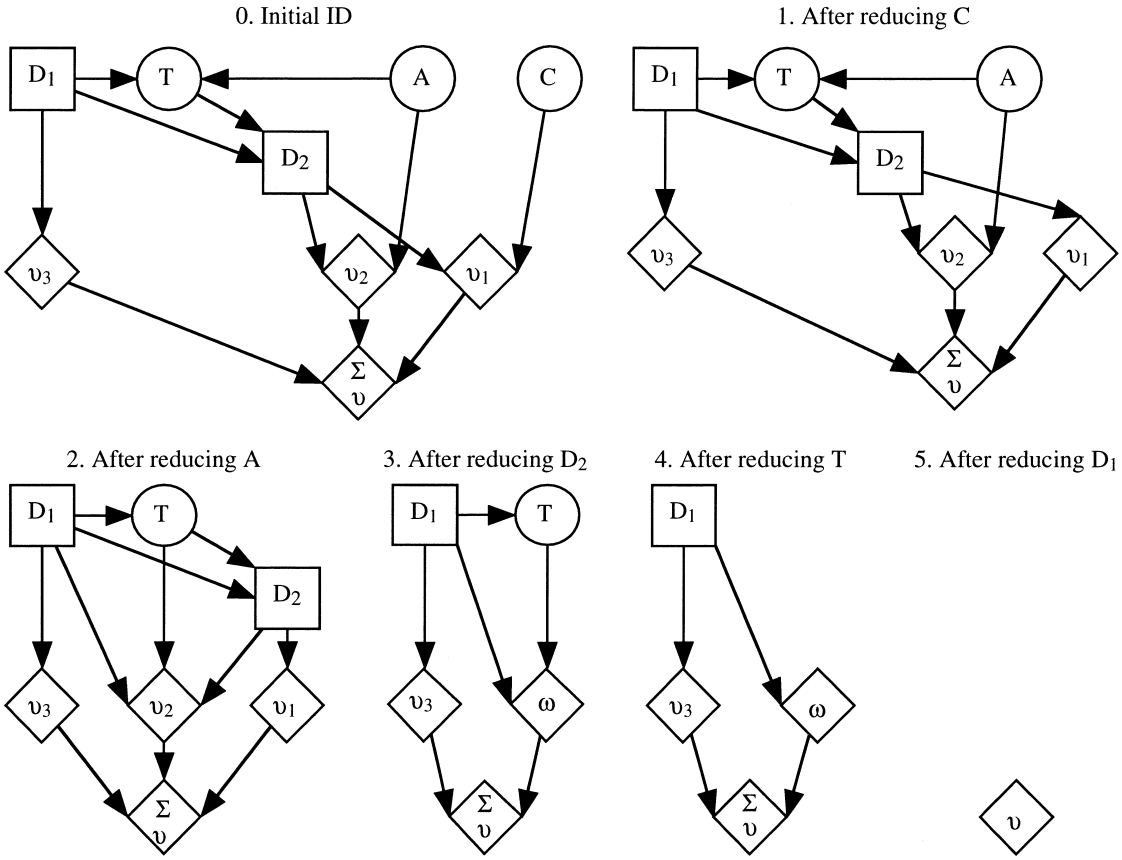
Next, we reduce T. Notice that ω is the only value node that has T in its domain. We absorb the conditional for T into the utility function ω using the expectation operation. Figure 4 shows the resulting ID. Finally, we need to reduce D_1. Since D_1 is in the domain of ν_3 and ω, first we combine ν_3 and ω obtaining ν, and then we reduce D_1 by maximizing ν over the possible states of D_1. This completes the solution of the Reactor ID representation. Complete details of the ID solution of the Reactor problem are found in Bielza and Shenoy (1998).

An optimal strategy can be pieced together from the optimal decision function for D_1 (obtained during reduction of D_1) and the optimal decision function for D_2 (obtained during reduction of D_2). Of course, we get the same optimal strategy and the same maximum expected utility as in the DT case.

3.3. Strengths of IDs

The main strength of IDs is their compactness. The size of an ID graphical representation grows linearly with the number of variables. Also, they are intuitive

Figure 4 **ID Solution at the Graphical Level**



to understand, and they encode conditional independence relations in the probability model.

The asymmetric extension of IDs captures asymmetry through the notion of distribution trees. These are easy to understand and specify. The sharing of scenarios, clipping of scenarios, collapsed scenarios, and unspecified-distribution features of distribution trees contribute to the expressiveness of the representation and to the efficiency of the solution technique.

In distribution trees one can mix the different kinds of atomic distributions—not only can one mix deterministic and stochastic atomic distributions for chance nodes, one can mix stochastic atomic distributions and atomic alternative sets for decision nodes. This feature may be useful if the decision-maker's ability to decide is determined by some previous decision or uncertainty (SHM 1993, p. 287).

The ID technique can detect the presence of unnecessary information in a problem by identifying *irrelevant* or *barren* nodes (Shachter 1988). This leads to a simplification of the original model and to a corresponding decrease in the computational burden of solving it.

3.4. Weaknesses of IDs

The ID technique is most suitable for problems in which we have a conditional probability model (also called a Bayesian network model) of the uncertainties. This is typical of problems in which the modeling of probabilities is done by a human expert. However, for problems in which a probability model is induced from data, the corresponding probability model is not always a Bayesian network model. In this case, the use of ID technique is problematic, i.e., it may require extensive and unnecessary preprocessing to complete an ID representation (Shenoy 1994a).

3.5. Some Open Issues

The asymmetric ID graphical representation does not distinguish between pure informational arcs and conditioning arcs for decision variables. Thus, we cannot predict the domain of the conditional for decision variables from the graphical representation alone. One of the most attractive aspects of graphical models (Bayes nets, symmetric IDs, VNs, etc.) is that one can determine domains of functions directly from the graphical level description. This is the essence of encoding conditional independence by graphs. This aspect of asymmetric IDs can be easily resolved by having two kinds of arcs that lead to decision variables. One kind can be interpreted as conditional as well as informational, and the other can be interpreted as purely informational.

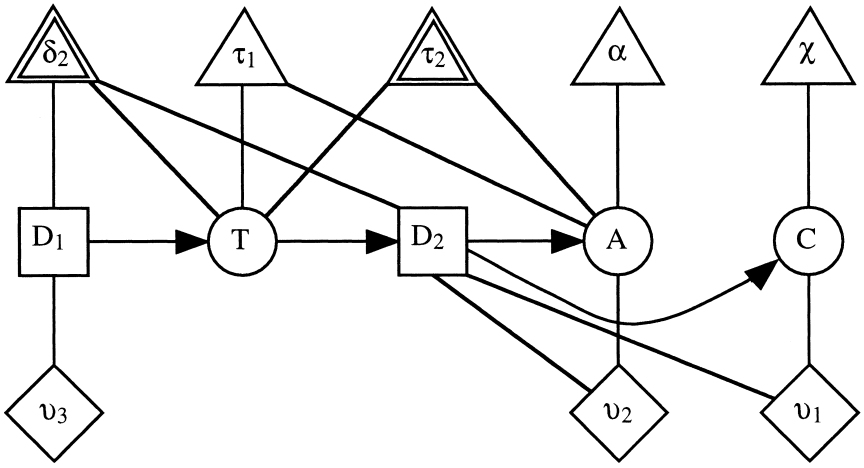
Our major concerns with the SHM representation center around having information about asymmetries in the problem stored in many distribution trees in the diagram. For example, in the Reactor problem, consider the distribution trees for T and D_2 (shown in Figure 3). Notice that, e.g., in the distribution tree for T, if $D_1 = nt$, we have $T = nr$ with probability 1. The clipping of T in the distribution tree for D_2 describes the same information. This repetition raises questions about the consistency and efficiency of the representation and the solution technique. First, the redundant specification of information may be inefficient when assessing these distributions; the user may have to repeatedly clip or collapse these same scenarios for many distributions. Second, if the user fails to clip or collapse scenarios in some distribution trees, he or she may do unnecessary calculations for these scenarios when solving the influence diagram. Third, even if the user represents all clipped/collapsed scenarios in all distribution trees, there is still the possibility of some unnecessary computation since the solution algorithm does not have access to all asymmetric information at all times. For example, in the Reactor problem, consider the situation immediately after reduction of node A shown in Figure 4. ν_2 has just inherited two new predecessors T and D_1, and its distribution tree has some conditioning scenarios that are not possible such as $D_1 = t$, $T = b$, $D_2 = a$. The absence of this scenario is encoded in the distribution tree for D_2 (see Figure 3), but we do not use this information until it is time to reduce D_2. Finally, the redundant specification creates a need for consistency: We need to somehow ensure that scenarios that are in fact possible are not inadvertently clipped in one of the distribution trees. Similarly, if we use the "unspecified distribution" feature, we need to be sure that we really do not need that distribution to answer particular questions. We can always check the consistency of an influence diagram by attempting to solve it (perhaps not carrying out the numeric calculations) and seeing if any required information has been clipped or left unspecified. It would be nice, however, if there were some simpler tests (perhaps along the lines of those used in VNs, Shenoy 1993b) that could determine whether the ID is sufficiently defined to answer specific questions.

The efficiency concerns can be at least partially addressed by "propagating" clipped scenarios during the assessment phase, as suggested by SHM (1993, p. 288). For example, if the user has specified the alternatives for D_1 and the distribution for T prior to specifying the distribution for D_2, then the system can figure out which combinations of D_1 and T are impossible (e.g., the combination $D_1 = t$ and $T = nr$ is impossible) and automatically clip the corresponding branches in the distribution tree for D_2. By propagating clipping in this way, we save the user the trouble of repeatedly doing this, thereby making the user more efficient and less likely to specify scenarios that should be clipped and less likely to clip scenarios that are possible. To propagate clipping in this way, the user must define distribution trees in a sequence consistent with the partial order defined by the influence diagram. Note, however, that this propagation does not make use of the numeric probabilities in the representation; it depends only on the specification of possible and impossible events. This propagation process is somewhat similar to the calculation of "effective state spaces" in VNs, as described in Shenoy (1993b).

4. Asymmetric Valuation Networks

In this section, we will represent and solve the Reactor problem using Shenoy's (1993b, 1996) asymmetric valuation network technique. The symmetric VN tech-

Figure 5 **A Valuation Network for the Reactor Problem**



nique is described in (Shenoy 1993a) for the case of a single undecomposed utility function, and in (Shenoy 1992) for the case of an additive decomposition of the joint utility function.

4.1. VN Representation

A valuation network representation is specified at three levels—graphical, dependence, and numerical. The graphical and dependence levels refer to qualitative (or symbolic) knowledge, whereas the numerical level refers to quantitative knowledge.

At the graphical level, we have a graph called a *valuation network*. Figure 5 shows a valuation network for the Reactor problem. A valuation network consists of two types of nodes—variable and valuation. Variables are further classified as either decision or chance, and valuations are further classified as indicator, probability, or utility. Thus, in all there are five different types of nodes—decision, chance, indicator, probability, and utility.

Decision nodes correspond to decision variables and are depicted by rectangles. Chance nodes correspond to chance variables and are depicted by circles. This part of VNs is similar to IDs.

Indicator valuations represent qualitative constraints on the joint state spaces of decision and chance variables and are depicted by double-triangular nodes. The set of variables directly connected to an indicator valuation by undirected edges constitutes the domain of the indicator valuation. In the Reactor problem, there are two indicator valuations labeled δ_2 and τ_2. δ_2's domain is $\{D_1, T, D_2\}$ and it represents the constraints that the test results are available only in the case we decide to do the test, and that the alternatives at D_2 depend on the choices at D_1 and the test results T. τ_2's domain is $\{T, A\}$ and it represents the constraint that if $A = as$, then $T = b$ is not possible.

Utility valuations represent additive factors of the joint utility function and are depicted by diamond-shaped nodes. The set of variables directly connected to a utility valuation constitutes the domain of the utility valuation. In the Reactor problem, there are three additive utility valuations labeled ν_1, ν_2, and ν_3, with domains $\{D_2, C\}$, $\{D_2, A\}$, and $\{D_1\}$, respectively.

Probability valuations represent multiplicative factors of the family of joint probability distributions for the chance variables in the problem, and are depicted by triangular nodes. Thus, in a VN, information about the current factorization of the joint probability distribution of chance variables is carried by the additional probability valuation nodes, rather than by the directions of arcs pointing to chance nodes. The set of all variables directly connected to a probability valuation constitutes the domain of the probability valuation. In the Reactor problem, there are three probability valuations labeled τ_1, α, and χ, with domains $\{A, T\}$, $\{A\}$, and $\{C\}$, respectively.

The specification of the valuation network at the

graphical level includes directed arcs between pairs of distinct variables. These directed arcs represent information constraints. Suppose R is a chance variable and D is a decision variable. An arc (R, D) means that the true state of R is known to the decision maker (DM) at the time the DM has to choose an alternative from D's state space (as in an ID). Conversely, an arc (D, R) means that the true state of R is not known to the DM at the time the DM has to choose an alternative from D's state space.

Next, we specify a valuation network representation at the dependence level. At this level, we specify the state spaces of all variables and we specify the details of the indicator valuations.

Associated with each variable X is a *state space* $\mathcal{W}_X$. As in the cases of IDs and SDDs, we assume that all variables have finite state spaces. Suppose s is a subset of variables. An *indicator* valuation for s is a function $\iota: \mathcal{W}_s \to \{0, 1\}$. An efficient way of representing an indicator valuation is simply to describe the elements of the state space that have value 1, i.e., we represent ι by Ω_ι where $\Omega_\iota = \{\mathbf{x} \in \mathcal{W}_s | \iota(\mathbf{x}) = 1\}$. Obviously, $\Omega_\iota \subseteq \mathcal{W}_s$. To minimize jargon, we also call Ω_ι an indicator valuation for s. In the Reactor problem, the details of the two indicator valuations are as follows: $\Omega_{\delta_2} = \{(nt, nr, n), (nt, nr, c), (nt, nr, a), (t, b, n), (t, b, c), (t, g, n), (t, g, c), (t, g, a), (t, e, n), (t, e, c), (t, e, a)\}$; $\Omega_{\tau_2} = \{(as, nr), (as, g), (as, e), (al, nr), (al, b), (al, g), (al, e), (am, nr), (am, b), (am, g), (am, e)\}$. Notice that the indicator valuation Ω_{δ_2} is identical to the scenarios in the distribution tree for D_2 depicted in Figure 3. The indicator valuation Ω_{τ_2} rules out the scenario $A = as$, $T = b$.

Before we can specify the valuation network at the numerical level, it is necessary to introduce the notion of effective state spaces for subsets of variables. Suppose that each variable is in the domain of some indicator valuation. (If not, we can create "vacuous" indicator valuations that are identically one for every state of such variables.) We define *combination* of indicator valuations as pointwise Boolean multiplication, and *marginalization* of an indicator valuation as Boolean addition over the state space of reduced variables. Then, the *effective state space* for a subset s of variables, denoted by Ω_s, is defined as follows: First we combine all indicator valuations that include some variable from s in their domains, and next we marginalize the combination so that only the variables in s remain in the marginal. Shenoy (1994b) has shown that these definitions of combination and marginalization satisfy the three axioms that permit local computation (Shenoy and Shafer 1990). Thus, the computation of the effective state spaces can be done efficiently using local computation. For example, to compute the effective state space for $\{A, T\}$, by definition $\Omega_{\{A,T\}} = (\delta_2 \otimes \tau_2)^{\downarrow\{A,T\}}$ (where $\delta_2 \otimes \tau_2$ denotes combination of valuations δ_2 and τ_2, and $(\delta_2 \otimes \tau_2)^{\downarrow\{A,T\}}$ denotes marginalization of the joint valuation $\delta_2 \otimes \tau_2$ down to the domain $\{A, T\}$). However, using local computation, it can be computed more efficiently as follows: $\Omega_{\{A,T\}} = \delta_2^{\downarrow\{T\}} \otimes \tau_2$. Details of local computation are found in Shenoy (1993b).

Finally, we specify a valuation network at the numerical level. At this level, we specify the details of the utility and probability valuations. A *utility* valuation ν for s is a function $\nu: \Omega_s \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. The values of ν are utilities. In the Reactor problem, there are three utility valuations. One of these is shown in Table 1, and we refer the reader to Bielza and Shenoy (1998) for the complete description.

A *probability* valuation π for s is a function $\pi: \Omega_s \to [0, 1]$. The values of π are probabilities. In the Reactor problem, there are three probability valuations. One of these is shown in Table 1. What do these probability

Table 1 **Utility Valuation ν_1 and Probability Valuation τ_1 in the Reactor Problem**

$\Omega_{\{D_2, C\}}$		ν_1	$\Omega_{\{A,T\}}$		τ_1
n	*cs*	0	*as*	*nr*	1
n	*cf*	0	*as*	*g*	.182
a	*cs*	0	*as*	*e*	.818
a	*cf*	0	*al*	*nr*	1
c	*cs*	8	*al*	*b*	.288
c	*cf*	−4	*al*	*g*	.565
			al	*e*	.147
			am	*nr*	1
			am	*b*	.313
			am	*g*	.437
			am	*e*	.250

valuations mean? χ is the marginal for C, α is the marginal for A, and $\delta_2^{2\{D_1,T\}} \otimes \tau_2 \otimes \tau_1$ is the conditional for T given A and D_1. Thus the conditional for T factors into three valuations such that τ_1 has the numeric information, and δ_2 and τ_2 include the structural information.

Notice that the utility and probability valuations are described only for effective state spaces that are computed using local computation from the specifications of the indicator valuations. There is no redundancy in the representation. However, in ν_1, unlike the ID representation, the irrelevance of C in scenarios $D_2 = n$ or a is not represented in the VN representation because we are unable to do so. Also, in ν_2, the irrelevance of A in scenarios where $D_2 = n$ or c is not represented. We will comment further on these issues in §4.5.

4.2. VN Solution

In this section, first we sketch the fusion algorithm for solving valuation network representations of decision problems, and then we solve the Reactor problem.

The fusion algorithm is essentially the same as in the symmetric case (Shenoy 1992). The main difference is in how indicator valuations are handled. Since indicator valuations are identically one on effective state spaces, there are no numeric computations involved in combining indicator valuations. Indicator valuations do contribute domain information and cannot be totally ignored. In the fusion algorithm, we reduce a variable by doing a fusion operation on the set of all valuations (utility, probability, and indicator) with respect to the variable. All numeric computations are done on effective state spaces only. This means that the effective state spaces may need to be computed prior to doing the fusion operation if the effective state space has not been already computed during the representation phase.

Fusion with respect to a decision variable D is defined as follows. The utility, probability, and indicator valuations whose domains do not include D remain unchanged. All utility valuations that include D in their domain are combined together, and the resulting utility valuation ν is marginalized such that D is eliminated from its domain. A new indicator valuation ζ_D corresponding to the decision function for D is created. All probability and indicator valuations that include D in their domain are combined together and the resulting probability valuation ρ is combined with ζ_D and the result is marginalized so that D is eliminated from its domain.

Fusion with respect to a chance variable C is defined as follows. The utility, probability, and indicator valuations whose domains do not include C remain unchanged. A new probability valuation, say ρ, is created by combining all probability and indicator valuations whose domain include C and marginalizing C out of the combination. Finally, we combine all probability and indicator valuations whose domains include C, divide the resulting probability valuation by the new probability valuation ρ that was created, combine the resulting probability valuation with the utility valuations whose domains include C, and finally marginalize the resulting utility valuation such that C is eliminated from its domain. In some special cases—such as if ρ is identically one, or if C is the only chance variable left—we can avoid creating a new probability valuation and the corresponding division.

The solution of the Reactor problem using the fusion algorithm is as follows.

Fusion with Respect to C. First we fuse valuations in $\{\delta_2, \tau_2, \nu_1, \nu_2, \nu_3, \chi, \alpha, \tau_1\}$ with respect to C. Since $\chi^{2\varnothing}$ is identically one,

$$\text{Fus}_C\{\delta_2, \tau_2, \nu_1, \nu_2, \nu_3, \chi, \alpha, \tau_1\} = \{\delta_2, \tau_2, \nu_2, \nu_3, (\nu_1 \otimes \chi)^{2D_2}, \alpha, \tau_1\}.$$

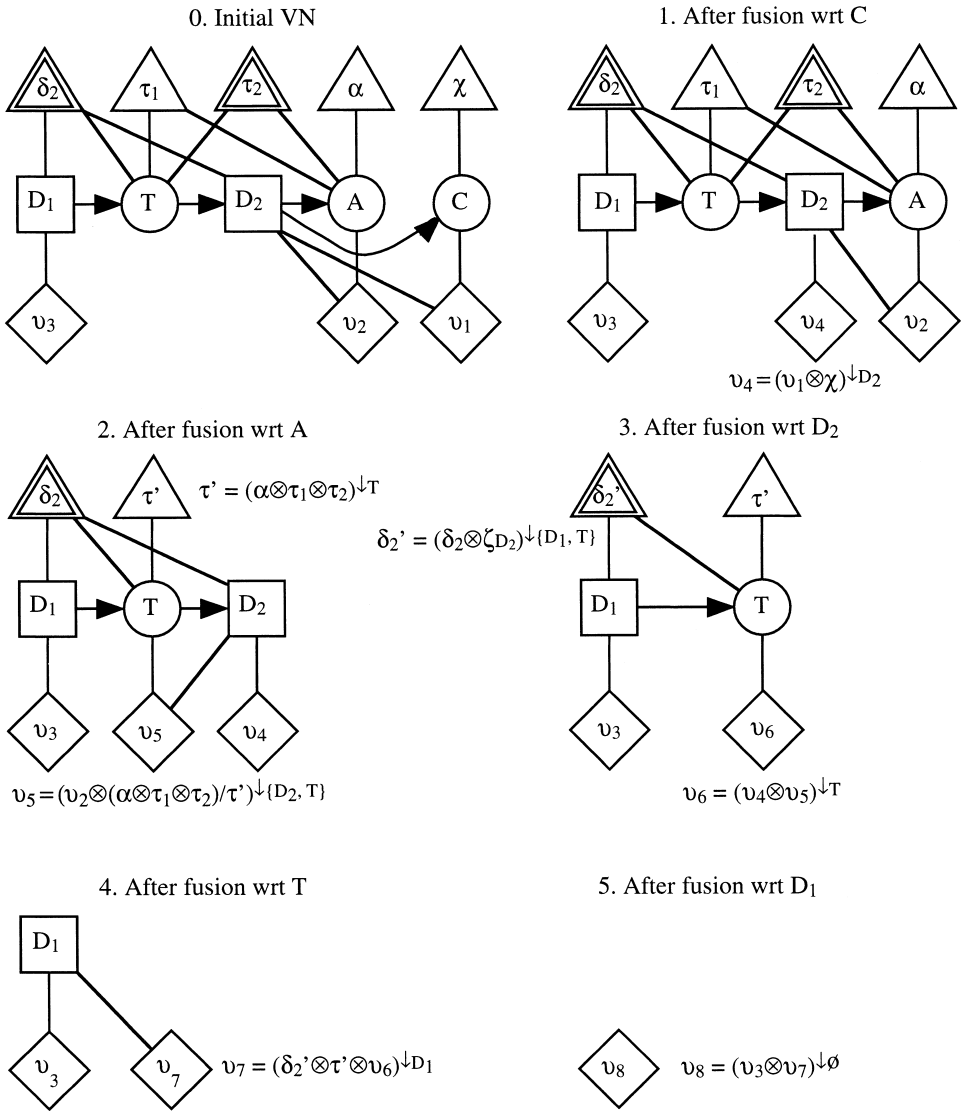
Let ν_4 denote $(\nu_1 \otimes \chi)^{2D_2}$. The details of the numerical computation involved are shown in Table 2. The result of fusion with respect to C is shown graphically in Figure 6.

Fusion with Respect to A. Next, we fuse the valu-

Table 2 **Details of Fusion with Respect to C**

$\Omega_{\{D_2,C\}}$		ν_1	χ	$\nu_1 \otimes \chi$	$(\nu_1 \otimes \chi)^{2D2} = \nu_4$
n	*cs*	0	0.98	0	0
n	*cf*	0	0.02	0	
c	*cs*	8	0.98	7.840	7.760
c	*cf*	−4	0.02	−0.080	
a	*cs*	0	0.98	0	0
a	*cf*	0	0.02	0	

Figure 6 **Fusion in VNs at the Graphical Level**



ations in $\{\delta_2, \tau_2, \nu_2, \nu_3, \nu_4, \alpha, \tau_1\}$ with respect to A. $\text{Fus}_A\{\delta_2, \tau_2, \nu_2, \nu_3, \nu_4, \alpha, \tau_1\} = \{\delta_2, \nu_3, \nu_4, (\nu_2 \otimes (\alpha \otimes \tau_1 \otimes \tau_2)/(\alpha \otimes \tau_1 \otimes \tau_2)^{2\,T})^{2\,\{D_2,T\}}, (\alpha \otimes \tau_1 \otimes \tau_2)^{2\,T}\}$. Let ν_5 denote $(\nu_2 \otimes (\alpha \otimes \tau_1 \otimes \tau_2)/(\alpha \otimes \tau_1 \otimes \tau_2)^{2\,T})^{2\,\{D_2,T\}}$, and let τ' denote $(\alpha \otimes \tau_1 \otimes \tau_2)^{2\,T}$. The result of fusion with respect to A is shown graphically in Figure 6. Notice that all computations are done on effective state spaces, and so we need to compute the effective state space of $\{T, D_2, A\}$ prior to doing the fusion since it has not been already computed during the representation stage (see Bielza and Shenoy (1998) for details).

Fusion with Respect to D_2. Next we fuse $\{\delta_2, \nu_3, \nu_4, \nu_5, \tau'\}$ with respect to D_2. Since D_2 is a decision variable, $\text{Fus}_{D_2}\{\delta_2, \nu_3, \nu_4, \nu_5, \tau'\} = \{(\delta_2 \otimes \zeta_{D_2})^{2\,\{D_1,T\}}, \nu_3, (\nu_4 \otimes \nu_5)^{2\,T}, \tau'\}$, where ζ_{D_2} is the indicator function representation of the decision function for D_2. Let ν_6 denote $(\nu_4 \otimes \nu_5)^{2\,T}$, and δ_2' denote $(\delta_2 \otimes \zeta_{D_2})^{2\,\{D_1,T\}}$. The result of fusion with respect to D_2 is shown graphically in Figure 6.

Fusion with Respect to T. Next we fuse $\{\delta_2', \nu_3, \nu_6, \tau'\}$ with respect to T. Since T is the only chance

variable, $\text{Fus}_R\{\delta_2', \nu_3, \nu_6, \tau'\} = \{\nu_3, (\delta_2' \otimes \tau' \otimes \nu_6)^{2^{D_1}}\}$. Let ν_7 denote $(\delta_2' \otimes \tau' \otimes \nu_6)^{2^{D_1}}$. The result of fusion with respect to T is shown graphically in Figure 6.

Fusion with Respect to D_1. Next we fuse $\{\nu_3, \nu_7\}$ with respect to D_1. Since D_1 is a decision variable, $\text{Fus}_{D_1}\{\nu_3, \nu_7\} = \{(\nu_3 \otimes \nu_7)^{2^{\varnothing}}\}$. Let ν_8 denote $(\nu_3 \otimes \nu_7)^{2^{\varnothing}}$. The result of fusion with respect to D_1 is shown graphically in Figure 6.

This completes the fusion algorithm. An optimal strategy can be pieced together from the decision functions for D_1 and D_2. The optimal strategy and maximum expected utility are the same as in the DT and ID cases. Complete details of the VN solution of the Reactor problem can be found in Bielza and Shenoy (1998).

4.3. Strengths of VNs

Like IDs, VNs are compact and they encode conditional independence relations in the probability model (Shenoy 1994c). Unlike IDs, the VN technique can represent directly every probabilistic model, without any preprocessing. All that is required is a factorization of the joint probability distribution for the chance variables.

The information constraints representation is more flexible in VNs than in IDs. In IDs, all decision nodes have to be completely ordered. This condition is called "no-forgetting" (Howard and Matheson 1981). In VNs, there is a weaker requirement called "perfect recall" (Shenoy 1992). The perfect recall requirement can be stated as follows. Given any decision variable D and any chance variable C, it should be clear whether the true state of C is known or unknown when a choice has to be made at D. The flexibility of information constraints will offer a greater number of allowable reduction sequences than the other techniques. Of course, the perfect recall condition can be easily adapted to the ID domain.

The VN representation technique captures asymmetry through the use of indicator valuations and effective state spaces. Indicator valuations encode structural asymmetry modularly with no duplication, and the effective state space for a subset of variables contains all structural asymmetry information that is relevant for that subset. This contributes to the parsimony of the representation.

In VNs, the joint probability distribution can be decomposed into functions with smaller domains than in IDs. This is so because IDs insist on working with conditionals. For example, the conditional for T has the domain $\{D_1, A, T\}$ in the ID (as seen in Figure 3), and the valuation τ_1 has the domain $\{A, T\}$ in the VN (as seen in Table 1). The distribution tree for T in the ID could be computed from the VN as $\delta_2^{2^{\{D_1,T\}}} \otimes \tau_2 \otimes \tau_1$. One implication of this decomposition is that during the solution phase, the computation is more local, i.e., it involves fewer variables, than in the case of IDs. For example, in the ID technique, reduction of A involves variables D_1, D_2, T, and A (as deduced from Figure 4), whereas in the VN technique, reduction of A only involves variables T, D_2, and A (as deduced from Figure 6).

VNs do not perform *unnecessary* divisions done in DTs, IDs, and in SDDs. In DTs, these unnecessary divisions are done during the preprocessing stage. In IDs and SDDs, the unnecessary divisions are done during arc reversal. For symmetric problems, Ndilikilikesha (1994) and Jensen et al. (1994) have suggested modifications to the ID solution technique to avoid these unnecessary divisions. These modifications need to be generalized to the asymmetric case. In general, with arbitrary potentials and an additive decomposition of the utility function, divisions are often necessary if we want to take advantage of local computation. In this case, VNs, IDs, and SDDs are similar. This is the situation in the Reactor problem, i.e., all divisions done in this problem are necessary.

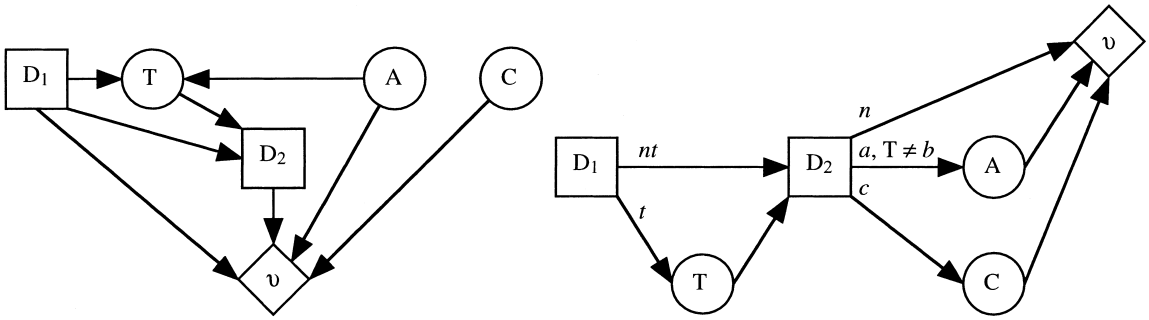
Finally, the VN technique includes conditions that tell us when a representation is well defined for computing an optimal strategy (Shenoy 1993b). These conditions are useful in automating the technique.

4.4. Weaknesses of VNs

The modeling of conditionals is not as intuitive in VNs as in IDs. For example, in the Reactor problem, the probability valuation τ_1 is not a true conditional; it is only a factor of the conditional, i.e., $\delta_2^{2^{\{D_1,T\}}} \otimes \tau_2 \otimes \tau_1$ is a conditional for T given D_2 and A. This factoring of conditionals into valuations with smaller domains makes it difficult to attach semantics for the probability valuations, and this may make it difficult or nonintuitive to represent.

In VNs, the specification of a decision problem is

Figure 7 The Initial ID and the SDD for the Reactor Problem



done sequentially as follows. First, the user specifies the VN diagram. Next, the user specifies the state spaces of all decision and chance nodes, and all indicator valuations. Finally, the user specifies the numerical details of each probability and utility valuations for configurations in the effective state spaces that are computed using local computation from the indicator valuations. Some users may find this sequencing too constraining.

VNs show explicitly the probability distributions as nodes, which implies a greater number of nodes and edges in the diagram and probably more confusion when representing problems with many variables.

4.5. Some Open Issues

A major issue of VNs is their inability to model some asymmetry. For example, in the Reactor problem, we are unable to model the irrelevance of node C for the scenarios $D_2 = n$ or a in the utility valuation ν_1. This issue perhaps can be resolved by adapting the collapsed scenario feature of IDs to VNs.

In comparison with IDs, VNs are unable to use sharing of scenarios and collapsed scenario features of IDs. Consequently, a VN representation may demand more space than a corresponding ID representation that can take advantage of these features. Also, the inability to use sharing and collapsed scenarios features has a computational penalty. For example, in the Reactor problem, reduction of C requires 9 arithmetic operations in VNs as compared to 3 in the case of IDs, and reduction of A requires 80 operations in VNs as compared to 39 in the case of IDs. This issue perhaps can be resolved by adapting the sharing and collapsed scenario features of IDs to VNs. However, VNs can and do represent clipping of scenarios through the use of effective state spaces. The elements of an effective state space include the unclipped conditioning scenarios. Also, VNs can represent partially unspecified distributions by simply not specifying the values for particular elements of the effective state space. However, to avoid the problem of determining when a representation is completely specified for computation of an optimal strategy, it may be better to not use this feature of IDs.

5. Sequential Decision Diagrams

In this section, we will represent and solve the Reactor problem using Covaliu and Oliver's (1995) sequential decision diagram technique. The SDD technique is described either for a problem in which the utility function is undecomposed, or for a problem in which the utility function decomposes into additive (or multiplicative) factors such that each factor has only one variable in its domain. Since our version of the Reactor problem is not in either of these two categories, first we combine the three utility factors and then we use the undecomposed version of the SDD technique to represent and solve the Reactor problem.

5.1. SDD Representation

In this technique, a decision problem is modeled at two levels, graphical and numerical. At the graphical level, we model a decision problem using two directed graphs—an ID to describe the probability model, and a new diagram, called a *sequential decision diagram*, which captures the asymmetric and the information constraints of the problem. Figure 7 shows an ID and

a SDD for the Reactor problem.

At the numerical level, we specify the conditionals for each chance node in the ID, and data built from both diagrams are organized in a *formulation table*, similar to the one used by Kirkwood (1993), in such a way that the recursive algorithm used in the solution process can easily access the data contained in it.

A SDD is a directed acyclic graph, with the same set of nodes as in the ID. However, its paths show all possible scenarios in a compact way, as if it were a schematic decision tree, i.e., a decision tree in which all branches from a decision or chance node leading to the same generic successor node are collapsed together. A SDD is said to be *proper* if (i) there is only one source node (a node with no arrows pointing to it), (ii) there is only one sink node (a node with no arrows emanating from it) and it is the value node, and (iii) there is a directed path that contains all decision nodes.

In the SDD for the Reactor problem, the arc (D_1, T) with the label t tells us that if we perform the test ($D_1 = t$) then we will observe its result ($T = b, g$ or e). Arc (D_1, D_2) with label *nt* tells us that we will not observe T when $D_1 = nt$. Arcs (D_2, ν), (D_2, C), and (D_2, A) show that A is relevant only if $D_2 = a$, and C is relevant only if $D_2 = c$. The label over the arc (D_2, A) also indicates dependence on realized states at predecessor nodes, i.e., the alternative $D_2 = a$ is available only if $T \neq b$. The six directed paths from D_1 to ν in the SDD are a compact representation of the twenty-one possible scenarios in the decision tree representation (Figure 1).

Notice that the partial order implied by the arrows in an ID may be different from the partial order implied by the arrows in a corresponding SDD. Let $<_D$ and $<_I$ denote the partial orders in SDD and ID respectively. If C is a chance node, D is a decision node, and $C <_I D$ implies $C <_D D$, then we say the ID and SDD are *compatible* (Covaliu and Oliver 1995). In Figure 7, for example, we have $A <_I D_2$ (since there is a directed path from A to D_2 in the ID), and $D_2 <_D A$ (since there is an arrow from D_2 to A in the SDD). Therefore the two diagrams are incompatible. The next step in completing the SDD representation is to transform the ID so that it is compatible with the SDD. In the Reactor problem, we must reverse the arc (A, T) in the ID to make the ID compatible with the SDD. The transformed ID is shown in Figure 8.

Figure 8 **The Transformed ID**

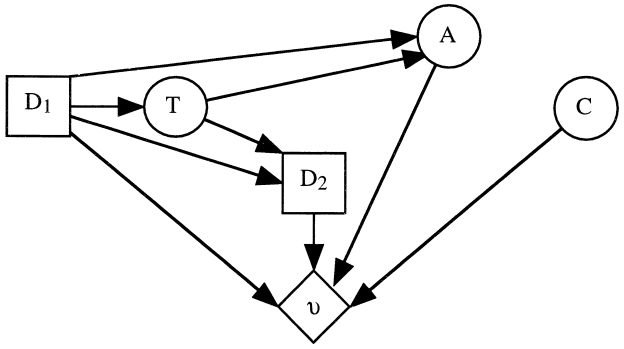


Table 3 **A Formulation Table for the Reactor Problem**

Node Name	Node Type	Standard Histories (Minimal in bold)	State Space	Probability Distribution	Next-Node Function
D_1	decision	$\varnothing$	*nt* *t*		D_2 T
T	chance	D_1			
		t	*b* *g* *e*	0.1 0.3 0.6	D_2 D_2 D_2
D_2	decision	D_1 $\boldsymbol{T}$			
		nt **–**	*n* *c* *a*		ν C A
		t ***b***	*n* *c*		ν C
		t ***g***	*n* *c* *a*		ν C A
		t ***e***	*n* *c* *a*		ν C A
A	chance	D_1 $\boldsymbol{T}$ D_2			
		nt **–** *a*	*as* *al* *am*	0.660 0.244 0.096	ν ν ν
		t ***g*** *a*		0.400 0.460 0.140	
		t ***e*** *a*		0.900 0.060 0.040	
C	chance	D_1 T D_2			
		nt – *c*	*cs* *cf*	0.98 0.02	ν ν
		t *b* *c*			
		t *g* *c*			
		t *e* *c*			
ν	value	$\boldsymbol{D_1}$ T $\boldsymbol{D_2}$ $\boldsymbol{A}$ $\boldsymbol{C}$			
		nt – ***n*** **–** **–**	0		
		nt – ***c*** **–** ***cs***	8		
		... ...	...		
		t *e* ***a*** ***am*** **–**	−11		

Next, we organize data in the formulation table, which contains the complete information the solution algorithm will require. Table 3 is the formulation table

for the Reactor problem. Not all details of the utility function v are shown here—see Bielza and Shenoy (1998) for details.

The formulation table has a row for each node in the SDD. If $X <_D Y$, then the row for X precedes the row for Y. Each row includes node name, node type, standard histories and minimal histories, state space, conditional distribution (for chance nodes only), and next-node function. It should be noted that the formulation table is part of the representation of the decision problem.

The term *history* refers to how one gets to a node through the directed paths in the SDD. It can be represented as a two-row matrix, the first row listing a node sequence of all nodes that precede it in the partial order, and the second row listing the corresponding realized states. The next-node function (in the last column) denotes the node that is realized after a node for each of its states and for each minimal history. There are different kinds of histories. Minimal histories are sufficient for defining node state spaces, probability distributions (for chance nodes), and next node functions. For a decision node, the minimal histories will include those variables that affect its state space, and its next-node function. For example, for D_2, variable T is the only one under these conditions. So, at node D_2 we have the minimal histories:

$$\begin{pmatrix} T \\ - \end{pmatrix}, \quad \begin{pmatrix} T \\ b \end{pmatrix}, \quad \begin{pmatrix} T \\ g \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} T \\ e \end{pmatrix},$$

where – denotes the absence of T in a path to D_2, i.e., when $D_1 = nt$. For a chance node, the minimal histories will include the nodes that suffice for defining its next-node function, and its conditional probability distribution. For example, for C, the set of minimal histories is the empty set. For a value node, the minimal histories will include the nodes that suffice to define the values of the corresponding utility function and they are the direct predecessors of v in the ID.

As we will see, minimal histories are not always sufficient to solve a decision problem. We need a new kind of history, called relevant history. The node sets of relevant histories contain the node sets of minimal histories and are contained in the node sets of full histories. Also, relevant histories can be computed from minimal and full histories. Therefore, we do not show relevant histories in the formulation, just full and minimal histories.

5.2. SDD Solution

Let $w_N(H_N) = E(u|H_N)$ denote the maximum expected utility at node N of the SDD given history H_N if optimal decisions are made at N (if N is a decision node) and from there onwards. Let $\mathcal{N}(H_N)$ denote the set of nodes in H_N. The solution technique is based on the same backward recursive relations used in decision trees, but here we use a new kind of history called relevant history. We cannot use only minimal histories because, when calculating $w_N(H_N)$, we may reference the next nodes n_N and their histories H_{n_N}, and $w_N(H_N)$ is not well defined if there exists at least one n_N such that $\mathcal{N}(H_{n_N}) - \{N\} \not\subset \mathcal{N}(H_N)$. We obtain the node sets in the relevant histories by enlarging the node sets in minimal histories by those SDD predecessors that appear in the node sets of relevant histories of any of the direct successors nodes. Covaliu and Oliver (1995) give a recursive definition of this term. The solution algorithm then follows a backward recursive method. We will describe in detail a part of the solution—the reduction of C.

Reduction of Node C. Let w_v denote the utility function associated with node v in the formulation table, e.g.,

$$w_v\left(\begin{pmatrix} D_1 & D_2 & C & A \\ nt & n & - & - \end{pmatrix}\right) = 0, \quad \text{etc.}$$

The relevant history for node C includes nodes D_1 and D_2 (since C's minimal history node set is $\varnothing$, C's successor is node v, v's minimal history node set is $\{D_1, D_2, C, A\}$, and the set of predecessors of C is $\{D_1, T, D_2\}$).

$$\begin{aligned} w_C\left(\begin{pmatrix} D_1 & D_2 \\ nt & c \end{pmatrix}\right) &= w_v\left(\begin{pmatrix} D_1 & D_2 & C \\ nt & c & cs \end{pmatrix}\right)(0.98) \\ &\quad + w_v\left(\begin{pmatrix} D_1 & D_2 & C \\ nt & c & cf \end{pmatrix}\right)(0.02) \\ &= w_v\left(\begin{pmatrix} D_1 & D_2 & C & A \\ nt & c & cs & - \end{pmatrix}\right)(0.98) \\ &\quad + w_v\left(\begin{pmatrix} D_1 & D_2 & C & A \\ nt & c & cf & - \end{pmatrix}\right)(0.02) \\ &= (8)(0.98) + (-4)(0.02) \\ &= 7.760. \end{aligned}$$

Similarly, $w_C(\binom{D_1 \quad D_2}{t \quad c}) = 6.76$.

For further details, see Bielza and Shenoy (1998).

5.3. Strengths of SDDs

The main strength of SDDs is their ability to represent asymmetry compactly. A SDD can be thought of as a compact (schematic) version of a DT. Thus, we get the intuitiveness of DTs without their combinatorial explosion.

Like DTs, SDDs model asymmetry without adding dummy states to variables. This is in contrast to IDs and VNs that enlarge the state spaces of some variables in order to represent asymmetry. For example, in the Reactor problem, the state space of T is $\{b, g, e\}$ in the DT and SDD representation, whereas it is $\{nr, b, g, e\}$ in the ID and VN representation.

In the solution phase, SDDs avoid working on the space of all scenarios (or histories) by using minimal and relevant histories. Thus, they can exploit coalescence automatically, which DTs cannot.

The SDD technique can detect the presence of unnecessary information in a problem by identifying *irrelevant* or *barren* nodes (Covaliu 1996). This leads to a simplification of the original model and to a corresponding decrease in the computational burden of solving it.

5.4. Weaknesses of SDDs

The main weakness of the SDD technique is its inability to represent probability models consistently. It uses one distribution in the ID representation that is compatible with the SDD and a different one in the formulation table. For example, in the Reactor problem, the state space of T in the ID representation in Figure 8 includes nr whereas the state space of T in the formulation table does not include nr.

Before one can complete a SDD formulation, including specifying a formulation table, it may be necessary to preprocess the probabilities. This means that the ID has to be modified to make it compatible with a corresponding SDD. In the reactor problem, there was only one incompatibility requiring one arc reversal. In other problems, there may be many incompatibilities requiring many arc reversals. In such problems, it is not clear which arcs one should reverse and in what sequence so as to achieve compatibility at minimum computational cost. In large problems, the lack of a formal method to translate a probability model from an ID to a formulation table may make the SDD technique unsuitable for large problems requiring Bayesian revision of probabilities.

In a formulation table, the nodes are linearly ordered by rows. This linear ordering is used during the solution process—the variable in a row is reduced only after all the variables in succeeding rows are reduced. Ideally, the ordering of nodes for reduction should belong to the solution phase and not to the formulation phase. If an arbitrary linear order is chosen (compatible with the partial order in SDD), there may be a computational penalty (see Shenoy (1994a) for an example of this phenomenon). This weakness is also shared by DTs.

5.5. Some Open Issues

As in DTs, SDDs require that a unique node be defined as a next node for each state of a variable in a formulation table even though the problem may allow several nodes to qualify as next nodes. The choice of which node should be a next node is a computational issue and it properly belongs to the solution phase, not to the formulation phase. This issue perhaps can be resolved by allowing the next node to be a subset instead of a single node. This strategy is advocated by Guo and Shenoy (1996) in the context of Kirkwood's algebraic method.

The SDD technique tells us how to compute minimal history node sets and relevant history node sets. However, once we have the node sets, we still have to generate the corresponding histories. Generating these from a list of standardized histories is one possibility. However, for large problems, the number of standardized histories is an exponential function of the number of variables. Thus, we need procedures for generating minimal histories and relevant histories from the corresponding node sets without actually listing all standardized histories. This is a task that remains to be done.

A complete SDD formulation of a problem consists of an ID and a SDD at the graphical level, and conditionals for the chance variables in the ID and a formulation table at the numerical level. The formulation table is built partly from information from a compatible ID (e.g., the conditional probability distributions), and partly from the SDD (e.g., the histories). Thus, there is duplication of information. A complete ID representation is sufficient for solving the problem. A corresponding SDD dupli-

cates some of this information. And a formulation table that includes standardized histories has all information required for solving the problem. Thus, a formulation table duplicates information contained in an ID and a SDD. This issue can be resolved by developing a solution technique that solves a decision problem directly from a SDD and a corresponding ID representation without having the user specify a formulation table.

As currently described, the SDD technique tells us only how to represent a problem with a single undecomposed utility function or a problem in which the utility function decomposes into factors whose domains include only one variable. The case of an arbitrary decomposition of the utility function is not covered. Also, it is not clear when a SDD representation is well defined. These are tasks that remain to be done.

6. Conclusion

The main goal of this work is to compare four distinct techniques proposed for representing and solving asymmetric decision problems—traditional decision trees, SHM influence diagrams, Shenoy's valuation networks, and Covaliu and Oliver's sequential decision diagrams. For each technique, we have identified the main strengths, intrinsic weaknesses, and some open issues that perhaps can be resolved with further research.

One conclusion is that no single technique stands out as always superior in all respects to the others. Each technique has some unmatched strengths. Another conclusion is that considerable work remains to be done to resolve the open issues of each technique. One possibility here is to borrow the strengths of a technique to resolve the issues of another. Also, there is need for automating each technique by building computer implementations, and there is very little literature on this topic.[2]

[2] There are several implementations of the decision tree technique, e.g., DATA [www.treeage.com], PrecisionTree [www.palisade.com], and Supertree [www.sdg.com]; several implementations of the symmetric influence diagram technique, e.g., Hugin [www.hugin.dk], Netica [www.norsys.com], and Analytica [www.lumina.com]; and one implementation of the asymmetric influence diagram technique based on Call and Miller's (1990) technique, namely DPL [www.adainc.com]. Currently, there are no implementations of either SHM's IDs or Shenoy's asymmetric VNs or Covaliu and Oliver's SDDs. For details of other software for decision analysis, see Buede (1996).

We conclude with some speculative comments about the class of problems for which each technique is appropriate. Decision trees are appropriate for small decision problems. Influence diagrams are appropriate for problems in which we have a Bayesian network model for the uncertainties. Valuation networks are appropriate for problems in which we have a non-Bayesian network model for the uncertainties such as undirected graphs, chain graphs, etc. Finally, sequential decision diagrams are appropriate for problems for which we have a Bayesian network model for the uncertainties such that no Bayesian revision of probabilities is required and for which the utility function decomposes into factors whose domains are singleton variable subsets.[3]

[3] We are grateful to Jim Smith, Zvi Covaliu, David Ríos Insua, Robert Nau, and an anonymous reviewer for extensive comments on earlier drafts.

References

Bielza, C., P. P. Shenoy. 1998. A comparison of graphical techniques for asymmetric decision problems: Supplement to Management Science paper. Working Paper No. 282, School of Business, University of Kansas, Lawrence, KS. Available by ftp from ⟨ftp.bschool.ukans.edu/home/pshenoy/wp282.ps⟩.

Buede, D. 1996. Aiding insight III. *OR/MS Today* **23**(4) 73–79.

Call, H. J., W. A. Miller. 1990. A comparison of approaches and implementations for automating decision analysis. *Reliability Engnrg. and System Safety* **30** 115–162.

Covaliu, Z. 1996. Sequential diagrams and influence diagrams: A complementary relationship for modeling and solving decision problems. *Bayesian Statistics 5* Oxford University Press, 521–532.

——, R. M. Oliver. 1995. Representation and solution of decision problems using sequential decision diagrams. *Management Sci.* **41**(12) 1860–1881.

Cowell, R. G. 1994. Decision networks: A new formulation for multistage decision problems. Research Report No. 132, University College, London, U.K.

Fung, R. M., R. D. Shachter. 1990. Contingent influence diagrams. Working Paper Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

Goutis, C. 1995. A graphical method for solving a decision analysis problem. *IEEE Trans. Systems, Man, & Cybernetics* **25**(8) 1181–1193.

Guo, R., P. P. Shenoy. 1996. A note on Kirkwood's algebraic method for decision problems. *European J. Oper. Res.* **93** 628–638.

Jensen, F., F. V. Jensen, S. L. Dittmer. 1994. From influence diagrams to junction trees. R. L. Mantaras, D. Poole, eds. *Uncertainty in*

Artificial Intelligence: *Proceedings of the Tenth Conference*. Morgan Kaufmann, San Francisco, CA. 367–373.

Kirkwood, C. W. 1993. An algebraic approach to formulating and solving large models for sequential decisions under uncertainty. *Management Sci*. **39**(7) 900–913.

Olmsted, S. M. 1983. On representing and solving decision problems, Ph.D. Thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

Howard, R. A., J. E. Matheson. 1981. Influence diagrams. R. A. Howard, J. E. Matheson, eds. *The Principles and Applications of Decision Analysis*. 1984 **2** Strategic Decisions Group, Menlo Park, CA. 719–762.

Ndilikilikesha, P. C. 1994. Potential influence diagrams. *Internat. J. Approximate Reasoning* **10**(3) 251–285.

Qi, R., L. Zhang, D. Poole. 1994. Solving asymmetric decision problems with influence diagrams. R. L. Mantaras, D. Poole, eds. *Uncertainty in Artificial Intelligence*: *Proceedings of the Tenth Conference*. Morgan Kaufmann, San Francisco, CA. 491–497.

Shachter, R. D. 1986. Evaluating influence diagrams. *Oper. Res*. **34**(6) 871–882.

——. 1988. Probabilistic inference and influence diagrams. *Oper. Res*. **36**(4) 589–604.

——, M. A. Peot. 1992. Decision making using probabilistic inference methods. D. Dubois, M. P. Wellman, B. D'Ambrosio, P. Smets, eds. *Uncertainty in Artificial Intelligence*: *Proceedings of the Eighth Conference*. Morgan Kaufmann, San Mateo, CA. 276–283.

Shenoy, P. P. 1992. Valuation-based systems for Bayesian decision analysis. *Oper. Res*. **40**(3) 463–484.

——. 1993a. A new method for representing and solving Bayesian decision problems. D. J. Hand, ed. *Artificial Intelligence Frontiers in Statistics*: *AI and Statistics III*. Chapman & Hall, London, UK. 119–138.

——. 1993b. Valuation network representation and solution of asymmetric decision problems. *European J. Oper. Res*. **121**(3) 146–174. Available by ftp from ⟨ftp.bschool.ukans.edu/home/pshenoy/wp246.ps⟩.

——. 1994a. A comparison of graphical techniques for decision analysis. *European J. Oper. Res*. **78**(1) 1–21.

——. 1994b. Consistency in valuation-based systems. *ORSA J. Comput*. **6**(3) 281–291.

——. 1994c. Representing conditional independence relations by valuation networks. *Internat. J. Uncertainty, Fuzziness and Knowledge-Based Systems* **2**(2) 143–165.

——. 1995. A new pruning method for solving decision trees and game trees. P. Besnard, S. Hanks, eds. *Uncertainty in Artificial Intelligence*: *Proceedings of the Eleventh Conference*. Morgan Kaufmann, San Francisco, CA. 482–490.

——. 1996. Representing and solving asymmetric decision problems using valuation networks. D. Fisher, H.-J. Lenz, eds. *Learning from Data*: *Artificial Intelligence and Statistics V*. *Lecture Notes in Statistics* **112** Springer-Verlag, New York. 99–108.

——. 1998. Game trees for decision analysis. *Theory and Decision* **44** 149–177.

——, G. Shafer. 1990. Axioms for probability and belief-function propagation. R. D. Shachter, T. S. Levitt, J. F. Lemmer, L. N. Kanal, eds. *Uncertainty in Artificial Intelligence* **4** North-Holland, Amsterdam. 169–198.

Smith, J. Q. 1989. Influence diagrams for Bayesian decision analysis. *European J. Oper. Res*. **40** 363–376.

Smith, J. E., S. Holtzman, J. E. Matheson. 1993. Structuring conditional relationships in influence diagrams. *Oper. Res*. **41**(2) 280–297.

Tatman, J. A., R. D. Shachter. 1990. Dynamic programming and influence diagrams. *IEEE Trans. Systems, Man, and Cybernetics* **20**(2) 365–379.

von Neumann, J., O. Morgenstern. 1944. *Theory of Games and Economic Behavior*, 1st ed. Princeton University Press, Princeton, NJ.

Zhang, N. L., R. Qi, D. Poole. 1994. A computational theory of decision networks. *Internat. J. Approximate Reasoning* **11**(2) 83–158.

Accepted by Robert Nau; *received November* 25, 1996. *This paper has been with the authors* 7 *months for* 4 *revisions*.