A New Perspective on Algorithms for Optimizing Policies Under Uncertainty

Rina Dechter
Department of Computer and Information Science
University of California, Irvine
Irvine, California, USA 92717
dechter@@ics.uci.edu

Abstract

The paper takes a fresh look at algorithms for maximizing expected utility over a set of policies, that is, a set of possible ways of reacting to observations about an uncertain state of the world. Using the bucket-elimination framework, we characterize the complexity of this optimization task by graph-based parameters, and devise an improved variant of existing algorithms. The improvement is shown to yield a dramatic gain in complexity when the probabilistic subgraph (of the influence diagram) is sparse, regardless of the complexity introduced by its utility subgraph.

Introduction

Influence diagram (IDs) (Howard & Matheson 1984) are a popular framework for decision analysis. They subsume finite horizon factored observable and partially observable, Markov decision processes (MDPs, POMDPs) used to model planning problems under uncertainty (Boutilier, Dean, & Hanks 1999).

The paper presents a bucket elimination algorithm for computing a sequence of policies which maximize the expected utility (meu) for an influence diagram. An earlier version of this algorithm (Dechter 1996) describes the algorithm only for the very restricted case where decision variables are roots in the network, namely for "blind" policies. Our derivation highlights topological properties of these algorithms, and in particular ties its time and space complexity to the induced width of an underlying graph.

In principle, the algorithm is similar to the variable elimination algorithm proposed by Shachter and others (Shachter 1986; 1988; 1990; Tatman & Shachter 1990; Shachter & Peot 1992; Shenoy 1992; Zhang 1998; F. Jensen & Dittmer 1994), and in particular, it is analogous to the join-tree clustering algorithm for evaluating influence diagrams (F. Jensen & Dittmer 1994). However, the new exposition using the bucket data-structure unifies the algorithm with a variety of inference algorithms in constraint satisfaction, propositional satisfiability, dynamic programming and probabilistic inference. Such algorithms can be expressed succinctly, are relatively easy to implement and the unification highlights ideas for improved performance, for incorporating variable elimination with search, for trading space for time and for approximation, all of which are applicable within the framework of bucket-elimination (Dechter 1996; 1999).

The derivation highlights the role of decision variables in increasing the algorithm's complexity, by forcing dependencies over independent reward components. However, we also show that performance can be improved exponentially relative to chance variables. In particular, we show, that computing expected utility can be improved dramatically in cases when the probabilistic subgraph is sparse, regardless of the complexity of the utility components. For example, when the belief subnetwork is singly connected (a poly-tree), computing the network's expected utility can be accomplished in linear time, even if the graph describing the utility components is fully connected. To our knowledge, this improvement is not yet incorporated in existing algorithms.

Background

Belief networks

Belief networks provide a formalism for reasoning about partial beliefs under conditions of uncertainty. It is defined by a directed acyclic graph over nodes representing random variables of interest The arcs are quantified by conditional probabilities that are attached to each cluster of parents-child nodes in the network.

A *directed graph* is a pair, $G = \{V, E\}$, where $V = \{X_1, ..., X_n\}$ is a set of elements and $E = \{(X_i, X_j)|X_i, X_j \in V, i \neq j\}$ is the set of edges. If $(X_i, X_j) \in E$, we say that X_i points to X_j. For each variable X_i, the set of parent nodes of X_i, denoted pa_{X_i} or pa_i, comprises the variables pointing to X_i in G. The family of X_i, F_i, includes X_i and its parent variables. A directed graph is *acyclic* if it has no directed cycles. In an *undirected graph*, the directions of the arcs are ignored: (X_i, X_j) and (X_j, X_i) are identical. The *moral graph* of a directed graph is the undirected graph obtained by connecting the parent nodes of each variable and eliminating direction.

Let $X = \{X_1, ..., X_n\}$ be a set of random variables over multivalued domains. A *belief network* is a pair (G, P) where G is a directed acyclic graph and $P = \{P(X_i|pa_i)\}$, denote conditional probability tables (CPTs). The belief network represents a probability distribution over X having the product form

$$P(x_1, ..., x_n) = \Pi_{i=1}^n P(x_i|x_{pa_i})$$

where an assignment $(X_1 = x_1, ..., X_n = x_n)$ is abbreviated to $x = (x_1, ..., x_n)$ and where x_S denotes the restriction of a tuple x to the subset of variables S. An evidence set e is an instantiated subset of variables. We use upper case letter for variables and nodes in a graph and lower case letters for values in variables' domains.

Influence diagrams

An *Influence diagram* extends belief networks by adding also *decision variables* and reward functional components. Formally, an influence diagram is defined by $ID = (X, D, P, R)$, where $X = \{X_1, ..., X_n\}$ is a set of chance variables on multi-valued domains (the belief network part) and $D = \{D_1, ..., D_m\}$ is a set of decision nodes (or actions). The chance variables are further divided into *observable* meaning they will be observed during execution, or *unobservable*. The discrete domains of decision variables denote its possible set of action. An action in the decision node D_i is denoted by d_i. Every chance node X_i is associated with a conditional probability table (CPT), $P_i = \{P(X_i|pa_i)\}$, $pa_i \subseteq X \cup D - \{X_i\}$. Each decision variable D_i has a parent set $pa_{D_i} \subseteq X \cup D$ denoting the variables, whose values will be known and may directly affect the decision. The reward functions $R = \{r_1, ..., r_j\}$ are defined over subsets of variables $Q = \{Q_1, ..., Q_j\}$, $Q_i \subseteq X \cup D$, called *scopes*, and the utility function is defined by $u(x) = \sum_j r_j(x_{Q_j})$.[1].

The graph of an ID contains nodes for chance variables (circled) decision variables (boxed) and for reward components (diamond). For each chance or decision node there is an arc directed from each of its parent variables towards it, and there is an arc directed from each variable in the scope of a reward component towards its reward node.

Let $D_1, ..., D_m$ be the decision variables in an influence diagram. A decision rule for a decision node D_i is a mapping

$$\delta_i : \Omega_{pa_{D_i}} \rightarrow \Omega_{D_i}$$

where for $S \subseteq X \cup D$, Ω_S is the cross product of the individual domains of variables in S. A policy is a list of decision rules $\Delta = (\delta_1, ..., \delta_m)$ consisting of one rule for each decision variable. To *evaluate an influence diagram* is to find an *optimal policy* that maximizes the expected utility (*meu*) and to compute the optimal expected utility. Assume that x is an assignment over both chance variables and decision variables $x = (x_1, ..., x_n, d_1, ..., d_m)$, The meu task is to compute

$$E = \max_{\Delta=(\delta_1,\ldots,\delta_m)} \sum_{x_1,\ldots,x_n} \Pi_{x_i} P(x_i, e|x^{\Delta}_{pa_i}) u(x^{\Delta}), \quad (1)$$

where x^{Δ} denotes an assignment $x = (x_1, ..., x_n, d_1, ...d_m)$ where each d_i is determined by $\delta_i \in \Delta$ as a functions of $(x_1, ..., x_n)$. Namely: $d_i = \delta_i(x)$.

Example 1 *Figure 1 describes the influence diagram of the oil wildcatter problem (adapted from (N. L. Zhang & Poole 1994)). The diagram shows that the* test decision (T) *is to be made based on no information, and the* drill (D) *decision is to be made based on the decision to* test (T) *and the* test results (R). *The test-results are dependent on* test *and* seismic-structure (S), *which depends on an unobservable variable* oil underground (O). *The decision regarding* oil-sales-policy (OSP) *is made once the* market information (MI) *and the* oil-produced (OP) *are available. There are four reward components: cost of test,* $r(T)$, *drill cost* $r(D, OP)$, *sales cost* $r(OP, OSP)$ *and oil sales* $r(OP, OSP, MI)$. *The meu task is to find the three decision rules* δ_T, δ_D *and* δ_{OSP}: $\delta_T :\rightarrow \Omega_T$, $\delta_D : \Omega_R \rightarrow \Omega_D$, $\delta_{OSP} : \Omega_{MI,OP} \rightarrow \Omega_{OSP}$ *such that:*

$$E = \max_{\delta_T,\delta_D,\delta_{OSP}} \sum_{t,r,op,mi,s,o} P(r|t, s)P(op|d, o)P(mi)P(s|o)\cdot$$
$$P(o)[r(t) + r(d, op) + r(op, osp) + r(op, osp, mi)]$$

IDs are required to satisfy several constraints. There must be a directed path that contains all the decision nodes and there must be no forgetting in the sense that a decision node and its parents be parents to all subsequent decision nodes. The rationale behind the no-forgetting constraint is that information available now should be available later if the decision-maker does not forget. In this paper, however we do not force these requirements. For a discussion of the implications of removing these restrictions see (N. L. Zhang & Poole 1994).

Definition 1 (elimination functions) *Given a function h defined over subset of variables S, where $X \in S$, the functions $(\sum_X h)$ is defined over $U = S - \{X\}$ as follows. For every $U = u$, $(\sum_X h)(u) = \sum_x h(u, x)$. Given a set of functions $h_1, ..., h_j$ defined over the subsets $S_1, ..., S_j$, the product function $(\Pi_j h_j)$ and $\sum_J h_j$ are defined over $U = \cup_j S_j$. For every $U = u$, $(\Pi_j h_j)(u) = \Pi_j h_j(u_{S_j})$, and $(\sum_j h_j)(u) = \sum_j h_j(u_{S_j})$.*

An elimination algorithm for MEU

We will first derive the bucket elimination algorithm using the well known car example (Howard 1976), and subsequently derive the general case.

[1]The original definition of ID had only one reward node. We allow multiple rewards as discussed in (Tatman & Shachter 1990)

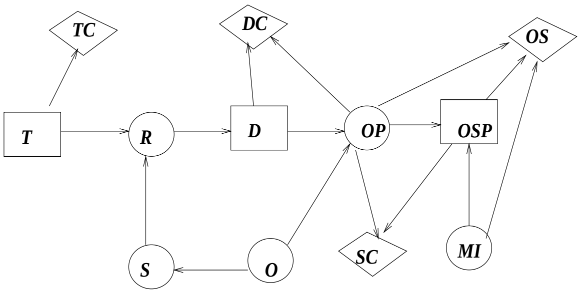

Figure 1: An influence diagram

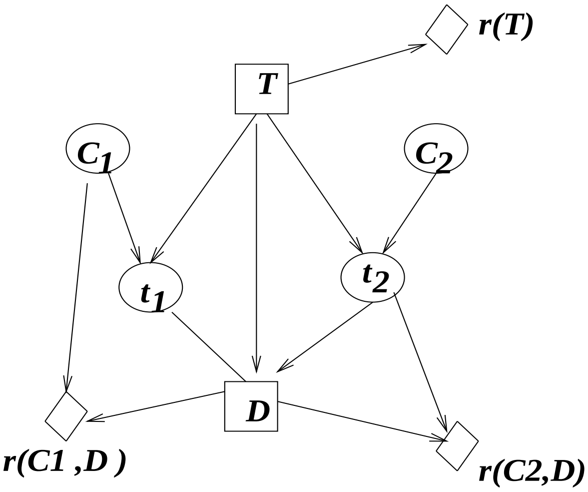

Figure 2: Influence diagram for car buyer example

The car buyer example

Consider a car buyer that needs to buy one of two used cars. The buyer of the car can carry out various tests with various costs, and then, depending on the test results, decide which car to buy. Figure 2 gives the influence diagram representing the car buyer example adapted from (Pearl 1988). T denotes the choice of test to be performed, $T \in \{t_0, t_1, t_2\}$ (t_0 means no test, t_1 means test car 1, and t_2 means test car 2). D stands for the decision of which car to buy, $D \in \{buy\ 1,\ buy 2\}$. C_i represents the quality of car i, $i \in \{1, 2\}$ and $C_i \in \{q_1, q_2\}$ (denoting good and bad qualities. t_i represents the outcome of the test on car i, $t_i \in \{pass,\ fail,\ null\}$. (The "null" value is for the irrelevant case when the Test is applied to car 1 and the results are related to car 2, and vice versa.) The cost of testing is given as a function of T, by $r(T)$, the reward in buying car 1 is defined by $r(C_1, D = buy\ 1)$ and the reward for buying car 2 is $r(C_2, D = buy\ 2)$. The rewards $r(C_2, D = buy\ 1) = 0$, and $r(C_1, D = buy\ 2) = 0$. The total reward or utility is given by $r(T) + r(C_1, D) + r(C_2, D)$.

The task is to determine T and D that maximize the expected utility. Namely,

$$E = \max_{T,D} \sum_{t_2,t_1,C_2,C_1} P(t_2|C_2,T)P(C_2)P(t_1|C_1,T) \cdot$$
$$P(C_1)[r(T) + r(C_2, D) + r(C_1, D)].$$

Reorganizing, and pushing operators as far to the right, while ordering decision variables to follow their parents, and unobservable chance variables (C_1, C_2) to be placed last in the order, we get

$$E = \max_T \sum_{t_2} \sum_{t_1} \max_D \sum_{C_2} P(t_2|C_2,T)P(C_2) \cdot$$
$$[\sum_{C_1} P(t_1|C_1,T)P(C_1)[r(T) + r(C_1, D) + r(C_2, D)] = \tag{2}$$

Performing the summation or maximization from right to left we get,
summing over C_1 first:

$$E = \max_T \sum_{t_2} \sum_{t_1} \max_D \sum_{C_2} P(t_2|C_2,T)P(C_2)\lambda_{C_1}(t_1, T) \cdot$$
$$[r(T) + r(C_2, D) + \theta_{C_1}(t_1, T, D)] \tag{3}$$

where

$$\lambda_{C_1}(t_1, T) = \sum_{C_1} P(t_1|C_1,T)P(C_1) \tag{4}$$

and

$$\theta_{C_1}(t_1, T, D) = \frac{1}{\lambda_{C_1}(t_1, T)} \cdot \sum_{C_1} P(t_1|C_1,T)P(C_1)r(C_1, D) \tag{5}$$

Therefore, eliminating C_1 creates a new probabilistic component (λ_{C_1}) and a new utility component (θ_{C_1}).

Pushing expression not involving C_2 and **summing over C_2, we get (from EQ. (3):**

$$E = \max_T \sum_{t_2} \sum_{t_1} \lambda_{C_1}(t_1, T) \max_D \sum_{C_2} P(t_2|C_2,T)P(C_2) \cdot$$
$$[r(T) + r(C_2, D) + \theta_{C_1}(t_1, T, D)]$$
$$E = \max_T \sum_{t_2} \sum_{t_1} \lambda_{C_1}(t_1, T) \max_D \lambda_{C_2}(t_2, T) \cdot \tag{6}$$
$$[r(T) + \theta_{C_1}(t_1, T, D) + \theta_{C_2}(t_2, T, D)]$$

where

$$\lambda_{C_2}(t_2, T) = \sum_{C_2} P(t_2|C_2,T)P(C_2) \tag{7}$$

and

$$\theta_{C_2}(t_2, T, D) = \frac{1}{\lambda_{C_2}(t_2, T)} \sum_{C_2} P(C_2|t_2,T)P(C_2)r(C_2, D). \tag{8}$$

Again, processing C_2 created a probabilistic component (EQ. 7) and a utility component (EQ. 8).

Reorganizing EQ. (6) and **maximizing over D, we get:**

$$E = \max_T \sum_{t_2} \lambda_{C_2}(t_2,T) \sum_{t_1} \lambda_{C_1}(t_1,T) \cdot \quad (9)$$

$$[r(T) + \max_D(\theta_{C_1}(t1,T,D) + \theta_{C_2}(t_2,T,D))] =$$

$$\max_T \sum_{t_2} \lambda_{C_2}(t_2,T) \sum_{t_1} \lambda_{C_1}(t_1,T)[r(T) + \theta_D(t_1,t_2,T)] \quad (10)$$

where

$$\theta_D(t_1,t_2,T) = \max_D[\theta_{C_1}(t_1,T,D) + \theta_{C_2}(t_2,T,D)] \quad (11)$$

Here, we generated only one utility component by maximizing over the relevant sum of utilities.
Next summing over t_1, from 10, we get:

$$E = \max_T \sum_{t_2} \lambda_{C_2}(t_2,T)\lambda_{t_1}(T)[r(T) + \theta_{t_1}(t_2,T)] \quad (12)$$

where

$$\lambda_{t_1}(T) = \sum_{t_1} \lambda_{C_1}(t_1,T) \quad (13)$$

and

$$\theta_{t_1}(t_2,T) = \frac{1}{\lambda_{t_1}(T)} \sum_{t_1} \lambda_{C_1}(t_1,T)\theta_D(t_1,t_2,T)] \quad (14)$$

Summing the result over t_2, from 12, we get:

$$E = \max_T \lambda_{t_1}(T) \sum_{t_2} \lambda_{C_2}(t_2,T)[r(T) + \theta_{t_1}(t_2,T)] = \quad (15)$$

$$\max_T \lambda_{t_1}(T)\lambda_{t_2}(T)[r(T) + \theta_{t_2}(T)] \quad (16)$$

where

$$\lambda_{t_2}(T) = \sum_{t_2} \lambda_{C_2}(t_2,T) \quad (17)$$

and

$$\theta_{t_2}(T) = \frac{1}{\lambda_{t_2}(T)} \sum_{t_2} \lambda_{C_2}(t_2,T)\theta_{t_1}(t_2,T) \quad (18)$$

Finally, we find T that maximizes:

$$E = \max_T \lambda_{t_1}(T)\lambda_{t_2}(T)[r(T) + \theta_{t_2}(T)]. \quad (19)$$

So, for each chance variable we generate a probability component, λ (by a local product and summation) and a utility component θ (by computing local normalized expected utility.) Decision variables were processed by maximizing over the relevant sum of utility functions, generating a θ component[2].

This whole algebraic manipulation can be conveniently organized using the bucket data structure as follows. We partition the probabilistic and reward components into ordered buckets in the usual manner (Dechter 1996), namely each functional component is placed in the latest bucket that mentions any of the variables in its scope. Using the ordering $o = T, t_2, t_2, D, C_2, C_1$, the initial partitioning for the car example is given in Figure 3.

$bucket(C_1)$: $P(C_1), P(t_1|C_1,T), r(C_1,D)$
$bucket(C_2)$: $P(C_2), P(t_2|C_2,T), r(C_2,D)$
$bucket(D)$:
$bucket(t_1)$:
$bucket(t_2)$:
$bucket(T)$: $r(T)$

Figure 3: Initial partitioning into buckets

$bucket(C_1)$: $P(C_1), P(t_1|C_1,T), r(C_1,D)$
$bucket(C_2)$: $P(C_2), P(t_2|C_2,T), r(C_2,D)$
$bucket(D)$: $||\ \theta_{C_1}(t_1,T,D), \theta_{C_2}(t_2,T,D)$
$bucket(t_1)$: $||\ \lambda_{C_1}(t_1,T), \quad \theta_D(t_1,t_2,T)$
$bucket(t_2)$: $||\ \lambda_{C_2}(t_2,T), \quad \theta_{t_1}(t_2,T)$
$bucket(T)$: $r(T)||\lambda_{t_1}(T), \quad \lambda_{t_2}(T), \quad \theta_{t_2}(T)$.

Figure 4: Schematic bucket evaluation of car example

We distinguish two types of functions in a bucket: probability components (denoted by λs) and utility components (denoted by θ). When processing a chance bucket a new probability component and a new utility component are created, each placed in a closest lower bucket of a variable in its scope. Figure 4 shows the recorded functions in the buckets after processing in reverse order of o. The new recorded functions are shown to the right of the bar. The subscript of each function indicates the bucket originating the function.

The optimizing policies for δ_T is the function computed in $bucket(T)$ (argmax of EQ. 19), a simple decision and $\theta_D(t_1,t_2,T)$ that is recorded in $bucket(t_1)$.

General derivation

The general derivation shows that processing decisions is, in general, more complex than what was demonstrated by the example. In fact, a decision variable may cause the decomposed rewards to be completely coupled. Nevertheless the simplified case appears quite frequently and can be identified easily.

Given an influence diagram, $ID = (X, D, P, R)$ and evidence e, where $R = \{r_1, ..., r_j\}$ defined over $Q = \{Q_1, ..., Q_j\}$, $Q_i \subseteq X \cup D$, and $u = \sum_j r_j$. The meu task is to compute

$$E = \max_{\Delta=(\delta_1,\ldots,\delta_m)} \sum_{x_1,\ldots,x_n} \prod_{X_i} P(x_i, e|x^{\Delta}{}_{pa_i})u(x^{\Delta}), \quad (20)$$

Where $\delta_1, ...\delta_m$ are decision rules, and x is an assignment over both chance variables and decision variables, $x = (x_1, ..., x_n, d_1, ..., d_m)$.

The operations of summation and maximization of EQ. (20) should be computed along *legal* orderings. We use the ordering criteria provided in (F. Jensen &

[2]We will see that this correspond to a simplification rather than to the general case

Dittmer 1994): assuming a given ordering for the decision variables, unobservable variables are last in the ordering, and chance variables between D_i and D_{i+1} are those that become observable following decision D_i. For any legal ordering, expression (20) can be written as:

$$E = \otimes_{\vec{x}_n} \prod_{X_i \in X} P(x_i, e|x_{pa_i}) \sum_j r_j(x_{Q_j}),$$

where $\vec{x}_i = (x_1, ..., x_i)$ and where the operation $\otimes$ is a summation, if it is applied to a chance variable, or a maximization if it is applied to a decision variable. For instance, $\otimes_{(x_1,d_1,d_2,x_2)} = \sum_{x_1} \max_{d_1} \max_{d_2} \sum_{x_2}$.

We will now isolate and assess the impact of variable X_n and derive the function that it induces on the rest of the problem. Variable X_n stands either for a chance variable or for a decision variable. Namely:

$$E = \otimes_{\vec{x}_{n-1}} \otimes_{x_n} \prod_{i=1}^{n} P(x_i, e|x_{pa_i}) \sum_{Q_j \in Q} r_j(x_{Q_j})$$

We can separate the components in the reward functions into those having X_n in their scope, denoted by the index set t_n, and those not mentioning X_n, labeled with indexes $l_n = \{1, ..., n\} - t_n$. Accordingly we produce

$$E = \otimes_{\vec{x}_{n-1}} \otimes_{x_n} \prod_{i=1}^{n} P(x_i, e|x_{pa_i}) \cdot (\sum_{j \in l_n} r_j(x_{Q_j}) + \sum_{j \in t_n} r_j(x_{Q_j})) = \tag{21}$$

$$= \otimes_{\vec{x}_{n-1}} \prod_{X_i \in X - F_n} P(x_i, e|x_{pa_i}) \otimes_{x_n} \prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) \cdot \tag{22}$$

$$[\sum_{j \in l_n} r_j(x_{Q_j}) + \sum_{j \in t_n} r_j(x_{Q_j})]$$

By migrating to the left of X_n all the functions that do not have X_n in their scope, we get the following expressions derived separately for the case when X_n is a chance variable or a decision variable.

Case of chance variable:
In this case $\otimes_{x_n} = \sum_{x_n}$.

$$E = \otimes_{\vec{x}_{n-1}} \prod_{X_i \in X - F_n} P(x_i, e|x_{pa_i}) \sum_{x_n} \prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) \tag{23}$$

$$[\sum_{j \in l_n} r_j(x_{Q_j}) + \sum_{j \in t_n} r_j(x_{Q_j})] =$$

$$\otimes_{\vec{x}_{n-1}} \prod_{X_i \in X - F_n} P(x_i, e|x_{pa_i}) \cdot [\sum_{j \in l_n} r_j(x_{Q_j}) \cdot$$

$$\sum_{x_n} \prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) + \sum_{x_n} \prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) (\sum_{j \in t_n} r_j(x_{Q_j})]$$

We denote by U_n the subset of decision and chance variables that appear with X_n in a probabilistic component, excluding X_n itself, and by W_n the union of variables (decisions or chance nodes) that appear in probabilistic and utility components with X_n, excluding X_n itself. We define λ_n over U_n (x is a tuple over $U_n \cup X_n$) as

$$\lambda_n(x_{U_n}) = \sum_{x_n} \prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) \tag{24}$$

We define θ_n over W_n as

$$\theta_n(x_{W_n}) = \frac{1}{\lambda_n(x_{U_n})} \sum_{x_n} \prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) \sum_{j \in t_n} r_j(x_{Q_j}). \tag{25}$$

After substituting EQs. (24) and (25) into EQ. (23), we get

$$E = \otimes_{\vec{x}_{n-1}} \prod_{X_i \in X - F_n} P(x_i, e|x_{pa_i}) \cdot \lambda_n(x_{U_n}) \cdot \tag{26}$$

$$[\sum_{j \in l_n} r_j(x_{Q_j}) + \theta_n(x_{W_n})].$$

The functions θ_n and λ_n compute the effect of eliminating the chance variable X_n. The result (EQ. (26)) is an expression which does not include X_n, where the product has a new (probabilistic) function, λ_n, and the utility components have a new element θ_n.

Case of decision variable X_n:
We start from expression (22), only now $\otimes_{x_n} = \max_{x_n}$.

$$E = \otimes_{\vec{x}_{n-1}} \prod_{X_i \in X - F_n} P(x_i, e|x_{pa_i}) \cdot \max_{x_n} \prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) \cdot$$

$$[\sum_{j \in l_n} r_j(x_{Q_j}) + \sum_{j \in t_n} r_j(x_{Q_j})] \tag{27}$$

At this point, unless $\prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) = 1$, we cannot migrate the maximization operation anymore to the right because summation and maximization do not commute. We define a function λ over $V_n = U_n \cup \{Q_1 \cup Q_2, ..., \cup Q_m\}$:

$$\lambda(x_{V_n}) = \max_{x_n} \prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) \cdot \sum_j r_j(x_{Q_j}) \tag{28}$$

Namely, we use *all* the reward function when computing λ. However, if $\prod_{X_i \in F_n} P(x_i, e|x_{pa_i}) = 1$, then maximization can be applied in stages only to the reward components containing X_n as we saw in the example. The simplified case often happens, either when the decision variable has no associated probabilistic components or when the probabilistic product is 1.

Applying similar algebraic manipulation to the rest of the chance and decision variables in order, yields the elimination algorithm *elim-meu-id* in Figure 5. Each bucket contains utility components θ_i and probability components, λ_i. The algorithm generates the λ_i of a bucket by multiplying all its probability components and summing (if the variable is a chance node) over X_i. The θ_i of a chance variable X_i is computed as the average utility of the bucket, normalized by the bucket's compiled λ. The computation is simplified when $\lambda = 1$, namely, when there is no evidence in the bucket, nor new probabilistic functions.

For a decision variable we compute a λ component by maximization as dictated by EQ. (28), and simplify when no probabilistic components appear in the decision bucket. We note therefore that processing a decision variable, does *not*, in general, allow exploiting a decomposition in the reward components.

It is known that observed variables frequently simplify computation by avoiding creating new dependencies among variables (although for probabilistic variables they would make a larger portion of the network relevant (Dechter 1999)). We see (EQ. (24)) that if $X_n = v$ is an evidence, λ_n^i can be computed by assigning the value $X_n = v$ to each probabilistic component, and each can be moved separately into a lower bucket. The effect of an observation on a utility component in a chance bucket is (substituting $X_n = v$ in the expression for θ_n):

$$\theta_n = \frac{1}{\lambda_n} \prod_{X_i \in F_n} P(x_i, e|x_n = v, x_{pa_i}) \sum_{j \in t_n} r_j(x_{Q_j}, x_n = v).$$

Since the multiplicands in the denominator and the dominator cancel out, we get

$$\theta_n = \sum_{j \in t_n} r_j(x_{Q_j}, x_n = v).$$

Again, each utility component can be processed independently.

Luckily, an instantiated *decision* bucket can also be simplified enough and can be processed in an identical manner to utilities computed in chance variables. Consequently, the utility components can also be assigned the observed decision value and each can be moved independently to a lower bucket; either to a chance bucket or to the closest decision bucket.

Example 2 *Consider the influence diagram of Figure 1 where* $u = r(t) + r(d, op) + r(op, mi, osp)$ *the ordering of processing is* $o = T, R, D, OP, MI, OSP, S, O$. *The chance variables* O *and* S *are unobservable and therefore placed last in the ordering. The bucket's partitioning and the schematic computation of this decision problem is given in Figure 7 and is explained next.*

Initially, bucket(O) contains $P(op|d, o), P(s|o), P(o)$. *Since this is a chance bucket having no reward components we generate*

$$\lambda_O(s, op, d) = \sum_o P(op|d, o)P(s|o)P(o)$$

placed in bucket(S). The bucket of S is processed next as a chance bucket. We compute

$$\lambda_S(r, op, d, t) = \sum_s P(r|t, s)\lambda_O(s, op, d)$$

placed in bucket(OP). The bucket of the decision variable OSP is processed next. It contains all the reward components $r(op, mi, osp)$, $r(op, osp)$, $r(t)$ *and* $r(d, op)$. *Since the decision bucket contains no probabilistic component,* $r(t)$ *is moved to the bucket of D and* $r(d, op)$

Algorithm elim-meu-id
Input: Influence diagram (X, D, P, R), a legal ordering of the variables, o; observations e.
Output: A set of decision rules $\delta_1, ..., \delta_k$ that maximizes the expected utility.
1. **Initialize:** Partition components into buckets where $bucket_i$ contains all probabilistic components whose highest variable is X_i. Reward components are partitioned using procedure *partition-rewards (P,R,o)*.
In each bucket, $\lambda_1, ..., \lambda_j$, denote probabilistic components and $\theta_1, ..., \theta_l$ utility components. Let $S_1, ..., S_j$ be the scopes of probability components, and $Q_1, ..., Q_l$ be the scopes of the utility components.
2. **Backward:** For $p \leftarrow n$ downto 1, do
Process $bucket_p$:
for all functions $\lambda_1, ..., \lambda_j, \theta_1, ..., \theta_l$ in $bucket_p$, do

- **If** (observed variable) $bucket_p$ contains $X_p = x_p$, assign $X_p = x_p$ to each λ_i, θ_i, and put each resulting function in appropriate bucket.
- **else**,
  - **If** X_p **is a chance variable** compute $\lambda_p = \sum_{X_p} \prod_i \lambda_i$ and $\theta_p = \frac{1}{\lambda_p} \sum_{X_p} \Pi_{i=1}^j \lambda_i \sum_{j=1}^l \theta_j$;
  - **If** X_p **is a decision variable**, then **If** bucket has only θ's , move each free from X_p θ, to its appropriate lower bucket, and for the rest compute $\theta_p = \max_{X_p} \sum_j \theta_j$ **else**, (the general case), compute $\lambda_p = \max_{X_p} \prod_{i=1}^j \lambda_i \sum_{j=1}^l \theta_j$
  - Add λ_p to the bucket of the largest-index variable in their scopes. Place θ_p in the closest chance bucket of a variable in its scope or in the closest decision bucket.

3. **Return :** decision rules computed in decision buckets.

Figure 5: Algorithm *elim-meu*

Procedure partition-rewards(P, R, o)
For i = n to 1
If X_i is a chance variable, put all rewards mentioning X_i in $bucket_i$.
else (decision variable) put all remaining rewards in current $bucket_i$
end.
Return ordered buckets

Figure 6: partition into buckets

$bucket(O)$: $P(o|op, D)P(s|o), P(o)$
$bucke(S)$: $P(r|s,t)$ || $\lambda_O(s, op, d)$
$bucket(OSP)$: $r(t), r(op, osp), r(op, mi, osp), r(d, op)$
$bucket(MI)$: $P(mi)$ || $\theta_{OSP}(op, mi)$
$bucket(OP)$: || $\lambda_S(r, t, op, d), r(op, d), \theta_{MI}(op)$
$bucket(D)$: || $\lambda_{OP}(r, t, d), \theta_{OP}(d, r, t) r(t)$
$bucket(R)$: || $\lambda_D(r, t)$
$bucket(T)$: || $\lambda_R(t)$

Figure 7: A schematic execution of elim-meu

is moved to the bucket of OP. We then create a utility component

$$\theta_{OSP}(op, mi) = \max_{osp}[r(op, mi, osp) + r(op, osp)]$$

Which is the decision rule for OSP, and place it in $bucket(MI)$. The bucket of MI is processed next as a chance bucket, generating a constant $\lambda = 1$ and

$$\theta_{MI}(op) = \sum_{mi} P(mi)\theta_{OSP}(op, mi),$$

placed in the bucket of OP. Processing the bucket of OP yields:

$$\lambda_{OP}(r, t, d) = \sum_{op} \lambda_S(r, t, op, d)$$

and

$$\theta_{OP}(r, t, d) = \frac{1}{\lambda_{OP}} \sum_{op} \lambda_S(r, t, op, d)[r(op, d) + \theta_{MI}(op)]$$

both placed in the bucket of decision variable D. Here we observe the case of a decision bucket that contains both probabilistic and utility component. The bucket has two reward components (one original $r(t)$ and one generated recently.) It computes a λ component:

$$\lambda_D(r, t) = \max_D \lambda_{OP}(r, t, d)[r(t) + \theta_{OP}(r, t, d)]$$

which provides the decision rule for D and is placed in the bucket of R. The bucket of R has only a probability component yielding:

$$\lambda_R(t) = \sum_r \lambda_D(r, t)$$

placed in $bucket(T)$, and

$$\lambda_T = \max_t \lambda_R(t)$$

is derived and provides the decision rule for T. Also, $\max_t \lambda_R(t)$ *is the optimal expected utility.*

The sequence of solution policies is given by the argmax functions that can be created in decision buckets. Once decision T is made, the value of R will be observed, and then decision D can be made based on T and the observed R.

In summary,

Theorem 1 *Algorithm elim-meu-id computes the meu of an influence diagram as well as a sequence of optimizing decision rules.* □

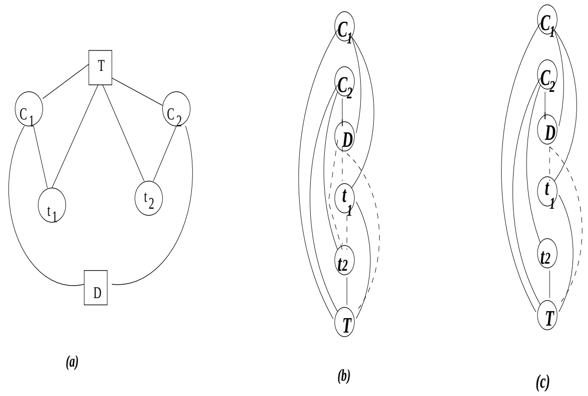

Figure 8: The augmented graph of the car example (a) and its induced order graph (b)

Complexity

As is usually the case with bucket elimination algorithms, their performance can be bounded as a function of the induced width of some graph that reflects the algorithm's execution.

An *ordered graph* is a pair (G, d) where G is an undirected graph and $d = X_1, ..., X_n$ is an ordering of the nodes. The *width of a node* in an ordered graph is the number of the node's neighbors that precede it in the ordering. The *width of an ordering* d, denoted $w(d)$, is the maximum width over all nodes. The *induced width of an ordered graph*, $w^*(d)$, is the width of the induced ordered graph obtained as follows: nodes are processed from last to first; when node X is processed, all its preceding neighbors are connected. The *induced width of a graph*, $w*$, is the minimal induced width over all its orderings.

The relevant graph for elim-meu-id, is the *augmented graph* obtained from the ID graph as follows. All the parents of chance variables are connected (moralizing the graph), all the parents of reward components are connected, and all the arrows are dropped. Value nodes and their incident arcs are deleted. Figure 8(a) gives the augmented graph of the car example. An ordered graph is depicted in Figure 8(b) with the solid lines. The broken lines are those added when creating the induced ordered graph. We see that the induced-width of this ordered graph is 3.

To capture the simplification associated with evidence we use the the notion of *adjusted induced graph.* The adjusted induced graph is created by processing the variables from last to first in the given ordering. Only parents of each none-evidence variable are connected. The adjusted induced-width is the width of the adjusted induced-graph. Figure 8c shows the adjusted induced-graph relative to the evidence in C_2 (assuming we are told about the quality of car C_2). Although the induced-width remains the same, we see that processing D will be easier since its induced-width is reduced by the evidence.

To capture the general case, when decision variables couple reward components, the induced graph should be computed in a slightly modified manner. We distinguish between chance arcs and reward arcs. A new arc, created using a chance arc, is a chance arc. Given the ordered augmented graph, we process the nodes from last to first. If the next node is a chance node, connect all its earlier neighbors (unless it is evidence). If the next node is a decision node, then, if it is connected to an earlier node by a chance arc, connect, not only all its earlier neighbors, but also all the earlier variables appearing in the scope of any reward component. The resulting induced graph is called the *compounded induced graph*.

Theorem 2 *Given an influence diagram defined on n variables, and given evidence e, algorithm elim-meu-id is time and space $O(n \cdot exp(w^*(o,e))$, where $w^*(o,e)$ is the width along o of the adjusted compounded induced graph.*

In the car example the compounded induced graph is the same as the (regular) induced graph since the decision nodes are not incident to chance arcs.

Improving elim-meu-id

In this section we show that dependencies created by multiple reward components can be avoided, whenever the reward functions are defined on chance variables only. This implies that computing the expected utility is not harder than the task of belief-updating over the underlying probabilistic subnetworks.

To illustrate the idea consider the following 6-variable belief network,

$$D \rightarrow H \rightarrow A \rightarrow B \rightarrow C \rightarrow E$$

where D is a decision variable while the rest are chance variables. Assume also that the reward functions are $r(A,B)$, $r(A,C)$ $r(A,E)$ $r(B,C)$ $r(B,E)$, $r(C,E)$ $r(H,A)$, $r(H,B)$, $r(H,C)$, $r(H,E)$. The task is to compute

$$\max_D \sum_{h,a,b,c,e} P(h|d)P(a|h)P(b|a)P(c|b)P(e|c)$$

$$[\sum_{x,y \in \{a,b,c,h,e\}} r(x,y)]$$

Let's execute elim-meu-id along the order D,H,A,B,C,E. The initial buckets are:
$bucket_E$: $P(e|c)$, $r(e,c)$, $r(e,b)$, $r(e,a)$, $r(e,h)$
$bucket_C$: $P(c|b)$, $r(c,b)$, $r(c,a)$,$r(c,h)$
$bucket_B$: $P(b|a)$, $r(b,a)$, $r(b,h)$
$bucket_A$: $P(a|h)$, $r(a,h)$
$bucket_H$: $P(h|d)$
$bucket_D$:

Processing bucket E by elim-meu-id generate the function (assuming no evidence)

$$\theta_E(a,b,c,h) = \sum_e P(e|c) \cdot [(r(e,c)+r(e,b)+r(e,a)+r(e,h))]$$

which will be placed in bucket C. However, by exchanging summation order we get:

$$\theta_E(a,b,c,h) = \sum_e P(e|c) \cdot r(e,c) + \sum_e P(e|c) \cdot r(e,b) +$$

$$\sum_e P(e|c) \cdot r(e,a) + \sum_e P(e|c) r(e,h).$$

Therefore, we can sum over variable E relative to each reward component, separately. Then, instead of recording $\theta_E(a,b,c,h)$ we record several smaller functions:
$\theta_E(c) = \sum_e P(e|c) \cdot r(e,c)$,
$\theta_E(b,c) = \sum_e P(e|c) \cdot r(e,b)$,
$\theta_E(c,a) = \sum_e P(c|e) r(e,a)$,
and $\theta_E(c,h) = \sum_e P(c|e) r(e,h)$,
each to be placed in bucket of C.

When processing $bucket_C$, instead of computing

$$\theta_C(b,a,h) = \sum_c P(c|b)[r(c,b)+r(c,a)+r(c,h)+\theta_E(c)+$$

$$\theta_E(c,b) + \theta_E(c,a) + \theta_E(c,h)]$$

we can, again, generate several summands:

$$\theta_C(b) = \sum_c P(c|b)(r(c,b) + \theta_E(c,b) + \theta_E(c))$$

$$\theta_C(a,b) = \sum_c P(c|b)(r(c,a) + \theta_E(c,a))$$

and

$$\theta_C(h,b) = \sum_c P(c|b)(r(c,h) + \theta_E(c,h))$$

and so on. The final bucket's structure is:

$bucket_E$: $P(e|c)$, $r(e,c)$, $r(e,b)$, $r(e,a)$, $r(e,h)$
$bucket_C$: $P(c|b)$,$r(c,b)$,$r(c,a)$,$r(c,h)$ || $\theta_E(c)$,$\theta_E(c,h)$ $\theta_E(c,b)$ $\theta_E(c,a)$
$bucket_B$: $P(b|a)$, $r(b,a)$, $r(b,h)$ || , $\theta_C(a,b)$, $\theta_C(b,h)$, $\theta_C(b)$
$bucket_A$: $P(a|h)$, $r(a,h)$ || $\theta_C(a,h)$
$bucket_H$: || $\theta_A(d,h)$
$bucket_D$:

Instead of:
$bucket_E$: $P(e|c)$, $r(e,c)$, $r(e,b)$, $r(e,a)$, $r(e,h)$
$bucket_C$: $P(c|b)$, $r(c,b)$, $r(c,a)$, $r(c,h)$ || $\theta_E(c,b,a,h)$
$bucket_B$: $P(b|a)$, $r(b,a)$, $r(b,h)$ || , $\theta_C(a,b,h)$
$bucket_A$: $P(a|h)$, $r(a,h)$ || $\theta_C(a,h)$
$bucket_H$: $P(h|d)$ || $\theta_A(h)$
$bucket_D$: || $\theta_H(d)$

We say that a function f, defined by an algebraic expression, is based-on a function r, if r appears in the algebraic expression that defines f. For example $\theta_E(c,h)$ is based on $r(e,h)$.

Proposition 3 *Given an influence diagram whose all decision nodes are root nodes, every utility function created by* improved elim-meu-id *in a chance bucket is based on a single original reward component.*

Algorithm improved elim-meu-id
Input: An Influence diagram (X, D, P, R).
Output: A set of policies $d_1, ..., d_k$ that maximizes the expected utility.
1. **Initialize:** Partition components into buckets, where $bucket_i$ contains all matrices whose highest variable is X_i. Call probability matrices $\lambda_1, ..., \lambda_j$ and utility matrices $\theta_1, ..., \theta_l$. Let $S_1, ..., S_j$ be the scopes of the probability components and $Q_1, ..., Q_l$ be the scopes of the reward components.
2. **Backward:** For $p \leftarrow n$ downto 1, do
for all matrices $\lambda_1, ..., \lambda_j, \theta_1, ..., \theta_l$ in $bucket_p$, do

- **If** (observed variable, or a predetermined decision policy) $bucket_p$ contains the observation $X_p = x_p$, then assign $X_p = x_p$ to each λ_i, θ_i, and put each resulting matrix in appropriate lower bucket.
- **else**,
  - **if** X_p **is a chance variable** then
    $\lambda_p = \sum_{X_p} \prod_i \lambda_i$ and
    for each $\theta_i \in bucket_p$ compute
    $\theta_p^i = \frac{1}{\lambda_p} \sum_{X_p} \theta_i \prod_{i=1}^{j} \lambda_i$
  - **else, if** X_i **is a decision variable**
    If bucket has only θ's , move each free from X_p θ, to its appropriate lower bucket, and for the rest compute
    $\theta_p = \max_{X_p} \sum_j \theta_j$
    else, (the general case), compute
    $\lambda_p = \max_{X_p} \prod_{i=1}^{j} \lambda_i \sum_{j=1}^{l} \theta_j$

  Add θ_p and λ_p to the bucket of the largest-index variable in their argument list.

3. **Forward:** Return the max value obtained in the first bucket and the set of decision rules.

Figure 9: Algorithm *elim-meu-id*

Consequently, the scopes of functions generated in chance buckets equals the the scope that will be created by the probabilistic subnetwork, extended by the scope of a single reward function at the most.

Theorem 4 *Given a Bayesian network and a set of reward functions whose scope is bounded by a constant c, the complexity of finding the expected utility is $O(exp(w^* + c))$ where w^* is the induced-width along o of the probabilistic subnetwork only.*

We augment this improvement in elim-meu-id, resulting in the improved algorithm of Figure 9. The improved version of elim-meu-id can sometime save exponential time. For instance, on singly-connected belief networks having binary reward functions on every pair of variables, the complexity of elim-meu-id is $O(exp(n))$ time and space, while the complexity of the improved

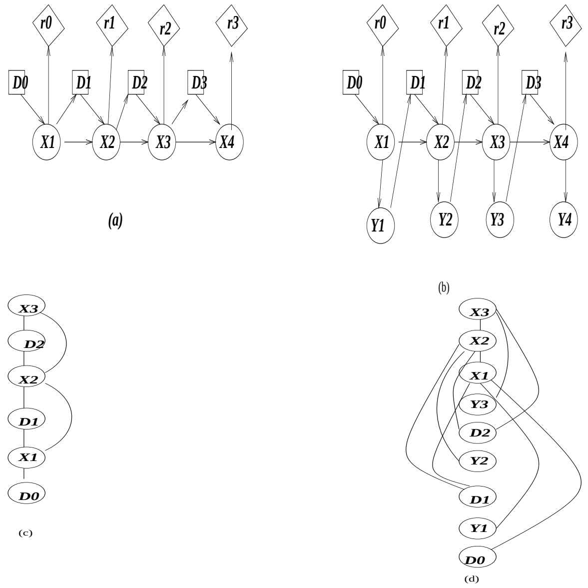

Figure 10: An influence diagram of an MDP (a) and a POMDP (b)

algorithm is just $n \cdot exp(3)$, namely $O(n)$. This is because in each bucket we process functions over 3 variables at the most.

Relation to MDPs and POMDPs

We next view the complexity associated with finite horizon MDPs and POMDPS, through the notion of augmented graph. In particular, MDPs correspond to simple influence diagrams where the decision variable allow decomposition of reward components, because probabilistic components in decision buckets are evaluated to constant 1. Figure 10(a) presents a network that corresponds to an MDP having 4 state variables, X_1, X_2, X_3, X_4, decision variables associated with each stage, D_0, D_1, D_2, D_3, reward components $r(X_i)$ for each state X_i. In Figure 10(b) the X_i variables are unobservable, and the decisions are made based on the observable Y_i implied (probabilistically) by X_i.

We see that using the ordering $o = D_0, X_1, D_1, X_2, D_2, X_3, D_3, X_4$ the augmented induced graph is simple. In particular decision variable can be processed by the simplified version since the corresponding probabilistic components will evaluate to 1. For the POMDP, all unobservable variables must appear at the end of the ordering and be processed first, namely $o = D_0, Y_1, D_1, Y_2, D_2, Y_3, D_3, Y_4, X_1, X_2, X_3, X_4$ yielding an ordered augmented graph that even its probabilistic subgraph is highly coupled (Figure 10(c,d).) The compound induced graph is obviously highly connected (not shown in the Figure).

Conclusion

The paper presented a bucket-elimination algorithm for maximizing the expected utility over a set of policies and analyzed its complexity using graph parameters such as induced width. A special focus is given to ways of exploiting decomposable reward functions, showing that while decision variables enforce dependencies on reward components, chance variables can fully exploit reward decompositions. Our analysis suggests that to avoid the complexity of inference associated with decision variables, combining conditioning search with bucket-elimination, by conditioning on decision variables only, could be very cost-effective, because instantiated decision variables can be processed efficiently. Finally, approximation algorithms like mini-bucket elimination (Dechter & Rish 1997) or time-space trading variants (El-Fattah & Dechter 1996), are applicable.

References

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverag. *Artificial Intelligence Researc (JAIR)* 11:1–94.

Dechter, R., and Rish, I. 1997. A scheme for approximating probabilistic inference. In *Proceedings of Uncertainty in Artificial Intelligence (UAI'97)*, 132–141.

Dechter, R. 1996. Bucket elimination: A unifying framework for probabilistic inference algorithms. In *Uncertainty in Artificial Intelligence (UAI'96)*, 211–219.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 41–85.

El-Fattah, Y., and Dechter, R. 1996. An evaluation of structural parameters for probabilistic reasoning: results on benchmark circuits. In *Uncertainty in Artificial Intelligence (UAI'96)*, 244–251.

F. Jensen, F. J., and Dittmer, S. 1994. From influence diagrams to junction trees. In *Tenth Conference on Uncertainty in Artificial Intelligence*, 367–363.

Howard, R. A., and Matheson, J. E. 1984. *Influence diagrams.*

Howard, R. A. 1976. The used car buyer. In *Readings in Decision Analysis*, 491–520.

N. L. Zhang, R. Q., and Poole, D. 1994. A computational theory of decision networks. *International Journal of Approximate Reasoning* 83–158.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann.

Shachter, R., and Peot, M. 1992. Decision making using probabilistic inference methods. In *Proceedings of Uncertainty in Artificial Intelligence (UAI92)*, 276–283.

Shachter, R. 1986. Evaluating influence diagrams. *Operations Research* 34.

Shachter, R. 1988. Probabilistic inference and influence diagrams. *Operations Research* 36.

Shachter, R. D. 1990. An ordered examination of influence diagrams. *Networks* 20:535–563.

Shenoy, P. 1992. Valuation-based systems for bayesian decision analysis. *Operations Research* 40:463–484.

Tatman, J., and Shachter, R. 1990. Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics* 365–379.

Zhang, N. L. 1998. Probabilistic inference in influence diagrams. *Computational Intelligence* 475–497.