A New Method for Influence Diagram Evaluation

by

Runping Qi and David Poole*

Technical Report 93-10
May 1993

Department of Computer Science
The University of British Columbia
Vancouver, B. C. V6T 1Z2
Canada

email: qi@cs.ubc.ca, poole@cs.ubc.ca

*Scholar of Canadian Institute for Advanced Research

Abstract

As Influence diagrams become a popular representational tool for decision analysis, influence diagram evaluation attracts more and more research interests. In this article, we present a new, two-phase method for influence diagram evaluation. In our method, an influence diagram is first mapped into a decision graph and then the analysis is carried out by evaluating the decision graph. Our method is more efficient than Howard and Matheson's two-phase method because, among other reasons, the size of the decision graph generated by our method from an influence diagram can be much smaller than that by Howard and Matheson's method for the same influence diagram. Like those most recent algorithms reported in the literature, our method can also exploit independence relationship among variables of decision problems, and provides a clean interface between influence diagram evaluation and Bayesian net evaluation, thus, various well-established algorithms for Bayesian net evaluation can be used in influence diagram evaluation. In this sense, our method is as efficient as those algorithms. Furthermore, our method has a few unique merits. First, it can take advantage of asymmetric processing in influence diagram evaluation. Second, by using heuristic search techniques, it provides an explicit mechanism for making use of heuristic information that may be available in a domain-specific form. These additional merits make our method more efficient than the current algorithms in general. Finally, by using decision graphs as an intermediate representation, the value of perfect information can be computed in a more efficient way.

1 Introduction

An influence diagram (Howard and Matheson 1984) is a graphical representation of decision problems. In comparison with decision trees, influence diagrams are arguably more intuitive, natural and compact.

The first method for influence diagram evaluation is Howard and Matheson's two–phase method. In this method, an influence diagram is first transformed into a decision tree and then the analysis is carried out by evaluating the decision tree. This two–phase method was largely abandoned after Shachter showed an influence diagrams can be evaluated directly (Shachter 1986). All of the recent algorithms for influence diagram evaluation (Cooper 1988, Zhang and Poole 1992b, Shachter and Peot 1992, Zhang *et al.* 1993b) carry out the evaluation without a secondary representation. One reason for abandoning the two–phase approach might be that people believe that direct evaluation is more efficient.

However, all those algorithms that evaluate influence diagrams directly suffer from a common shortcoming. That is they do not perform asymmetric processing (Shachter 1986). Another weakness of these algorithms is that they fail to provide any explicit mechanism to make use of domain dependent information (e.g. a heuristic function estimating the optimal expected values of influence diagrams), even when it is available.

In this paper, we will show that the two–phase approach to influence diagram evaluation need not be inefficient. The inefficiency of Howard and Matheson's method is not due to the two–phase approach. Rather, it is due to the *way* to generate the secondary representation. We will present a two–phase method for influence diagram evaluation. In our method, we map an influence diagram into a decision graph (Qi 1993, Qi and Poole 1993) in such a way that an optimal solution graph of the decision graph corresponds to an optimal strategy of the influence diagram. Thus, the problem of computing an optimal strategy is reduced to the problem of searching for an optimal solution graph of a decision graph, which can be accomplished by various algorithms (Qi 1993, Qi and Poole 1993). The size of the decision graph generated by our method from an influence diagram can be much smaller than that generated by Howard and Matheson's method for the same influence diagram. Like those most recent algorithms (Zhang and Poole 1992b, Zhang *et al.* 1993a, Shachter and Peot 1992, Zhang 1993), our method can also exploit independence relationship among variables of decision problems, and provides a clean interface between influence diagram evaluation and Bayesian net evaluation, thus, various well–established algorithms for Bayesian net evaluation can be used in influence diagram evaluation. In this sense, our method is essentially as efficient as the one proposed by Zhang and Poole (Zhang and Poole 1992b), which rivals any other algorithms in terms of efficiency (Zhang 1993). Furthermore, our method has a few additional merits. First, it can take advantage of asymmetric processing in influence diagram evaluation. Second, by using heuristic search techniques, it provides an explicit mechanism for making use of heuristic information that may be available in a domain–specific form. Finally, by using decision graphs as an intermediate representation, the value of perfect information (Matheson 1990) can be computed in a more efficient way (Zhang *et al.* 1993b).

The remainder of this paper is organized as follows. We introduce the basic concepts of influence diagrams and influence diagram evaluation in the next section, and review the current algorithms for influence diagram evaluation in Section 3. In Section 4, we point out why those recent algorithms fail to perform asymmetric processing. In Section 5, we present our two–phase

method for the evaluation of influence diagram with a single value node. We first establish a stochastic dynamic programming formulation for influence diagram evaluation and show that the problem of influence diagram evaluation can be reduced to a decision graph search problem. Then we point out how to exploit asymmetry in a decision problem for improving evaluation efficiency. In Section 6, we extend our method to the influence diagrams with multiple value nodes. Conclusions are given in Section 7.

2 Influence Diagrams

An influence diagram is a direct acyclic graph with three types of nodes: *random nodes*, *decision nodes* and *value nodes*. Each random node represents a random variable whose value is determined according to some probability distribution, and each decision node represents a decision variable whose value is to be chosen by the decision maker. In this paper, we will use the term decision (random) variables and decision (random) nodes interchangeably. The arcs into a random node, called *conditional arcs*, indicate the probabilistic dependency of the random node. The arcs into a decision node, called *informational arcs*, indicate the information available to the decision maker at the time he must choose a value for the decision variable. The lack of an arc from a node a to a decision node d means that the value of the variable a is not known to the decision maker when decision d is to be made.

An influence diagram is *regular* (Howard and Matheson 1984, Shachter 1986) if there is a direct path containing all of the decision variables. Since the diagram is acyclic, such a path defines an order for the decision nodes. This is the order in which the decisions are made. We refer to it as the *regularity order* of the influence diagram. An influence diagram is "no–forgetting" if each decision node d and its parents are also parents of those decision nodes which are descendents of d (Howard and Matheson 1984, Shachter 1986). Intuitively, the "no–forgetting" property means that a decision maker will remember all the information that was earlier available to him and remember all the previous decisions he made. A regular and "no–forgetting" influence diagram represents the decision maker's view of the world.

We first consider the regular influence diagrams with exactly one value node. We come to the more general influence diagrams in Section 6.

In a given influence diagram, let v be the value node, let D and C denote the set of decision nodes and the set of random nodes respectively. Suppose $a \rightarrow b$ is an arc in an influence diagram. The node a is called a *parent* of node b, and node b is a *child* of node a. We will also use other standard graph terminologies such as descendents and ancestors in this paper. Without loss of generality, we assume that the value node in any influence diagram has no children.

For any node x, let $\pi(x)$ denote the set of all parents of x in the diagram. Associated with each node x is set Ω_x, which is the set of values the variable can take. This set is called the *frame* of the variable. For any subset $J \subseteq C \cup D$, let $\Omega_J = \prod_{x \in J} \Omega_x$. Each random node x is characterized by a conditional probability distribution, written $P\{x|\pi(x)\}$. For each $o \in \Omega_x$ and $c \in \Omega_{\pi(x)}$, the distribution specifies the conditional probability of event $x = o$, given $\pi(x) = c$ [1]. The value node is characterized by a function f, called the *value function* of the value node. The

[1] In this paper, for any variable set J and any element $e \in \Omega_J$, we use $J = e$ to denote the set of assignments which assign an element of e to the corresponding variable in J.

value function is a mapping from $\Omega_{\pi(v)}$ to Ω_v which is usually assumed to be the real domain $\mathcal{R}$.

Let $D = \{d_1, ..., d_n\}$ be the set of decision nodes of influence diagram I. A *strategy* Δ for the influence diagram is a collection of *decision functions*, $\Delta = (\delta_1, \delta_2, ..., \delta_n)$, where function δ_k, $1 \leq k \leq n$, corresponds to decision node d_k and is a mapping from $\Omega_{\pi(d_k)}$ to Ω_{d_k}. Note that the set $\Omega_{\pi(d_k)}$ represents all the possible states of the information available to the decision maker when he needs to choose a value for the decision variable d_k, and the set Ω_{d_k} represents all the possible choices for d_k. Thus, a decision function prescribes a decision choice based on the information available to the decision maker when the decision is to be made.

Given a strategy $\Delta = (\delta_1, \delta_2, ..., \delta_n)$, each decision node d_i can be regarded as a deterministic random node whose probability distribution is given as follows:

$$P\{d_i = x | \pi(d_i) = c\} = \begin{cases} 1 & \text{if } \delta_i(c) = x, \\ 0 & \text{otherwise.} \end{cases}$$

With the decision nodes treated this way, the influence diagram I, together with a particular strategy Δ, becomes a Bayesian net. We use I_Δ to denote the net and use $P_\Delta\{\cdot\}$ to denote the joint distribution determined by I_Δ.

Let $P_\Delta\{\pi(v) = o\}$, for any $o \in \Omega_{\pi(v)}$, denote the probability for the event $\pi(v) = o$, in I_Δ. The expected value of the value node *w.r.t.* strategy Δ, written $E_\Delta[v]$, is given by

$$E_\Delta[v] = \sum_{o \in \Omega_{\pi(v)}} f(o) * P_\Delta\{\pi(v) = o\} \tag{1}$$

Since we presently assume that v is the only value node of the influence diagram, $E_\Delta[v]$ is actually the expected value of the influence diagram *w.r.t.* strategy Δ. The decision objective is to find an optimal strategy maximizing the expected value. i.e. to find Δ^0 such that

$$E_{\Delta^0}[v] = max\{E_\Delta[v] \mid \Delta \text{ is a strategy}\} \tag{2}$$

The computational problem related to an influence diagram is to compute the optimal expected value and an optimal strategy for the influence diagram. In this paper, we refer to this problem as *influence diagram evaluation*.

As an example, consider the oil wildcatter problem from (Raiffa 1968). In the problem, an oil wildcatter must decide whether to drill or not to drill an oil well on a particular site. He is not certain whether the site is `dry`, `wet` or `soaking`. Before making this decision, he can either order a test on the seismic structure or not. If he does, the test result will be available to him at the time when he decides whether or not to drill. The profit depends on the cost of the test, the amount of oil and the cost of drilling, which can be either low or medium or high.

An influence diagram representation of this problem is shown Fig. 1. In the diagram, boxes are decision nodes, circles are random nodes and the diamond is the value node. `T` and `D` are decision variables, corresponding to the test decision and the drill decision. `O`, `S`, `R` and `CD` are random variables, representing the amount of oil, the seismic structure, the test result, and the cost of the drill, respectively.

Decision variables have the same frame consisting of two alternatives: `yes` and `no`. The frame of random variable `O` has three values: `dry`, `wet` and `soaking`. The frame of random variable

Table 1: The function of the value node

T	D	O	CD	V
no	no	--	--	0
no	yes	dry	l	-40
no	yes	dry	m	-50
no	yes	dry	h	-70
no	yes	wet	l	80
no	yes	wet	m	70
no	yes	wet	h	50
no	yes	soaking	l	230
no	yes	soaking	m	220
no	yes	soaking	h	200

T	D	O	CD	V
yes	no	--	--	-10
yes	yes	dry	l	-50
yes	yes	dry	m	-60
yes	yes	dry	h	-80
yes	yes	wet	l	70
yes	yes	wet	m	60
yes	yes	wet	h	40
yes	yes	soaking	l	220
yes	yes	soaking	m	210
yes	yes	soaking	h	190

Table 2: The conditional probability distributation of R

T	S	R	prob
no	--	nobs	1.0
no	--	others	0
yes	--	nobs	0
yes	ns	ns	1.0
yes	cs	cs	1.0
yes	os	os	1.0

S has three values: ns for no-structure, cs for close-structure and os for open-structure. The frame of random variable R has four values: ns for no-structure, cs for close-structure, os for open-structure, and nobs for no observation. The frame of random variable CD has three values: l for low, m for medium and h for high.

The value function of the value node and the probability distributions of the random variables are given in Tables 1 — 5.

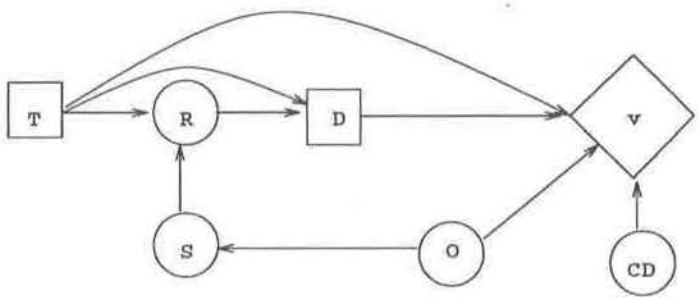


Figure 1: An influence diagram for the oil wildcatter's problem

Table 3: The conditional probability distributation of CD

CD	prob
l	0.2
m	0.7
h	0.1

Table 4: The prior probability distributation of O

O	prob
dry	0.5
wet	0.3
soaking	0.2

Table 5: The conditional probability distributation of S

O	S	prob
dry	ns	0.6
	cs	0.1
	os	0.3
wet	ns	0.3
	cs	0.3
	os	0.4
soaking	ns	0.1
	cs	0.5
	os	0.4

3 A Review of Methods for Influence Diagram Evaluation

3.1 Howard and Matheson's two–phase method

Influence diagrams were first proposed as a representational tool for decision analysis (Miller *et al.* 1976, Howard and Matheson 1984). They served as a "front–end" for the automation of decision making process. The actual analysis of a decision problem was carried out in two–phases. An influence diagram is first transformed into a decision tree and then the decision tree is evaluated.

Howard and Matheson (Howard and Matheson 1984) discussed a way to transform a regular and no–forgetting influence diagram into a decision tree. The transformation involves two steps. An influence diagram is first transformed into a *decision tree network* and then a decision tree can be constructed from the decision tree network by sequentially "expanding" variables in the network. An influence diagram is a *decision tree network* if it is regular and no–forgetting, and if all predecessors of each decision node are direct predecessors (parents) of the decision node (Howard and Matheson 1984). The basic operation for transforming a regular, no–forgetting influence diagram into a decision network is *arc reversal* (Howard and Matheson 1984, Shachter 1986).

The *arc reversal* operation is illustrated as in Fig. 2. Suppose $a \rightarrow b$ is an arc in an influence diagram such that both a and b are random nodes and there is no other directed path from node a to node b, then, the direction of the arc can be reversed and both nodes inherit each other's parents. This operation is an application of Bayesian theorem. In Fig. 2, we begin with conditional probability distributions $P\{b|a,\cdot\}$ and $P\{a|\cdot\}$, and end up with conditional probability distributions $P\{a|b,\cdot\}$ and $P\{b|\cdot\}$. Formally, we have:

$$P\{b|x,y,z\} = \sum_b P\{a,b|x,y,z\} = \sum_b P\{b|a,y,z\} * P\{a|x,y\}$$

$$P\{a|b,x,y,z\} = \frac{P\{a,b|x,y,z\}}{P\{b|x,y,z\}} = \frac{P\{b|a,y,z\} * P\{a|x,y\}}{P\{b|x,y,z\}}.$$

As an example, consider the influence diagram shown in Fig. 1. That influence diagram is regular and no–forgetting, but is not a decision tree network, since node S and O are two predecessors of node D but they are not parents of D. In order to transform that influence diagram into a decision network, we can first reverse the arc O $\rightarrow$ S and then reverse the arc S $\rightarrow$ R. The resultant decision network is shown in Fig. 3. In the course of this transformation, we have also to compute the new conditional probability distributions for nodes O and S. More specifically, when we reverse the arc O $\rightarrow$ S, we need to compute probability distributions $P\{\mathtt{O}|\mathtt{S}\}$ and $P\{\mathtt{S}\}$ from probability distributions $P\{\mathtt{S}|\mathtt{O}\}$, and $P\{\mathtt{O}\}$; when we reverse the arc S $\rightarrow$ R, we need to compute probability distributions $P\{\mathtt{R}|\mathtt{T}\}$ and $P\{\mathtt{S}|\mathtt{T},\ \mathtt{R}\}$ from probability distributions $P\{\mathtt{S}\}$ and $P\{\mathtt{R}|\mathtt{T},\ \mathtt{S}\}$.

A decision tree can be constructed from a decision tree network as follows. First, define an order $\prec$ over the set $C \cup D \cup \{v\}$ such that $a \prec b$ if either $b = v$ or there is a directed path from a to b in the network or a is a decision node and there is no directed path from b to a in the network. Then, construct a decision tree by considering variables one by one according to the order. Each layer in the decision tree corresponds to a variable. For the decision network in Fig. 3, we can obtain the following order:

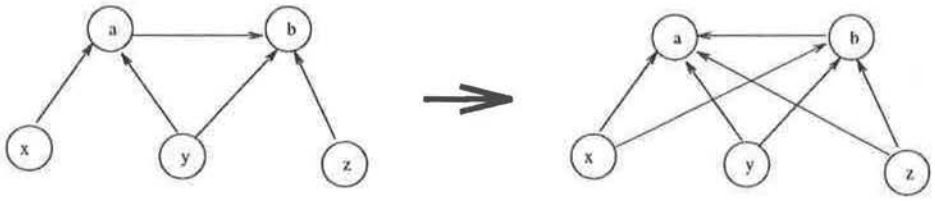


Figure 2: An illustration of arc reversal operation: reversing arc a → b

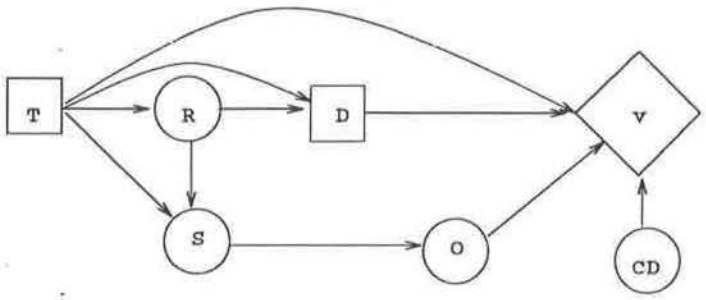


Figure 3: A decision tree network derived from the influence diagram for the oil wildcatter's problem

$$T \prec R \prec D \prec S \prec O \prec CD \prec V$$

Using Howard and Matheson's notation (Howard and Matheson 1984), the decision tree for the oil wildcatter problem is shown in Fig. 4. In the figure, boxes correspond to decision variables, circles to random variables, and diamonds to the value node. This is a compact representation of a full decision tree. A layer of the decision tree is indicated by the corresponding variable and its possible values (alternatives). In the case of a random variable, its probability distribution is also included in the layer. The full decision tree can be obtained by systematically expanding each layer and adding necessary connections in the expanded graph. Fig. 5 shows a partial decision tree resulting from the expansion of the first three layers of the compact representation.

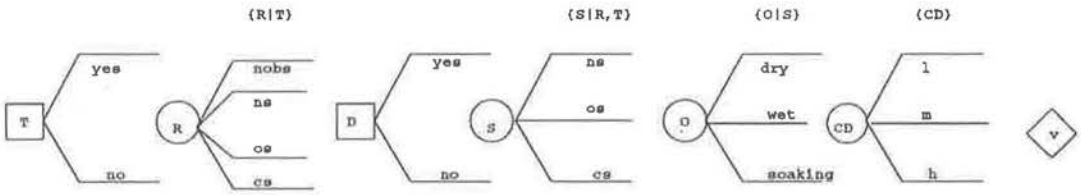


Figure 4: A compact representation of the decision tree derived for the oil wildcatter problem

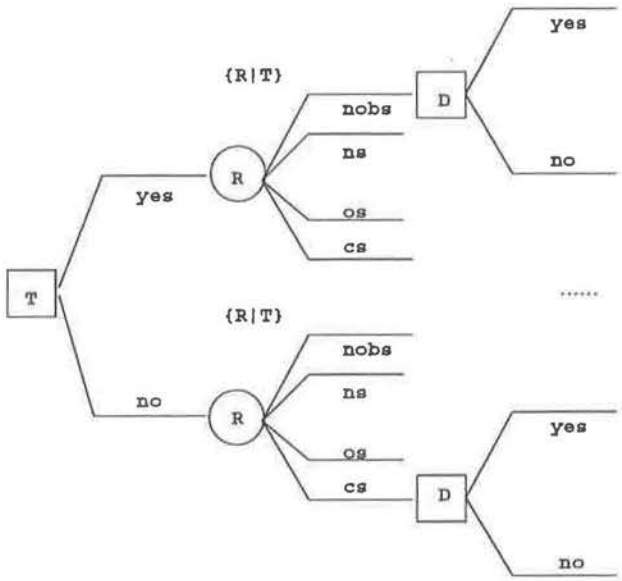


Figure 5: A partial decision tree after the expansion of the first three layers

The major problem of this approach is that the resultant decision tree tends to be very large and involve much redundancy. The depth of the decision tree so obtained from an influence diagram is equal to the number of the variables in the influence diagram. Thus, the size of the decision tree is exponential in the number of the variables in the influence diagram.

3.2 Methods for evaluating influence diagrams directly

The idea of evaluating influence diagrams directly was proposed in (Olmsted 1983). The first complete algorithm for influence diagram evaluation was developed in (Shachter 1986).

3.2.1 Shachter's algorithm

Shachter's algorithm takes a reduction approach. For a given influence diagram, the algorithm evaluates the influence diagram by applying a series of *value-preserving reductions*. A value-preserving reduction is an operation that can transform an influence diagram into another one with the same optimal expected value.

Shachter identifies four basic value-preserving reductions, namely, *barren node removal, random node removal, decision node removal* and *arc reversal*. The arc reversal operation has been illustrated in the previous section. The other reductions are illustrated as follows.

Barren node removal. A node in an influence diagram is called *barren node* if it has no children in the diagram. The barren node removal reduction states that any barren node which is not a value node, can be removed together with their incoming arcs.

Random node removal. If the value node is the only child of a random node x in an influence diagram, then the node x can be removed by conditional expectation. Afterwards, the value node inherits all of the parents of node x. The reduction is illustrated in Fig. 6 where the value function f' of the value node v' in the resultant influence diagram is given by:

$$f'(a,b,c) = \sum_x f(x,b,c) * P\{x|a,b\}.$$

Decision node removal. A decision node is called a *leaf* decision node if it has no decision node descendent. If a leaf decision node d has the value node v as its only child and $\pi(v) \subseteq \{d\} \cup \pi(d)$, then the decision node can be removed by maximization. The reduction is illustrated in Fig. 7 where the value function f' of the value node v' in the resultant influence diagram is given by:

$$f'(b) = max_d f(d,b).$$

The maximizing operation also results in an optimal decision function δ for the leaf decision node through

$$\delta(b) = arg \ \ max_d f(d,b).$$

Note that the value node does not inherit any parents from the leaf decision node. Thus, some of the parents of d may become barren nodes as a result of this reduction. In Fig. 7, node a becomes a barren node. The arc from such a node to d represents information available to the decision maker, but the information has no effects on the optimal expected value and the optimal strategy of the influence diagram. We call this kind of arcs (such as $a \rightarrow d$ in Fig. 7) *irrelevant* arcs. Later, we will see that irrelevant arcs can be identified and removed at a pre-processing step.

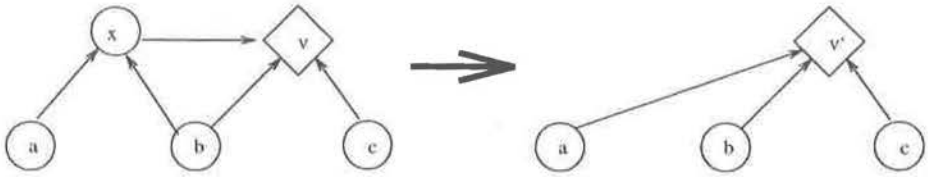


Figure 6: An illustration of random node removal: x is removed by expectation

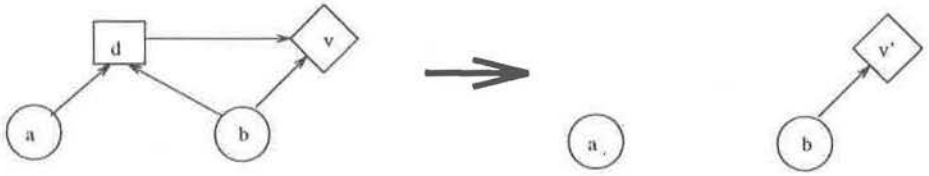


Figure 7: An illustration of decision node removal: d is removed by maximization

3.2.2 Other developments

After Shachter's algorithm, research on influence diagram evaluation has advanced in the following directions: *exploiting structural independency*, *making use of Bayesian net evaluation methods*, and *exploiting separability of the value function*. In this section, we discuss the first two directions. We will come to the third direction in Section 6.

As we have seen, during the evaluation of an influence diagram, some arcs to a decision node may turn out to be *irrelevant* to the decision node. An interesting question is that, given an influence diagram, can we identify which arcs are irrelevant in general? Shachter studied this problem in (Shachter 1988, Shachter 1990) through examining the d-separation among nodes in an influences diagram. Zhang and Poole (Zhang and Poole 1992b) recently gave a simple algorithm to identify irrelevant arcs. This algorithm also removes all barren nodes resulting from removing the irrelevant arcs. The resultant influence diagram is simpler than the original one in the sense that it has fewer nodes and arcs.

Influence diagrams are closely related to Bayesian nets (Pearl 1988), which have been a very active research area. A number of algorithms have been developed in the literature (Lauritzen and Spiegelhalter 1988, Shachter *et al.* 1990, Zhang and Poole 1992a, Jensen *et al.* 1990, Pearl 1988) for computing marginal probabilities and posterior probabilities in Bayesian nets. Thus, it is natural to ask whether we can make use of these Bayesian net algorithms for influence diagram evaluation. This problem is examined in (Shachter 1990, Shachter and Peot 1992, Cooper 1988, Zhang and Poole 1992b, Ndilikilikesha 1991, Shenoy 1990, Shenoy 1991), and the answer is affirmative.

Recall that a decision function for decision node d in an influence diagram is a mapping from $\Omega_{\pi(d)}$ to Ω_d. It is observed in (Cooper 1988, Shachter 1990, Shachter and Peot 1992, Zhang and Poole 1992b) that given a no-forgetting influence diagram, the optimal strategy can be computed by sequentially computing the optimal decision functions for decision nodes, one at a time, starting from the last one backwards. The computation of the optimal decision function

of a decision node is independent of those decision nodes which precede the decision node.

In (Cooper 1988), a recursive formula is given for computing the maximal expected values and optimal policies of influence diagrams. To some extent, the formula serves as a bridge between the evaluations of Bayesian nets and influence diagrams.

In (Shachter and Peot 1992), the problem of influence diagram evaluation is reduced to a series of Bayesian net evaluations. To this end, the value node of an influence diagram is first replaced with an "observed" probabilistic *utility* node v'. The frame of this utility node is $\{0, 1\}$. The probability distribution of v' is defined by:

$$P\{v' = 1 | \pi(v') = x\} = \frac{f(x) - f_{min}}{f_{max} - f_{min}}$$

where f_{min} and f_{max} are the smallest and the largest values of the value function V. Now, suppose that the decision nodes in the influence diagram is ordered as $d_1, ..., d_n$ such that d_n is the last decision node, d_{n-1} is the second last, etc. and d_1 is the first decision node. Then, the optimal decision function δ_n for the last decision node d_n can be computed as follows: for each element $e \in \Omega_{\pi(d_n)}$,

$$\delta_n(e) = arg \ \ max_{a \in \Omega_{d_n}} P\{d_n = a, \pi(d_n) = e | v' = 1\}.$$

In general, the optimal decision function δ_i of the decision node d_i is computed after the optimal decision functions δ_{i+1} ..., δ_n have been obtained. The decision nodes d_{i+1}, ..., d_n are first replaced with their corresponding deterministic random nodes in the influence diagram. The decision function δ_i is then computed as follows: for each element $e \in \Omega_{\pi(d_i)}$,

$$\delta_i(e) = arg \ \ max_{a \in \Omega_{d_i}} P\{d_i = a, \pi(d_i) = e | \delta_{i+1}, ..., \delta_n, v' = 1\}.$$

Thus, the problem of influence diagram evaluation is reduced to problems of computing posterior probabilities in Bayesian nets. In the same paper, Shachter and Peot pointed out that an influence diagram can be converted into a *cluster tree* which is similar to the clique trees of Bayesian nets, and the problem of evaluating the influence diagram is reduced to the problem of evaluating the cluster tree. This approach was previously used by Shenoy for his *valuation based systems* (Shenoy 1990).

In (Zhang and Poole 1992b), an influence diagram is called *stepwise solvable* if its optimal strategy can be computed by considering one decision node at a time. Zhang and Poole provide a sufficient and necessary condition on the stepwise solvability of influence diagrams. The condition is called *stepwise-decomposability* (Zhang and Poole 1992b).

Stepwise–decomposability is defined in terms of graph separation. Informally, an influence diagram is *stepwise decomposable* if for each decision node, its parents divide the influence diagram into two parts. In order to define the property formally, we need some notations and concepts.

We use $nond(x)$ to denote the set of nodes which are not descendents of x in the influence diagram. Thus, $nond(d) \cap D$ is the set of decision nodes which are not descendents of node d. For a node set Z, let $\pi(Z) = \cup_{z \in Z} \pi(z)$ and $\pi^*(Z) = Z \cup \pi(Z)$.

The *moral graph* of a directed graph G is an undirected graph $m(G)$ with the same node set such that there is an edge between node x and node y in $m(G)$ if and only if either there

is an arc $x \rightarrow y$ or $y \rightarrow x$ in G, or there are two arcs $\langle x, z \rangle$ and $\langle y, z \rangle$ in G. A node x is *m–separated* from a node y by a node set Z in a directed graph G if every path between x and y in the moral graph $m(G)$ contains at least one node in set Z.

Let d be a decision node in G, $m(G)$ be the moral graph of G and G_d be the undirected graph obtained from $m(G)$ by removing all the nodes in $\pi(d)$. The *downstream* Y_d of d is the set of all the nodes which are connected to d in G_d, with d excluded. The *upstream* X_d of d is the set of all the nodes that are not connected to d in G_d.

An influence diagram is stepwise decomposable if for each decision node d, and any node $x \in \pi^*(nond(d) \cap D)$, $\{x\} \cup \pi(x) \subseteq X_d \cup \pi(d)$. Note that a no–forgetting influence diagram is a stepwise decomposable influence diagram. The no–forgetting property is defined in terms of information availability, while stepwise decomposability is defined in terms of graph separation. In a stepwise decomposable influence diagram, an arc into a decision node indicates both informational availability and functional dependency. More precisely, for any decision node d and any other node a in a stepwise decomposable influence diagram, the presence of an arc $a \rightarrow d$ implies that the value of variable a is available at the time when decision d is to be made, and it is *not known* that the information is *irrelevant* to the decision. On the other hand, the absence of an arc $a \rightarrow d$ in an stepwise decomposable influence diagram implies that either the value of variable a is not available at the time when decision d is to be made, or it is *known* that the information is *irrelevant* to the decision. Thus, one of the advantages of stepwise decomposability over no–forgetting is that it allows the representation of the knowledge that a piece of information carried by a (no–forgetting) informational arc to a decision node in an influence diagram is irrelevant to the optimal decision function of the decision node.

Like Shachter and Peot's algorithm, Zhang and Poole's algorithm also deals with one decision node at a time. Unlike Shachter and Peot's algorithm, Zhang and Poole's algorithm takes a reduction approach. Suppose node d is a leaf decision node of a stepwise decomposable influence diagram I, then, $\pi(d)$ separates the influence diagram into two parts, namely a *body* and a *tail*. The tail is a simple influence diagram with only one decision node (d). The body's value node is a new node whose value function is obtained by evaluating the tail. A reduction step *w.r.t.* the decision node d reduces I to the body. The main computation involved in a reduction step, however, is for evaluating the tail.

Since the tail is a simple influence diagram with only one decision node, its evaluation can be directly reduced to a problem of computing posterior probabilities in a Bayesian net, as suggested in (Shachter and Peot 1992, Zhang *et al.* 1993a). The result of the evaluation consists of two parts: a function $f' : \Omega_{\pi(d)} \rightarrow \mathcal{R}$, and an optimal decision function $\delta_d : \Omega_{\pi(d)} \rightarrow \Omega_d$ for decision node d. The function f' is used in the body as the value function of the value node.
A reduction step can be described as follows.

Input: I — a stepwise decomposable influence diagram.

Output: I' — a new decomposable influence diagram, and
δ_d — the optimal decision function of a leaf decision node d.

1. Identify a leaf decision node d, and divide the influence diagram into two parts: a body and a tail.

2. Evaluate the tail to obtain a new value function f' and an optimal decision δ_d for decision node d.
3. Use f' as the value function of the value node in the body. Return the decision function and the body.

It is proved in (Zhang and Poole 1992b) that this reduction is a value preserving reduction and the resultant influence diagram is also stepwise decomposable if the original one is. Thus, this reduction can be applied recursively till the optimal expected value of the original influence diagram and an optimal decision function for each decision node are obtained. Here we can see that Zhang and Poole's algorithm and Shachter and Peot's algorithm are somehow complementary. A better algorithm can be readily obtained by combining them together. The combined algorithm will use Zhang and Poole's overall reduction framework and use Shachter and Peot's algorithm to evaluate the tail at each reduction step.

4 Some Common Weaknesses of the Previous Algorithms

One common weakness of the influence diagram evaluation algorithms we reviewed in the previous section is that they fail to provide any explicit mechanism to make use of domain dependent information (e.g. a heuristic function estimating the optimal expected values of influence diagrams), even when it is available for some problems.

Another notable and common shortcoming of these algorithms is that they do not perform asymmetric processing. This was also observed in (Shachter 1986).

In an influence diagram, all relations of variables are symmetric. Thus an influence diagram corresponds to a symmetric decision tree. The asymmetric relations of variables in a decision problem are extended into symmetric ones by introducing artificial outcomes to the frames of relevant variables. This may lead to significant inefficiency in evaluation when the asymmetry in a decision problem is substantial.

For example, the oil wildcatter problem is asymmetric in the following two aspects: (1) if the test decision is 'no', then no test result will be observed while if the test decision is 'yes', the test result can have one value out of three possibilities; (2) if the drill decision is 'no', the profit will be independent of the amount of oil and the cost of drilling.

Fung and Shachter made a first attempt to tackle the asymmetry issue. They proposed in (Fung and Shachter 1990) a modified representation, called *contingent influence diagrams*, as a tool to explicitly represent the asymmetric aspects of decision problems. However, it seems quite hard to do so without compromising the elegance of influence diagrams in representing decision problems.

In the next section, we present a new method to influence diagram evaluation. In our method, all the representational advantages of influence diagrams are retained. The asymmetric aspects can be recovered at evaluation time.

Before presenting our method, let us examine, by using an example, some of the previously discussed algorithms again, to see how and why they fail to exploit asymmetry. Observe that a common property of these algorithms is that they all compute the decision functions one by one

in the reverse order of the decision nodes (that is why we generally refer to them as reduction methods).

For the oil wildcatter problem, Shachter's algorithm will transform the original influence diagram into the one shown in Fig. 8. At this time, in order to reduce decision variable D into the value node, the following parameterized maximization operation is performed:

$$max_{\mathtt{D}}\{E[\mathtt{v}|\mathtt{D},\ \mathtt{R},\ \mathtt{T}]\}$$

In performing the above operation, a maximization is done conditioned on each of the eight elements in $\Omega_{\{\mathtt{T},\ \mathtt{R}\}}$. The result of this maximization is of two parts: a decision function $\delta_{\mathtt{D}}$ for variable D and a new value node. Both the decision function and the new value node are usually represented as a table containing one entry for each element in $\Omega_{\{\mathtt{T},\ \mathtt{R}\}}$. However, as we know, half of the elements in $\Omega_{\{\mathtt{T},\ \mathtt{R}\}}$ (e.g., T=no, R=o) represent impossible events (their marginal probabilities are zero). Therefore, the maximization conditioned on these events and the entries corresponding to these events in the decision table for variable D waste computational resources.

For the same problem, both Zhang and Poole's algorithm and Shachter and Peot's recent algorithm (Shachter and Peot 1992) will first compute the optimal decision function $\delta_{\mathtt{D}}$ for decision node D. Zhang and Poole's algorithm does not specify a particular way for this computation. Shachter and Peot's algorithm computes the function as follows. For *each element* $e \in \Omega_{\{\mathtt{T},\ \mathtt{R}\}}$, $\delta_{\mathtt{D}}(e)$ is obtained by:

$$\delta_{\mathtt{D}}(e) = arg\ \ max_{a\in\Omega_{\mathtt{D}}} P\{\mathtt{D} = a, \pi(\mathtt{D}) = e|v' = 1\}.$$

This algorithm computes $\delta_{\mathtt{D}}(e)$ even when e represents an impossible event.

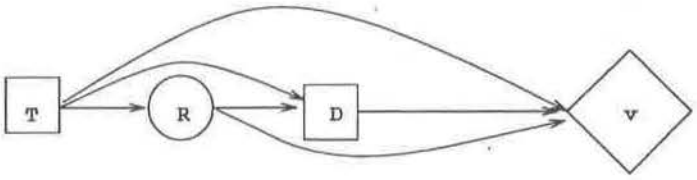


Figure 8: An intermediate influence for the oil wildcatter problem

This example shows that the problem with the current algorithms is that, for each decision node d, they will perform a maximization operation conditioned on each element in $\Omega_{\pi(d)}$, even though the marginal probabilities of some elements are zero. The reason for this is that, at the time to perform the maximization operations, the marginal probabilities of the elements in $\Omega_{\pi(d)}$ are not computed yet. This problem arises from the fact that these algorithms compute the decision functions in the reverse order of the decision nodes in the influence diagram.

5 A Search Oriented Algorithm

In this section, we present our method for influence diagram evaluation. We first formulate the influence diagram evaluation problem as a stochastic dynamic programming problem. Then we give

a graphical depiction of the computational structure of the optimal expected value by mapping an influence diagram into a decision graph. Finally, we point out how to avoid wasteful computation in computing the optimal strategy, and propose a search oriented approach to influence diagram evaluation.

5.1 Preliminaries

A decision node d *directly precedes* another decision node d' if d precedes d' and there is no other decision node d'' such that d precedes d'' and d'' precedes d'. In a regular influence diagram, a decision node can directly precede at most one decision node.

A decision node that is preceded by no other decision node is called a *root* decision node.

Let I be a regular, stepwise influence diagram with a single value node v. Suppose the decision nodes in I are d_1 ..., d_n in the regularity order, then, d_1 is the only root decision node and d_n is the only leaf decision node. For each k, $1 \leq k < n$, d_k directly precedes d_{k+1}. For each k, $1 \leq k < n$ let $I(d_k, d_{k+1})$ denote the subgraph consisting of d_k, $\pi(d_k)$ and those non-descendents of d_{k+1} which are not m-separated from d_k by $\pi(d_k)$. Procedurally, $I(d_k, d_{k+1})$ can be obtained from I as follows: (1) remove all nodes that are m-separated from d_k by $\pi(d_k)$, excluding the nodes in $\pi(d_k)$; (2) remove all the descendents of d_{k+1}; (3) remove all the arcs among the nodes in $\pi(d_k) \cup \{d_k\}$ and assign uniform distribution to the root nodes[2] in $\pi(d_k) \cup \{d_k\}$.

$I(d_k, d_{k+1})$ is called the *section* of I from d_k to d_{k+1}. For the root decision node d_1, the section $I(-, d_1)$ contains only non-descendents of d_1. For the leaf decision node d_n, the section $I(d_n, -)$ contains those nodes in $\pi(d_n) \cup \{d_n\}$ and those nodes which are not m-separated from d_n by $\pi(d_n)$.

It is easy to see that the section $I(-, d_1)$ is a Bayesian net. Furthermore, because I is stepwise-decomposable, it is easy to see that d_k is the only decision node in the section $I(d_k, d_{k+1})$ that are not in $\pi(d_k)$. Therefore $I(d_k, d_{k+1})$ is a Bayesian network. Similarly, d_n is the only decision node in the section $I(d_n, -)$. Thus, $I(d_n, -)$ is a Bayesian net with a value node attached.

As an example, consider again the oil wildcatter problem. The section $I(-, \mathtt{T})$ is empty. The sections $I(\mathtt{T}, \mathtt{D})$ and $I(\mathtt{D}, -)$ are shown in Fig. 9.

Let $\Delta = (\delta_1, ..., \delta_n)$ be any strategy for I. We have the following results on I_Δ.

Lemma 1 *For any k, $1 < k \leq n$, any j, $1 \leq j < k$, $\pi(v)$ is independent of $\pi(d_j)$ and d_j, given $\pi(d_k)$. Formally, the following relations hold:*

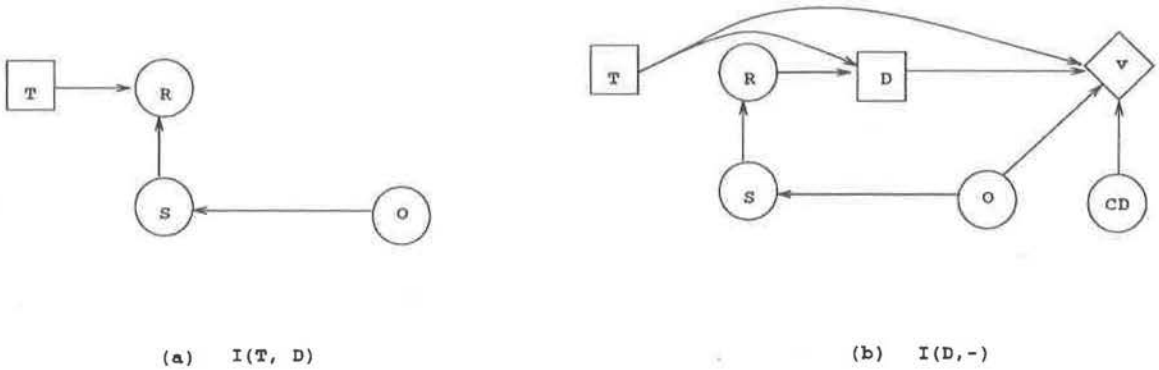
$$P_\Delta\{\pi(v) = o | \pi(d_k) = y\} = P_\Delta\{\pi(v) = o | \pi(d_k) = y, \pi(d_j) = x, d_j = a\}$$

for any $o \in \Omega_{\pi(v)}$, $x \in \Omega_{\pi(d_j)}$, $a \in \Omega_{d_j}$, and $y \in \Omega_{\pi(d_k)}$.

Proof Immediately follows the m-separation property of a stepwise decomposable influence diagram.

---

[2]The assignment of uniform distributions to the root nodes in $\pi(d_k) \cup \{d_k\}$ is only to make $I(d_k, d_{k+1})$ a Bayesian network. Since we shall only be considering probabilities conditioned on $\pi(d_k) \cup \{d_k\}$, the distributions of those nodes are irrelevant.



(a) I(T, D) (b) I(D,-)

Figure 9: Two sections of the influence diagram for the oil wildcatter problem.

Lemma 2 *For any* k, $1 \leq k \leq n$, *and any* $o \in \Omega_{\pi(v)}$, $x, \in \Omega_{\pi(d_k)}$,

$$P_\Delta\{\pi(v) = o|\pi(d_k) = x, d_k = \delta_k(x)\} = P_\Delta\{\pi(v) = o|\pi(d_k) = x\}$$

Proof. Recall that, *w.r.t.* a strategy $\Delta = (\delta_1, ...\delta_n)$, the decision node d_i, for $i = 1, ..., n$, is characterized by the following probability distribution:

$$P\{d_i = x|\pi(d_i) = c\} = \begin{cases} 1 & \text{if } \delta_i(c) = x, \\ 0 & \text{otherwise.} \end{cases}$$

Thus,

$$\begin{aligned} & P_\Delta\{\pi(v) = o|\pi(d_k) = x\} \\ & \quad = \sum_{a \in \Omega_(d_k)} P_\Delta\{\pi(v) = o|\pi(d_k) = x, d_k = a\} * P_\Delta\{d_k = a|\pi(d_k) = x\} \\ & \quad = P_\Delta\{\pi(v) = o|\pi(d_k) = x, d_k = \delta_k(x)\} \end{aligned}$$

□

Lemma 3 *For any* $x \in \Omega_{\pi(d_1)}$, *the probability* $P_\Delta\{\pi(d_1) = x\}$ *depends on only those nodes in the section* $I(-, d_1)$, *and is independent of* Δ. *Consequently, for any other strategy* Δ',

$$P_\Delta\{\pi(d_1) = x\} = P_{\Delta'}\{\pi(d_1) = x\}$$

Proof. Since all the nodes not in $I(-, d_1)$ are descendents of the nodes in $\pi(d_1)$, thus, they are irrelevant to the marginal probabilities of $\pi(d_1)$. Since there is no decision node in $I(-, d_1)$, then $P_\Delta\{\pi(d_1) = x\}$ is independent of Δ. □

Lemma 4 (1) *For any* $o \in \Omega_{\pi(v)}$, $x, \in \Omega_{\pi(d_n)}$ *and* $a \in \Omega_{d_n}$, *the conditional probability* $P_\Delta\{\pi(v) = o|\pi(d_n) = x, d_n = a\}$ *depends on only those nodes in the section* $I(d_n, -)$, *and is independent of* Δ. *In other words, for any other strategy* Δ',

$$P_\Delta\{\pi(v) = o|\pi(d_n) = x, d_n = a\} = P_{\Delta'}\{\pi(v) = o|\pi(d_n) = x, d_n = a\}.$$

(2) *For any* $y \in \Omega_{\pi(d_{k+1})}$, $x \in \Omega_{\pi(d_k)}$ *and* $a \in \Omega_{d_k}$, *the conditional probability* $P_\Delta\{\pi(d_{k+1}) = y|\pi(d_k) = x, d_k = a\}$ *depends on only those nodes in the section* $I(d_k, d_{k+1})$, *and is independent of* Δ. *In other words, for any other strategy* Δ',

$$P_\Delta\{\pi(d_{k+1}) = y|\pi(d_k) = x, d_k = a\} = P_{\Delta'}\{\pi(d_{k+1}) = y|\pi(d_k) = x, d_k = a\}.$$

(3) *Suppose* $\Delta' = (\delta'_1, ..., \delta'_n)$ *is another strategy for* I *such that* $\delta'_1 = \delta_1, ..., \delta'_{k-1} = \delta_{k-1}$ *for some* k, $1 \leq k \leq n$, *then, for any* j, $1 \leq j \leq k$, *and any* $x \in \Omega_{\pi(d_j)}$,

$$P_\Delta\{\pi(d_j) = x\} = P_{\Delta'}\{\pi(d_j) = x\}.$$

Proof. Follows the definition of sections and the m–separation property of a stepwise decomposable influence diagram. □

Lemmas 3 and 4 indicate that some conditional probabilities in an influence diagram is independent of the strategies for the influence diagram, and can be computed in a section of the influence diagram. The computations can be carried out by any one of the well established algorithms for Bayesian nets. This fact facilitates a clean interface between influence diagram evaluation and Bayesian net evaluation.

5.2 Influence diagram evaluation vs. stochastic dynamic programming

In this section, we establish a stochastic dynamic programming formulation for influence diagram evaluation. We accomplish this by studying the relationship among the conditional expected values of influence diagrams.

Let e be any event in I_Δ and let $E_\Delta[v|e]$ be defined as follows:

$$E_\Delta[v|e] = \sum_{o \in \Omega_{\pi(v)}} f(o) * P_\Delta\{\pi(v) = o|e\}.$$

For each k, $1 \leq k \leq n$, let U_k be a function defined as follows.

$$U_k(x, \Delta) = E_\Delta[v|\pi(d_k) = x] \tag{3}$$

Informally, $U_k(x, \Delta)$ is the expected value of the influence diagram *w.r.t.* strategy Δ, conditioned on $\pi(d_k) = x$.

Lemma 5 *The expected value of the influence diagram* w.r.t. *strategy* Δ *can be expressed in terms of* U_k *as:*

$$E_\Delta[v] = \sum_{x \in \Omega_{\pi(d_k)}} U_k(x, \Delta) * P_\Delta\{\pi(d_k) = x\}$$

Proof By the definition of $E_\Delta[v]$, we have:

$$E_\Delta[v] = \sum_{o \in \Omega_{\pi(v)}} f(o) * P_\Delta\{\pi(v) = o\}.$$

Since

$$P_\Delta\{\pi(v) = o\} = \sum_{x \in \Omega_{\pi(d_k)}} P_\Delta\{\pi(v) = o | \pi(d_k) = x\} * P_\Delta\{\pi(d_k) = x\},$$

thus,

$$E_\Delta[v] = \sum_{o \in \Omega_{\pi(v)}} f(o) * \sum_{x \in \Omega_{\pi(d_k)}} P_\Delta\{\pi(v) = o | \pi(d_k) = x\} * P_\Delta\{\pi(d_k) = x\}.$$

By reordering the order of the two summations, we have:

$$E_\Delta[v] = \sum_{x \in \Omega_{\pi(d_k)}} P_\Delta\{\pi(d_k) = x\} * \sum_{o \in \Omega_{\pi(v)}} f(o) * P_\Delta\{\pi(v) = o | \pi(d_k) = x\}.$$

By the definition of U_k, we have:

$$E_\Delta[v] = \sum_{x \in \Omega_{\pi(d_k)}} U_k(x, \Delta) * P_\Delta\{\pi(d_k) = x\}.$$

□

Lemma 6 *The following relation between functions U_k and U_{k-1} holds.*

$$U_{k-1}(x, \Delta) = \sum_{y \in \Omega_{\pi(d_k)}} U_k(y, \Delta) * P_\Delta\{\pi(d_k) = y | \pi(d_{k-1}) = x\}$$

for any $x \in \Omega_{\pi(d_{k-1})}$.

Proof By the definition of U_{k-1}, we have:

$$U_{k-1}(x, \Delta) = \sum_{o \in \Omega_{\pi(v)}} f(o) * P_\Delta\{\pi(v) = o | \pi(d_{k-1}) = x\}.$$

Since

$$P_\Delta\{\pi(v) = o | \pi(d_{k-1}) = x\} = \sum_{y \in \Omega_{\pi(d_k)}} P_\Delta\{\pi(v) = o | \pi(d_{k-1}) = x, \pi(d_k) = y\} * P_\Delta\{\pi(d_k) = y | \pi(d_{k-1}) = x\},$$

then, by Lemma 1, we have:

$$U_{k-1}(x, \Delta) = \sum_{o \in \Omega_{\pi(v)}} f(o) * \sum_{y \in \Omega_{\pi(d_k)}} P_\Delta\{\pi(v) = o | \pi(d_k) = y\} * P_\Delta\{\pi(d_k) = y | \pi(d_{k-1}) = x\}.$$

By reordering the two summations, we obtain:

$$U_{k-1}(x, \Delta) = \sum_{y \in \Omega_{\pi(d_k)}} P_\Delta\{\pi(d_k) = y | \pi(d_{k-1}) = x\} * \sum_{o \in \Omega_{\pi(v)}} f(o) * P_\Delta\{\pi(v) = o | \pi(d_k) = y\}.$$

By the definition of U_k, we have:

$$U_{k-1}(x, \Delta) = \sum_{y \in \Omega_{\pi(d_k)}} U_k(y, \Delta) * P_\Delta\{\pi(d_k) = y | \pi(d_k) = x\}.$$

□

Lemma 7 *Let* $\Delta' = (\delta'_1, ..., \delta'_n)$ *be another strategy for* I *such that* $\delta'_k = \delta_k, ..., \delta'_n = \delta_n$, *for some* k, $1 \leq k \leq n$. *Then,* $U_j(x, \Delta) = U_j(x, \Delta')$ *for each* j, $k \leq j \leq n$ *and each* $x \in \Omega_{\pi(d_j)}$.

Proof By induction.

Basis: Consider U_n. By the definition of U_n, we have:

$$U_n(x, \Delta) = \sum_{o \in \Omega_{\pi(v)}} f(o) * P_\Delta\{\pi(v) = o | \pi(d_n) = x\}$$

and

$$U_n(x, \Delta') = \sum_{o \in \Omega_{\pi(v)}} f(o) * P_{\Delta'}\{\pi(v) = o | \pi(d_n) = x\}.$$

By Lemma 2, we have:

$$P_{\Delta'}\{\pi(v) = o | \pi(d_n) = x\} = P_{\Delta'}\{\pi(v) = o | \pi(d_n) = x, d_n = \delta'_n(x)\}$$

and

$$P_\Delta\{\pi(v) = o | \pi(d_n) = x\} = P_\Delta\{\pi(v) = o | \pi(d_n) = x, d_n = \delta_n(x)\}.$$

Since $\delta_n = \delta'_n$, then by Lemma 4-(1), we have:

$$P_\Delta\{\pi(v) = o | \pi(d_n) = x, d_n = \delta_n(x)\} = P_{\Delta'}\{\pi(v) = o | \pi(d_n) = x, d_n = \delta'_n(x)\}.$$

Thus, $U_n(x, \Delta) = U_n(x, \Delta')$. Therefore, the basis holds.

Induction: Suppose $U_i(x, \Delta) = U_i(x, \Delta')$ for all i, $k < i \leq n$. By Lemma 6, we have:

$$U_{i-1}(x, \Delta) = \sum_{y \in \Omega_{\pi(d_i)}} U_i(y, \Delta) * P_\Delta\{\pi(d_i) = y | \pi(d_{i-1}) = x\}$$

and

$$U_{i-1}(x, \Delta') = \sum_{y \in \Omega_{\pi(d_i)}} U_i(y, \Delta') * P_{\Delta'}\{\pi(d_i) = y | \pi(d_{i-1}) = x\}.$$

By the induction hypothesis, we have:

$$U_i(y, \Delta) = U_i(y, \Delta').$$

By Lemma 2, we have:

$$P_\Delta\{\pi(d_i) = y | \pi(d_{i-1}) = x\} = P_\Delta\{\pi(d_i) = y | \pi(d_{i-1}) = x, d_{i-1} = \delta_{i-1}(x)\}$$

and

$$P_{\Delta'}\{\pi(d_i) = y | \pi(d_{i-1}) = x\} = P_{\Delta'}\{\pi(d_i) = y | \pi(d_{i-1}) = x, d_{i-1} = \delta'_{i-1}(x)\}.$$

Since $\delta_{i-1} = \delta'_{i-1}$, then by Lemma 4-(2), we obtain:

$$P_{\Delta'}\{\pi(d_i) = y | \pi(d_{i-1}) = x, d_{i-1} = \delta'_{i-1}(x)\} = P_\Delta\{\pi(d_i) = y | \pi(d_{i-1}) = x, d_{i-1} = \delta_{i-1}(x)\}.$$

Thus,

$$P_\Delta\{\pi(d_i) = y | \pi(d_{i-1}) = x\} = P_{\Delta'}\{\pi(d_i) = y | \pi(d_{i-1}) = x\}.$$

Therefore,

$$U_{i-1}(x, \Delta) = U_{i-1}(x, \Delta').$$

Therefore, the lemma holds in general. □

So far, we developed some results on the expected values of an influence diagram *w.r.t.* an arbitrary strategy. Now, let us examine the properties of an optimal strategy. Let $\Delta^0 = (\delta_1^0, ..., \delta_n^0)$ be an optimal strategy for influence diagram I and let V_k be a function defined as

$$V_k(x) = U_k(x, \Delta^0).$$

Intuitively, $V_k(x)$ is the optimal expected value of the influence diagram I conditioned on $\pi(d_k) = x$. In other words, $V_k(x)$ is the expected value a decision maker can obtain if he starts to make optimal decisions in the situation represented by the event $\pi(d_k) = x$.

Let $V_k'(x, a)$ be an auxiliary function defined as:

$$V_k'(x, a) = \sum_{y \in \Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\Delta^0}\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = a\} \tag{4}$$

Intuitively, $V_k'(x, a)$ is the optimal expected value of the influence diagram I conditioned on $\pi(d_k) = x, d_k = a$. In other words, $V_k'(x, a)$ is the expected value a decision maker can obtain if he starts to make decisions in the situation represented by the event $\pi(d_k) = x$, and first chooses a for d_k (in this situation) and then follows an optimal strategy for the rest decisions.

The next two lemmas characterize the relationship between V_k and V_k'.

Lemma 8 *For all* $k = 1, ..., n$,

$$V_k'(x, \delta_k^0(x)) = V_k(x).$$

Proof

$$\begin{aligned}
&V_k'(x, \delta_k^0(x)) \\
&= \sum_{y \in \Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\Delta^0}\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = \delta_k^0(x)\} \\
&= \sum_{y \in \Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\Delta^0}\{\pi(d_{k+1}) = y | \pi(d_k) = x\} \quad \text{by Lemma 2} \\
&= \sum_{y \in \Omega_{\pi(d_{k+1})}} U_{k+1}(y, \Delta^0) * P_{\Delta^0}\{\pi(d_{k+1}) = y | \pi(d_k) = x\} \quad \text{by the definition of } V_k \\
&= U_k(x, \Delta^0) \quad \text{by Lemma 6} \\
&= V_k(x) \quad \text{by the definition of } V_{k-1}.
\end{aligned}$$

□

Lemma 9 *For all* $k = 1, ..., n$,

$$V_k'(x, \delta_k^0(x)) \geq V_k'(x, a) \quad \textit{for each } x \in \Omega_{\pi(d_k)} \textit{ and each } a \in \Omega_{d_k}.$$

$$U_k(x, \Delta^0) \geq U_k(x, \Delta) \quad \textit{for every strategy } \Delta \textit{ and each } x \in \Omega_{\pi(d_k)}.$$

Proof (1) Suppose the first inequality does not hold. Thus, there exist $x_0 \in \Omega_{\pi(d_k)}$ and $a_0 \in \Omega_{d_k}$ such that

$$V'_k(x_0, a_0) > V'_k(x_0, \delta^0_k(x_0)).$$

Construct a strategy $\Delta' = (\delta'_1, ..., \delta'_n)$ such that $\delta_i = \delta^0_i$ for all i, $1 \leq i \leq n$, $i \neq k$ and $\delta'_k(x_0) = a_0$, and $\delta'_k(x) = \delta^0_k(x)$, for all $x \in \Omega_{\pi(d_k)}, x \neq x_0$. It can be proved that $E_{\Delta'}[v] > E_{\Delta^0}[v]$. A contradiction to the optimality assumption of Δ^0.

(2) The second inequality can be proved by a simple induction on k, starting from $k = n$ and backwards. □

The results we have obtained so far can be summarized into the following theorem.

Theorem 1 *Let I be a regular and stepwise decomposable influence diagram, let functions V_k and V'_k be defined as before, and let $\Delta^0 = (\delta^0_1, ..., \delta^0_n)$ be an optimal strategy for I. For any $k, 1 \leq k < n$, $x \in \Omega_{\pi(d_k)}$ and $a \in \Omega_{d_k}$, the following relations hold.*

$$V'_k(x, a) = \sum_{y \in \Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = a\} \tag{5}$$

$$V_k(x) = V'_k(x, \delta^0_k(x)) = max_{a \in \Omega_{d_k}} V'_k(x, a) \tag{6}$$

$$\delta^0_k(x) = arg \ \ max_{a \in \Omega_{d_k}} \{V'_k(x, a)\} \tag{7}$$

$$E_{\Delta^0}[v] = \sum_{x \in \Omega_{d_1}} V_0(x) * P\{\pi(d_1) = x\}. \tag{8}$$

where $P\{\pi(d_1) = x\} = P_{\Delta^0}\{\pi(d_1) = x\}$ can be computed in the section $I(-, d_1)$ and $P\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = a\} = P_{\Delta^0}\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = a\}$ can be computed in the section $I(d_k, d_{k+1})$, and they are independent of Δ^0.

Equations 5, 6 and 7 establish the computational structure of influence diagram evaluation in the form of finite–stage stochastic dynamic programming (Ross 1983). They essentially describe an expectation–maximization iteration for computing the optimal strategy and the optimal expected value. Equations 5 and 6 collectively form a variation of Bellman's optimality principle (Bellman 1957) for stochastic dynamic programming.

According to Theorem 1, if we can compute function V_n, then we can compute functions $V_1, ..., V_{n-1}$ and $\delta^0_1, ..., \delta^0_{n-1}$, as well as $E_{\Delta^0}[v]$. The computation process is similar to the one implied in the recursive formula given in (Cooper 1988). It is not hard to observe that the amount of computation involved is comparable to that involved in the other algorithms such as those in (Zhang and Poole 1992b, Shachter and Peot 1992).

Now, let us consider how to compute the functions V_n and δ^0_n. Recall that

$$V_n(x) = U_n(x, \Delta^0)$$

and

$$U_n(x, \Delta) = \sum_{o \in \Omega\pi(v)} f(o) * P_\Delta\{\pi(v) = o | \pi(d_n) = x\}.$$

By Lemma 2, we have:

$$U_n(x,\Delta) = \sum_{o\in\Omega\pi(v)} f(o) * P_\Delta\{\pi(v) = o|\pi(d_n) = x, d_n = \delta_n(x)\}.$$

According to Lemma 4-(1), the probability term in the above equation is independent of the other decision functions of Δ other than δ_n. Thus,

$$U_n(x,\Delta) = \sum_{o\in\Omega\pi(v)} f(o) * P\{\pi(v) = o|\pi(d_n) = x, d_n = \delta_n(x)\}.$$

Since $U_n(x,\Delta^0) \geq U_n(x,\Delta)$ for every strategy Δ, we have:

$$\sum_{o\in\Omega\pi(v)} f(o) * P\{\pi(v) = o|\pi(d_n) = x, d_n = \delta_n^0(x)\}$$
$$\geq \sum_{o\in\Omega\pi(v)} f(o) * P\{\pi(v) = o|\pi(d_n) = x, d_n = \delta_n(x)\}$$

for any decision function δ_n of d_n. This is equivalent to:

$$\sum_{o\in\Omega\pi(v)} f(o) * P\{\pi(v) = o|\pi(d_n) = x, d_n = \delta_n^0(x)\}$$
$$\geq \sum_{o\in\Omega\pi(v)} f(o) * P\{\pi(v) = o|\pi(d_n) = x, d_n = a\}$$

for any $a \in \Omega_{d_n}$. Therefore, we have:

$$\delta_n^0(x) = arg\ \ max_{a\in\Omega_{d_n}}\{\sum_{o\in\Omega\pi(v)} f(o) * P\{\pi(v) = o|\pi(d_n) = x, d_n = a\}\}$$

and

$$V_n(x) = \sum_{o\in\Omega\pi(v)} f(o) * P\{\pi(v) = o|\pi(d_n) = x, d_n = \delta_n^0(x)\}.$$

The computation of functions δ_n^0 and V_n involves only the section $I(d_n,-)$ of I. These functions can be computed directly from the above formulas or be more efficiently computed by the methods as suggested in (Shachter and Peot 1992, Zhang *et al.* 1993a).

5.3 Decision graphs

From the structural point of view, a *decision graph* (Qi 1993, Qi and Poole 1993) is an acyclic AND/OR graph (Pearl 1984, Nilsson 1982) with maximization-expectation evaluation function. More precisely, a decision graph is a directed acyclic graph whose nodes are classified into two types: *choice nodes* and *chance nodes*. Each decision graph has exactly one *root*. A subset of nodes is designated as terminals. Each terminal has a value associated with it. A value is associated with each arc emanating from a choice node. Each chance node has a (discrete) probability distribution over its children. In other words, a probability is associated with each of

the children of a chance node, and the probabilities of all the children of a chance node sum to unit.

A *solution graph* SG, *w.r.t.* a node M, of a decision graph DG is a graph with the following characteristics:

1. M is in SG;
2. If a non–terminal chance node of DG is in SG, then all of its children are in SG;
3. If a non–terminal choice node of DG is in SG, then exactly one of its children is in SG.

A solution graph *w.r.t.* the root of a decision graph is simply referred to as a solution graph of the decision graph.

5.4 Representing the computational structure by decision graphs

Before we discuss how to construct decision graphs for influence diagrams, we need some terminologies.

Let d be a decision node in an influence diagram. For each $x \in \Omega_{\pi(d)}$, we call an assignment in the form of $\pi(d) = x$ a *parent situation* of the decision node d. For each alternative $a \in \Omega_d$, we call an assignment in the form of $\pi(d_i) = x, d = a$ an *inclusive situation* of the decision node d.

For an influence diagram, we can define a decision graph in terms of situations. In the graph, a choice node represents a parent situation and a chance node represents an inclusive situation. The following is a specification of such a decision graph.

- The empty situation is the root, which is a chance node, of the decision graph.
- For each decision node d_i, $1 \leq i < n$, and each $x \in \Omega_{\pi(d_i)}$, there is a choice node in the decision graph representing the parent situation $\pi(d_i) = x$; for each $a \in \Omega_{d_i}$, there is a chance node in the decision graph representing the inclusive situation $\pi(d_i) = x, d_i = a$.
- Let s be a choice node representing a parent situation $\pi(d_i) = x$, $1 \leq i < n$, the chance nodes representing the inclusive situations $\pi(d_i) = x, d_i = a$ for all $a \in \Omega_{d_i}$ constitute the children of node s. The value associated with the arcs emanating from the choice node are all zero.
- For each $x \in \Omega_{d_n}$, there is a terminal node representing the parent situation $\pi(d_n) = x$. The value of the terminal is $V_n(x)$.
- Let s be a chance node representing an inclusive situation $\pi(d_i) = x, d_i = a$, and let $\mathcal{S}_{i+1}$ denote the set of choice nodes representing the parent situations of d_{i+1}, s has the nodes in $\mathcal{S}_{i+1}$ as its children. In other words, for each $x \in \Omega_{\pi(d_i)}$, $a \in \Omega_{d_i}$ and $y \in \Omega_{\pi(d_{i+1})}$ there is an arc from the chance node representing the inclusive situation $\pi(d_i) = x, d_i = a$ to a choice node representing the parent situation $\pi(d_{i+1}) = y$. The arc is labeled by the probability $P\{\pi(d_{i+1}) = y | \pi(d_i) = x, d_i = a\}$.

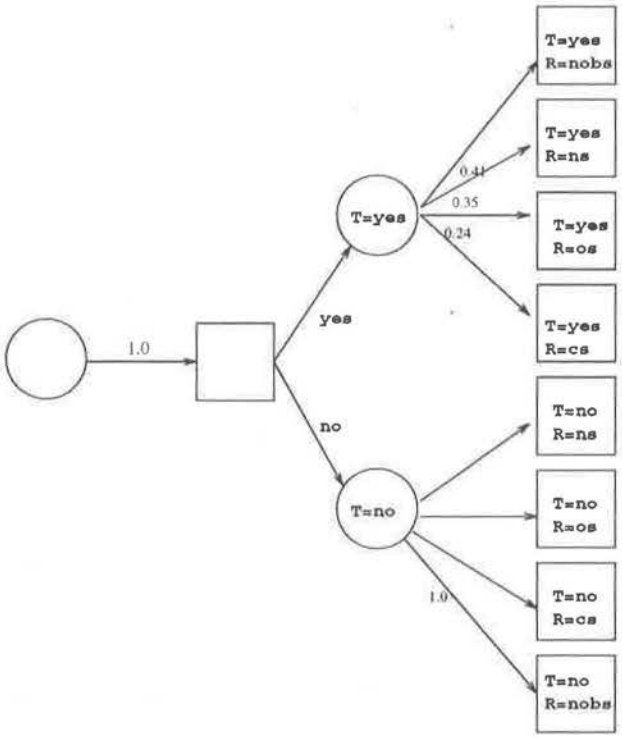


Figure 10: A complete decision graph for the oil wildcatter problem

- For each choice node representing a parent situation of d_1 in form of $\pi(d_1) = x$, there is an arc from the root node to the choice node. On the the arc is a probability $P\{\pi(d_1) = x\}$.

In such a decision graph, a choice node represents a parent situation in the form of $\pi(d_k) = x$ for some k and $x \in \Omega_{\pi(d_k)}$. In such a situation, the decision agent needs to decide which alternative among Ω_{d_k} should be selected for d_k. Thus, the choice node has $|\Omega_{d_k}|$ children, each for an alternative value in Ω_{d_k}. The child corresponds to alternative a is a chance node, representing the inclusive situation $\pi(d_k) = x, d_k = a$. From this inclusive situation, one of the parent situations of d_{k+1} may be reached, according to a probability distribution. The probability of reaching the parent situation $\pi(d_{k+1}) = y$ is $P\{\pi(d_{k+1}) = y|\pi(d_k) = x, d_k = a\}$.

As an example, consider the oil wildcatter problem. A complete decision graph for the oil wildcatter problem is shown in Fig. 10. As we know, a probability is associated with each arcs from a chance node to a choice node. These probabilities can be computed in the section I(T, D) as shown in Fig. 9.

In Fig. 10, those arcs without labels are associated with zero probability. Those non–zero probabilities are computed as follows.

$$
\begin{aligned}
&P\{\texttt{T = yes, R = ns|T=yes}\} \\
&\quad = P\{\texttt{R = ns| T = yes}\} \\
&\quad = P\{\texttt{R = ns| T = yes, S=ns}\} * P\{\texttt{S = ns}\} \\
&\qquad\quad + P\{\texttt{R = ns| T = yes, S=os}\} * P\{\texttt{S = os}\} \\
&\qquad\quad + P\{\texttt{R = ns| T = yes, S=cs}\} * P\{\texttt{S = cs}\} \\
&\quad = 1 * P\{\texttt{S = ns}\} + 0 * P\{\texttt{S = os}\} + 0 * P\{\texttt{S = cs}\} \\
&\quad = P\{\texttt{S = ns}\}
\end{aligned}
$$

Similarly, we have:

$$P\{\texttt{T = yes, R = os|T=yes}\} = P\{\texttt{S = os}\}$$

and

$$P\{\texttt{T = yes, R = cs|T=yes}\} = P\{\texttt{S = cs}\}.$$

The marginal probabilities of S are computed as follows.

$$
\begin{aligned}
P\{\texttt{S = ns}\} \;=\; & P\{\texttt{S=ns|O = dry}\} * P\{\texttt{O=dry}\} \\
& + P\{\texttt{S=ns|O = wet}\} * P\{\texttt{O=wet}\} \\
& + P\{\texttt{S=ns|O = soaking}\} * P\{\texttt{O=soaking}\} \\
=\; & 0.6 * 0.5 + 0.3 * 0.3 + 0.1 * 0.2 = 0.41
\end{aligned}
$$

Similarly, we can obtain that $P\{\texttt{S = os}\} = 0.35$ and $P\{\texttt{S = cs}\} = 0.24$.

Let DG be such a decision graph, we can define a *maximization–expectation* evaluation function, u_1, on DG as follows:

- If s is a terminal representing a situation $\pi(d_n) = x$, then

$$u_1(DG, s) = V_n(x)$$

- If s is a chance node, with children $s_1, ..., s_l$, then

$$u_1(DG, s) = \sum_{i=1}^{l} P_i * u_1(DG, s_i)$$

where P_i is the probability on the arc from node s to node s_i.

- If s is a choice node, with children $s_1, ..., s_l$, then

$$u_1(DG, s) = max_{i=1}^{l}\{u_1(DG, s_i)\}.$$

A solution graph SG of the decision graph DG is *optimal w.r.t.* the evaluation function u_1 if $u_1(SG, s) = u_1(DG, s)$ for every node s in SG. The following lemma can be easily proved by induction on nodes in the decision graph.

Lemma 10 *(1) If s is a choice node representing a parent situation $\pi(d_k) = x$, then $u_1(\mathrm{DG}, s) = V_k(x)$.*

(2) If s is a chance node representing an inclusive situation $\pi(d_k) = x, d_k = a$, then $u_1(\mathrm{DG}, s) = V_k'(x, a)$.

(3) If s is the root, then $u_1(\mathrm{DG}, s)$ is equal to the optimal expected value of the influence diagram.

Now, the correspondence between the optimal strategies of the influence diagrams and the optimal solution graphs should be apparent. As a matter of fact, an optimal solution graph of the decision graph can be viewed as a representation of decision tables in which all the unreachable situations are removed (Zhang *et al.* 1993c). Thus, the problem of influence diagram evaluation is reduced to the problem of decision graph search.

5.5 Computing the optimal solution graph

The optimal solution graph of a decision graph can be computed in a "bottom–up" way or in a "top–down" way. The bottom–up computation will compute the values for all leaves first. In the course of computing leaves, δ_n, the optimal decision function for decision d_n is also computed. Then, the max–exp values of interior nodes can be computed when the max–exp values of all children of the node are available. The computational complexity of this process is linear in the size of the decision graph[3]. This method also has the weaknesses we mentioned in Section 4.

5.5.1 Recovering the asymmetric property

We observe that the asymmetric property of an influence diagram is reflected by the arcs with zero probability in the corresponding decision graph. As we know, the value of a chance node in a decision graph is the expectation of the values of its children. If the probability on the arc to a child is known in advance to be zero, then there is no need to compute the value of the child (as far as this chance node is concerned). In case the probabilities on all the arcs to a node are

[3] Note that the size of the decision graph is normally exponential. See the analysis in the next section.

all zero, the value of the node will never be required, thus some computation effort can be saved. Unfortunately, this saving cannot be exploited in bottom–up computations as described above. One way to exploit asymmetry is to use the following procedure:

(1) generate a decision graph from an influence diagram;

(2) compute the probabilities of all arcs emanating from chance nodes;

(3) delete all arcs with zero probability;

(4) delete all those nodes, along with the arcs incident to and from them, which are not reachable from the root node in the decision graph;

(5) compute an optimal solution from the resultant graph.

Consider the decision graph shown in Fig. 10. In the figure, those arcs, emanating from circle nodes and without labels, have zero probability. After removing from the graph those arcs with zero probability and those nodes not reachable from the root we obtain a simpler decision graph as shown in Fig. 11. With this decision graph, we will no longer need to compute the values for the parent situations of D which represent the impossible situations such `T=no, R = ns`.

The values for the rest terminals can be computed locally in the section $I(\mathtt{D}, -)$ (as shown in Fig. 9) by various algorithms. The optimal choices for the decision variable `D` in the situations corresponding to these terminals are also computed in the process of computing the values of these terminals. The values and the choices for the terminals in Fig. 11 are as follows.

For the node `T=yes, R = ns`, the optimal choice for `D` is `no`; the value for the node is -10, the cost of the test. For the node `T=yes, R = os`, the optimal choice for `D` is `yes`; the the value for the node is 52.5. For the node `T=yes, R = cs`, the optimal choice for `D` is `yes`; the value for the node is 97.5. For the node `T=no, R=nobs`, the optimal choice for `D` is `yes`; the value for the node is 40.

The decision graph has two solution graphs as shown in Fig. 12-(a) and Fig. 12-(b) respectively. The solution graph in Fig. 12-(a) corresponds to the strategy of no test and drill. The expected value of the influence diagram *w.r.t.* the strategy is 40. The solution graph in Fig. 12-(b) corresponds to the strategy of test and drill unless the test result is no structure. The expected cost of the influence diagram *w.r.t.* the strategy is 37.675. Thus, the optimal expected value of the influence diagram is 40, and the optimal strategy is not to test and drill.

It seems that, when compared to the algorithms in (Zhang and Poole 1992b, Shachter and Peot 1992), the above procedure involves some extra work in steps (1), (3) and (4) for processing decision graphs. We argue that the extra effort will be paid back. Our argument is as follows. First, we note that the procedure can be improved by combining the first four steps into one step, and generating the decision graph starting from the root. This way, those arcs with zero probability will not be included in the graph at the first place. Consequently, those parts that will be deleted by steps (3) and (4) need not be generated either. Thus the overhead can be reduced. Second, the values of the nodes and the probabilities of the arcs in the parts deleted by steps (3) and (4) need not be computed at all. If the asymmetry involved in the decision problem is substantial, the deleted part can constitute quite a large portion of the total decision graph, and this may mean a big saving. Furthermore, the decision graph representation generated by this

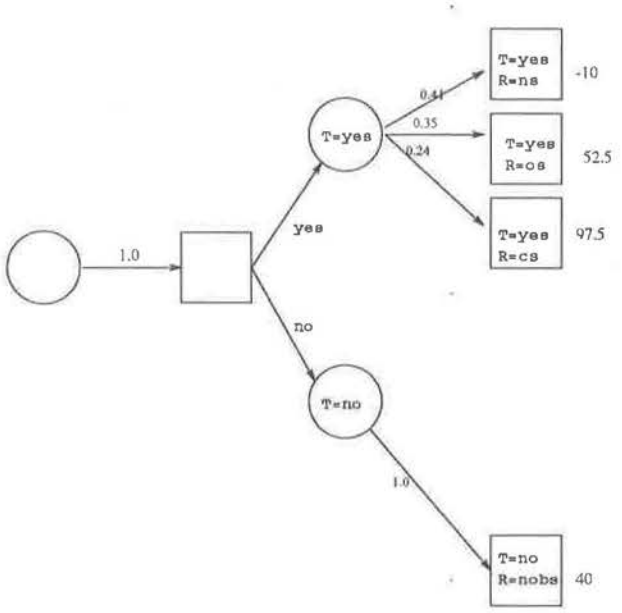


Figure 11: A simplified decision graph

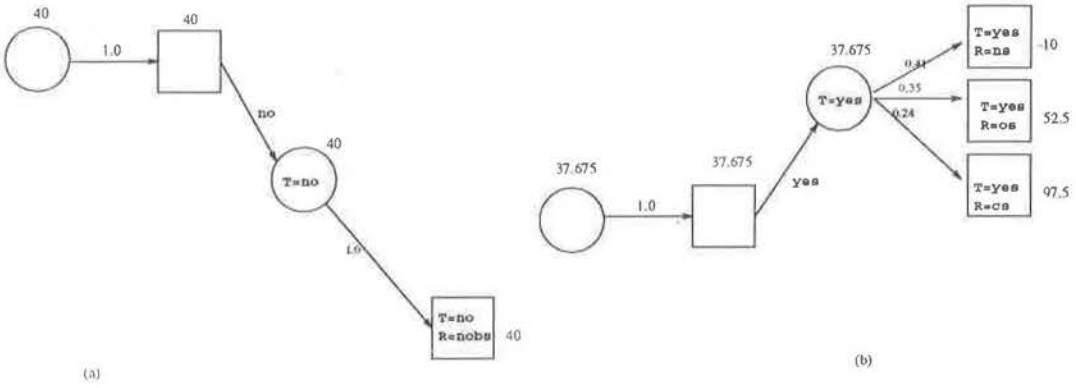


Figure 12: The two solution graphs for the oil wildcatter problem

procedure can be used for efficiently computing the value of perfect information. This possibility is explored in (Zhang *et al.* 1993b).

5.5.2 Using heuristic algorithms

The computation method just described is successful in recovering the asymmetric property of the original problem. Better performance can be achieved if we use the algorithms developed in (Qi 1993, Qi and Poole 1993) for decision graph search. By using these algorithms, some subgraphs may not be processed at all, due to pruning.

To use these algorithms, we need a domain dependent function that can give admissible estimation on $u_1(s)$ for any situation s. Note that the admissibility of a heuristic function here is different from the one in the traditional case. Because we are maximizing merits instead of minimizing costs, we define a heuristic function to be admissible if it never under–estimates for any context s. Formally, a function h is admissible if $h(s) \geq u_1(s)$ for any situation s.

5.6 A comparison with Howard and Matheson's method

The relationship between our method and Howard and Matheson's method should now be clear. In both methods, an influence diagram is first transformed into a decision tree from which an optimal strategy is computed. However, there are a few notable differences between the two methods.

First, Howard and Matheson's method works only for no–forgetting influence diagrams while ours is applicable to stepwise decomposable influence diagrams.

Second, the size of the decision trees generated by the two methods are different. For a given influence diagram, the depth of the decision tree obtained by Howard and Matheson's method is equal to the number of the variables in the influence diagram, while the depth of the decision tree obtained by our method is $2n$, where n is the number of the decision variables in the influence diagram. Typically, there are more random variables than decision variables in a decision problem, thus the depth of a decision tree obtained by Howard and Matheson's method from an influence diagram is larger than the depth of a decision tree obtained by our method for the same influence diagram. Furthermore, the number of the nodes in the decision tree obtained by Howard and Matheson's method from an influence diagram is exponential in the depth of the tree, but this is not necessarily true for the decision tree obtained by our method. In fact, the number of nodes in a decision tree obtained by our method is:

$$1 + |\Omega_{\pi(d_n)}| + \sum_{i=1}^{n-1}(|\Omega_{\pi(d_i)}| + |\Omega_{\pi(d_i)}| * |\Omega_{d_i}|).$$

Finally, our method provides a clear interface to those algorithms developed for Bayesian net evaluation.

6 Extension to Influence Diagrams with Multiple Value Nodes

In the previous section, we developed a method for influence diagram evaluation. We have assumed that the concerned influence diagrams are regular and have only one value node. As pointed out

in (Tatman and Shachter 1990), if the value function of an influence diagram is separable, then the separable nature can be exploited for increasing the efficiency of influence diagram evaluation. The separability of a value function can be represented by multiple value nodes. Thus, it is desirable to develop algorithms that can be used to evaluate influence diagrams with multiple value nodes.

An generalization of Shacter's algorithm has been developed in (Tatman and Shachter 1990) that can exploit the separability of value functions. Zhang and Poole's algorithm (Zhang and Poole 1992b) is actually developed for influence diagrams with multiple value nodes. In this section, we generalize the method we presented in the previous section so that it can be applicable to regular influence diagrams with multiple value nodes as well.

6.1 Separable value functions

If the value function of the value node in an influence diagram can be expressed as the sum of two or more functions with fewer variables, we say the value function is *separable*. More precisely, let $f(X)$ be a value function with variable set X, let f_1 ..., f_q are functions with variable sets X_1 ... X_q respectively, and X_1 ... X_q are all proper subsets of X. V is separable if

$$f(X) = \sum_{i=1}^{q} f_i(X_i).$$

Consider again the oil wildcatter problem. The value node depends on four variables: T, D, O and CD. The value function can be separated into three parts: a function f_1 on the cost of the seismic structure test, a function f_2, on the drill cost, and a function f_3 on the value of the oil. Formally, this can be expressed as

$$f(\mathtt{T},\ \mathtt{D},\ \mathtt{O},\ \mathtt{CD}) = f_1(\mathtt{T}) + f_2(\mathtt{D},\ \mathtt{CD}) + f_3(\mathtt{D},\ \mathtt{O}).$$

With this separation of the value function, the value node can be *split* into three value nodes, v_1, v_2 and v_3, with f_1, f_2 and f_3 as their value functions respectively. This separation results in a new influence diagram as shown in Fig. 13.

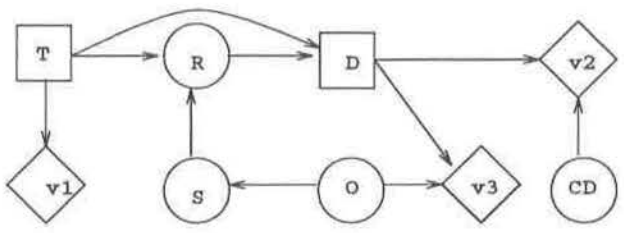


Figure 13: A new representation of the oil wildcatter problem by an influence diagram with multiple value nodes

The semantics of influence diagrams with multiple value nodes is the same as that of influence diagrams with single value node except that, for an influence diagram with multiple value nodes,

we interpret the sum of the values of all individual value nodes as the value of the influence diagram. Therefore, all the terminologies we have used before can be used here for influence diagrams with multiple value nodes. Furthermore, the expected value and the optimal strategies of a influence diagram with multiple value nodes can be defined in a similar way as follows. Let $U = \{v_1, ..., v_q\}$ be the set of the value nodes of influence diagram I. Let Δ be a strategy for I. Let P_Δ denote the joint probability distribution determined by the Bayesian net I_Δ obtained from I and the strategy Δ.

Let $P_\Delta\{\pi(v_i) = x\}$, for any $x \in \Omega_{v_i}$, denote the probability for the event $\pi(v_i) = x$, in the Bayesian net I_Δ. The expected value of the influence diagram *w.r.t.* strategy Δ, and the value node v^i, written $E_\Delta[v_i]$, is given by

$$E_\Delta[v_i] = \sum_{x \in \Omega_{\pi(v_i)}} f_i(x) * P_\Delta\{\pi(v_i) = x\} \tag{9}$$

where f_i is the value function of the value node v_i. The expected value of the influence diagram *w.r.t.* strategy Δ, written E_Δ, is given by

$$E_\Delta = \sum_{i=1}^{q} E_\Delta[v_i].$$

The decision objective is to find an optimal strategy maximizing the expected value. i.e. to find Δ^0 such that

$$E_{\Delta^0} = max\{E_\Delta | \ \Delta \text{ is a strategy}\} \tag{10}$$

The computational problem related to an influence diagram is to compute the optimal expected value and an optimal strategy for the influence diagram.

6.2 Decision graphs for influence diagrams with multiple value nodes.

Like the case for influence diagrams with single value node, the computational structure of the optimal expected value of an influence diagram with multiple value nodes can also be represented as a decision graph. The difference is that in the case for influence diagrams with single value node, the values are associated only with the terminals in the decision graph while in the case for influence diagrams with multiple value nodes, values can be associated with interior nodes as well. In order to illustrate this, let us consider the influence diagram shown in Fig. 13, which is an influence diagram with three value nodes for the oil wildcatter problem. Since the value node v_1 depends on only node `T`, its value can be determined in any situation where the variable `T` is instantiated. Thus, at the nodes representing the situations `T = yes` and `T = no`, the value of v_1 can be determined. In particular, the value of f_1 for the situation `T = yes` is -10 and the value of f_1 for the situation `T = no` is 0. The value -10 can be viewed as the value resulting directly from performing the test action in the parent situation, while the value 0 can be viewed as the value resulting directly from performing the no-test action in the parent situation.

In order to deal with general cases, we introduce a new concept. Let $\pi(d_k) = x$ be a parent situation of d_k, $\pi(d_k) = x, d_k = a$, be an inclusive situation of d_k and let $I(d_k, d_{k+1})$ be the section of I from d_k to d_{k+1}. Without loss of generality, suppose nodes $v_i, ..., v_j$ are the value

nodes in $I(d_k, d_{k+1})$. For $i \leq l \leq j$, let $E_{I(d_k,d_{k+1})}[v_l|\pi(d_k) = x, d_k = a]$ denote the expected value of the value node v_l, conditioned on $\pi(d_k) = x, d_k = a$, in the section $I(d_k, d_{k+1})$. We call the sum

$$\sum_{l=i}^{j} E_{I(d_k,d_{k+1})}[v_l|\pi(d_k) = x, d_k = a]$$

the *value of the inclusive situation* $\pi(d_k) = x, d_k = a$. This value can be computed more efficiently by a method in (Zhang *et al.* 1993b). Intuitively, the value can be viewed as the utility directly resulting from selecting a as the choice for decision node d_k in the parent situation $\pi(d_k) = x$. Using the terminologies for decision graphs, the value can be associated with the arc from the choice node representing the parent situation $\pi(d_k) = x$ to the chance node representing to the inclusive situation $\pi(d_k) = x, d_k = a$. The following is a new specification of the decision graph of an influence diagram with multiple value nodes.

- The empty situation is the root, which is a chance node, of the decision graph.
- For each decision node d_i, $1 \leq i < n$, and each $x \in \Omega_{\pi(d_i)}$, there is a choice node in the decision graph representing the parent situation $\pi(d_i) = x$; for each $a \in \Omega_{d_i}$, there is a chance node in the decision graph representing the inclusive situation $\pi(d_i) = x, d_i = a$.
- Let s be a choice node representing a parent situation $\pi(d_i) = x$, $1 \leq i < n$, the nodes representing the inclusive situations $\pi(d_i) = x, d_i = a$ for all $a \in \Omega_{d_i}$ constitute the children of node s. The arc from the node representing the parent situation to a child representing an inclusive situation $\pi(d_i) = x, d_i = a$ is labeled with the value of the inclusive situation.
- For each $x \in \Omega_{d_n}$, there is a terminal node representing the parent situation $\pi(d_n) = x$. The value of the terminal is $V_n(x)$.
- Let $\mathcal{S}_{i+1}$ denote the set of choice nodes representing the parent situations of d_{i+1}. For each chance node representing an inclusive situation $\pi(d_i) = x, d_i = a$, it has the nodes in $\mathcal{S}_{i+1}$ as its children. In other words, for each $x \in \Omega_{\pi(d_i)}$, $a \in \Omega_{d_i}$ and $y \in \Omega_{\pi(d_{i+1})}$ there is an arc from the chance node representing the inclusive situation $\pi(d_i) = x, d_i = a$ to a choice node representing the parent situation $\pi(d_{i+1}) = y$. The arc is labeled by the probability $P\{\pi(d_{i+1}) = y|\pi(d_i) = x, d_i = a\}$.
- For each choice node representing a parent situation of d_1 in form of $\pi(d_1) = x$, there is an arc from the root node to the choice node. On the the arc is a probability $P\{\pi(d_1) = x\}$.

As an example, consider the influence diagram as shown in Fig. 13. A decision graph for the influence diagram is shown in Fig. 14 (all arcs with zero probability are removed).

Let DG be a decision graph derived from an influence diagram with multiple value nodes, we can define an evaluation function, u_2, on it as follows:

- If s is a terminal, corresponding to a situation $\pi(d_k) = x$, then

$$u_2(DG, s) = V_n(x)$$

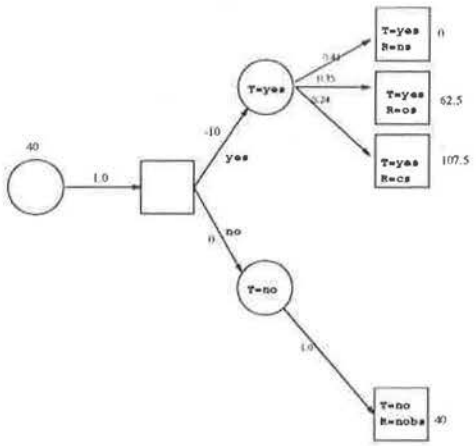


Figure 14: A decision graph for the influence diagram shown in Fig. 13

- If s is a chance node, with children $s_1, ..., s_l$, then

$$u_2(DG, s) = \sum_{i=1}^{l} P_i * u_2(DG, s_i)$$

  where P_i is the probability on the arc from node s to node s_i.

- If s is a choice node, with children $s_1, ..., s_l$,then

$$u_2(DG, s) = max_{i=1}^{l}\{c(s, s_i) + u_2(DG, s_i)\}.$$

The reader may have noticed that the decision graphs corresponding to the examples we have considered so far are actually decision trees. This need not be true in general. Here we consider another example whose decision graph has shared structure.

Consider the following variation of the oil wildcatter problem. In the previous examples, we implicitly assumed that the amount of oil the oil wildcatter can obtain is equal to the amount of the oil underground. Now we replace this assumption by a more realistic one, namely, the amount of oil the oil wildcatter can obtain depends on the amount of the oil underground and the equipment status as well. Thus, the oil wildcatter needs also to decide whether to upgrade his equipment. Furthermore, suppose the profit by selling oil also depends on market information and the sale policy. This more elaborated problem can be represented by the influence diagram as shown in Fig. 15.

Suppose the amount of obtained oil can be either zero, or low, or medium or high. The decision graph corresponding to the problem is shown in Fig. 16. Suppose the decision problem is asymmetric in the following sense: if the drill decision is yes, then the amount of obtained oil must not be zero and if the drill decision is no, the amount of obtained oil must be zero. Therefore, Some of the arcs to the nodes representing the situations of `oil-obtained` are labeled with zero probability. After removing these zero-probability arcs, the decision graph becomes the one in Fig. 17, which is indeed a graph (not a tree).

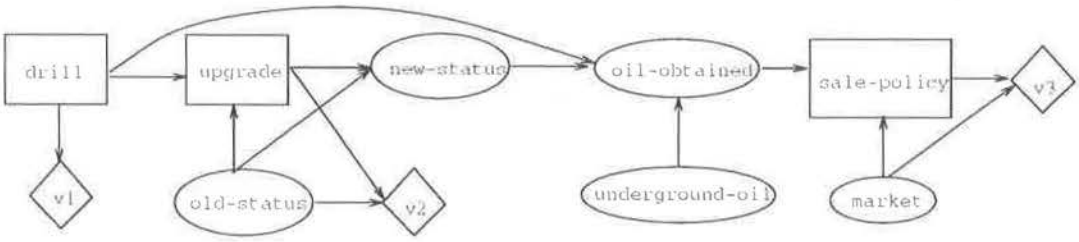


Figure 15: A more elaborated decision problem

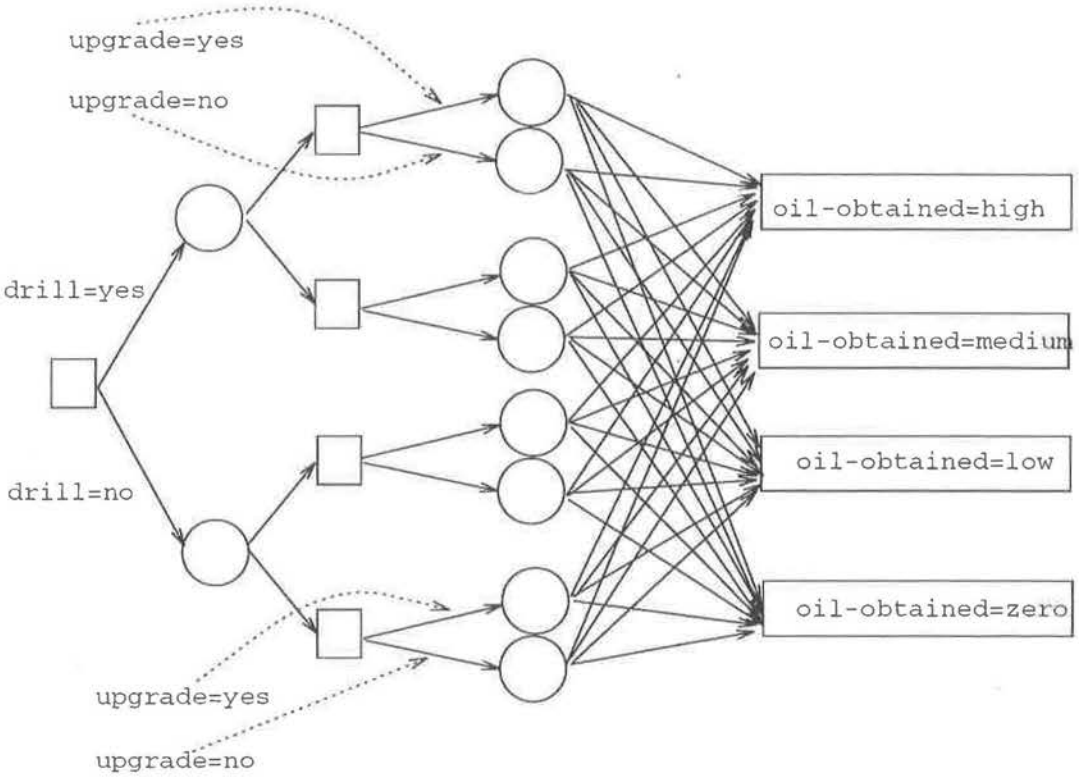


Figure 16: A decision graph for the influence diagram in Fig. 15

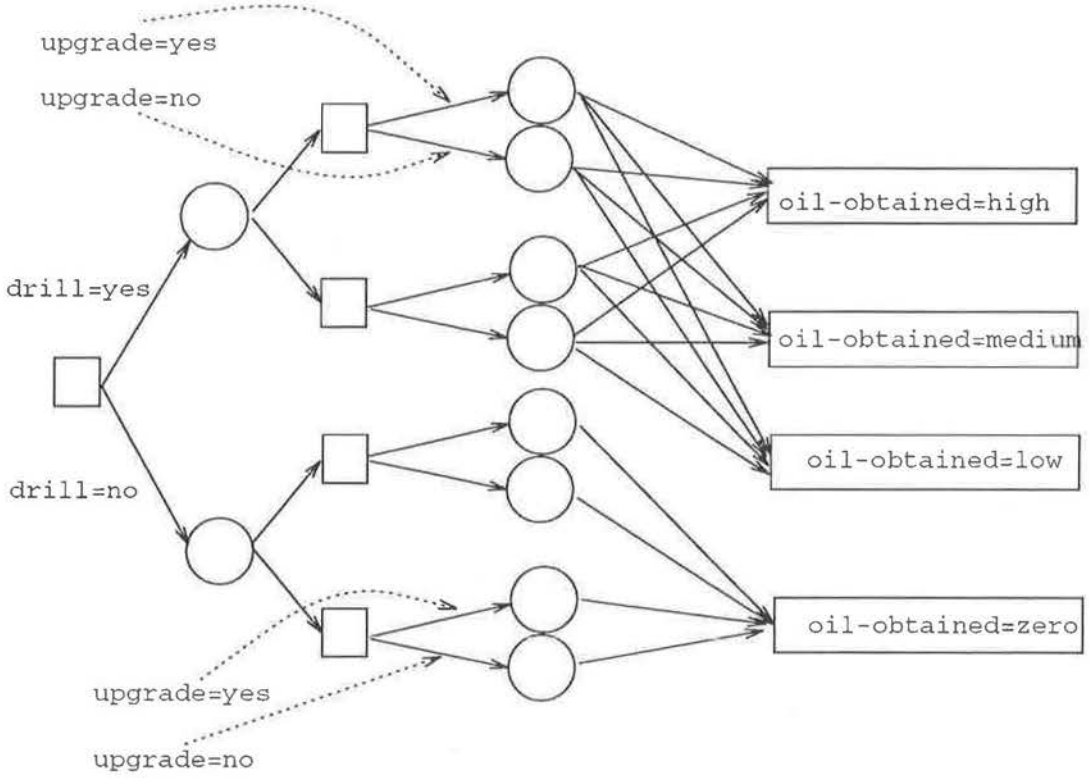


Figure 17: A decision graph, with zero probability arcs removed, for the influence diagram in Fig. 15

A possible heuristic function for this problem can be defined as follows: For any node s, if s represents a situation in which the drill decision is no, return zero, otherwise, return M where M is a large integer. Obviously, if M is large enough, this heuristic function is admissible. Suppose we use algorithm A1 to search the decision graph in Fig. 17. If the algorithm uses the heuristic function, and searches the branch corresponding to `drill = yes` first, then the subgraph under the branch corresponding to `drill = no` could be pruned altogether, provided that, according to the actual numerical setting, it is profitable to drill. Furthermore, if we have a stronger heuristic function such that it asserts that the optimal expected value for the node representing the situation `drill = no` *equals* zero, then lower half of the decision graph in Fig. 17 need not to be expanded at all. When one builds a decision tree for the decision problem, exactly the same heuristic information is used so that the branch corresponding to `drill = no` will not be expanded.

7 Summary

In this paper, we have presented a new method for influence diagram evaluation. The basic idea of the method is to transform an influence diagram into a decision graph in such a way that the optimal strategies of the influence diagram correspond to the the optimal solution graphs of the decision graphs. In this aspect, our method is similar to Howard and Metheson's two–phase method. However, our method is more efficient than theirs.

To the best of our knowledge, our method is the only one enjoying all of the following merits simultaneously.

(1). It is applicable to a class of influence diagrams that are more general than the class of no–forgetting influence diagrams.

(2). It provides an interface to the algorithms developed for Bayesian net evaluation.

(3) It can make use of heuristic search techniques and domain dependet knowledge.

(4) It can take the advantage of asymmetry in decision problems.

Acknowledgement The research reported in this paper is partially supported under NSERC grant OGPOO44121 and Project B5 of the Institute for Robotics and Intelligent Systems. The authors wish to thank Craig Boutilier, Andrew Csinger, Mike Horsch, Keiji Kanazawa, (Nevin) Lianwen Zhang and Ying Zhang for their valuable comments on this paper.

References

[Bellman, 1957] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.

[Cooper, 1988] G. F. Cooper. A method for using belief networks as influence diagrams. In *Proc. of the Fourth Conference on Uncertainty in Artificial Intelligence*, pages 55–63, Univ. of Minnesota, Minneapolis, USA, 1988.

[Fung and Shachter, 1990] R. M. Fung and R. D. Shachter. Contingent influence diagrams, 1990.

[Howard and Matheson, 1984] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *The Principles and Applications of Decision Analysis, VolumII*, pages 720–761. Strategic Decision Group, Mento Park, CA., 1984.

[Jensen *et al.*, 1990] F. V. Jensen, K. G. Olesen, and K. Anderson. An algebra of Bayesian belief universes for knowledge based system. *Networks*, 20:637–659, 1990.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Statist. Soc. Ser. B*, 50:157–224, 1988.

[Matheson, 1990] J. E. Matheson. Using influence diagrams to value information and control. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Nets and Decision Analysis*, pages 25–63. John Wiley and Sons, 1990.

[Miller *et al.*, 1976] A . C. Miller, M. M. Merkhofer, R. A. Howard, J. E. Matheson, and T. T. Rice. Development of automated aids for decision analysis. Technical report, Stanford Research Institute, 1976.

[Ndilikilikesha, 1991] P. Ndilikilikesha. Potential influence diagrams. Technical report, Business School, University of Kansas, 1991. Working paper No. 235.

[Nilsson, 1982] Nils J. Nilsson. *Principles of Artificial Intelligence.* Springer-Verlag Berlin Heidelberg New York, 1982.

[Olmsted, 1983] S. M. Olmsted. *On representing and Solving Decision Problems.* PhD thesis, Engineering Economics Department, Stanford University, 1983.

[Pearl, 1984] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Publishing Company, 1984.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, Los Altos, CA, 1988.

[Qi and Poole, 1993] R. Qi and D. Poole. Decision graph search. *submitted to a journal, also available as a technique report 93-9, Department of Computer Science, UBC*, 1993.

[Qi, 1993] R. Qi. *Decision Graphs: algorithms and applications.* PhD thesis, Department of Computer Science, University of British Columbia, 1993. Forthcoming.

[Raiffa, 1968] H. Raiffa. *Decision Analysis.* Addison–Wesley Publishing Company, 1968.

[Ross, 1983] S. M. Ross. *Introduction to Stochastic Dynamic Programming.* Academic Press, 1983.

[Shachter and Peot, 1992] R. D. Shachter and M. A. Peot. Decision making using probabilistic inference methods. In *Proc. of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 276–283, San Jose, CA., USA, 1992.

[Shachter *et al.*, 1990] R. D. Shachter, B. D'Ambrosio, and B. A. Del Favero. Symbolic probabilistic inference in belief networks. In *Proc. of AAAI-90*, pages 126–131, 1990.

[Shachter, 1986] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.

[Shachter, 1988] R. D. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 36:589–605, 1988.

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

[Shenoy, 1990] P. P. Shenoy. Valuation–based systems for Bayesian decision analysis. Technical Report working paper No. 220, School of Business, University of Kansas, April 1990.

[Shenoy, 1991] P. P. Shenoy. A fusion algorithm for solving bayesian decision problems. In B. D. D'Ambrosio, P. Smet, and P. P. Bonissone, editors, *Proc. of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 361–369, UCLA, Los Angeles, USA, 1991. Morgan Kaufmann.

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

[Zhang and Poole, 1992a] L. Zhang and D. Poole. Sidestepping the triangulation problem. In *Proc. of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 360–367, Stanford University, Standford, USA, 1992.

[Zhang and Poole, 1992b] L. Zhang and D. Poole. Stepwise decomposable influence diagrams. In B. Nebel, C. Rich, and W. Swartout, editors, *Proc. of the Fourth International Conference on Knowledge Representation and Reasoning*, pages 141–152, Cambridge, Mass., USA, Oct. 1992. Morgan Kaufmann.

[Zhang *et al.*, 1993a] L. Zhang, R. Qi, and D. Poole. A comoutational theory of decision networks. *submitted to a journal, also available as a technique report 93-6, Department of Computer Science, UBC*, 1993.

[Zhang *et al.*, 1993b] L. Zhang, R. Qi, and D. Poole. Incremental computation of the value of perfect information in stepwise-decomposable influence diagrams. In *Proc. of the Ninth Conference on Uncertainty in tificial Intelligence, to appear*, 1993.

[Zhang *et al.*, 1993c] L. Zhang, R. Qi, and D. Poole. Minimizing decision tables in stepwise-decomposable influence diagrams. In *Proc. of the Fourth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, Florida, USA, Jan. 1993.

[Zhang, 1993] L. Zhang. *A Computational Theory of Decision Network*. PhD thesis, Department of Computer Science, University of British Columbia, 1993. in preparation.