Computational Intelligence, Volume 11, Number 1, 1995

A NEW METHOD FOR INFLUENCE DIAGRAM EVALUATION

RUNPING QI AND DAVID POOLE[1]

Department of Computer Science, University of British Columbia, Vancouver B. C. Canada

As influence diagrams become a popular representational tool for decision analysis, influence diagram evaluation attracts more and more research interest. 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 because, among other reasons, our method generates a much smaller decision graph for the same influence diagram. Like those most recent algorithms reported in the literature, our method also provides a clean interface between influence diagram evaluation and Bayesian net evaluation. Consequently, various well–established algorithms for Bayesian net evaluation can be used in influence diagram evaluation. Furthermore, our method has a few unique merits. First, it takes advantage of asymmetry in influence diagrams to avoid unnecessary computation. 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.

Key words: influence diagrams, influence diagram evaluation, asymmetric decision problems, Bayesian net evaluation, decisin trees (graphs).

1. INTRODUCTION

Decision trees were used as a simple tool both for problem modeling and optimal policy computation in the early days of decision analysis (Raiffa 1968). A decision tree explicitly depicts all scenarios of a problem and specifies the "utility" the agent can get in each scenario. An optimal policy for the decision problem can be computed from the decision tree representation of the problem by a simple "average–out–and–fold–back" method (Raiffa 1968, Smith 1988).

Though conceptually simple, decision trees have a number of drawbacks. First, the dependency/independency relationships among the variables in a decision problem cannot be represented in a decision tree. Second, a decision tree specifies a particular order for the assessment on the probability distributions of the random variables in the decision problem. This order is in most cases not a natural assessment order. Third, the size of a decision tree for a decision problem is exponential in the number of variables of the decision problem. Finally, a decision tree is not easily adaptable to changes in a decision problem. If a slight change is made in a problem, one may have to draw a decision tree anew.

Influence diagrams were proposed as an alternative to decision trees for decision analysis (Howard and Matheson 1984, Miller *et al.* 1976). As a representation framework, influence diagrams do not have the aforementioned drawbacks of decision trees. The influence diagram representation is expressive enough to explicitly describe the dependency/independency relationships among the variables in the decision problem; it allows a more natural assessment order on the probabilities of the random variables; it is compact; and it is easy to adapt to the changes in the problem.

The first method for influence diagram evaluation was proposed by Howard and Matheson (1984). In this method, an influence diagram is first transformed into a decision tree and then an optimal policy is computed from the decision tree. After

[1]Scholar, Canadian Institute for Advanced Research

Shachter published his algorithm that evaluates influence diagrams directly (Shachter 1986), the two–phase approach was largely abandoned. None of the recent algorithms (Cooper 1988, Shachter and Peot 1992, Zhang *at al.* 1993a, Zhang and Poole 1992b) needs 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 in handling asymmetric decision problems (Covaliu and Oliver 1992, Fung and Shachter 1990, Phillips 1990, Shachter 1986, Smith *et al.* 1993). Decision problems are usually asymmetric in the sense that the set of possible outcomes of a random variable may vary depending on different conditioning states, and the set of legitimate alternatives of a decision variable may vary depending on different information states. To be represented as an influence diagram, an asymmetric decision problem must be "symmetrized" by adding artificial states and assuming degenerate probability distributions (Smith *et al.* 1993). This symmetrization results in two problems. First, the number of information states of decision variables are increased. Among the information states of a decision variable, many are "impossible" to reach (having zero probability). The optimal choices for these states need not be computed at all. However, conventional influence diagram evaluation algorithms (Shachter 1986, Shachter and Peot 1992, Smith *et al.* 1993, Zhang *at al.* 1993a, Zhang and Poole 1992b) cannot avoid such computations. Second, for an information state of a decision variable, some of the alternatives of the decision variable may not be legitimate, thus they need not be considered at all when computing an optimal choice for the variable in the information state. However, conventional influence diagram algorithms have to consider all of the alternatives in order to compute an optimal choice for a decision variable in any of its information states (including those impossible states). Thus, it is evident that conventional influence diagram evaluation algorithms involve unnecessary computation.

In this paper, we present an approach for overcoming the aforementioned disadvantages. Our approach consists of two independent components: a simple extension to influence diagrams and a two–phase method for influence diagram evaluation. Our extension allows explicitly expressing the fact that some decision variable have different frames in different information states. Our method, similar to Howard and Matheson's, evaluates an influence diagram in two conceptual steps: it first maps an influence diagram into a decision graph (Qi 1994, Qi and Poole 1993) in such a way that an optimal solution graph of the decision graph corresponds to an optimal policy of the influence diagram. Thus the problem of computing an optimal policy is reduced to the problem of searching for an optimal solution graph of a decision graph, which can be accomplished by various algorithms (Qi 1994, 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 (Shachter and Peot 1992, Zhang 1994, Zhang *et al.* 1993a, Zhang and Poole 1992b), our method provides a clean interface between influence diagram evaluation and Bayesian net evaluation so that 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 (1992b), which rivals existing algorithms in terms of efficiency (Zhang 1994). Furthermore, our method has a few additional merits. First, it avoids computing decision choices for decision variables in impossible states. We will show that avoiding such computation may lead to an exponential factor of savings for typical decision problems. 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 *at al.* 1993b). This method works for influence diagrams with or without our extension.

The rest of this paper is organized as follows. The next section introduces influence diagrams. Section 3 reviews the existing algorithms and illustrates their disadvantages. In Section 4, we present our approach for overcoming these disadvantages. Section 5 gives an analysis on how much can be saved by exploiting asymmetry in decision problems. Section 6 discusses related work in dealing with asymmetric decision problems. Section 7 concludes the paper.

2. INFLUENCE DIAGRAMS AND INFLUENCE DIAGRAM EVALUATION

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 dictated by 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/she must choose a value for the decision variable.

The following definition for influence diagrams is borrowed from Zhang *et al.* 1993a). An influence diagram $\mathcal{I}$ is defined as a quadruple $\mathcal{I} = (X, A, \mathcal{P}, \mathcal{U})$ where

- (X, A) is a directed acyclic graph with node set X and arc set A. The node set X is partitioned into a random node set C, a decision node set D and a value node set U. All the nodes in U have no children.
  Each decision node or random node has a set, called the *frame*, associated with it. The frame of a node consists of all the possible outcomes of the (decision or random) variable denoted by the node. For any node $x \in X$, we use $\pi(x)$ to denote the parent set of node x in the graph and use Ω_x to denote the frame of node x. For any subset $J \subseteq C \cup D$, we use Ω_J to denote the Cartesian product $\prod_{x \in J} \Omega_x$.
- $\mathcal{P}$ is a set of probability distributions $P\{c|\pi(c)\}$ for all $c \in C$. For each $o \in \Omega_c$ and $s \in \Omega_{\pi(c)}$, the distribution specifies the conditional probability of event $c = o$, given[2] $\pi(c) = s$.
- $\mathcal{U}$ is a set $\{g_v : \Omega_{\pi(v)} \rightarrow \mathcal{R} | v \in U\}$ of *value functions* for the value nodes, where $\mathcal{R}$ is the set of the real.

For a decision node d_i, a value $x \in \Omega_{\pi(d_i)}$ is called an information state of d_i, and a mapping $\delta_i : \Omega_{\pi_{d_i}} \rightarrow \Omega_{d_i}$ is called a *decision function* for d_i. The set of all the decision functions for d_i, denoted by Δ_i, is called the *decision function space* for

[2]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 that assign an element of e to the corresponding variable in J.

d_i. Let $D = \{d_1, ..., d_n\}$ be the set of decision nodes in influence diagram $\mathcal{I}$. The Cartesian product $\Delta = \prod_{i=1}^{n} \Delta_i$ is called the *policy space* of $\mathcal{I}$.

Given a policy $\delta = (\delta_1, \ldots, \delta_k) \in \Delta$ for $\mathcal{I}$, a probability P_δ can be defined over the random nodes and the decision nodes as follows:

$$P_\delta\{C, D\} = \prod_{c \in C} P\{c|\pi(c)\} \prod_{i=1}^{k} P_{\delta_i}\{d_i|\pi(d_i)\} \tag{1}$$

where $P\{c|\pi(c)\}$ is given in the specification of the influence diagram, while $P_{\delta_i}\{d_i|\pi(d_i)\}$ is given by δ_i as follows:

$$P_{\delta_i}\{d_i|\pi(d_i)\} = \begin{cases} 1 & \text{if } \delta_i(\pi(d_i)) = d_i, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For any value node v, $\pi(v)$ must consist of only decision and random nodes, since value nodes do not have children. Hence, we can talk about marginal probabilities $P_\delta\{\pi(v)\}$. The *expectation of the value node v under policy δ*, denoted by $E_\delta[v]$, is defined as follows:

$$E_\delta[v] = \sum_{\pi(v)} P_\delta\{\pi(v)\} g_v(\pi(v)).$$

The summation $E_\delta = \sum_{v \in U} E_\delta[v]$ is called the *expected value* of $\mathcal{I}$ under the policy δ. The maximum of E_δ over all the possible policies is the *optimal expected value* of $\mathcal{I}$. An *optimal policy* is a policy that achieves the optimal expected value. To *evaluate* an influence diagram is to determine its optimal expected value and to find an optimal policy.

2.1. No–forgetting and stepwise decomposable influence diagrams

In this section, we introduce two classes of influence diagrams, which are the focus of the literature.

An influence diagram is *regular* (Howard and Matheson 1984, Shachter 1986) if there is a directed 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. An influence diagram is "no–forgetting" if each decision node d and its parents are also parents of those decision nodes that are descendants of d (Howard and Matheson 1984, Shachter 1986). Intuitively, the "no–forgetting" property means that a decision maker remembers all the information that was earlier available to him/her and remembers all the previous decisions he/she made. The lack of an arc from a node a to a decision node d in a no–forgetting influence diagram 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 called *stepwise solvable* (Zhang *et al.* 1993a, Zhang and Poole 1992b) if its optimal policy can be computed by considering one decision node at a time. A necessary and sufficient syntactic condition for the stepwise solvability of influence diagrams, called *stepwise–decomposability*, is provided in (Zhang 1994, Zhang *et al.* 1993a).

Stepwise–decomposability is defined in terms of graph separation. Informally, an influence diagram is *stepwise decomposable* if the parents of each decision node divide

the influence diagram into two parts. In order to define the property formally, we need some notation and concepts.

Let $nond(x)$ denote the set of nodes that are not descendants of x in the influence diagram. Thus, $nond(d) \cap D$ is the set of decision nodes that are not descendants 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* (Lauritzen and Spiegelhalter 1988) 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 $x \rightarrow z$ and $y \rightarrow z$ in G and $x \neq y$. 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 the set Z. Because the "m–separated" relation is symmetric, we sometimes simply say that two nodes x and y are m–separated by Z if x is m–separated from y by Z. Two sets X and Y are m–separated by set Z if x and y are m–separated by set Z for each $x \in X$ and each $y \in Y$.

Let d be a decision node in G, let $m(G)$ be the moral graph of G and let 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 that 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. The upstream X_d and the downstream Y_d of d are m–separated by $\pi(d)$. This property is important for influence diagrams because m–separation implies conditional independence.

An influence diagram is *stepwise decomposable* if, for each decision node d and for any node $x \in \pi^*(nond(d) \cap D)$, the following holds: $\{x\} \cup \pi(x) \subseteq X_d \cup \pi(d)$. This definition implies that for each decision node d and any node $x \in \pi^*(nond(d) \cap D)$, $\{x\} \cup \pi(x)$ and Y_d are m–separated by $\pi(d)$.

In a stepwise decomposable influence diagram, an arc into a decision node indicates both information availability and functional dependence. More precisely, for any decision node d and any other node x in a stepwise decomposable influence diagram, the presence of an arc $x \rightarrow d$ implies that the value of variable x 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 $x \rightarrow d$ in a stepwise decomposable influence diagram implies that either the value of variable x 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.

Stepwise decomposable influence diagrams have a number of advantages over no–forgetting influence diagrams (Zhang 1994). First, any no–forgetting influence diagram is stepwise decomposable, thus, stepwise decomposability is more general than no–forgetting. Second, stepwise decomposability allows representing the knowledge that a piece of information (carried by a no–forgetting informational arc) is irrelevant to the optimal decision function of the decision. Third, stepwise decomposability allows to model problems where the decision maker may "forget" some information (Zhang 1994). Finally, all irrelevant informational arcs can be automatically detected and unharmfully removed from an influence diagram (Zhang 1994, Zhang *et al.* 1993c) so that it opens a possibility of simplifying influence diagrams in a preprocessing step.

The results presented in this paper are applicable to regular *stepwise decomposable influence diagrams* (Qi 1994), but we shall limit the exposition only to regular influence diagrams with a single value node for simplicity.

2.2. An example — the used car buyer problem

In order to illustrate how to use influence diagrams to represent decision problems, let us consider the used car buyer problem (Howard 1984). Joe is considering buying a used car. The marked price is \$1000, while a three year old car of this model is worth \$1100, if it has no defect. Joe is uncertain whether the car is a "peach" or a "lemon". But Joe knows that, of the ten major subsystems in the car, a peach has a defect in only one subsystem whereas a lemon has a defect in six subsystems. Joe also knows that the probability for the used car being a peach is 0.8 and the probability for the car being a lemon is 0.2. Finally, Joe knows that it will cost him \$40 to repair one defect and \$200 to repair six defects.

Observing Joe's concern about the possibility that the car may be a lemon, the dealer offers an "anti–lemon guarantee" option. For an additional \$60, the anti–lemon guarantee will cover the full repair cost if the car is a lemon, and cover half of the repair cost otherwise. At the same time, a mechanic suggests that some mechanical examination should help Joe determine the car's condition. In particular, the mechanic gives Joe three alternatives: test the steering subsystem alone at a cost of \$9; test the fuel and electrical subsystems at a total cost of \$13; a two–test sequence in which, the transmission subsystem will be tested at a cost of \$10, and after knowing the test result, Joe can decide whether to test the differential subsystem at an additional cost of \$4. All tests are guaranteed to detect a defect if one exists in the subsystem(s) being tested.

An influence diagram for the used car problem is shown in Fig. 1. The random variable `CC` represents the car's condition. The frame for `CC` has two elements: `peach` and `lemon`. The variable has no parent in the graph, thus, we specify its prior probability distribution in Table 1.

The decision variable $\mathtt{T}_1$ represents the first test decision. The frame for $\mathtt{T}_1$ has four elements: `nt, st, f&e` and `tr`, representing respectively the options of performing no test, testing the steering subsystem alone, testing the fuel and electrical subsystems, and testing the transmission subsystem with a possibility of testing the differential subsystem next.

The random variable $\mathtt{R}_1$ represents the first test results. The frame for $\mathtt{R}_1$ has four elements: `nr, zero, one` and `two`, representing respectively the four possible outcomes of the first test: no result, no defect, one defect and two defects. The probability distribution of the variables, conditioned on $\mathtt{T}_1$ and `CC`, is given in Table 2.

The decision variable $\mathtt{T}_2$ represents the second test decision. The frame for $\mathtt{T}_2$ has two elements: `nt` and `diff`, denoting the two options of performing no test and testing the differential subsystem.

The random variable $\mathtt{R}_2$ represents the second test results. The frame for the random variable $\mathtt{R}_2$ has three elements: `nr, zero` and `one`, representing respectively the three possible outcomes of the second test: no result, no defect and one defect. The probability distribution of the variables, conditioned on $\mathtt{T}_1$, $\mathtt{R}_1$, $\mathtt{T}_2$ and `CC`, is given in Table 3.

The decision variable `B` represents the purchase decision. The frame for `B` has three elements: $\tilde{\mathtt{b}}$, `b` and `g`, denoting respectively the options of not buying the car, buying the car without the anti–lemon guarantee and buying the car with the anti–lemon guarantee.

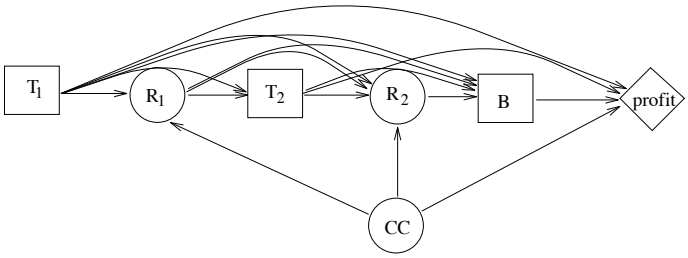

FIGURE 1. An influence diagram for the used car buyer problem

TABLE 1. The prior probability distribution of the car's condition $P\{\texttt{CC}\}$

CC	prob
peach	0.8
lemon	0.2

3. REVIEW OF ALGORITHMS FOR INFLUENCE DIAGRAM EVALUATION

In this section, we review related research efforts on influence diagram evaluation, and classify them into two categories. Those in the first category use an intermedi-

TABLE 2. The probability distribution of the first test result $P\{\texttt{R}_1|\texttt{T}_1, \texttt{CC}\}$

$\texttt{T}_1$	CC	$\texttt{R}_1$	prob
nt	–	nr	1.0
nt	–	others	0
st	–	nr	0
st	–	two	0
st	peach	zero	0.9
st	peach	one	0.1
st	lemon	zero	0.4
st	lemon	one	0.6
f&e	–	nr	0
f&e	peach	zero	0.8
f&e	peach	one	0.2
f&e	peach	two	0
f&e	lemon	zero	0.13
f&e	lemon	one	0.53
f&e	lemon	two	0.33

TABLE 3. The probability distribution of the second test result $P\{\mathtt{R}_2|\mathtt{T}_1, \mathtt{R}_1, \mathtt{T}_2, \mathtt{CC}\}$

$\mathtt{T}_1$	$\mathtt{R}_1$	$\mathtt{T}_2$	CC	$\mathtt{R}_2$	prob
nt	–	–	–	nr	1.0
nt	–	–	–	others	0
st	–	–	–	nr	1.0
st	–	–	–	others	0
f&e	–	–	–	nr	1.0
f&en	–	–	–	others	0
tr	nr	–	–	nr	1.0
tr	nr	–	–	others	0
tr	two	–	–	nr	1.0
tr	two	–	–	others	0
tr	–	nt	–	nr	1.0
tr	–	nt	–	others	0
tr	zero	diff	peach	zero	0.89
tr	zero	diff	peach	one	0.11
tr	zero	diff	lemon	zero	0.67
tr	zero	diff	lemon	one	0.33
tr	one	diff	peach	zero	1.0
tr	one	diff	peach	one	0
tr	one	diff	lemon	zero	0.44
tr	one	diff	lemon	one	0.56

ate representation and evaluate an influence diagram in two conceptual steps: first transforming the influence diagram into its intermediate representation and then computing an optimal policy from the intermediate representation. Those in the second category compute optimal policies directly from influence diagrams. Our method belongs to the first category.

3.1. Howard and Matheson's two–phase method

Howard and Matheson's method belongs to the first category. It first transforms an influence diagram into a decision tree and then evaluates an optimal policy from the decision tree.

Howard and Matheson (1984) discuss a way to transform a regular 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 is constructed from the decision tree 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 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), which will be illustrated in the next subsection.

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

decision tree is exponential in the number of 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 by Shachter (1986).

Shachter's algorithm. Shachter's algorithm takes a reduction approach. The algorithm evaluates an 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, *arc reversal*, *barren node removal, random node removal* and *decision node removal*.

The *arc reversal* operation is illustrated 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. The direction of the arc can be reversed and both nodes inherit each other's parents. This operation is an application of Bayes Theorem. In Fig. 2, we begin with conditional probability distributions $P\{b|a,y,z\}$ and $P\{a|x,y\}$, and end up with conditional probability distributions $P\{a|b,x,y,z\}$ and $P\{b|x,y,z\}$. Formally, we have:

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

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

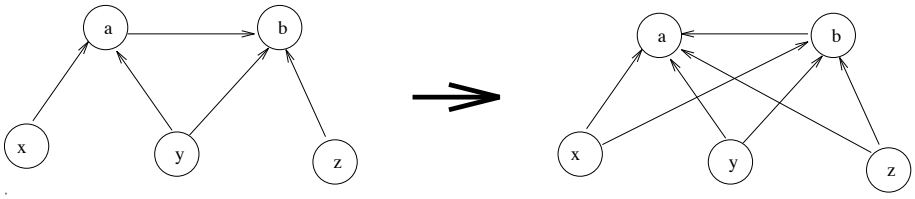

FIGURE 2. An illustration of the arc reversal operation: reversing arc a $\rightarrow$ b

Barren node removal. A node in an influence diagram is called a *barren node* if it has no child in the diagram. The barren node removal reduction states that any barren node that is not a value node can be removed together with its 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. As a result of this operation, the random node x is removed and the old value node is replaced with a new one that inherits all of the parents of both the old value node and the random node. The reduction is illustrated in Fig. 3 where the value function

g' of the new value node v' in the resultant influence diagram is given by:

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

Decision node removal. A decision node is called a *leaf* decision node if it has no decision node descendant. 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. 4 where the value function g' of the new value node v' in the resultant influence diagram is given by:

$$g'(b) = max_d\ g(d,b).$$

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

$$\delta_d(b) = arg\ \ max_d\ g(d,b).$$

Note that the new value node has the same parent as the old value node. Thus, some of the parents of d may become barren nodes as a result of this reduction. In Fig. 4, node a becomes a barren node. The arc from such a node represents information available to the decision maker, but the information has no effect on either the optimal expected value or the optimal policy of the influence diagram. This kind of arc (such as $a \rightarrow d$ in Fig. 4) is called an *irrelevant* arc. Irrelevant arcs can be identified and removed in a pre–processing step (Zhang 1994, Zhang *et al.* 1993a, Zhang *et al.* 1993c).

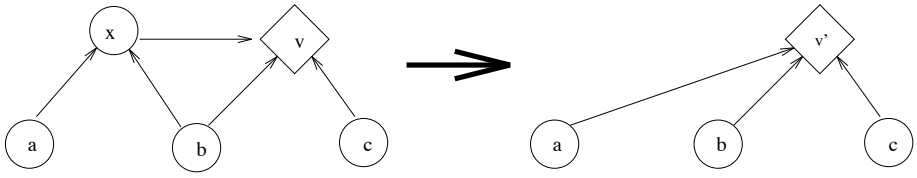

FIGURE 3. An illustration of random node removal: x is removed by expectati on

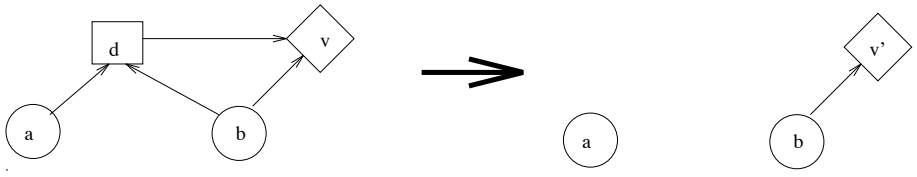

FIGURE 4. An illustration of decision node removal: d is removed by maximization

Other developments. Influence diagrams are closely related to Bayesian nets (Pearl 1988). Quite a few algorithms have been developed in the literature (Jensen *et al.* 1990, Lauritzen and Spiegelhalter 1988, Pearl 1988, Shachter *et al.* 1990, Zhang and Poole 1992a) 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 (Cooper 1988, Ndilikilikesha 1991, Shachter 1990, Shachter and Peot 1992, Shenoy 1990, Shenoy 1991, Zhang *et al.* 1993a, Zhang and Poole 1992b), 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 regular, no–forgetting influence diagram, the optimal policy 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 that precede the decision node.

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

Shachter and Peot (1992) show that the problem of influence diagram evaluation can be reduced to a series of Bayesian net evaluations. To this end, the decision nodes are transformed into random nodes with a uniform distribution, and the value node of an influence diagram is replaced with an "observed" probabilistic *utility* node v' with frame $\{0, 1\}$ and a normalized probability distribution. The optimal decision function δ_n for the last decision node d_n can be computed as follows: for each element $e \in \Omega_{\pi(d_n)}$,

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

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 $e \in \Omega_{\pi(d_i)}$,

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

The problem of influence diagram evaluation is then reduced to a series of problems of computing posterior probabilities in Bayesian nets. Shachter and Peot (1992) also point out that an influence diagram can be converted into a *cluster tree*, which is similar to the clique trees (Lauritzen and Spiegelhalter 1988) of Bayesian nets, and that the problem of evaluating the influence diagram can thus be reduced to evaluating the cluster tree. A similar approach has also been used by Shenoy (1990) for his *valuation based systems* and by Ndilikilikesha (1991) for evaluating *potential influence diagrams*.

Zhang and Poole (1992b) propose a divide–and–conquer method for evaluating stepwise decomposable influence diagrams. This method is further studied in (Zhang *et al.* 1993a, Zhang 1994). Like Shachter and Peot's algorithm, Zhang and Poole's method also deals with one decision node at a time. Unlike Shachter and Peot's algorithm, Zhang and Poole's method takes a reduction approach. Suppose node d

is a leaf decision node of a stepwise decomposable influence diagram $\mathcal{I}$. The set $\pi(d)$ 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 with respect to the decision node d transforms $\mathcal{I}$ 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 value function $g' : \Omega_{\pi(d)} \rightarrow \mathcal{R}$, and an optimal decision function $\delta_d : \Omega_{\pi(d)} \rightarrow \Omega_d$. The same reduction is applicable to the resulting body.

3.3. Some common weaknesses of the previous algorithms

One common weakness of the algorithms that evaluate influence diagrams directly 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 inherited from the disadvantages of conventional influence diagrams for asymmetric decision problems. To be represented by an influence diagram, an asymmetric decision problem must be "symmetrized." This symmetrization results in many "impossible" information states (they have zero probability). The optimal choices for these impossible states need not be computed at all. We use the used car problem to illustrate this point.

The used car buyer problem is asymmetric in a number of aspects. First, the set of the possible outcomes of the first test result varies, depending on the choice for the first test. If the choice for the first test is `nt`, then there is only one possible outcome for the first test result — `nr` (representing no result). If the choice for the first test is `st` or `tr`, then there are two possible outcomes for the first test result — `zero` and `one` (representing no defect and one defect, respectively). If the choice for the first test is `f&e`, then there are three possible outcomes for the first test result — `zero`, `one` and `two` (representing no defect, one defect and two defects, respectively). However, in the influence diagram representation, the frame of the variable $\mathtt{R}_1$ is a common set of outcomes for all the three cases. The impossible combinations of the test choices and the test results are characterized by assigning zero probability to them (as shown in Table 2). A similar discussion is applicable to the variable $\mathtt{R}_2$. Second, from the problem statement we know that testing the differential subsystem is possible only in the states where the first test performed is on the transmission subsystem. However, in the influence diagram representation, it appears that the second test is possible in any situation, while the fact that the option of testing the differential subsystem is not available in some situations is characterized by assigning unit probability to outcome `nr` of the variable $\mathtt{R}_2$ conditioned on these situations. Third, when we examine the information states of the decision variable $\mathtt{T}_2$, we will see many combinations of test options and test results are impossible. For example, if Joe first tests the transmission subsystem, it is impossible to observe `nr` and `two`. If the influence diagram is evaluated by conventional algorithms, an optimal choice for the second test will be computed for each of the information states, including many impossible states. Similar argument is applicable to the decision variable `B`. Because

it is not necessary to compute optimal choices of a decision variables for impossible states, it is desirable to avoid the computation.

Existing algorithms for directly evaluating influence diagrams have two weaknesses, stemming from the symmetrization. The first one is that, for each decision node d, they will perform a maximization operation conditioned on each information state in $\Omega_{\pi(d)}$, even though the marginal probabilities of some states are zero. This weakness arises from the fact that these algorithms compute the decision functions in the reverse order of the decision nodes in the influence diagram. At the time of computing the decision function for decision d, the marginal probabilities of the information states in $\Omega_{\pi(d)}$ are not computed yet. The second weakness is that optimal choices for a decision node are chosen from the whole frame Ω_d, instead of from the corresponding effective frames. Thus, it is evident that these algorithms involve unnecessary computation.

4. OUR SOLUTION

In this section, we present an approach for overcoming the aforementioned weeknesses. Our approach consists of two independent components: a simple extension to influence diagrams and a two–phase method for influence diagram evaluation.

Our extension allows explicitly expressing the fact that some decision variables have different frames in different information states. We achieve this by introducing a *framing function* for each decision variable, which characterizes the available alternatives for the decision variable in different information states. With the help of framing functions, our solution algorithm effectively ignores the unavailable alternatives when computing an optimal choice for a decision variable in any information state. Our extension is inspired by the concepts of *indicator valuations* and *effective frames* proposed by Shenoy (1993).

Conceptually, our evaluation method, similar to Howard and Matheson's (Howard and Matheson 1984), consists of two steps: in order to evaluate an influence diagram, a decision graph is generated and the evaluation is then carried out on the decision graph. The first step will be described in this section. The second step can be carried out by the search algorithms presented in (Qi 1994, Qi and Poole 1993). By using those search algorithms, the two steps of decision graph generation and optimal policy computation can be combined into one, and only a portion of the decision graph needs to be generated, due to heuristic search. Our method successfully avoids the unnecessary computations by pruning those impossible states and ignoring those unavailable alternatives for the decision variables.

4.1. Extending influence diagrams

We extend influence diagrams by introducing *framing functions* to the definition given in Section 2. With this extension, an influence diagram $\mathcal{I}$ is a tuple $\mathcal{I} = (X, A, \mathcal{P}, \mathcal{U}, \mathcal{F})$ where $X, A, \mathcal{P}, \mathcal{U}$ have the same meaning as before, and $\mathcal{F}$ is a set $\{f_d : \Omega_{\pi(d)|d\in D} \rightarrow 2^{\Omega_d}\}$ of *framing functions* for the decision nodes.

A framing function expresses the fact that the legitimate alternative set for a decision variable may vary in different information states. More specifically, for a decision variable d and an information state $s \in \Omega_{\pi(d)}$, $f_d(s)$ is the set of the legitimate alternatives the decision maker can choose for d in information state s. Following Shenoy (1993), we call $f_d(s)$ the *effective frame* of decision variable d in

information state s.

With this extension, a decision function δ_i must satisfy the following constraint: For each $s \in \Omega_{\pi(d_i)}$, $\delta_i(s) \in f_{d_i}(s)$. In words, the choice prescribed by a decision function for a decision variable d in an information state must be a legitimate alternative.

In the used car problem, the framing functions for the first test decision and the purchase decision are simple — they map every information state to the corresponding frames. The frame function for the second test decision can be specified as follows:

$$f_{T_2}(X) = \begin{cases} \{\texttt{nt, diff}\} & \text{if } \sigma_{T_1}(X) = \texttt{tr} \\ \{\texttt{nt}\} & \text{otherwise.} \end{cases}$$

4.2. Decision graphs

Decision graphs (Qi 1994, Qi and Poole 1993) are a generalization of decision trees (Qi and Poole 1992, Raiffa 1986), allowing structure sharing. A decision graph can be viewed as an acyclic AND/OR graph (Pearl 1984, Nilsson 1980) with a maximization–expectation evaluation function. Nodes in a decision graph 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. A probability is associated with each arc emanating from a chance node and the probabilities of all the arcs from 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.

Let DG be a decision graph. A *max–exp evaluation* of DG is a real–valued function u defined as follows:

1. If N is a terminal: $u(DG, N) = V(N)$, where $V(N)$ denotes the value associated with the terminal N.
2. If N is a chance node with k children $N_1, ..., N_k$ in DG:
   $u(DG, N) = \sum_{i=1}^{k} p(N, N_i) * u(DG, N_i)$, where $p(N, N_i)$ is the probability on the arc from node N to node N_i.
3. If N is a choice node with k children $N_1, ..., N_k$ in DG:
   $u(DG, n) = max_{i=1}^{k}\{v(N, N_i) + u(DG, N_i)\}$, where $v(N, N_i)$ is the value on the arc arc from node N to node N_i.

The value given by $u(DG, N)$ is called the *max–exp* value of the node N. Note that the above definition is applicable to a solution graph as well since a solution graph is a special decision graph.

4.3. Influence Diagram Evaluation via Stochastic Dynamic Programming

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

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'. A decision node that is preceded by no other decision node is called a *root* decision node. A decision node that precedes no other decision node is called a *leaf* decision node. In a regular influence diagram, a decision node can directly precede at most one decision node.

Let $\mathcal{I}$ be a regular, stepwise decomposable influence diagram with a single value node v and with decision nodes d_1, ..., d_n in the precedence order. Let Y_{d_k} denote the downstream of d_k. Let $\mathcal{I}(d_k, d_{k+1})$ denote the subgraph consisting of d_k, $\pi(d_k)$, $\pi(d_{k+1})$ and those nodes in Y_{d_k} that are not m–separated from d_{k+1} by $\pi(d_{k+1})$. Procedurally, $\mathcal{I}(d_k, d_{k+1})$ can be obtained from $\mathcal{I}$ 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 descendants of d_{k+1}; (3) remove all barren nodes not in $\pi(d_{k+1})$; (4) remove all arcs among nodes in $\pi(d_k) \cup \{d_k\}$ and assign uniform distributions to the root nodes[3] in $\pi(d_k) \cup \{d_k\}$.

$\mathcal{I}(d_k, d_{k+1})$ is called the *sector*[4] of $\mathcal{I}$ from d_k to d_{k+1}. The sector $\mathcal{I}(-, d_1)$ contains only the nodes in $\pi(d_1)$ and those nodes that are not m–separated from d_1 by $\pi(d_1)$. The sector $\mathcal{I}(d_n, -)$ contains those nodes in $\pi(d_n) \cup \{d_n\}$ and those nodes in the downstream Y_{d_n} of d_n.

Note that the sector $\mathcal{I}(-, d_1)$ is a Bayesian net. Furthermore, because $\mathcal{I}$ is stepwise decomposable, d_k is the only decision node in the sector $\mathcal{I}(d_k, d_{k+1})$ that is not in $\pi(d_k)$. Similarly, d_n is the only decision node in the sector $\mathcal{I}(d_n, -)$ that is not in $\pi(d_n)$.

Let e be any event in $\mathcal{I}_\delta$ and let $E_\delta[v|e]$ be defined as follows:

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

For each k with $1 \leq k \leq n$, 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 with respect to policy δ, conditioned on $\pi(d_k) = x$.

Let $\delta^0 = (\delta_1^0, ..., \delta_n^0)$ be an optimal policy for influence diagram $\mathcal{I}$ 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 $\mathcal{I}$ conditioned on $\pi(d_k) = x$. In other words, $V_k(x)$ is the expected value a decision maker can obtain if he/she starts to make optimal decisions in the situation represented by the event $\pi(d_k) = x$.

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

[4]Previously, such a part of an influence diagram has been called a *section* (Zhang *et al.* 1993a).

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

$$V'_n(x,a) = \sum_{y \in \Omega_{\pi(v)}} g(y) * P_{\delta^0}\{\pi(v) = y | \pi(d_n) = x, d_n = a\} \tag{4}$$

$$V'_k(x,a) = \sum_{y \in \Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\delta^0}\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = a\} \tag{5}$$

Intuitively, $V'_k(x,a)$ is the optimal expected value of the influence diagram $\mathcal{I}$ 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 policy for the remaining decisions.

The following theorem, taken from (Qi 1994), establishes the computational structure of influence diagram evaluation in the form of finite–stage stochastic dynamic programming (Ross 1983). A proof of this theorem is given in Appendix A.

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

$$V'_k(x,a) = \sum_{y \in \Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = a\} \tag{6}$$

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

$$\delta^0_k(x) = arg \;\; max_{a \in \Omega_{d_k}} \{V'_k(x,a)\} \tag{8}$$

$$E_{\delta^0}[v] = \sum_{x \in \Omega_{d_1}} V_0(x) * P\{\pi(d_1) = x\}. \tag{9}$$

where $P\{\pi(d_1) = x\} = P_{\delta^0}\{\pi(d_1) = x\}$ and $P\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = a\} = P_{\delta^0}\{\pi(d_{k+1}) = y | \pi(d_k) = x, d_k = a\}$, both independent of δ^0, can be computed in the sector $\mathcal{I}(d_k, d_{k+1})$ by any Bayesian net algorithm.

Equations 6 and 7 collectively form a variation of Bellman's optimality equation (Bellman 1957) for stochastic dynamic programming.

Theorem 1 shows that functions $V_1, ..., V_n$ and $\delta^0_1, ..., \delta^0_n$, as well as $E_{\delta^0}[v]$ can be computed from V'_n. The computation process is similar to the one implied in the recursive formula given in (Cooper 1988). For an influence diagram, the computation can be roughly divided into two parts: one for computing conditional probabilities, which can be computed by using Bayesian net algorithms, and one for the summations and maximizations as specified in Equations 6 and 7. The definition of V'_n is given by Equation 4.

4.4. Representing the Computational Structure by Decision Graphs

In this section, we use *decision graphs* to depict the computational structures of the optimal expected values of influence diagrams. For each influence diagram, we can construct a decision graph and define a *max–exp* evaluation function for the

decision graph in such a way that the solution graphs of the decision graph correspond to the policies for the influence diagram, and the optimal solution graphs correspond to the optimal policies.

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

Let d be a decision variable in an influence diagram. For each $x \in \Omega_{\pi(d)}$, we call assignments of the form $\pi(d) = x$ *parent situations* for the decision variable d. For each alternative $a \in \Omega_d$, we call assignments of the form $\pi(d_i) = x, d = a$ *inclusive situations* for the decision variable d. Two situations are *consistent* if the variables common to the two situations are assigned the same values.

For an influence diagram, we 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 for an influence diagram.

- A chance node denoting the empty situation is the *root* of the decision graph.
- For each information state $x \in \Omega_{\pi(d_1)}$, there is a choice node, denoting the parent situation $\pi(d_1) = x$, as a child of the root in the decision graph. The arc from the root to the node is labeled with the probability $P\{\pi(d_1) = x\}$.
- Let N be a choice node in the decision graph denoting a parent situation $\pi(d) = x$ for some decision variable d and some $x \in \Omega_{\pi(d)}$. Let $f_d(x)$ be the effective frame for the decision variable d in the information state x. Then, N has $|f_d(x)|$ children, each being a chance node corresponding to an alternative[5] in $f_d(x)$. The node corresponding to alternative a denotes the inclusive situation $\pi(d) = x, d = a$.
- The node denoting an inclusive $\pi(d_n) = x, d_n = a$ is a terminal in the decision graph, having value $V'_n(x, a)$.
- Let N be a chance node denoting an inclusive situation $\pi(d_{i-1}) = x, d_{i-1} = a$, and let $\mathcal{A}$ be the subset of the parent situations for decision variable d_i which are consistent with $\pi(d_{i-1}) = x, d_{i-1} = a$. Node N has $|\mathcal{A}|$ children, each being a choice node denoting a parent situation in $\mathcal{A}$. The arc from N to the child denoting a parent situation $\pi(d_i) = y$ is labeled with the conditional probability[6] $P\{\pi(d_i) = y | \pi(d_{i-1}) = x, d_{i-1} = a\}$.

In such a decision graph, a choice node represents a situation of the form $\pi(d_i) = x$ for some i and $x \in \Omega_{\pi(d_i)}$. In such a situation, the decision agent needs to decide which alternative among $f_{d_i}(x)$ should be selected for d_i. Thus, the choice node has $|f_{d_i}(x)|$ children, each for an alternative in $f_{d_i}(x)$. The child corresponding to alternative a is a chance node, representing the probabilistic state $\pi(d_i) = x, d_i = a$. From this probabilistic state, one of the information states of d_{i+1} may be reached. The probability of reaching information state y is $P\{\pi(d_{i+1}) = y | \pi(d_i) = x, d_i = a\}$.

Let DG be a decision graph derived from an influence diagram. We can define an evaluation function, u, on DG as follows:

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

$$u(DG, N) = V'_n(x, a)$$

[5]Note that here we are using the effective frame $f_d(x)$ instead of the frame Ω_d.

[6]Note that probabilities of this kind on arcs are not computed unless necessary. This point will be clear in Section 4.5.

- If N is a chance node with children $N_1, ..., N_l$, then

$$u(DG, N) = \sum_{i=1}^{l} p(N, N_i) * u(DG, N_i)$$

  where $p(N, N_i)$ is the probability on the arc from node N to node N_i.
- If N is a choice node with children $N_1, ..., N_l$, then

$$u(DG, N) = max_{i=1}^{l}\{u(DG, N_i)\}.$$

The following theorem can be easily proved by induction on nodes in the decision graph.

Theorem 2. (1) If N is a choice node representing a parent situation $\pi(d_k) = x$, then $u(N) = V_k(x)$.

(2) If N is a chance node representing an inclusive situation $\pi(d_k) = x, d_k = a$, then $u(N) = V'_k(x, a)$.

(3) If N is the root, then $u(N)$ is equal to the optimal expected value of the influence diagram.

The correspondence between the optimal policies of the influence diagrams and the optimal solution graphs becomes 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).

4.5. Avoiding unnecessary computations

We observe that the asymmetry of an influence diagram is reflected by arcs with zero probability in the corresponding decision graph. As we know, the value of a chance node in a decision graph is the weighted sum 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 the arcs to a choice node are all zero, the value of the node will never be required. Thus, we need not compute the max–exp value of, and the optimal choice for, the node. In this case, the probability of the information state denoted by the choice node is zero. Thus, it is equivalent to say we need not compute the optimal choice for the corresponding decision variable for the impossible information state. One way to avoid such computation for the impossible states is by pruning those choice nodes corresponding to impossible states. The following procedure for generating a decision graph for an influence diagram effectively realizes this objective:

(1) generate the root node and put it into set G;
(2) if G is empty then stop;
(3) pick a node N from G and set G to $G - \{N\}$;
(4) if N is a terminal node then go to (2);
(5) generate the child set $C(N)$ of node N; if N is a choice node, set G to $G \cup C(N)$, otherwise let $C'(N)$ be the subset of $C(N)$ such that the probabilities of the arcs from N to the nodes in $C'(N)$ are non–zero; set G to $G \cup C'(N)$;
(6) goto (2);

The above procedure will not expand a choice node unless its probability is not zero. Thus, the procedure effectively ignores subgraphs rooted at nodes corresponding to impossible states. Thus, the computation (for computing optimal choices for choice nodes and for computing the conditional probabilities) involved in the subtrees is totally avoided.

When applying the above procedure to the used car buyer problem, a decision graph (tree) shown in Fig. 5 is generated. In the graph, the leftmost box represents the only situation in which the first test decision is to be made. The boxes in the middle column correspond to the information states in which the second test decision is to be made. Similarly, the boxes in the right column correspond to the information states in which the purchase decision is to be made. From the figure we see that among those nodes corresponding to the information states of the second test, all but two have only one child because the effective frames of the second test in the those information states have only a single element. Making use of the framing function this way is equivalent to six prunings, each cutting a subtree under a node corresponding to an information state of the second test. Those shadowed boxes correspond to the impossible states. Our algorithm effectively detects those impossible states and prune them when they are created. Each of such pruning amounts to cutting a subtree under the corresponding node. Consequently, our algorithm does not compute optimal choices for a decision node for those impossible states. For the used car buyer problem, our algorithm computes optimal choices for the purchase decision for only 12 information states, and optimal choices for the second test for only 8 information states (among which six can be computed trivially). In contrast, whereas those algorithms that do not exploit asymmetry will compute the optimal choices for the purchase decision for 96 ($4 \times 4 \times 2 \times 3$) information states and will compute optimal choices for the second test for 16 information states.

Better performance will be achieved by using decision graph search algorithms (Qi 1994, Qi and Poole 1993). By using these algorithms, the step for generating decision graph and the step for computing optimal solution graph are combined, and some subgraphs may not be processed at all.

To use these algorithms, we need a heuristic function that gives admissible estimation on $u(s)$ for any situation s. Note that the notion of admissibility for a heuristic function here is different from the traditional one. Because we are maximizing merit instead of minimizing cost, we define a heuristic function to be admissible if it never under–estimates for any situation s. Formally, a function h is admissible if $h(s) \geq u(s)$ for any situation s. An obvious admissible heuristic function for an influence diagram evaluation problem is the one returning $+\infty$ for each situation. Performance can be further enhanced if we can obtain a more *informed* heuristic function by using domain–dependent knowledge.

4.6. A comparison with Howard and Matheson's method

The relationship between our method and Howard and Matheson's should now be clear. In both methods, an influence diagram is first transformed into a secondary representation from which an optimal policy 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 regular stepwise decomposable influence diagrams.

Second, the sizes of the secondary representation generated by the two methods are different. For a given influence diagram, the depth of the decision tree obtained by

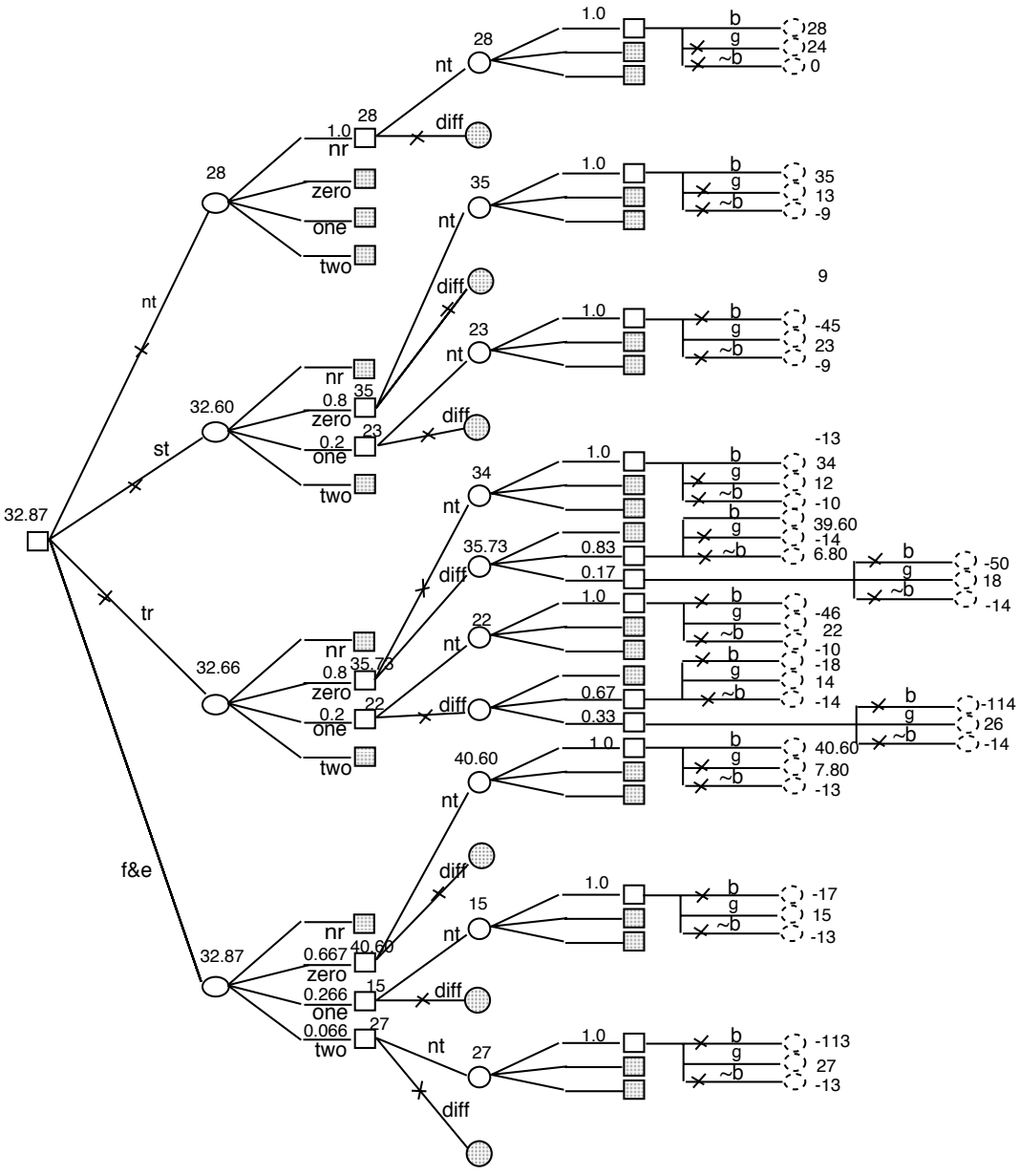

FIGURE 5. A decision graph (tree) generated for the used car buyer problem

Howard and Matheson's method is equal to the number of variables in the influence diagram, while the depth of the decision graph obtained by our method is $2n$, where n is the number of decision variables in the influence diagram. Typically, there are more random variables than decision variables in a decision problem, thus the

depth of the decision tree obtained by Howard and Matheson's method is larger than the depth of the decision graph obtained by ours for the same influence diagram. Furthermore, the number of 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 graph obtained by our method. In fact, the number of nodes in a decision graph obtained by our method is:

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

Third, both methods need to compute some conditional probabilities. In Howard and Matheson's method, these probabilities are computed by directly applying Bayes theorem, while in our method, we can use those algorithms developed for Bayesian net evaluation.

5. HOW MUCH CAN BE SAVED?

In this section, we give a general analysis of how much computation can be saved by exploiting asymmetry for a typical class of decision problems. Since the number of optimal choices to be computed is a relative measure on the time an algorithm takes for evaluating an influence diagram, we compare the number of optimal choices to be computed by our method against other methods. Consider the following generalized buying problem:

> Suppose we have to decide whether to buy an expensive and complex item. Before making the decision, we can have n tests, denoted by $T_1, ..., T_n$, on the item. Suppose test T_i has k_i alternatives and j_i possible outcomes for $i = 1, ..., n$. Among the k_i alternatives for test T_i, one stands for the option of no–testing. Correspondingly, among the j_i possible outcomes of test T_i, one stands for no observation, resulting from the no–testing alternative.

An influence diagram for this problem is shown in Fig. 6. In this influence diagram, decision variable T_i has a frame of size k_i, including an alternative `nt` for not testing, and random variable R_i has a frame of size j_i, including an outcome `nr` for no observation. Let $H_i = k_i j_i$. The size of $\Omega_{\pi(T_i)}$ is $\Pi_{l=1}^{i-1} H_l$. Thus, the decision T_i has $\Pi_{l=1}^{i-1} H_l$ information states, the decision B has $\Pi_{l=1}^{n} H_l$ information states. If we do not exploit the asymmetry of the problem, we will have to compute an optimal choice for every decision and each of its information states. The total number is $\sum_{i=1}^{n+1} \Pi_{l=1}^{i-1} H_l$.

Let us consider for how many information states our method will compute optimal choices for each decision variable. To do so, we simply count the number of choice nodes in the decision graph (actually, decision tree) generated by our method from the influence diagram.

Before counting, let us first make some notes on the conditional probabilities we will use in the process of decision graph construction. During the process, we need the conditional probabilities $P\{\pi(d_{i+1})|\pi(d_i), d_i\}$, where d_i denotes T_i for $i = 1, ..., n$ and d_{n+1} denotes B, the purchase decision. First, because $\pi(d_{i+1}) = \pi(d_i) \cup \{d_i, R_i\}$, we have:

$$P\{\pi(d_{i+1})|\pi(d_i), d_i\} = P\{\pi(d_i), d_i, R_i|\pi(d_i), d_i\} = P\{R_i|\pi(d_i), d_i\}.$$

Second, for any given information state $y \in \Omega_{\pi(d_i)}$ with $1 \leq i \leq n$, the total number of test–choice/test–result combinations is $k_i j_i$. Among these combinations, some have zero probability, as illustrated by the partial decision tree shown in Fig. 7, where shadowed boxes correspond to the zero probability combinations. The number of condition/outcome combinations with zero probability determines the degree of asymmetry of the problem. Let $J_{ir}(y)$ denote the set of possible outcomes of R_i if the choice for T_i in situation $\pi(d_i) = y$ is $a_r \in \Omega_{T_i}$. In terms of probability distributions, this is equivalent to saying that, for all $y \in \Omega_{\pi(d_i)}$, for all $a_r \in \Omega_{d_i}$ and for all $x \in \Omega_{R_i} - J_{ir}(y)$

$$P\{R_i = x | \pi(d_i) = y, d_i = a_r\} = 0$$

and for all $y \in \Omega_{\pi(d_i)}$ and for all $a_r \in \Omega_{d_i}$

$$\sum_{x \in J_{ir}(y)} P\{R_i = x | \pi(d_i) = y, d_i = a_r\} = 1.$$

Thus, the total number of non–zero probability combinations of test–choice/test–result, conditioned on $\pi(d_i) = y$, is bounded from above by $\sum_{r=1}^{k_i} |J_{ir}(y)|$. Let $h_i(y) = \sum_{r=1}^{k_i} |J_{ir}(y)|$ and $h_i = \max_y h_i(y)$.

At the most conservative extreme, we can assume that for any $y \in \Omega_{\pi(d_i)}$, there exist some $a \in \Omega_{d_i}$ and some $x \in \Omega_{R_i}$ such that

$$P\{R_i = x | \pi(d_i) = y, d_i = a\} = 0.$$

In this case, the total number of non–zero probability combinations of test–choices/test–results is bounded from above by $k_i j_i - 1$.

A reasonable assumption we can make about a practical decision problem is that, for any decision variable, the choice of do–nothing will always lead to no result while a choice other than do–nothing will always lead to some results. We call this assumption the "no action, no result" assumption. For the generalized buying problem, this assumption means that no-test choice `nt` for decision node T_i will lead to no observation `nr` as the only possible outcome of R_i, and other test choices will not lead to no observation `nr`. The partial decision tree in Fig. 7 depicts this case, where shaded nodes correspond to the states with zero probability. In terms of probability distributions, we have:

$$P\{R_i = x | \pi(d_i) = y, d_i = \texttt{nt}\} = 0$$

$$P\{R_i = \texttt{nr} | \pi(d_i) = y, d_i = \texttt{nt}\} = 1$$

for any $x \in \Omega_{R_i}, x \neq \texttt{nr}$, and any $y \in \Omega_{\pi(d_i)}$, and

$$P\{R_i = \texttt{nr} | \pi(d_i) = y, d_i = a\} = 0$$

for any $a \in \Omega_{d_i}, a \neq \texttt{nt}$, and $y \in \Omega_{\pi(d_i)}$. In this case, the total number of non–zero probability combinations of test–choices/test–results is bounded from above by $1 + (k_i - 1)(j_i - 1)$.

Now, let us count the number of choice nodes in the decision graph that correspond to information states with non–zero probabilities. Based on the above analysis of the probability distributions, we have the following: T_1 has one parent situation;

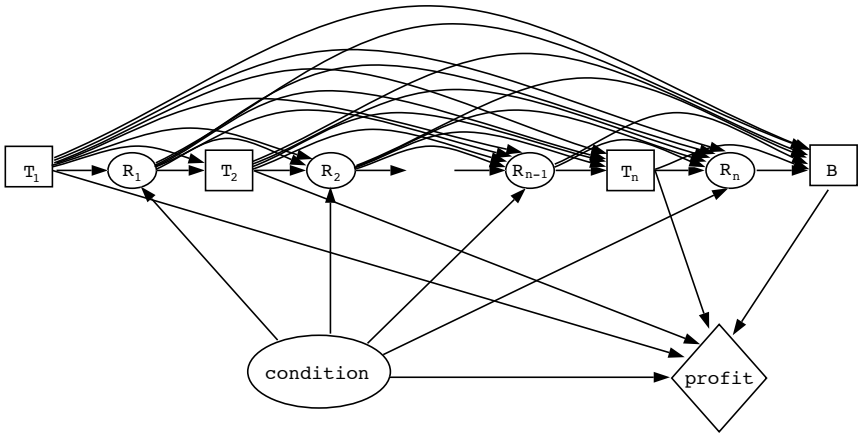

FIGURE 6. An influence diagram for a generalized buying problem

T_2 has at most h_1 parent situations with non–zero probabilities; T_2 has at most $h_1 h_2$ parent situations with non–zero probabilities; etc., and B has at most $\Pi_{l=1}^{n} h_l$ parent situations with non–zero probabilities. Thus, the total number of choice nodes in the decision graph is bounded from above by $\sum_{i=1}^{n+1} \Pi_{l=1}^{i-1} h_l$.

Let $\rho_i = H_i/h_i$. We call ρ_i the "savings factor" with respect to decision T_i . Since $h_i \leq 1 + (k_i - 1)(j_i - 1) < H_i$, we have $\rho_i \geq (k_i j_i)/(1 + (k_i - 1)(j_i - 1)) > 1$.

The overall savings factor is given by

$$\rho = \frac{\sum_{i=1}^{n+1} \Pi_{l=1}^{i-1} H_l}{\sum_{i=1}^{n+1} \Pi_{l=1}^{i-1} h_l} \geq \frac{\Pi_{l=1}^{n} H_l}{\sum_{i=1}^{n+1} \Pi_{l=1}^{i-1} h_l} = \frac{\Pi_{l}^{n} \rho_l}{\sum_{i=1}^{n+1} \frac{1}{\Pi_{l=i}^{n} h_l}}.$$

It is reasonable to assume that $h_i \geq 2$ for $i = 1, ..., n$. Then we have $\rho \geq \frac{\Pi_{i=1}^{n} \rho_i}{2}$.

For example, suppose $k_i = 4$ (each test has three alternatives plus the alternative of no test) and $j_i = 4$ for $i = 1, ..., n$. Then we have $\rho_i \geq 16/15$ in the most conservative extreme, and $\rho_i \geq 16/10$ under the "no action, no result" assumption. Then, the overall savings factor is bounded from below by $\frac{(16/15)^n}{2}$ in the most conservative extreme, and by $\frac{(16/10)^n}{2}$ under the "no action, no result" assumption.

In the above analysis, we assume that test T_i has exactly k_i alternatives in every information state. In fact, the set of legitimate alternatives for test decision T_i in an information state s is $f_{T_i}(s)$ and may have fewer than k_i elements. Thus, the actual overall savings factor could be much greater than $\Pi_{i=1}^{n} \rho_i/2$.

Finally, let us note that the above analysis is applicable to any decision problem: as long as it satisfies the "no action, no result" assumption. Exploiting asymmetry will lead to a savings factor that is exponential in the number of asymmetric decisions in the decision problem.

It is worth pointing out that an exponential savings factor for an exponential problem may not change the exponential nature of the problem. More specifically, suppose that problem P will take algorithm A $O(c^n)$ time units and take algorithm

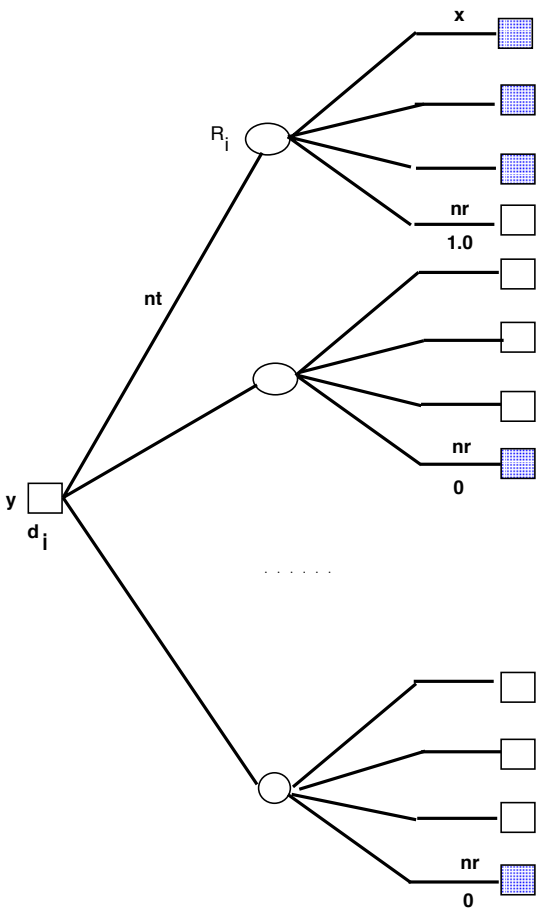

FIGURE 7. An illustration of the asymmetry of a decision variable

B $O(c/\alpha)^n$ time units with $\alpha > 1$. We say that algorithm B has an exponential savings factor in comparison with algorithm A because algorithm B runs α^n times faster than algorithm A. Although algorithm B is still an exponential algorithm, for a fixed time resource, algorithm B may be able to handle larger problems than algorithm A. Let n_a and n_b be the size of the largest problems algorithms A and B can handle, respectively. Then n_a and n_b satisfy:

$$c^{n_a} = (c/\alpha)^{n_b}$$

or

$$\frac{n_b}{n_a} = \frac{\log c}{\log c - \log \alpha}.$$

For example, let us consider again the generalized buying problem. Suppose that each decision in the problem has two alternatives and each random node has three

outcomes, i.e., $k_i = 2$ and $j_i = 3$ for $i = 1, .., n$. It takes those algorithms that do not exploit asymmetry $O(2 * 3)^n$ time units and it takes our algorithm $O(3^n)$ time units. In this case, we have $c = 6$ and $\alpha = 2$. Thus, we have $n_b = n_a \log 6 / \log 3 = 1.63 n_a$. Note that for the same α, the bigger the value of c, the smaller the ratio n_b/n_a.

By applying the analysis results to the used car buyer problem, we have: $H_1 = 16$, $H_2 = 6$, $h_1 = 8$, and $h_2 = 3$. Thus, $\rho_1 = 2$ and $\rho_2 = 2$. $\rho = (1+16+96)/(1+8+24) = 3.7$.

6. RELATED WORK FOR HANDLING ASYMMETRIC DECISION PROBLEMS

Recognizing that influence diagrams are not effective for asymmetric decision problems, several researchers have recently proposed alternative representations.

Fung and Shachter (1990) propose *contingent influence diagrams* for explicitly expressing asymmetry of decision problems. In that representation, each variable is associated with a set of contingencies, and associated with one relation for each contingence. These relations collectively specify the conditional distribution of the variable.

Covaliu and Oliver (1992) propose a different representation for representing decision problems. This representation uses a *decision diagram* and a *formulation table* to specify a decision problem. A decision diagram is a directed acyclic graph whose directed paths identify all possible sequences of decisions and events in a decision problem. In a sense, a decision diagram is a degenerate decision tree in which paths having a common sequence of events are collapsed into one path (Covaliu and Oliver 1992). Numerical data are stored in the formulation table.

Shenoy (1993) proposes a "factorization" approach for representing degenerate probability distributions. In that approach, a degenerate probability distribution over a set of variables is decomposed into several factors over subsets of the variables such that the their "product" is equivalent to the original distribution.

Smith *et al.* (1993) present some interesting progress towards exploiting asymmetry of decision problems. They observe that an asymmetric decision problem often has some degenerate probability distributions, and that the influence diagram evaluation can be speed up if these degenerate probability distributions are used properly. Their philosophy is analogous to the one behind various algorithms for sparse matrix computation. In their proposal, a conventional influence diagram is used to represent a decision problem at the level of relation. In addition, they propose to use a decision tree–like representation to describe the conditional probability distributions associated with the random variables in the influence diagram. The decision tree–like representation is effective for economically representing degenerate conditional probability distributions. They propose a modified version of Shachter's algorithm (Shachter 1986) for influence diagram evaluation, and show how the decision tree–like representation can be used to increase the efficiency of *arc reversal*, a fundamental operation used in Shachter's algorithm. However, their algorithm cannot avoid computing optimal choices for decision variables with respect to impossible information states.

7. CONCLUSIONS

In this paper we analyzed some drawbacks of influence diagrams and existing evaluation algorithms with asymmetric decision problems and we presented an approach for overcoming the drawbacks. Our approach consists of a simple extension to influence diagrams and a two–phase method for influence diagram evaluation. The extension facilitates expressing asymmetry in influence diagrams. The two–phase method effectively avoids unnecessary computation, which leads to exponential savings for asymmetric decision problems.

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 makes use of heuristic search techniques and domain dependent knowledge.

(4) It takes the advantage of asymmetry in decision problems to avoid unnecessary computation.

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, Jim Little, Alan Mackworth, Maurice Queyranne, Jack Snoeyink, (Nevin) Lianwen Zhang and Ying Zhang for their valuable comments on this paper. The first author benefits a lot from many long discussions with Jack Snoeyink and (Nevin) Lianwen Zhang.

REFERENCES

BELLMAN, R. 1957. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey.

COOPER, G. F. 1988. 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.

COVALIU, Z. and R. M. OLIVER. 1992. Formulation and solution of decision problems using decision diagrams. Technical report, University of California at Berkeley.

FUNG, R. M. and R. D. SHACHTER. 1990. Contingent influence diagrams.

HOWARD, R. A. and J. E. MATHESON. 1984. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *The Principles and Applications of Decision Analysis, Volume II*, pages 719–762. Strategic Decision Group, Mento Park, CA.

HOWARD, R. A. 1984. The used car buyer problem. In R. A. Howard and J. E. Matheson, editors, *The Principles and Applications of Decision Analysis, Volume II*, pages 690–718. Strategic Decision Group, Mento Park, CA.

JENSEN, F. V., K. G. OLESEN, and K. ANDERSON. 1990. An algebra of Bayesian belief universes for knowledge based systems. *Networks*, **20**:637–659.

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

Matheson, J. E. 1990. 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.

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

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

Nilsson, Nils J. 1980. *Principles of Artificial Intelligence*. Springer-Verlag Berlin Heidelberg New York.

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

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

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

Phillips, L. D. 1990. Discussion of 'From Influence to Relevance to Knowledge by R. A. Howard'. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Nets and Decision Analysis*, page 22. John Wiley and Sons.

Qi, R. and D. Poole. 1992. Two algorithms for decision tree search. In *Proc. of the Second Pacific Rim International Conference on Artificial Intelligence*, pages 121–127, Seoul, Korea.

Qi, R. and D. Poole. 1993. Decision graph search. *technical report 93-9, Department of Computer Science, UBC*.

Qi, R. 1994. *Decision Graphs: algorithms and applications to influence diagram evaluation and high level path planning with uncertainty*. PhD thesis, Department of Computer Science, University of British Columbia.

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

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

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

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

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

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

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

Shenoy, P. P. 1991. 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. Morgan Kaufmann.

Shenoy, P. P. 1993. Valuation network representation and solution of asymmetric decision problems. working paper No. 246, School of Business, University of Kansas.

Smith, J. E., S. Holtzman and J. E. Matheson. Structuring conditional relationships in influence diagrams. *Operations Research*, **41**(2):280–297, 1993.

Smith, J. Q. 1988. *Decision analysis : a Bayesian approach*. London ; New York : Chapman and Hall.

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

ZHANG, L. and D. POOLE. 1992b. 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. Morgan Kaufmann.

ZHANG, L., R. QI and D. POOLE. 1993a. A computational theory of decision networks. *accepted by International Journal of Approximate Reasoning, also available as a technical report 93-6, Department of Computer Science, UBC.*

ZHANG, L., R. QI and D. POOLE. 1993b. Incremental computation of the value of perfect information in stepwise-decomposable influence diagrams. In *Proc. of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 400–410, Washington, DC.

ZHANG, L., R. QI and D. POOLE. 1993c. Minimizing decision tables in stepwise-decomposable influence diagrams. In *Proc. of the Fourth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, Florida.

ZHANG, L. 1994. *A Computational Theory of Decision Networks.* PhD thesis, Department of Computer Science, University of British Columbia.

A. PROOF OF THEOREM 1

In this appendix, we present a proof of Theorem 1. To prove the theorem, we prove a number of intermediate results first.

Let $\delta = (\delta_1, ..., \delta_n)$ be any policy for $\mathcal{I}$. We have the following results on $\mathcal{I}_\delta$.

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

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

for any $o \in \Omega_{\pi(v)}$, $y \in \Omega_{\pi(d_k)}$, $a \in \Omega_{d_j}$, and for any $x \in \Omega_{\pi(d_j)}$ consistent with y (i.e., the projections of x and y on the common variable are the same).

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

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

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

Proof. Recall that, with respect to a policy $\delta = (\delta_1, ...\delta_n)$, 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 only on nodes in the sector $\mathcal{I}(-, d_1)$, and is independent of δ. Consequently, for any other policy δ',

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

Proof. Since all nodes not in $\mathcal{I}(-, d_1)$ are non–ancestors of the nodes in $\pi(d_1)$, they are irrelevant to the marginal probabilities of $\pi(d_1)$. Since there is no decision node in $\mathcal{I}(-, d_1)$, 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 only on those nodes in the sector $\mathcal{I}(d_n, -)$, and is independent of δ. In other words, for any other policy 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 sector $\mathcal{I}(d_k, d_{k+1})$, and is independent of δ. In other words, for any other policy 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 policy for $\mathcal{I}$ such that $\delta'_1 = \delta_1, ..., \delta'_{k-1} = \delta_{k-1}$ for some k, $1 \le k \le n$, then, for any j, $1 \le j \le k$, and any $x \in \Omega_{\pi(d_j)}$,

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

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

Lemma 5. The expected value of the influence diagram with respect to policy 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:

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

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,

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

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

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

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:

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

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:

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

By reordering the two summations, we obtain:

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

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 policy for $\mathcal{I}$ 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:

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

and

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

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. □

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

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

$$V'_k(x,\delta^0_k(x)) = V_k(x).$$

Proof

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

□

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

$$V'_k(x,\delta^0_k(x)) \geq V'_k(x,a) \quad \text{for each } x \in \Omega_{\pi(d_k)} \text{ and each } a \in \Omega_{d_k}\,.$$

Proof Suppose the 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 policy $\delta' = (\delta'_1, ..., \delta'_n)$ 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 $x \in \Omega_{\pi(d_k)}, x \neq x_0$. For any $x \in \Omega_{\pi(d_k)}$, any $a \in \Omega_{d_k}$ and any $y \in \Omega_{\pi(d_{k+1})}$, by Lemma 7, we have

$$U_{k+1}(x,\delta') = U_{k+1}(x,\delta^0) = V_{k+1}(x);$$

by Lemma 3, we have

$$P_{\delta'}\{\pi(d_k) = x\} = P_{\delta^0}\{\pi(d_k) = x\};$$

by Lemma 4, we have

$$P_{\delta'}\{\pi(d_{k+1}) = y|\pi(d_k) = x, d_k = a\} = P_{\delta^0}\{\pi(d_{k+1}) = y|\pi(d_k) = x, d_k = a\}.$$

We can prove $E_{\delta'}[v] > E_{\delta^0}[v]$ by the following derivation:

$$
\begin{aligned}
&E_{\delta'}[v] \\
&\quad = \sum_{x\in\Omega_{\pi(d_k)}} U_k(x,\delta') * P_{\delta'}\{\pi(d_k) = x\} \\
&\quad = \sum_{x\in\Omega_{\pi(d_k)}} \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} U_{k+1}(y,\delta') * P_{\delta'}\{\pi(d_{k+1}) = y|\pi(d_k) = x\}\Big) * P_{\delta'}\{\pi(d_k) = x\} \\
&\quad = \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} U_{k+1}(y,\delta') * P_{\delta'}\{\pi(d_{k+1}) = y|\pi(d_k) = x_0\}\Big) * P_{\delta'}\{\pi(d_k) = x_0\} \\
&\qquad + \sum_{x\in\Omega_{\pi(d_k)}, x\neq x_0} \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} U_{k+1}(y,\delta') * P_{\delta'}\{\pi(d_{k+1}) = y|\pi(d_k) = x\}\Big) * P_{\delta'}\{\pi(d_k) = x\}
\end{aligned}
$$

$$
\begin{aligned}
&= \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} U_{k+1}(y,\delta^0) * P_{\delta'}\{\pi(d_{k+1})=y|\pi(d_k)=x_0\}\Big) * P_{\delta^0}\{\pi(d_k)=x_0\} \\
&\quad + \sum_{x\in\Omega_{\pi(d_k)},x\neq x_0} \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} U_{k+1}(y,\delta^0) * P_{\delta^0}\{\pi(d_{k+1})=y|\pi(d_k)=x\}\Big) * P_{\delta^0}\{\pi(d_k)=x\} \\
&= \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\delta'}\{\pi(d_{k+1})=y|\pi(d_k)=x_0, d_k=\delta'(x_0)\}\Big) * P_{\delta^0}\{\pi(d_k)=x_0\} \\
&\quad + E_{\delta^0}[v] - \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\delta^0}\{\pi(d_{k+1})=y|\pi(d_k)=x_0\}\Big) * P_{\delta^0}\{\pi(d_k)=x_0\} \\
&= \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\delta'}\{\pi(d_{k+1})=y|\pi(d_k)=x_0, d_k=a_0\}\Big) * P_{\delta^0}\{\pi(d_k)=x_0\} \\
&\quad + E_{\delta^0}[v] \\
&\quad - \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\delta^0}\{\pi(d_{k+1})=y|\pi(d_k)=x_0, d_k=\delta_0(x_0)\}\Big) * P_{\delta^0}\{\pi(d_k)=x_0\} \\
&= \Big(\sum_{y\in\Omega_{\pi(d_{k+1})}} V_{k+1}(y) * P_{\delta^0}\{\pi(d_{k+1})=y|\pi(d_k)=x_0, d_k=a_0\}\Big) * P_{\delta^0}\{\pi(d_k)=x_0\} \\
&\quad + E_{\delta^0}[v] - V'_k(x_0,\delta^0(x_0)) * P_{\delta^0}\{\pi(d_k)=x_0\} \\
&= V'_k(x_0,a_0) * P_{\delta^0}\{\pi(d_k)=x_0\} - V'_k(x_0,\delta^0(x_0)) * P_{\delta^0}\{\pi(d_k)=x_0\} + E_{\delta^0}[v] \\
&= (V'_k(x_0,a_0) - V'_k(x_0,\delta^0(x_0)) * P_{\delta^0}\{\pi(d_k)=x_0\} + E_{\delta^0}[v] \\
&> E_{\delta^0}[v]
\end{aligned}
$$

This contradicts the optimality assumption of δ^0. □

Proof of Theorem 1.

The theorem follows from definitions of V and V', and Lemmas 1, 4, 5, 8 and 9 directly. □

LIST OF FIGURES

LIST OF TABLES