A Computational Theory of Decision Networks

(Nevin) Lianwen Zhang Runping Qi David Poole

Technical Report 93-6
March 1993

Department of Computer Science
The University of British Columbia
Vancouver, B. C. V6T 1Z2
Canada
e-mail: {lzhang, qi, poole}@cs.ubc.ca

A Computational Theory of Decision Networks

(Nevin) Lianwen Zhang Runping Qi David Poole

Technical Report 93-6
March 1993

Department of Computer Science
The University of British Columbia
Vancouver, B. C. V6T 1Z2
Canada
e-mail: {lzhang, qi, poole}@cs.ubc.ca

Abstract

This paper is about how to apply Bayesian Decision Theory to problems that involve multiple decisions and multiple variables. Our emphasis is on the computational aspects.

Decision trees (Raiffa 1968) are the first paradigm where an agent can deal with multiple decisions. The non-forgetting influence diagram formalism (Howard and Matheson 1983, Shachter 1986) improves decision trees by exploiting random variables' independencies of decision variables and other random variables. In this paper, we introduce a notion of decision networks that further explore decision variables' independencies of random variables and other decision variables. We also drop the semantic constraints of total ordering of decisions and of one value node. Only the fundamental constraint of acyclicity is kept.

From a computational point of view, it is desirable if a decision network is stepwise-solvable, i.e if it can be evaluated by considering one decision at a time. However, decision network in the most general sense need not to be stepwise-solvable. A syntactic constraint called stepwise-decomposability is therefore imposed. We show that stepwise-decomposable decision networks can be evaluated not only by considering one decision at a time, but also by considering one portion of the network at a time.

Key word: decision analysis, influence diagrams, decision networks, Bayesian networks, stepwise-decomposability, evaluation.

1 Introduction

Influence diagrams were introduced by Howard and Matheson (1981) as a way to graphically describe dependencies among random variables and decisions of a decision analysis problem. Influence diagrams without decision nodes and value nodes are called Bayesian networks (Pearl 1988). In this paper, we shall refer to influence diagrams with decision nodes and/or value nodes *decision networks* [1].

In comparison with decision trees (Raiffa 1968), decision networks are arguably more intuitive, hence are preferred for modeling; they are more structured and compact, hence allow more efficient treatments by computers. In this paper, we shall slightly modify the semantics of decision networks so that they also allow us to explore independencies among decisions and between decisions and information, while decision trees do not.

Specifically, decision networks are directed graphs with three types of nodes: decision nodes, random nodes and value nodes. Decision nodes represent decisions one wants to make. Random nodes represent random quantities relevant to the decisions. Value nodes represent the decision maker's utilities.

The basic constraint imposed on influence diagrams is acyclicity. A Bayesian network is required to be acyclic because conceptually it originates from a particular order for expanding a joint probability of all the variables of interest by the chain rule of probabilities. A decision network is required to be acyclic because it can be viewed as a Bayesian network with deterministic nodes, and because arcs into decision nodes indicate time precedence.

In addition to the basic constraint, traditionally decision networks are also required to:

1. be *regular*, i.e. all the decision nodes be ordered;
2. be *no-forgetting*, i.e. each decision node and it parents be the parents of all subsequent decision nodes; and
3. have only one value node.

[1] In the literature, influence diagrams with decision nodes and/or values nodes are again called influence diagrams. Here, we want to distinguish between the two notions. Also note that the concept of decision network defined here is different from the concept with the same name in (Howard and Matheson 1981).

Howard and Matheson (1984) need the additional constraints because of their way of computing optimal decisions. In their framework, decision networks are first transformed into decision trees and then optimal decisions are obtained for the decision trees by the standard averaging out and folding back algorithm. The additional constraints assure that the transformation from a decision network to a decision tree is possible.

The semantic justifications for the three additional constraints are as follows. Firstly, regularity follows if the decision network is to represent a single decision maker's view of the world. Secondly, arcs into decision nodes indicate information availability. If the decision maker does not forget information, it is reasonable to impose the no-forgetting constraint. Finally, the one-value-node constraint is due to the standard setup of Bayesian decision theory.

Shachter (1986) found a way to compute optimal decisions without transforming decision networks into decision trees. Other algorithms were discovered later (Shenoy 1990, Ndilikilikesha 1991). However, the additional constraints are still used in proving the correctness of the algorithms. The semantic justifications for the constraints remain the same.

In this paper, we lift the three additional constraints and present a general theory of decision networks.

Semantically, we want to lift those additional constraints for the following reasons. Firstly, we want to allow more than one decision maker. Even in the case of one decision maker, we want allow her/him the freedom of making certain decisions in an order of her/his own choice. These defeat the regularity constraint. Secondly, we shall re-interpret arcs into decision nodes as indication of both dependency and information availability, not merely of information availability. This defeats the no-forgetting constraint. Finally, we need more than one value node to maximally explore independencies between decisions and information.

Syntactically, lifting the additional constraints enables us to develop a more insightful theory of decision networks. In particular, it leads us to discover the concept of stepwise-decomposable decision networks, which proves to be a good tradeoff between representativeness and computational efficiency. It leads us to find efficient algorithms for computing optimal decisions. It also leads us to successfully exploit the possibility that some decisions are independent of some information by a preprocessing step.

The paper consists of three parts. In part I, decision networks are intu-

itively illustrated and formally defined. To gain precise understanding of the semantics, we go through the exercise of developing the concept of decision networks from the standard Bayesian decision theory setup by considering multiple decision problems. Semantic constraints are discussed to motivate the particular notion of decision networks we shall be dealing with in the rest of the paper.

Part II proposes stepwise-decomposable decision networks as a tradeoff between representation adequacy and computational efficiency. We show that stepwise-decomposable decision networks can be evaluated by considering a series of what we shall call *simple semi-Bayesian networks*, each corresponding to one decision node. The problem of evaluating simple semi-Bayesian networks are investigated and an algorithm is proposed.

All previous research on decision networks has been about no-forgetting decision networks. In this paper, we propose and study stepwise-decomposable decision networks, which are more general than no-forgetting decision networks. However, the algorithms for evaluating no-forgetting decision networks can be used to evaluate stepwise-decomposable decision networks with minor modifications. In part III, we argue that our algorithm enjoys the advantage of modularity in comparison to other algorithms. We also discuss related work and provide a summary of the paper and a plan for future research.

Part I
Bayesian networks and decision networks

In this first part of this paper, we exhibit the concepts of Bayesian networks and decision networks. The concept of decision networks is intuitively illustrated through an example in section 2 and is formally defined in section 6. Sections 3-5 are more foundational. Section 3 develops the concept of Bayesian networks from joint probabilities by means of the chain rule of probabilities, and by using the concept of conditional independencies. Section 4 arrives at decision networks from the standard Bayesian decision theory setups by considering multiple decision problems. Section 5 discusses semantic constraints on decision networks and motivates the particular notion of decision networks of interest to us in this paper.

2 Decision networks intuitively

In this section, we illustrate the concept of decision networks through an example.

Decision networks can be viewed from the qualitative level and the quantitative level. At the qualitative level, decision networks are directed graphs consisting of three types of nodes: decision nodes, random nodes and value nodes. They are used to graphically represent the structures of Bayesian decision problems.

Consider the following extension to the oil wildcatter problem of Raiffa (1968).

> The oil wildcatter is deciding whether or not to drill in a new area. To aid his decision, he can order a seismic structure test. His decision about drill will depend on the test results if a test is ordered. If the oil wildcatter does decide to drill, crude oil and natural gas will be produced. Then, the oil wildcatter will decide his gas sale policy and oil sale policy on the basis of the quality

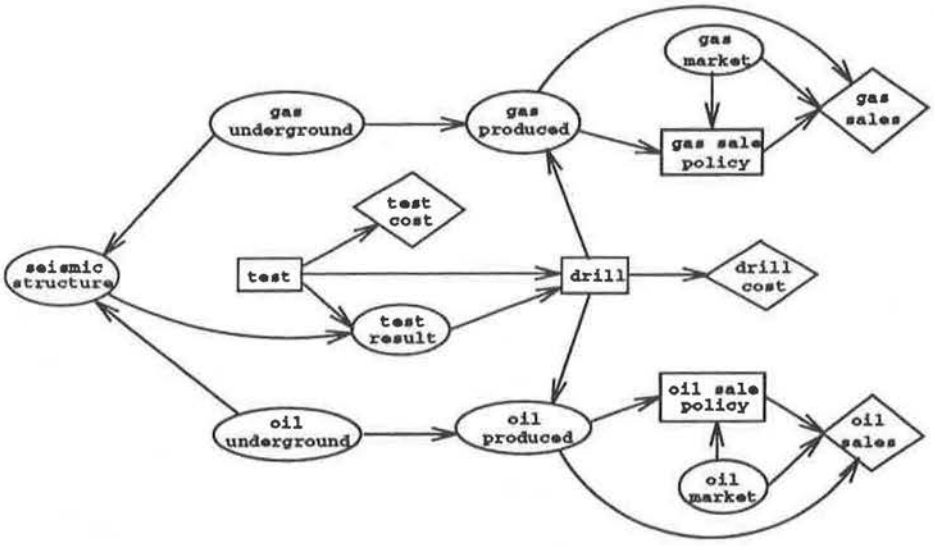

Figure 1: A decision network for the extended oil wildcatter problem.

> and quantity of crude oil and natural gas produced, and on the basis of market information.

The structures of this decision problem can be represented by the decision network shown in Figure 1, where decision nodes are drawn as rectangles, random nodes as ovals, and value nodes as diamonds.

Briefly, here is the semantics of a decision network. Arcs into random nodes indicate probabilistic dependencies. A random node depends on all its parents, and is independent of all its non-descendants given the values of its parents. In the extended oil wildcatter problem, `test-result`, for instance, probabilistically depends on `seisimic-structure` and the decision to `test`, but is independent of `gas-underground` and `oil-underground` if `seismic-structure` is given and the `test` decision has been made.

Arcs into decision nodes indicate both information availabilities and functional dependencies. In our example, the arc from `oil-produced` to `oil-sale-policy` means that the oil wildcatter will have learned the quantity and quality of crude `oil-produced` when he decides his `oil-sale-policy`, and he thinks that the quantity and quality of `oil-produced` should affect

his oil-sale-policy. There is no arc from oil-underground to oil-sale-policy because information about oil-underground is not directly available. There is no arc from test-result to oil-sale-policy, because the oil wildcatter figures that the information about the test-result should not affect his oil-sale-policy since that he will already have learned the quality and quantity of crude oil-produced at the time the policy is to be made.

Arcs into value nodes indicate functional dependencies. A value node is characterized by a function of its parents. The values of the function are real numbers representing the decision maker's utilities about the alternatives. In the extended oil wildcatter problem, oil-sales is a (utility) function of oil-produced, oil-market and oil-sale-policy. It depends on no other node. Given the quality and quantity of oil-produced, a state of oil-market, and an oil-sale-policy, the utility function gives us the expected oil-sales. The overall utility is the sum of all the value nodes. In our example, the overall utility is the sum of test-cost, drill-cost, oil-sale and gas-sale, where the costs are counted as negative utilities.

At the quantitative level, a decision network contains the conditional probabilities of all the random nodes given their respective parents and prior probabilities of the random nodes without parents, and contains a utility function for each of the value nodes.

In a decision network, decisions are made knowing the values of their parents. Optimal decisions are decisions that maximize the expected overall utility. The goal of decision analysis is to find the optimal decisions and to determine the optimal expected overall utility.

3 Bayesian networks

One way to understand decision networks is to think of them as developed from the standard Bayesian Decision Theory setup. We shall explain this in the next section. In this section, we first develop the concept of Bayesian networks from joint probabilities by means of the chain rule of probabilities and the concept of conditional independency.

Let X be a set of random variables. Let $P(X)$ be the joint probability of the variables in X. It is usually difficult, if possible at all, to assess the joint probability directly. One way to assess the joint probability indirectly is first to choose an ordering over the variable set X, say $x_1, x_2, \ldots, x_n$, then to

expand the joint probability by the chain rule as follows:

$$P(x_1, x_2, \ldots, x_n) = P(x_1)P(x_2|x_1)\ldots P(x_n|x_1, \ldots, x_{n-1}). \quad (1)$$

The ordering is called an *expansion ordering.* Because of equation (1), to assess the joint probability $P(X)$, it suffices to assess $P(x_i|x_1, \ldots, x_{i-1})$ for each $i \in \{1, \ldots, n\}$.

Often a decision maker is able to determine a proper subset $\pi(x_i)$ of $\{x_1, \ldots, x_{i-1}\}$ that are "directly related" to x_i such that other variables in $\{x_1, \ldots, x_{i-1}\}$ are only "indirectly related" to x_i via $\pi(x_i)$. Translating into the language of the probability theory, this means that x_i is independent of other variables in $\{x_1, \ldots, x_{i-1}\}$ given $\pi(x_i)$. Formally

$$P(x_i|x_1, \ldots, x_{i-1}) = P(x_i|\pi(x_i)). \quad (2)$$

This equation further reduces the assessment task.

Given an expansion ordering $x_1, \ldots, x_n$ and a π, we can construct a directed graph with the variables in X as nodes by the following rule:

> For any x_i and x_j, draw an arc from x_i to x_j if and only if $x_i \in \pi(x_j)$.

The acyclic directed graph such constructed, together with the conditional probabilities $P(x_i|\pi(x_i))$, is called a *Bayesian network* for the joint probability $P(X)$.

As an example, consider the following decision scenario which is borrowed from (Poole and Neufeld 1991). The scenario involves five variables: `alarm`, `fire`, `tampering`, `smoke`, and `leaving`, denoting respectively the following propositions: the alarm is on, there is a fire, somebody is tempering; there is smoke and people leaving. An expansion ordering for the joint probability P(`alarm`, `fire`, `tampering`, `smoke`, `leaving`) can be `fire`, `tampering`, `alarm`, `leaving`, `smoke`. Suppose it is reasonable to set π(`tampering`) $= \emptyset$, π(`alarm`) $= \{$`fire`, `tampering`$\}$, π(`leaving`) $= \{$`alarm`$\}$, and π(`smoke`) $= \{$`fire`$\}$. Then we get the Bayesian network `net2` shown in Figure 2-(1). Another expansion ordering can be `leaving`, `alarm`, `smoke`, `fire`, `tampering`. Suppose it is reasonable to set π(`alarm`) $= \{$`leaving`$\}$, π(`smoke`) $= \{$`alarm`$\}$, π(`fire`) $= \{$`alarm`, `smoke`$\}$, and π(`tampering`) $= \{$`fire`, `alarm`$\}$. Then we get the Bayesian network `net3` shown in Figure 2-(2). This network has more arcs than the one in `net2`.

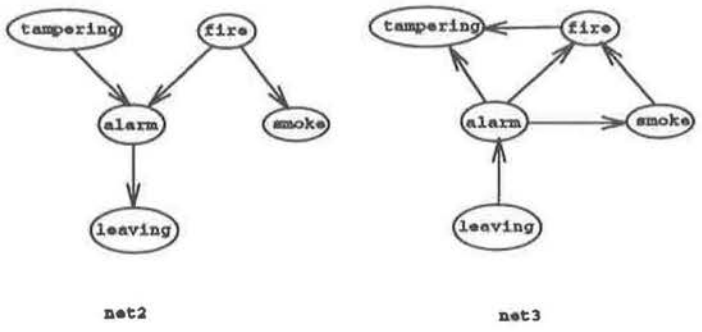

Figure 2: Two Bayesian networks for the joint probability $P(alarm, fire, tampering, smoke, leaving)$.

How should one choose an expansion ordering? The answer provided by Howard and Matheson (1984) is that the ordering should be chosen such that the decision maker would feel natural and comfortable in assessing the $\pi(x_i)$'s and the $P(x_i|\pi(x_i))$'s. For example, it probably is easier to assess $P(\texttt{alarm}|\texttt{fire}, \texttt{tampering})$ than to assess $P(\texttt{tampering}|\texttt{fire}, \texttt{alarm})$. Smith (1989) says that one should choose the ordering to minimize the number of arcs in the resulting directed graph. In our example, `net2` is preferred. Pearl (1988, pp. 50-51) claims that when there are cause-effect relationships among the variables, the structure of a Bayesian network can be directly determined from the cause-effect relationships. For example, `tampering` and `fire` cause `alarm`, `fire` causes `smoke`, `alarm` causes `leaving`.

4 Decision networks

In this section, the concept of decision networks is derived from an attempt to solve multiple decision problems in the framework of the Bayesian decision theory.

The standard setup of decision theory (Gärdenfors 1988b, Fishburn 1988) includes a set of (random and decision) variables X, a utility function $\mu(X)$, and a set of probabilities $\{P_\delta(X)\}$ indexed by a parameter δ [2]. The decision is

[2]If there is only one decision variable, δ is a mapping from the the Cartesian product of the frames of the observed variables to the frame of that decision variable. Otherwise,

to choose a value for δ. The principle of maximizing expected utility states that the decision maker should choose a value δ^0 such that the expected utility is maximized:

$$\sum_X P_{\delta^0}(X)\mu(X) = max_\delta\{\sum_X P_\delta(X)\mu(X)\}. \quad (3)$$

In the above equation, summation is used instead of integration because we deal only with discrete variables in this paper. However, most of the results can be easily extended to the case of continuous variables.

Consider the case when the decision maker needs to decide on the values for a number of variables $d_1, \ldots, d_k$. Let $OBS(d_i)$ denote the set of all the variables whose values will become known to the decision maker at the time of decision d_i is to be made. Let $\pi_0(d_i)$ be a subset of $OBS(d_i)$, such that the variables in $OBS(d_i) - \pi_0(d_i)$ are, according to the decision maker, irrelevant to the decision given $\pi_0(d_i)$. The value for each d_i is to be set based on the values of variables in $\pi_0(d_i)$. We call such a problem a *multiple decision problem*, and write it as as $\mathcal{D} = \{< d_i, \pi_0(d_i) > | 1 \leq i \leq k\}$.

The extended oil wildcatter problem is a multiple decision problem. The decision maker needs to set the value of `test`, the value of `drill` based on `test-result`, the value of `gas-sale-policy` based on the values of `gas-produced` and `gas-market`, and the value of `oil-sale-policy` based on the values of `oil-produced` and `oil-market`.

Given a multiple decision problem $\mathcal{D}$, let X be the set of all the variables in $\mathcal{D}$ and other variables that are relevant to the problem. For any variable $x \in X$, let Ω_x be the *frame* of x, i.e. the set of all possible values of x. For any subset $B \subseteq X$, let $\Omega_B = \prod_{x \in B} \Omega_x$.

To determine a value for d_i based on the values of the variables in $\pi_0(d_i)$ is to choose a function $\delta_i : \Omega_{\pi_0(d_i)} \rightarrow \Omega_{d_i}$. Such a function is called a *decision function (table)* for d_i. Let Δ_i denote the set of all the decision functions for d_i. The *policy space* of the problem is the Cartesian product $\Delta = \prod_{i=1}^k \Delta_i$. An element of Δ is called a *policy*.

A probability P is *consistent* with a policy δ if for each d_i, $P(d_i|\pi(d_i))$ is 1 when $d_i = \delta(\pi(d_i))$, and 0 otherwise.

A policy $\delta \in \Delta$ determines a set of probabilities over X, namely the set of probabilities that are consistent with δ. To solve the multiple decision

it is a tuple of such mappings, each of which corresponds to one decision variable. Each δ determines a probability P_δ over X.

problem in the foregoing Bayesian decision theory setup, one needs one single probability. An interesting question is: what knowledge and beliefs it takes for one to to be able to single out one probability P_δ from the set?

Since δ asserts that d_i is a function of $\pi_0(d_i)$, $P_\delta(X)$ must fulfill:

$$P_\delta(d_i|B, \pi_0(d_i)) = P_\delta(d_i|\pi_0(d_i)) \tag{4}$$

for any subset $B \subseteq X$ that does not contain any descendents of d_i.

An expansion ordering for $P_\delta(X)$ *conforms to* $\mathcal{D}$ if for each d_i, variables in $\pi_0(d_i)$ precede d_i in the ordering. As we shall see in the next section, such an ordering is possible due to the semantic constraints on multiple decision problems.

Given an expansion ordering $x_1, \ldots, x_n$ that conforms to $\mathcal{D}$, we can expand $P_\delta(X)$, determine the π function, and construct a Bayesian network, denoted by $\mathcal{N}_\delta$, as in the previous section.

Because of equation (4), it must be the case that $\pi(d_i) = \pi_0(d_i)$. In words, the parents of d_i in $\mathcal{N}_\delta$ are the nodes in $\pi_0(d_i)$.

In the Bayesian network $\mathcal{N}_\delta$, the conditional probability of each d_i given its parents $\pi(d_i)$ is given by the policy δ. However, the network structure and the conditional probabilities of variables other than the decision variables need to be elicited by the decision maker from his knowledge and beliefs. We call this part of $\mathcal{N}_\delta$ a *network of beliefs* [3] for the multiple decision problem. In other words, the network of beliefs is the "difference" between $\mathcal{N}_\delta$ and δ.

In order to solve a multiple decision problem in the framework of Bayesian decision theory, the decision maker needs also to express his preferences by a utility function $\mu(X)$. Let $B \subseteq X$ be the set of variables of which U is an explicit function. We add $\mu(B)$ into the network of beliefs by creating a node v for $\mu(X)$ and drawing an arc to v from each of the nodes in B. This results in a *decision network* for the multiple decision problem. It specifies the beliefs and the utilities that have to be elicited before the problem can be solved in the framework of Bayesian decision theory.

As an example, consider a decision scenario where a decision maker needs to decide whether to `bring-umbrella` based on weather `forecast`. An additional variable, `rain`, which takes the value "yes" if it does turn out to

[3] In the literature, authors sometimes refer to Bayesian networks as belief networks. Our concept of networks of beliefs is consistent with the usage of the term belief network except that our concept is more general. To us, a network of beliefs contains not only random variables, but also decision variables.

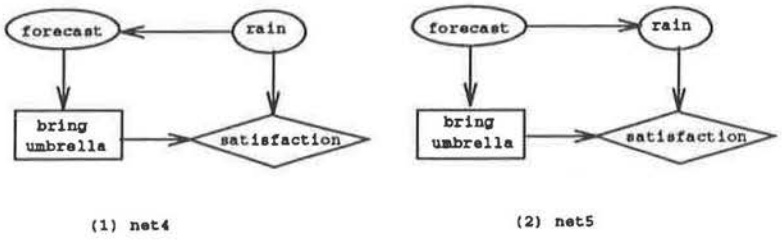

Figure 3: Two decision networks for the rain and umbrella problem.

rain and "no" otherwise, is believed to be relevant to the decision and is included in our analysis. A decision function $\delta : \Omega_{\texttt{forecast}} \rightarrow \Omega_{\texttt{bring-umbrella}}$ determines a joint probability $P_\delta(\texttt{rain}, \texttt{forecast}, \texttt{bring} - \texttt{umbrella})$. The expansion ordering `rain`, `forecast`, `bring-umbrella` conforms to the decision problem. It gives rise to the decision network `net4` shown in Figure 3 (1). The expansion ordering `forecat`, `rain`, `bring-umbrella` also conforms to the decision problem. It gives rise to the decision network `net5` also shown on Figure 3. It is easy to see that one can go between those two networks by reversing the arc between `forecast` and `rain` using Bayesian theorem (see Howard and Matheson 1984 and section 11).

To end this section, let us point out that there may be more than one expansion ordering for $P_\delta(X)$ that conform to $\mathcal{D}$. Thus there may be more than one network of beliefs for $\mathcal{D}$. A question is whether two such networks demand the same amount of information. The answer is positive. Because we can start from one and arrive at another by fitting in an arbitrary policy δ to get the joint probability $P_\delta(X)$, and expanding $P_\delta(X)$ using the ordering of the second network, and removing the $P(d_i|\pi(d_i))$'s.

5 Semantic constraints

In this section, we discuss semantic constraints on decision networks. We begin with acyclicity.

5.1 Acyclicity

A Bayesian network is a directed graph, and it originates from an expansion ordering for a joint probability. If there is an arc $x \rightarrow y$, then x comes earlier than y in the expansion ordering. Therefore a Bayesian network must be acyclic.

A decision network is also a directed graph, and it is obtained from a network of beliefs by adding one value node. As Bayesian networks, a network of beliefs originates from an expansion ordering for a joint probability. So, it must be acyclic. Consequently, decision networks must also be acyclic.

There is one issue about the acyclicity of decision networks. In section 4, we have assumed the existence of an expansion ordering for $P_\delta(X)$ that conforms to a given multiple decision problem $\mathcal{D} = \{< d_i, \pi_0(d_i) > | 1 \leq i \leq k\}$. We need to justify this assumption.

Construct a directed graph G over X from $\mathcal{D}$ by, for each d_i, drawing an arc from each variable in $\pi_0(d_i)$ to d_i. This directed graph must be acyclic because of the time precedence relationships implied by $\mathcal{D}$. In fact, the value for d_i is to be set knowing the values of the variables in $\pi_0(d_i)$. Thus the values of the variables in $\pi_0(d_i)$ must be determined, either by nature or by the decision maker, before the decision about d_i is made. In this sense we say variables in $\pi_0(d_i)$ precede d_i in time. If there were a cycle $d'_1, d'_2, \ldots, d'_j$ in G, then d'_1 precedes d'_2 in time, $\ldots$, d'_{j-1} precedes d'_j in time, and d'_j precedes d'_1 in time. This is impossible. Therefore, G must be acyclic.

It can be easily shown by induction that in a directed acyclic graph G, there exists at least one ordering of the nodes of G such that if $x \rightarrow y$ is an arc of G, then x precedes y in the ordering. Let s be such an ordering for the directed graph defined in the previous paragraph. Then s is an expansion ordering for $P_\delta(X)$ that conforms to $\mathcal{D}$.

5.2 Ordering of decisions

Given a multiple decision problem $\mathcal{D} = \{< d_i, \pi_0(d_i) > | 1 \leq i \leq k\}$, there is a natural partial ordering among the decision nodes. A decision node d_j *precedes* another decision node d_i in $\mathcal{D}$ if $d_j \in \pi_0(d_i)$, or d_j precedes d_k and $d_k \in \pi_0(d_i)$. This ordering is not necessarily a total ordering, for there may exist d_i and d_j such that neither d_i precedes d_j nor d_j precedes d_i.

If there is only one decision maker, then the decisions are made sequen-

tially. If the order by which the decisions are to be made is known beforehand, then it is reasonable to require that the decision network be *regular*, i.e all the decision nodes be ordered. The previous literature only deals with regular decision networks.

However, there may be cases where there are more than one decision maker. In the extended oil wildcatter example, it may well be the case that the `test` and `drill` decisions are made by the company headquarter, while the `gas sale policy` is made by the gas department and the `oil-sale-policy` is made by the oil department. If there are more than one decision maker, it is not reasonable to require the decision network be regular.

Even in the case of one decision maker, one may not be sure about the order by which the decisions are to be made a priori, but it is known that the order in which these two decisions are made is irrelevant to the optimal expected value. In the extended oil wildcatter example, the decision maker may choose to make the `gas-sale-policy` first, or he may choose to make the `oil-sale-policy` first, because he knows that the order is not important. In such case, it is more appropriate to leave the choice to the decision maker himself rather than make the choice for him by imposing the regularity constraint.

In this paper, the regularity constraint is lifted. A merits of doing so is that it allows one to explore more independencies between decisions and information. The reader will see this in the next subsection.

5.3 Dependencies of decisions on information

In a multiple decision problem $\mathcal{D} = \{< d_i, \pi_0(d_i) > | 1 \leq i \leq k\}$, $\pi_0(d_i)$ is the set of all the variables whose values are known at the time the decision d_i is to be made and whose values are, according to the decision maker, relevant to the decision d_i. In this sense, we say that π_0 indicates both information availability and dependency.

In the previous literature (Howeard and Matheson 1984), π_0 indicates only information availability. In other word, $\pi_0(d_i)$ is defined to be the set of all the variables whose values are known at the time the decision d_i is to be made. If there is only one decision maker and the decision maker does not forget information, then for any decision d_j that precedes d_i in $\mathcal{D}$, it must be the case that $d_j \in \pi_0(d_i)$ and $\pi_0(d_j) \subset \pi_0(d_i)$. A multiple decision problem

with this property is said to be *no-forgetting*. A decision network for a no-forgetting multiple decision problem is a *no-forgetting* decision network. The previous literature only deals with no-forgetting decision networks.

However, there are reasons for letting π_0 indicate both information availability and dependency. First of all, the decision maker sometimes is able to tell that a decision does not depend on a particular variable. In the extended oil wildcatter problem, the decision maker may declare that `test-result` is irrelevant to the decision `oil-sale-policy` if the quality and quantity of `oil-produced` are known. In general, letting π_0 to indicate both information availability and dependency enables the representation of knowledge and belief about a piece of information being irrelevant to a decision.

Secondly, it is more difficult to identify the set *OBS* itself than to identify a subset π_0 such that variables in $OBS - \pi_0$ are (subjectively) irrelevant to the decision under consideration. In the extended oil wildcatter example, the decision maker may have difficulties in determining whether `gas-sale-policy` will be available at the time `oil-sale-policy` is to be made. Nonetheless, the decision maker may judge that `gas-sale-policy` is irrelevant, and he can thus leave `gas-sale-policy` out side $\pi_0(\texttt{oil} - \texttt{sale} - \texttt{policy})$ regardless its availability.

Thirdly, if we require a multiple decision problem to be no-forgetting, then when we have a long chain of decision nodes, the $\pi_0(d)$ for decision nodes toward the end of the chain may consist of a large number of variables. This implies huge decision tables. However, as has already been shown (Shachter 1988, Zhang and Poole 1992), many arcs in a no-forgetting decision network are irrelevant and hence can be harmlessly removed. Letting π_0 indicate both information availability and dependency provides a semantics for the remaining networks after removing the irrelevant arcs from a no-forgetting decision network.

Finally, letting π_0 indicate both information availability and potential dependency uniforms the semantics for arcs into decision nodes and arcs into random nodes (Smith 1988).

In this paper, we choose to let π_0 indicate both information availability and dependency. This necessitates the lifting of the no-forgetting constraint.

5.4 Multiple value nodes

In section 4, one value node v was introduced to represent the decision maker's utility function $mu(X)$. As pointed out by Tatman and Shachter (1991), in many applications, $\mu(X)$ can usually be decomposed into the sum of a number of components, each of which only depends on a small number of variables. In the extended oil wildcatter example, the overall utility function can be decomposed into four components, namely `test-cost`, `drill-cost`, `gas-sales`, and `oil-sales`.

Suppose $\mu(X)$ decomposes into m components $v_1(Z_1) + \ldots + v_m(Z_m)$, we replace the value node v by m value nodes $v_1, \ldots, v_m$, where v_i corresponds to $f_{v_i}(Z_i)$, and we draw arcs from each of the variables of Z_i to v_i. This results in a decision network with m value nodes.

One advantage of having multiple value nodes is that it allow us to explore more independencies. In Figure 1, for instance, the node `oil-produced` separates `oil-sale-policy`, `oil-market`, and `oil-sales` from the rest of the network. The same would not be true if only one value node is allowed.

6 Formal definitions

In this section, we given the formal definitions of Bayesian networks and decision networks. Before getting started, let us note that in this paper, standard graph theory terms such as connected acyclic directed graphs, parents (direct predecessors), children (direct successors), predecessors, descendants (successors), leaves (nodes with no children), and roots (nodes with no parents) will be used without giving the definitions. The reader is directed to Lauritzen *et al* (1990) for exact definitions. We shall use $\pi(x)$ to denote the set of parents of a node x in a directed graph.

A *Bayesian network* $\mathcal{N}$ is a triplet $\mathcal{N} = (Y, A, \mathcal{P})$, where

1. Y is a set of variables (nodes) and A is a set of arcs over Y such that (Y, A) is a connected acyclic directed graph,

2. $\mathcal{P}$ is a set $\{P(x|\pi(x))|x \in Y\}$ of conditional probabilities of the variables given their respective parents[4].

[4]Note that when x is a root, $\pi(x)$ is empty. When it is the case, $P(x|\pi(x))$ stands for the prior probability of x.

The *prior joint probability* $P_{\mathcal{N}}(Y)$ of a Bayesian network $\mathcal{N} = (Y, A, \mathcal{P})$ is defined by

$$P(Y) = \prod_{x \in Y} P(x|\pi(x)). \tag{5}$$

For any subset $B \subseteq Y$, the *marginal probability* $P(B)$ is defined by

$$P(B) = \sum_{Y-B} P(Y). \tag{6}$$

For any two subset $B_1, B_2 \subseteq Y$, the *conditional probability* $P(B_1|B_2)$ is a number such that

$$P(B_1) = P(B_2)P(B_1|B_2). \tag{7}$$

A *decision network skeleton* is a connected acyclic directed graph $G = (X, A)$ whose nodes are are partitioned into *random nodes*, *decision nodes*, and *values nodes*, and that the value nodes have no children.

A *decision network* $\mathcal{N}$ is a quadruplet $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F})$ where

1. (X, A) is a decision network skeleton with random node set C, decision node set D and value node set U.

2. $\mathcal{P}$ is a set $\{P(c|\pi(c))|c \in C\}$ of conditional probabilities of the random nodes given their respective parents.

3. $\mathcal{F}$ is a set $\{f_v : \Omega_{\pi(v)} \rightarrow R^1 | v \in U\}$ of *value (utility) functions* for the value nodes, where R^1 stands for the real line.

A *decision function (table)* for a decision node d_i is a mapping $\delta_i : \Omega_{\pi(d_i)} \rightarrow \Omega_{d_i}$. The *decision function space* Δ_i for d_i is the set of all the decision functions for d_i. Let $D = \{d_1, \ldots, d_k\}$ be the set of all the decision nodes. The Cartesian product $\Delta = \prod_{i=1}^{k} \Delta_i$ is called the *policy space* for $\mathcal{N}$, and a member $\delta = (\delta_1, \ldots, d_k) \in \Delta$ is called a *policy*.

The relationship between a decision node d_i and its parents $\pi(d_i)$ as indicated by a decision function $\delta_i : \Omega_{\pi(d_i)} \rightarrow \Omega_{d_i}$ is equivalent to the relationship as represented by the conditional probability $P_{\delta_i}(d_i|\pi(d_i))$ given by

$$P_{\delta_i}(d_i = \alpha|\pi(d_i) = \beta) = \begin{cases} 1 & \text{if } \delta_i(\beta) = \alpha \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

for all $\alpha \in \Omega_{d_i}$ and $\beta \in \Omega_{\pi(d_i)}$. One can see that if viewed as a function of d_i, $P_{\delta_i}(d_i|\pi(d_i) = \beta)$ is the characteristic function of the set $\{d_i|d_i = \delta_i(\beta)\}$. In formula that is

$$P_{\delta_i}(d_i|\pi(d_i) = \beta) = \chi_{\{d_i|d_i=\delta_i(\beta)\}}(d_i). \tag{9}$$

Since $\delta = (\delta_1, \ldots, \delta_k)$, we sometimes write $P_\delta(d_i|\pi(d_i))$ for $P_{\delta_i}(d_i|\pi(d_i))$. Because of equation (8), we will abuse the symbol δ by letting it also denote the set $\{P_\delta(d_i|\pi(d_i))|d_i \in D\}$ of conditional probabilities of the decision nodes.

In a decision network $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F})$, let $Y = C \cup D$. Let A_Y be the set of all the arcs of A that lie completely in Y. Then the triplet $(Y, A_Y, \mathcal{P} \cup \delta)$ is a Bayesian network [5]. We shall refer to this Bayesian network the *Bayesian network induced from $\mathcal{N}$ by the policy δ*, and write it as $\mathcal{N}_\delta$. The prior joint probability $P_\delta(Y)$ is given by

$$P_\delta(Y) = \prod_{x \in C} P(x|\pi(x)) \prod_{x \in D} P_\delta(x|\pi(x)). \tag{10}$$

Because the value nodes do not have children, for any value node v, $\pi(v)$ contains no value nodes. Hence $\pi(v) \subseteq Y$. Write Z for $\pi(v)$. The expectation $E_\delta[v]$ of the value function $f_v(Z)$ under P_δ is given by

$$E_\delta[v] = \sum_Y P_\delta(Y) f_v(Z).$$

The *expected value* $E_\delta[\mathcal{N}]$ of $\mathcal{N}$ under the policy δ is defined by

$$E_\delta[\mathcal{N}] = \sum_{v \in U} E_\delta[v] \tag{11}$$

$$= \sum_Y P_\delta(Y) \sum_{v \in U} f_v(Z). \tag{12}$$

The *optimal expected value* $E[\mathcal{N}]$ of $\mathcal{N}$ is defined by

$$E[\mathcal{N}] = max_{\delta \in \Delta} E_\delta[\mathcal{N}]. \tag{13}$$

The optimal value of a decision network that does not have any value nodes is zero. An *optimal policy* δ° is one that satisfies

$$E_{\delta^\circ}[\mathcal{N}] = E[\mathcal{N}]. \tag{14}$$

[5] Remember that δ denotes a set of conditional probabilities for the decision nodes.

For a decision network that does not have value nodes, all policies are optimal.

In this paper we shall only consider variables with finite frames. Hence there are only finite possible policies. Consequently, there always exists at least one optimal policy. *To evaluate a decision network* is to

1. find an optimal policy, and
2. find the optimal expected value.

Part II
Evaluating decision networks

To evaluate a decision network $\mathcal{N}$, one needs, in the general case, to compute the expected value $E_\delta[\mathcal{N}]$ of $\mathcal{N}$ under each policy δ, and to search through the entire policy space Δ for a policy that yields the maximum expectation. This may be computationally expensive, because the policy space may be huge, and for each policy δ, the computation of $E_\delta[\mathcal{N}]$ is global in the sense that it potentially involves all the nodes of the network.

So, it is desirable to impose some technical constraints on decision networks for computational efficiency. However, more constraints means less representation power. In this part, we propose the concept of stepwise-decomposable decision networks as a tradeoff between representation power and computational efficiency and present algorithms for solving stepwise-decomposable decision networks.

The organization of this part is as follows. Section 7 defines the technical concepts of semi-Bayesian networks and semi-decision networks. Stepwise-decomposable decision networks are introduced in section 8. In section 9, we shall show that a smooth stepwise-decomposable decision network can be evaluated by dealing with a number of simple semi-decision networks, each corresponding to one decision node of the original network. Section 10 discusses the problem of evaluating simple semi-decision networks. Section 11 demonstrates how to transform a decision network that is not smooth into one that is smooth.

7 Semi-Bayesian networks and semi-decision networks

In this section, we define the technical concepts of semi-Bayesian networks and semi-decision networks. We will be needing them later. The reader may wish to skip this section for now and come back to it when necessary.

A *semi-Bayesian network* is a Bayesian network except that the prior probabilities of some of the root nodes are missing. More precisely, a semi-Bayesian network is a quadruplet $\mathcal{N} = (Y, A, \mathcal{P} : S)$, where (Y, A) is a

connected acyclic directed graph, $\mathcal{P} = \{P(x|\pi(x))|x \in Y - S\}$ is set of conditional probabilities, and S is a set of root nodes whose prior probabilities are missing.

As in Bayesian networks, we can define $P_{\mathcal{N}}(Y)$ as follows,

$$P_{\mathcal{N}}(Y) = \prod_{x \in (Y-S)} P(x|\pi(x)). \tag{15}$$

Unlike in Bayesian networks, here $P_{\mathcal{N}}(Y)$ usually is not a probability. It may not sum to one. Thus, it is called the *prior joint potential* instead of the prior joint probability. Marginal potentials can be defined from the joint potential in the same way as marginal probabilities are defined from joint probabilities.

Note that as there are no arcs from $Y - S$ to S, the prior joint potential $P_{\mathcal{N}}(Y)$ is nothing but the conditional probability of the variables in $Y - S$ given S.

A *semi-decision network* is a decision network except that the prior probabilities of some of the random root nodes are missing. We use $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F} : S)$ to denotes a semi-decision network, where S is the set of random root nodes whose prior probabilities are missing.

Let $Y = C \cup D$ be the set of random and decision nodes. Similar to the case of decision networks, a policy δ induces a semi-Bayesian network $(Y, A_Y, \mathcal{P} \cup \delta : S)$, which we shall refer to as the *semi-Bayesian network induced from $\mathcal{N}$ by the policy δ*, and which will be written as $\mathcal{N}_\delta$. Let $P_\delta(Y)$ be the prior joint potential of $\mathcal{N}_\delta$.

For any value node $v \in U$, $\pi(v) \subseteq Y$. The *expected value* $E_\delta[\mathcal{N}]$ of $\mathcal{N}$ under the joint potential $P_\delta(Y)$ is defined by

$$E_\delta[\mathcal{N}] = \sum_Y P_\delta(Y) \sum_{v \in U} f_v(\pi(v)). \tag{16}$$

The *conditional expected value* [6] $E_\delta[\mathcal{N}|S]$ of $\mathcal{N}$ given S is defined by

$$E_\delta[\mathcal{N}|S] = \sum_{Y-S} P_\delta(Y) \sum_{v \in U} f_v(\pi(v)). \tag{17}$$

A semi-decision network $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F} : S)$ is *simple* if it contains exactly one decision node d and $\pi(d) = S$.

[6] This choice of term is because of the note we made earlier in this section right after the definition of semi-Bayesian networks.

Lemma 1 *Suppose $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F} : S)$ is a simple semi-decision network. Then for any value $\beta \in \Omega_S$, $E_\delta[\mathcal{N}|S = \beta]$ only depends on the value $\delta(\beta)$, and is independent of $\delta(\alpha)$ for any other $\alpha \in \Omega_S$.*

Proof: Since $\pi(d) = S$, we can write $P_\delta(d|S)$ for $P_\delta(d|\pi(d))$. The joint potential $P_\delta(Y)$ is given by

$$P_\delta(Y) = P_\delta(d|S = \beta) \prod_{x \in Y-S, x \neq d} P(x|\pi(x)).$$

Therefore

$$E_\delta[\mathcal{N}|S = \beta] = \sum_{Y-S} P_\delta(d|S = \beta) \prod_{x \in Y-S, x \neq d} P(x|\pi(x)) \sum_{v \in U} f_v(\pi(v)). \quad (18)$$

The only term that contains δ is $P_\delta(d|S = \beta)$, which only depends on the value of δ at β according to equation (9). Therefore, the lemma is proved. □

Because of the lemma, we sometimes write $E_\delta[\mathcal{N}|S = \beta]$ as $E_{\delta:\delta(\beta)=\alpha}[\mathcal{N}|S = \beta]$ to signify the fact that it only depends on the value α of δ at β.

Proposition 1 *Suppose $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F} : S)$ is a simple semi-decision network. For any $\delta^\circ \in \Delta$,*

$$E_{\delta^\circ}[\mathcal{N}] = max_{\delta \in \Delta} E_\delta[\mathcal{N}], \quad (19)$$

if and only if for any value $\beta \in \Omega_S$

$$E_{\delta^\circ}[\mathcal{N}|S = \beta] = max_\delta E_{\delta \in \Delta}[\mathcal{N}|S = \beta]. \quad (20)$$

Proof: From equations (16) and (17), we see that

$$E_\delta[\mathcal{N}] = \sum_{\alpha \in \Omega_S} E_\delta[\mathcal{N}|S = \alpha].$$

So, equation (20) implies (19).

We now show that (19) implies (20). For the sake of contradiction, assume there were a policy δ° that satisfies (19) which did not satisfy (20). Then, there must exist another policy δ_1 and a value $\beta \in \Omega_S$ such that

$$E_{\delta^\circ}[\mathcal{N}|S = \beta] < E_{\delta_1}[\mathcal{N}|S = \beta].$$

Construct a new policy δ_2 which is the same as δ_1 at β and which is the same as δ° at all other values of S. By Lemm 1, $E_{\delta_2}[\mathcal{N}|S=\beta] = E_{\delta_1}[\mathcal{N}|S=\beta]$, and $E_{\delta_2}[\mathcal{N}|S=\alpha] = E_{\delta_0}[\mathcal{N}|S=\alpha]$ for any other $\alpha \in \Omega_S$. Hence, we have

$$\begin{aligned}
E_{\delta_2}[\mathcal{N}] &= \sum_{\alpha\in\Omega_S} E_{\delta_2}[\mathcal{N}|S=\alpha]. \\
&= E_{\delta_1}[\mathcal{N}|S=\beta] + \sum_{\alpha\in\Omega_S,\alpha\neq\beta} E_{\delta^\circ}[\mathcal{N}|S=\alpha] \\
&> E_{\delta^\circ}[\mathcal{N}|S=\beta] + \sum_{\alpha\in\Omega_S,\alpha\neq\beta} E_{\delta^\circ}[\mathcal{N}|S=\alpha] \\
&= \sum_{\alpha\in\Omega_S} E_{\delta^\circ}[\mathcal{N}|S=\alpha] \\
&= E_{\delta^\circ}[\mathcal{N}].
\end{aligned}$$

A contradiction. Therefore, it must be the case that (19) implies (20). The proposition is proved. □

Corollary 1 *Let $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F} : S)$ be a simple semi-decision network and let d be the only decision node. Then an optimal policy δ° for $\mathcal{N}$ can be found pointwise by*

$$\delta^\circ(\beta) = arg\ max_{\alpha\in\Omega_d} E_{\delta:\delta(\beta)=\alpha}[\mathcal{N}|S=\beta]. \tag{21}$$

8 Stepwise-decomposable decision networks

In this section, we introduce stepwise-decomposable decision networks and define some relevant terms. We begin with the concept of moral graph.

Let $G = (X, A)$ be a directed graph. An arc from x to y is written as an ordered pair (x, y). The *moral graph* $m(G)$ of G is an undirected graph $m(G) = (X, E)$ whose edge set E is given by

$$E = \{\{x, y\} | (x, y) \text{ or } (y, x) \in A, \text{ or } \exists z \text{ such that } (x, z) \text{ and } (y, z) \in A\}.$$

In words, $\{x, y\}$ is an edge in the moral graph if either there is an arc between the two vertices or they share a common child. The term moral graph was chosen because two nodes with a common child are "married" into an edge (Lauritzen and Speigehalter 1988).

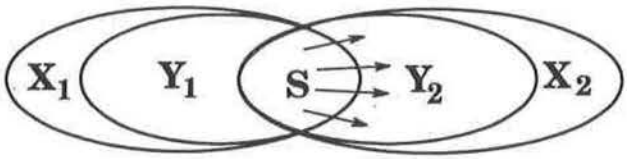

Figure 4: The relationships among the sets X, X_1, X_2, Y_1, Y_2, and S. The three sets X_1, X_2 and S constitute a partition of X — the set of all the nodes, while Y_1, Y_2 and S constitute a partion of $C \cup D$ — the set of all the random and decision nodes. When the network is smooth at d, there are no arcs going from Y_2 to S.

In an undirected graph, two nodes x and y are *separated* by a set of nodes S if every path connecting them contains at least one node in S. In a directed graph G, x and y are *m-separated* by S if they are separated by S in the moral graph $m(G)$. Note that for any node set S, it separates itself from any other set.

Suppose $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F})$ is decision network and d is a decision node of $\mathcal{N}$. Let X_1 be the set of all the nodes that are m-separated from d by $\pi(d)$, with $\pi(d)$ excluded, and X_2 be the set of all the nodes that are not m-separated from d by $\pi(d)$. We observe that $X_2 = X - (\pi(d) \cup X_1)$. Let $Y_1 = X_1 \cap (C \cup D)$ and $Y_2 = X_2 \cap (C \cup D)$. For the sake of simplicity, we shall write S for $\pi(d)$. The relationships among the sets are illustrated in Figure 4. In the sequel, we shall refer to X_1 as the *upstream set* of $\pi(d)$, X_2 as the *downstream set* of $\pi(d)$. We shall also refer to Y_1 as the set of random and decision nodes in the upstream of $\pi(d)$, and Y_2 as the set of random and decision nodes in the downstreams of $\pi(d)$.

M-separation implies conditional independence (Lauritzen *et al* 1990). Since S m-separates Y_1 and Y_2, we have for any policy δ of $\mathcal{N}$ that $P_\delta(Y_2|Y_1, S) = P_\delta(Y_2|S)$. Therefore

$$\begin{aligned} P_\delta(Y_1, S, Y_2) &= P_\delta(Y_1, S) P_\delta(Y_2|Y_1, S) \\ &= P_\delta(Y_1, S) P_\delta(Y_2|S). \end{aligned} \tag{22}$$

A decision network is *smooth* at the decision node d, if there are no arcs going from nodes in Y_2 to nodes in S. In other words, arcs between S and Y_2 only go from S to Y_2.

The decision network skeleton for the extended oil wildcatter problem (Figure 1) is not smooth because of the arc from `seismic-structure` to `test-result`.

Let $d_1, \ldots, d_j$ be all the decision nodes in $Y_1 \cup S$ and $d_{j+1}, \ldots, d_k$ be all the decision nodes in Y_2. Note that $d \in \{d_{j+1}, \ldots, d_k\}$. For a policy $\delta = (\delta_1, \ldots, \delta_k)$, let $\delta_I = (\delta_1, \ldots, \delta_j)$ and $\delta_{II} = (\delta_{j+1}, \ldots, \delta_k)$.

Lemma 2 *If the decision network $\mathcal{N}$ is smooth at d, then we have*

$$P_\delta(Y_1, S) = \prod_{x \in C \cap (Y_1 \cup S)} P(x|\pi(x)) \prod_{i=1}^{j} P_{\delta_i}(d_i|\pi(d_i)), \tag{23}$$

$$P_\delta(Y_2|S) = \prod_{x \in C \cap Y_2} P(x|\pi(x)) \prod_{i=j+1}^{k} P_{\delta_i}(d_i|\pi(d_i)). \tag{24}$$

So, $P_\delta(Y_1, S)$ only depends on δ_I, and $P_\delta(Y_2|S)$ only depends on δ_{II}. We shall write them as $P_{\delta_I}(Y_1, S)$ and $P_{\delta_{II}}(Y_2|S)$ respectively.

Let d be a decision node in a decision network $\mathcal{N}$. Denoted by $\mathcal{N}_{II}$, the *tail* of $\mathcal{N}$ w.r.t d is a semi-decision network over $S \cup X_2$. The graphical structure among the nodes is the same as in $\mathcal{N}$ except that all the arcs among nodes in S are removed. The conditional probabilities and the value functions for nodes in X_2 are the same as in $\mathcal{N}$. The nodes in S are viewed as random nodes, and their prior probabilities are missing.

Denoted by $\mathcal{N}_I$, the *body* of $\mathcal{N}$ w.r.t d is a decision network over $X_1 \cup S \cup \{v\}$. The graphical structure among nodes in $X_1 \cup S$ is the same as in $\mathcal{N}$, so are the conditional probabilities and the values functions. The newly introduced node v is a value node, whose parent set is $\pi(v) = S$, and whose value function $f_v(S)$ is set to be the optimal conditional expected value $E[\mathcal{N}_{II}|S]$ of the tail $\mathcal{N}_{II}$.

In Figure 5, the concepts of body and tail are illustrated by using the extended oil wildcatter problem.

Given a policy $\delta_{II} = (\delta_{j+1}, \ldots, \delta_k)$, the tail $\mathcal{N}_{II}$ becomes a semi-Bayesian network. Let $P_{\mathcal{N}_{II}, \delta_{II}}(S, Y_2)$ denote the joint potential of this semi-Bayesian network. From the definition of tails one can see that $P_{\mathcal{N}_{II}, \delta_{II}}(S, Y_2)$ is qual to the right hand side of equation (24). Therefore

$$P_{\mathcal{N}_{II}, \delta_{II}}(S, Y_2) = P_{\delta_{II}}(Y_2|S). \tag{25}$$

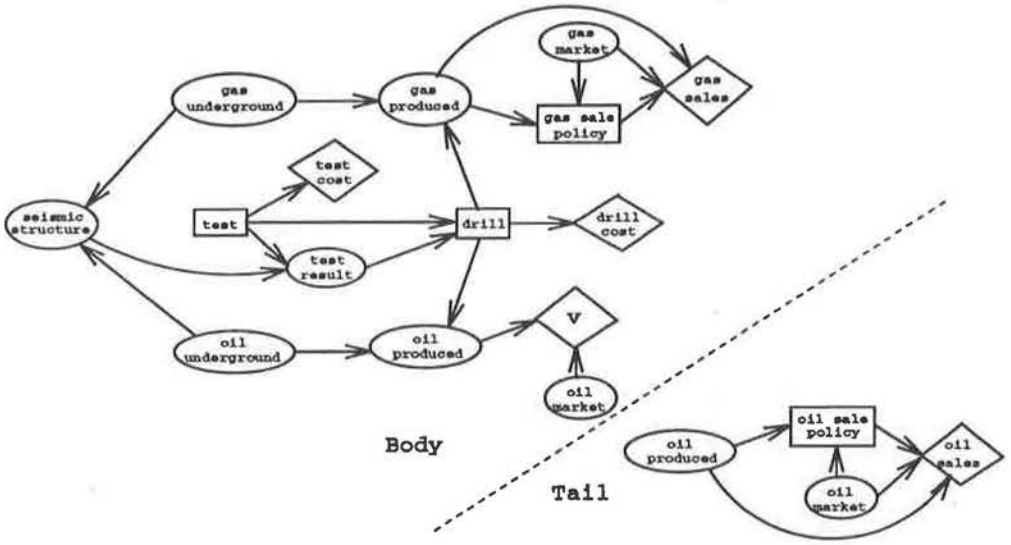

Figure 5: Tail and body: The tail is a semi-decision network, where the prior probabilities for oil-produced and oil-market are missing. In the body, v is a new value node, whose value function is the optimal conditional expected value of the tail.

Similarly, given a policy $\delta_I = (\delta_1, \ldots, \delta_j)$, the body $\mathcal{N}_I$ becomes Bayesian network. Let $P_{\mathcal{N}_I,\delta_I}(Y_1, S)$ denote the joint probability of this Bayesian network. From the definition of bodies one can see the $P_{\mathcal{N}_I,\delta_I}(Y_1, S)$ is equal to the right hand side of equation (23). Therefore

$$P_{\mathcal{N}_I,\delta_I}(Y_1, S) = P_{\delta_I}(Y_1, S). \tag{26}$$

Let $v_1, \ldots, v_m$ be all the value nodes of $\mathcal{N}$. We write Z_i for $\pi(v_i)$. Suppose $v_1, \ldots, v_l \in X_1 \cup S$, and $v_{l+1}, \ldots, v_m \in X_2$. Because of equations (25) and (26), we have

$$\begin{aligned} E_{\delta_{II}}[\mathcal{N}_{II}|S] &= \sum_{Y_2} P_{\mathcal{N}_{II},\delta_{II}}(S, Y_2) \sum_{i=l+1}^{m} f_{v_i}(Z_i) \\ &= \sum_{Y_2} P_{\delta_{II}}(Y_2|S) \sum_{i=l+1}^{m} f_{v_i}(Z_i), \end{aligned} \tag{27}$$

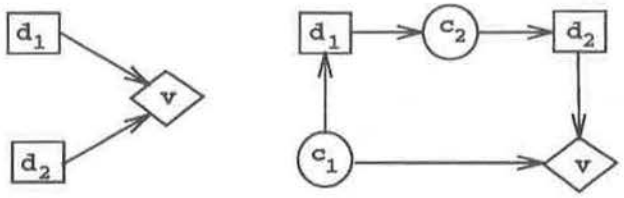

Figure 6: Two decision networks that are not stepwise-decomposable.

and

$$E_{\delta_I}[\mathcal{N}_I] = \sum_{Y_1,S} P_{\mathcal{N}_I,\delta_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i) + E[\mathcal{N}_{II}|S]\}$$
$$= \sum_{Y_1,S} P_{\delta_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i) + E[\mathcal{N}_{II}|S]\}. \quad (28)$$

A decision network skeleton is *stepwise-decomposable* if either it contains no decision nodes or

1. There exists a decision node d whose parents m-separate d itself from all other decision nodes and their parents. The node d will be referred to as a *candidate decision node*[7].

2. The body $\mathcal{N}_I$ of $\mathcal{N}$ w.r.t d is stepwise-decomposable.

A (semi-)decision network is *stepwise-decomposable* if its skeleton is.

The decision network skeleton for the extended oil wildcatter problem (Figure 1) is stepwise-decomposable. Figure 6 shows two typical decision network skeletons that are not stepwise-decomposable.

A stepwise-decomposable decision network $\mathcal{N}$ is *smooth* if either it contains no decision nodes, or if it is smooth at some candidate decision node d and the body $\mathcal{N}_I$ of $\mathcal{N}$ w.r.t d is smooth.

[7]Note that a candidate decision node has no decision node successors, i.e. it is a leaf decision node.

Proposition 2 *Suppose $\mathcal{N}$ is a smooth stepwise-decomposable decision network and d is a candidate decision node. Then*

1. *The tail $\mathcal{N}_{II}$ of $\mathcal{N}$ w.r.t d is a simple semi-decision network, and*
2. *The body $\mathcal{N}_I$ of $\mathcal{N}$ w.r.t d is a again a smooth stepwise-decomposable decision network.* □

9 Evaluating smooth stepwise-decomposable decision networks

In this section, we show that an optimal policy for a smooth stepwise-decomposable decision network can be computed by recursively evaluating the tail semi-decision networks.

Theorem 1 *Let $\mathcal{N}$ be a stepwise-decomposable decision network and d be a candidate decision node. Let $\mathcal{N}_{II}$ be the tail of $\mathcal{N}$ w.r.t d, and $\mathcal{N}_I$ be the body. If $\mathcal{N}$ is smooth at d, then*

1. *If δ^o_{II} is an optimal policy for $\mathcal{N}_{II}$ and δ^o_I is an optimal policy for $\mathcal{N}_I$, then $\delta^o =_{def} (\delta^o_I, \delta^o_{II})$ is an optimal policy for $\mathcal{N}$.*
2. *The optimal expected value $E[\mathcal{N}_I]$ of the body $\mathcal{N}_I$ is the same as the optimal expected value $E[\mathcal{N}]$ of $\mathcal{N}$.*

Proof: For any policy δ of $\mathcal{N}$, we have

$$\begin{aligned}
E_\delta[\mathcal{N}] &= \sum_{Y_1,S,Y_2} P_\delta(Y_1,S,Y_2)\sum_{i=1}^{m} f_{v_i}(Z_i) && \text{(By definition)}\\
&= \sum_{Y_1,S,Y_2} P_{\delta_I}(Y_1,S)P_{\delta_{II}}(Y_2|S)\sum_{i=1}^{m} f_{v_i}(Z_i) && \text{(By equations 22 and Lemma 2)}\\
&= \sum_{Y_1,S} P_{\delta_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i)\sum_{Y_2} P_{\delta_{II}}(Y_2|S) + \sum_{Y_2} P_{\delta_{II}}(Y_2|S)\sum_{i=l+1}^{m} f_{v_i}(Z_i)\}\\
&= \sum_{Y_1,S} P_{\delta_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i) + E_{\delta_{II}}[\mathcal{N}_{II}|S]\} && \text{(By equation 27).} \qquad (29)
\end{aligned}$$

Since δ^o_{II} is an optimal policy for $\mathcal{N}_{II}$ and $\mathcal{N}_{II}$ is a simple semi-decision network, it follows from Proposition 1 that

$$E_{\delta_{II}}[\mathcal{N}_{II}|S] \leq E_{\delta^o_{II}}[\mathcal{N}_{II}|S]. \qquad (30)$$

Noticing that $P_{\delta_I}(Y_1,S)$ is non-negative, we have

$$E_\delta[\mathcal{N}] = \sum_{Y_1,S} P_{\delta_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i) + E_{\delta_{II}}[\mathcal{N}_{II}|S]\} \quad \text{(By equation 29)}$$

$$
\begin{aligned}
&\leq \sum_{Y_1,S} P_{\delta_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i) + E_{\delta^o_{II}}[\mathcal{N}_{II}|S]\} && \text{(By equation 30)}\\
&= \sum_{Y_1,S} P_{\delta_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i) + E[\mathcal{N}_{II}|S]\} && \text{(By definition)}\\
&= E_{\delta_I}[\mathcal{N}_I] && \text{(By equation 28)}\\
&\leq E_{\delta^o_I}[\mathcal{N}_I] && \text{(Optimality of } \delta^o_I\text{)}\\
&= \sum_{Y_1,S} P_{\delta^o_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i) + E[\mathcal{N}_{II}|S]\} && \text{(By equation 28)}\\
&= \sum_{Y_1,S} P_{\delta^o_I}(Y_1,S)\{\sum_{i=1}^{l} f_{v_i}(Z_i) + E_{\delta^o_{II}}[\mathcal{N}_{II}|S]\} && \text{(Optimality of } \delta^o_{II}\text{)}\\
&= E_{\delta^o}[\mathcal{N}]. && \text{(By equation 29)}
\end{aligned}
$$

Therefore, the first statement of the theorem is proved.

The foregoing derivation has also shown that

$$E_{\delta}[\mathcal{N}] \leq E_{\delta^o_I}[\mathcal{N}_I] \leq E_{\delta^o}[\mathcal{N}].$$

Letting δ be δ^o, we get

$$E_{\delta^o}[\mathcal{N}] = E_{\delta^o_I}[\mathcal{N}_I].$$

Therefore $E[\mathcal{N}] = E[\mathcal{N}_I]$. The proof is completed. □

The theorem reduces the task of evaluating a smooth stepwise-decomposable decision network $\mathcal{N}$ into the task of evaluating the tail $\mathcal{N}_{II}$, which is a simple semi-decision network and the task of evaluating the body $\mathcal{N}_I$, which contains one less decision node than $\mathcal{N}$.

Let N-EVALUATE be a procedure that computes the (optimal) expected values of decision networks that do not have decision nodes. Let S-EVALUATE be a procedure that computes an optimal policy for, and the conditional optimal expected value of, any simple semi-decision network. Theorem 1 and Proposition 2 suggest the following procedure for evaluating a smooth stepwise-decomposable decision network.

Procedure EVALUATE:

- Input: $\mathcal{N}$ — a smooth stepwise-decomposable decision network.

- Output: An optimal policy and the optimal expected value.

1. **If** there are no decision nodes, Call N-EVALUATE to compute the expected value, and stop.
2. **Else**
   - Find a candidate decision node d,
   - Construct the tail $\mathcal{N}_{II}$ of $\mathcal{N}$ w.r.t d,
   - Call S-EVALUATE to compute an optimal policy for and the optimal conditional expected values of $\mathcal{N}_{II}$,
   - Construct the body $\mathcal{N}_I$ of $\mathcal{N}$ w.r.t d, and
   - Make a recursive call to EVALUATE to evaluate $\mathcal{N}_I$.

The procedure N-EVALUATE is essentially about Bayesian network computations, which is beyond the scope of this paper. We now turn to discuss how to write the procedure S-EVALUATE, which evaluates simple semi-decision networks.

10 Evaluating simple semi-decision networks

Let $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F} : S)$ be a simple semi-decision network. Let d be the only decision node. By definition $\pi(d) = S$. We want to find a decision function $\delta^o : \Omega_S \rightarrow \Omega_d$ such that

$$E_{\delta^o}[\mathcal{N}] = max_{\delta \in \Delta} E_\delta[\mathcal{N}].$$

According to Corollary 1, such a δ^o can be found in a pointwise fashion. More precisely, for each value $\beta \in \Omega_S$, the value $\delta^o(\beta)$ can be determined through

$$\delta^o(\beta) = arg\ max_{\alpha \in \Omega_d} E_{\delta:\delta(\beta)=\alpha}[\mathcal{N}|S = \beta]. \tag{31}$$

An advantage of computing an optimal decision function δ^o by equation (31) is that the conditional optimal expected values $E[\mathcal{N}|S]$, which are needed in defining a body, are computed as a by-product.

The term $E_{\delta:\delta(\beta)=\alpha}[\mathcal{N}|S = \beta]$ is a function of α and β. Let us write it as $f(\alpha, \beta)$ for the time being. Equation (31) states that to evaluate $\mathcal{N}$ we need to compute $f(\alpha, \beta)$ for each $\alpha \in \Omega_d$ and each $\beta \in \Omega_S$.

A pointwise strategy for computing the function $f(\alpha, \beta)$ may involve redundancy, because certain parts of the computation may be the same for different (α, β). In this sense, we say that there exist *repetitions of computation due to a strategy pointwise on* (α, β).

Now, we transform the task of evaluating a simple semi-decision network into a tasks of computing marginal potentials in the semi-Bayesian network. This way, the function $f(\alpha, \beta)$ is computed as a whole instead of pointwise. Consequently the redundancy can be avoided. Another motivation for the transformation is that we can then make use of methods for Bayesian network computations. What we are doing here is closely related to (Peot and Shachter 1992). The difference is that we deal with more than one value node and we do not change the value nodes into observed node

Let $\mathcal{N}_0$ be the semi-Bayesian network that is same as $\mathcal{N}$ except that it does not contain a conditional probability $P_\delta(d|S)$ for d. Let P_0 denote the joint potential of $\mathcal{N}_0$ and let P_δ denote the joint potential of $\mathcal{N}_\delta$. Then, we have

$$P_\delta(Y) = P_0(Y)P_\delta(d|S). \tag{32}$$

Consequently, we have

$$\begin{aligned}
E_\delta[\mathcal{N}|S] &= \sum_{Y-S} P_\delta(Y) \sum_{v\in U} f_v(\pi(v)) && \text{(By definition)} \\
&= \sum_{Y-S} P_\delta(d|S)P_0(Y) \sum_{v\in U} f_v(\pi(v)) && \text{(By equation (32))} \\
&= \sum_{v\in U}\sum_{d} P_\delta(d|S) \sum_{Y-(S\cup\{d\})} P_0(Y)f_v(\pi(v)) \\
&= \sum_{v\in U}\sum_{d} P_\delta(d|S) \sum_{\pi(v)}\sum_{Y-(\pi(v)\cup S\cup\{d\})} P_0(Y)f_v(\pi(v)) \\
&= \sum_{v\in U}\sum_{d} P_\delta(d|S) \sum_{\pi(v)} P_0(\pi(v), S, d)f_v(\pi(v)) && \text{(Marginal potential)}
\end{aligned}$$

As indicated out by equation (9), given S, $P_\delta(d|S)$ is nothing but the characteristic function $\chi_{\{d|d=\delta(S)\}}(d)$ of the set $\{d|d = \delta(S)\}$. Therefore

$$E_\delta[\mathcal{N}|S] = \sum_{v\in U}\sum_{\pi(v)} P_0(\pi(v), S, \delta(S))f_v(\pi(v)). \tag{33}$$

Let us now define the *evaluation function* $e(\alpha, \beta)$ of $\mathcal{N}$ by

$$e(\alpha, \beta) = \sum_{v \in U} \sum_{\pi(v)} P_0(\pi(v), S = \beta, d = \alpha) f_v(\pi(v)). \forall \alpha \in \Omega_d, \forall \beta \in \Omega_S. \quad (34)$$

Equations (31) and (33) together have established the following theorem.

Theorem 2 *Let $\mathcal{N} = (X, A, \mathcal{P}, \mathcal{F} : S)$ be a simple semi-decision network. Let d be the only decision node. Let $e(\alpha, \beta)$ be the evaluation function. Then*

1. *The conditional optimal expected values $E[\mathcal{N}|S]$ are given*

$$E[\mathcal{N}|S = \beta] = max_{\alpha \in \Omega_d} e(\alpha, \beta), \forall \beta \in \Omega_S. \quad (35)$$

2. *An optimal decision function δ^o can be found by*

$$\delta^o(\beta) = arg\ max_{\alpha \in \Omega_D} e(\alpha, \beta), \forall \beta \in \Omega_S. \quad (36)$$

The theorem suggests that we can write the S-EVALUATE procedure as follows.

Procedure S-EVALUATE

- $\mathcal{N}$ — A simple semi-decision network.
- An optimal policy and the optimal conditional expected values.

1. Construct the semi-Bayesian network $\mathcal{N}_0$.
2. Compute the marginal potentials $P_0(\pi(v), S, d)$ for all value nodes v.
3. Obtain the evaluation function $e(\alpha, \beta)$ by equation (34).
4. Compute the optimal conditional expected value by equation (35) and an optimal policy by equation (36).

Note that because the marginal potential $P_0(\pi(v), S, d)$ is computed as whole, the repetitions of computation due to a strategy pointwise on (α, β) are avoided.

11 Evaluating non-smooth stepwise-decomposable decision networks

We have presented in section 8 an algorithm for evaluating smooth stepwise-decomposable decision networks. In this section we show how to deal with non-smooth stepwise-decomposable decision networks. The idea is that we revise the evaluation algorithm so that if a decision network $\mathcal{N}$ is not smooth at a candidate decision node d, it will automatically smooth $\mathcal{N}$ at d by calling the subroutine SMOOTH($\mathcal{N}, d$), which is what this section is about. First of all, here is the revised algorithm.

Procedure EVALUATE:

- Input: $\mathcal{N}$ — a stepwise-decomposable decision network (which is not necessarily smooth).
- Output: An optimal policy and the optimal expected value.

1. **If** there are no decision nodes, Call N-EVALUATE to compute the expected value, and stop.
2. **Else**
   - Find a candidate decision node d.
   - If $\mathcal{N}$ is not smooth at d, call SMOOTH($\mathcal{N}, d$) and let $\mathcal{N}'$ be the resulting decision network. Else let $\mathcal{N}'$ be $\mathcal{N}$.
   - Construct the tail $\mathcal{N}'_{II}$ of $\mathcal{N}'$ w.r.t d,
   - Call S-EVALUATE to compute an optimal policy for and the optimal conditional expected values of $\mathcal{N}'_{II}$.
   - Construct the body $\mathcal{N}'_{I}$ of $\mathcal{N}'$ w.r.t d.
   - Make a recursive call to EVALUATE to evaluate $\mathcal{N}'_{I}$.

Before we can discuss the subroutine SMOOTH($\mathcal{N}, d$), we need to make some preparations. Two decision networks $\mathcal{N}$ and $\mathcal{N}'$ are *equivalent* if

1. They have the same sets of random, decision and value nodes.
2. Each decision node has the same parents in both networks. In other words, the two networks have the same policy space.

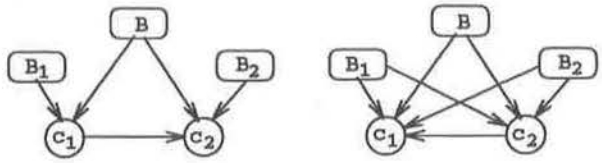

Figure 7: The concept of arc reversal: At the beginning the parent sets of c_1 and c_2 are $B \cup B_1$ and $B \cup B_2 \cup \{c_1\}$ respectively. After reversing the arc $c_1 \rightarrow c_2$, the parent sets become $B \cup B_1 \cup B_2 \cup \{c_2\}$ and $B \cup B_1 \cup B_2$.

3. For each policy δ, the induced Bayesian networks $\mathcal{N}_\delta$ and $\mathcal{N}'_\delta$ have the same prior joint probability.

If two decision networks are equivalent, then they have the same policy space, and their expected values are the same given the same policies. Consequently, they have the same optimal policy and the same optimal expected value.

Arc reversal is an operation that transforms one decision network into another different but equivalent decision network (Howard and Mahteson 1984, Shachter 1986). Let c_1 and c_2 be two random nodes in a decision network $\mathcal{N}$. Suppose there is an arc from c_1 to c_2, and there is no other directed path from c_1 to c_2. Let $B = \pi(c_1) \cap \pi(c_2)$, $B_1 = \pi(c_1) - \pi(c_2)$, and $B_2 = \pi(c_2) - (\pi(c_1) \cup \{c_1\})$. *To reverse the arc* $c_1 \rightarrow c_2$ is to:

1. Reverse the arc between c_1 to c_2, draw an arc from each node in B_2 to c_1, and draw an arc from each node in B_1 to c_2; and

2. Define the conditional probabilities $P(c_2|B, B_1, B_2)$ and $P(c_1|B, B_1, B_2, c_2)$ by

$$\begin{aligned} P(c_2|B, B_1, B_2) &= \sum_{c_1} P(c_1, c_2|B, B_1, B_2), \\ P(c_1|B, B_1, B_2, c_2) &= \frac{P(c_1, c_2|B, B_1, B_2)}{P(c_2|B, B_1, B_2)}, \end{aligned}$$

where $P(c_1, c_2|B, B_1, B_2) = P(c_1|\pi(c_1))P(c_2|\pi(c_2))$, and $P(c_1|\pi(c_1))$ and $P(c_2|\pi(c_2))$ are in turn given in the specification of $\mathcal{N}$.

Proposition 3 *Let $\mathcal{N}$ be a decision network. Let a $\mathcal{N}'$ be the decision network obtained from $\mathcal{N}$ by reversing the arc between two random nodes. Then $\mathcal{N}'$ and $\mathcal{N}$ are equivalent.*

Proof: It is evident that the two networks have the same sets of random, decision and value nodes. It is also obvious that each decision node has the same parents in both $\mathcal{N}$ and $\mathcal{N}'$. The conditional probabilities of random nodes other than c_1 and c_2 are the same in both networks. It follows from the definition of arc reversal that the product of the conditional probabilities of c_1 and c_2 is also $P(c_1, c_2|\pi(c_1) \cup \pi(c_2))$ in both networks. Consequently $\mathcal{N}'$ and $\mathcal{N}$ have the same expected value given the same policy. The proposition is proved. □

We now start to discuss the subroutine SMOOTH. Let $\mathcal{N}$ be a stepwise-decomposable decision network and let d be a candidate decision node. Let Y_2 be the set of random and decision nodes in the downstream of $\pi(d)$. A node $c \in Y_2$ is *disturbing w.r.t d* if there is a directed path from c to at least one node in $\pi(d)$.

Consider applying procedure EVALUATE to the network in Figure 1. The network is smooth at the candidate decision nodes `gas-sale-policy` and `oil-sale-policy`. After it evaluates those two decision nodes, the network becomes the one shown in Figure 8 (1). In this network, `drill` is the candidate decision node, but the network is not smooth at `drill`. The nodes `gas-underground`, `oil-underground` and `seismic-structure` in the network shown are disturbing.

Lemma 3 *Let $\mathcal{N}$ be a stepwise-decomposable decision network and d be a candidate decision node. For any $c_1 \in Y_2$ and any $c_2 \in \pi(d)$,*

1. *$\pi(c_1) \subseteq Y_2 \cup \pi(d)$.*
2. *If there is an arc from c_1 to c_2 in $\mathcal{N}$, then c_1 and c_2 are both random nodes. So are the predecessors of c_1 in Y_2.*

Proof: The lemma follows immediately from the definition of stepwise-decomposability and the definition of the set Y_2. □

A *root disturbing node w.r.t d* is a disturbing node whose parents, if any, are all in $\pi(d)$. In our example, both `gas-underground` and `oil-underground` are root disturbing nodes.

Now, here is the subroutine SMOOTH.

Procedure SMOOTH($\mathcal{N}, d$)

- Input: $\mathcal{N}$ — a stepwise-decomposable decision network.
  d — a candidate decision node of $\mathcal{N}$.
- Output: $\mathcal{N}'$ — an equivalent stepwise-decomposable decision network that is smooth at d.

While there are disturbing nodes:

1. Find a root disturbing node, say c.
2. Reverse, one after another, all the arcs emanating from c and on a directed path from c to $\pi(d)$.

Theorem 3 *The procedure SMOOTH terminates. The resulting decision network remains stepwise-decomposable, is smooth at d, and is equivalent to the input network.*

Proof: Let us first note that when an arc $c \rightarrow c'$ between two random nodes is reversed, arcs need to be introduced from the parents of c' to c and from the parents of c to c', and that there are no other graphical changes.

We now show that the algorithm does not destroy stepwise-decomposability. In SMOOTH all the arc reversals happen at step 2. Consider one pass through step 2. Suppose the arcs reversed are $c \rightarrow c_1, \ldots, c \rightarrow c_k$. Some of those c_i's may be in $\pi(d)$. However, since they all have c as a parent, their other parents can only be in $\pi(d) \cup Y_2$. Thus after step 2, all the new arcs into c are from nodes in $\pi(d) \cup Y_2$. Since all the parents of c, old or new, are in $\pi(d) \cup Y_2$, the new arcs into the c_i's are also in $\pi(d) \cup Y_2$. Consequently, stepwise-decomposability is not destroyed.

After this step 2, c is no longer a disturbing node. Thus the number of disturbing nodes is reduced by one. Therefore, an execution of the algorithm will eventually leave the while loop. When that happens, there are no longer any disturbing nodes w.r.t d. Consequently, the algorithm terminates and when it does, the resulting decision network is smooth at d.

Finally, the equivalence between the resulting network and the input network was established by proposition 3. □

As an example, consider the stepwise-decomposable decision network shown in Figure 8 (1). It is not smooth at the candidate decision node `drill`. Both `gas-underground` and `oil-underground` are root disturbing

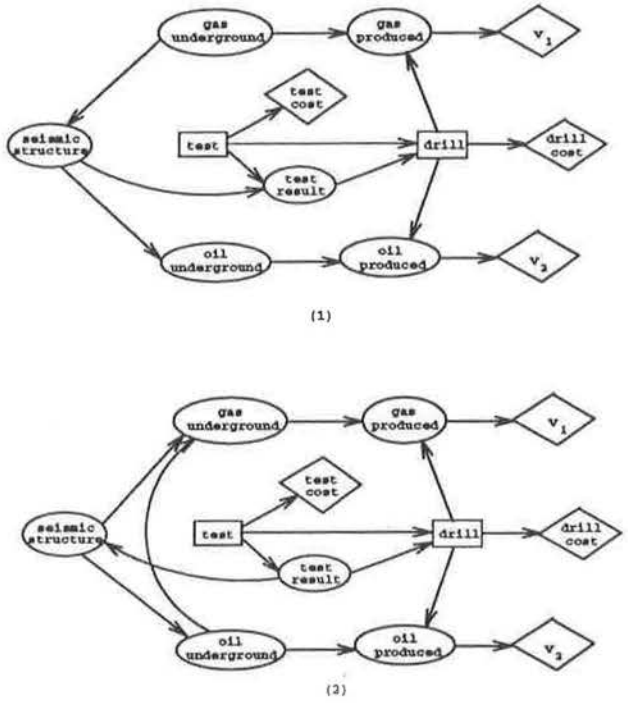

Figure 8: Arc reversal and smoothness.

nodes. So, either the arc from `gas-underground` to `seismic-structure` or the arc from `oil-underground` to `seismic-structure` can be reversed first. Suppose the former of those two is reversed first. This introduces a new arc from `oil-underground` to `gas-underground`. Then the arc from `oil-underground` to `seismic-structure` and the arc from `seismic-structure` to `test-result` are reversed in order, which produces no more extra arcs. The resulting network is shown in Figure 8 (2).

We notice in the example that the reversals do not change the moral graph. This is always true if none of the root disturbing nodes have parents (in $\pi(d)$). Hence, arc reversals for the sake of smoothness usually do not introduces many arcs.

Part III
Related work and Summary

12 Related work

All the previous research on decision networks has been about no-forgetting decision networks. In this paper, we have proposed and studied stepwise-decomposable decision networks, which is more general than no-forgetting decision networks. In this section, we restrict ourselves to no-forgetting decision networks and discuss the pros and cons of our algorithm as compared to the previous algorithms for evaluating no-forgetting decision networks.

There are at least four well developed algorithms for evaluating no-forgetting decision networks. The most popular one probably remains to be the one developed by Shachter (1986) based on the work of Howard and Matheson (1984) and Olmsted (1983). The algorithm evaluates no-forgetting decision networks by using four operations, namely barren node removal, arc reversal, random node removal, and decision node removal. We shall refer to this algorithm as Shachter's first algorithm.

Ndilikilikesha (1991) proposed no-forgetting potential decision networks as a generalization to no-forgetting decision networks, and designed an algorithm to evaluate such decision networks by using three operations, namely barren node removal, random node absorption, and decision node absorption. Since the operation of arc reversal is avoided, this algorithm is believed to be more efficient than Shachter's first algorithm. We shall refer to this algorithm as Ndilikilikesha's algorithm.

Let d be a leaf decision node in a no-forgetting decision network $\mathcal{N}$, and let $v_1, \ldots, v_k$ be the value nodes in the tail of $\mathcal{N}$ w.r.t d. Shachter (1988, 1990) have noticed that an optimal decision function $\delta^o : \Omega_{\pi(d)} \rightarrow \Omega_d$ can be obtained through

$$\delta^o(\beta) = arg\, max_{\alpha \in \Omega_d} E_{\delta:\delta(\beta)=\alpha}[v_1 + \ldots + v_k | \pi(d) = \beta], \qquad (37)$$

for each $\beta \in \Omega_{\pi(d)}$. After this optimal decision function is computed, the decision node d is replace by a deterministic node characterized by the function δ^o, resulting in a no-forgetting decision node with one less decision nodes.

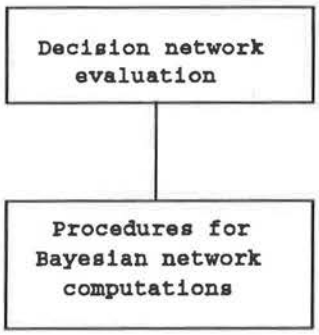

Figure 9: Modularity.

Thus, the decision network can be evaluated by repeatedly using formula (37) . We shall refer to this algorithm Shachter's second algorithm.

Shenoy's algorithm (Shenoy 1990) is a combination of the clique tree propagation approach for computing marginal probabilities and the clique tree propagation approach to dynamic programming. The algorithm works on a directed clique trees build from the original no-forgetting decision network. The directionality is due to the time precedence relationships encoded in the decision network. The evaluation process is carried out by eliminating nodes from the clique tree one by one. Random nodes are eliminated by means summation while decision nodes by means of maximization.

One property of our evaluation algorithms is that it explicitly identifies the Bayesian network computation tasks that arise in the evaluation process, and dispatches them to subroutines that live in a module for Bayesian network computations. In other words, our algorithm can be implemented on top of a module for Bayesian network computations. This is shown in Figure 9.

This modularity is important for three reasons. Firstly, researchers have long noticed the close relationship between decision network evaluation and Bayesian network computations (Cooper 1989, Peot and Shachter 1992). But no one has been able to clearly described the relationship. By explicitly identifying Bayesian network computations in decision network evaluation, our algorithm provides a clear picture about the relationship.

Secondly, the modularity is desirable from an implementation point of

view.

Thirdly, in both decision network evaluation and Bayesian network computations, efficiency heavily depends on the elimination ordering (EO) chosen. There have been a few heuristics for finding "good" EO's for Bayesian network computations, but none for decision network evaluation. The reason is not all possible EO's are valid for the decision network evaluation. Our algorithm enables one to directly use the heuristics of Bayesian networks to evaluate decision networks.

However, Shachter's first algorithm, Pierre's algorithm and Shenoy's algorithm do not clearly identify the Bayesian network computations involved in evaluating a decision network. Thus, they do not enjoy the modularity mentioned above. Consequently, heuristics for finding "good" EO's for Bayesian network computations have to be modified in order to be used for decision network evaluations. So far, there have been no research on how this can be done.

Shachter's second algorithm does enjoy the modularity mentioned above. However, it is not as efficient as our algorithm, because computations in the tail are repeated.

On the other hand, our algorithms carries an overhead when compared to the other four algorithms. When the network is not smooth, we need to convert it into a smooth one by a series of arc reversals. We also need to identify tails and bodies.

Finally we would like to point out that Cooper (1989) contains a method for transforming the task of evaluating a no-forgetting decision network into tasks of computing conditional probabilities. The basic idea is similar to ours. However, we deal with stepwise-decomposable decision networks, which are more general than no-forgetting decision networks. Also, we have combined this idea with irrelevance results to give rise to notions like tail, body, and section.

13 Summary

This paper has been about how to apply Bayesian Decision Theory to problems that involve multiple decisions and multiple variables. Here is a summary of our contributions.

First of all, the concept of decision networks has been developed from

a Bayesian Decision Theory setup by considering the so-called conditional multiple decision problems. This gives precise semantics to decision networks. A discussion on semantic constraints has led to the conclusion that acyclicity is the only constraint that is indispensable to decision networks. We thus make acyclicity the only semantic constraint our theory relies upon.

Secondly from a computational point of view, it is desirable if a decision network is stepwise-solvable, i.e if it can be evaluated by considering one decision at a time. However, decision networks in the most general sense need not be stepwise-solvable. A syntactic constraint called stepwise-decomposability has therefore been imposed. We have shown that stepwise-decomposability implies stepwise-solvability.

Thirdly, the problem of evaluating stepwise-decomposable decision networks has been studied in detail. A number of important concepts, such as simple semi-decision networks, body, tail, and smoothness have been identified. We have shown that a stepwise-decomposable decision network can be evaluated by recursively considering the tails of the network, which are simple semi-decision networks, and each of which corresponds to one decision node in the original network. An algorithm has been given to efficiently evaluate simple semi-decision networks.

Could there be a condition that is weeker than stepwise-decomposability, but it still implies stepwise-solvability? The answer is negative, because it can be shown that stepwise-solvability also implies stepwise-decomposability. Another question is, does stepwise-decomposability severely limit the representation power of decision networks? The answer is again negative. The reader is referred to Zhang (1993) for reasons.

In Zhang (1993), we have introduced the concept of decision nodes being conditionally independent of part of available information. We have shown the equivalence between such independencies and removable arcs. A graphical criterion, namely the criterion of potential unaccompanied arcs, has been discovered for finding all such independencies implied by decision networks skeletons. A linear time algorithms has been designed to prune all the potential unaccompanied arcs and the potential barren nodes before evaluation takes place.

Acknowledgements

The first author would like to thank Glenn Shafer and Prakash Shenoy for getting him interested in this area of research. Discussions with Glenn Shafer have been valuable for this paper per se. All the authors would like to thank Mike Horsch, Keiji Kanazawa, and Craig Boutilier for their useful comments. Research is supported by NSERC Grant OGP0044121.

References

[1] D. E. Bell, H. Raiffa, and A. Tversky (1988), *Decision Making: Descriptive, Normative, and prescriptive interactions*, Cambridge University Press.

[2] J. Breese (1991). Construction of belief and decision networks, Technical Report, Rockwell International Science Center.

[3] J. R. Clarke and G. M. Provan (1992), A dynamic decision-theoretic approach to the management of GVHD, in *Proc. AAAI Spring Symposium on AI and Medicine.*

[4] G. F. Cooper (1989), A method for using belief networks as influence diagrams, in *Proceedings of the fifth Conference on Uncertainty in Artificial Intelligence*, Windsor, Ontario.

[5] T. Dean and K. Kanazawa (1989), A model for reasoning about persistence and causation, *Computational Intelligence*, **5**(3), pp. 142-150.

[6] P. C. Fishburn (1988), Normative theories of decision making under risk and under uncertainty, in Bell *et al* 1988.

[7] R. M. Fung and R. D. Shachter (1990), Contingent Influence Diagrams, Advanced Decision Systems, 1500 Plymouth St., Mountain View, CA 94043, USA.

[8] P. Gärdenfors and N. Sahlin (1988a), *Decision, Probability, and Utility: Selected Readings*, Cambridge University Press.

[9] P. Gärdenfors and N. Sahlin (1988b), Introduction: Bayesian decision theory – foundations and problems, in [9].

[10] D. Heckerman and E. Horvitz (1990), Problem formulation as the reduction of a decision model, in *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, July, Cambridge, Mass. , pp. 82-89.

[11] R. A. Howard, and J. E. Matheson (1984), Influence Diagrams, in *The principles and Applications of Decision Analysis*, Vol. II, R. A. Howard

and J. E. Matheson (eds.). Strategic Decisions Group, Menlo Park, California, USA.

[12] R. A. Howard (1990), From influence to relevance to knowledge, in [15].

[13] J. E. Matheson (1990), Unsing influence diagrams to value information and control, in [15].

[14] P. Ndilikilikesha (1991), Potential Influence Diagrams, Working Paper No. 235, Business School, University of Kansas.

[15] R. M. Oliver and J. Q. Smith eds. (1990), *Influence Diagrams, Belief Nets and Decision Analysis*, John Wiley and Sons.

[16] J. Pearl (1988), *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, Los Altos, CA.

[17] D. Poole and E. Neufeld (1991), Sound probabilistic inference in Prolog: An executable specification of Bayesian networks, Department of Computer Science, University of British Columbia.

[18] G. M. Provan (1991), Dynamic network updating techniques for diagnostic reasoning, in *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence.* influence diagram evaluation, submitted to IJCAI'93.

[19] H. Raiffa, (1968), *Decision Analysis*, Addison-Wesley, Reading, Mass.

[20] R. Shachter (1986), Evaluating Influence Diagrams, *Operations Research*, 34, pp. 871-882.

[21] R. Shachter (1988), Probabilistic Inference and Influence Diagrams, *Operations Research*, 36, pp. 589-605.

[22] G. Shafer and P. Shenoy (1988), Local computation in hypertrees, Working Paper No. 201, Business School, University of Kansas.

[23] G. Shafer (1993), *Probabilistic Expert Systems*, to be published by SIAM.

[24] P. P. Shenoy, (1990), Valuation-Based Systems for Bayesian Decision Analysis, Working Paper No. 220, Business School, University of Kansas.

[25] J. Q. Smith (1989), Influence Diagrams for Statistical Modeling, *Ann. Stat.*, 17, pp. 654-672.

[26] T. P. Speed (1991), Complexity, calibration and causality in influence diagrams, in [15].

[27] S. Srinivas and J. Breese (1990), IDEAL: A software package for analysis of influence diagrams, in *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, July, Cambridge, Mass. , pp. 212-219.

[28] J. A. Tatman and R. Shachter (1990), Dynamic programming and influence diagrams, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, pp. 265-279.

[29] M. Wellman (1990), *Formulation of Tradeoffs in Planning under Uncertainty*, Pitman, London.

[30] M. Wellman, J. Breese, and R. Goldman (1992), From knowledge bases to decision models, *Knowledge Engineering Review*, **7**(1).

[31] L. Zhang and D. Poole (1992) Sidestepping the triangulation problem in Bayesian net computations, in *Proc. of 8th Conference on Uncertainty in Artificial Intelligence*, July 17-19, Standford University, pp. 360-367.

[32] L. Zhang and D. Poole (1992), Stepwise-Decomposable Influence Diagrams, in *The Proc. of the 3rd Conference on the principles of Knowledge representation*, Cambridge, Mass. USA, October 26-29, 1992.

[33] L. Zhang, Runping Qi and D. Poole (1993), Minimizing Decision Tables in Influence Diagrams, in *The Proc. of the Fourth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, Florida, January 3-6, 1993.

[34] L. Zhang (1993), A computational theory of decision networks, Ph.D Dissertation, under preparation.