DEPARTMENT OF
ARTIFICIAL INTELLIGENCE

Universidad Nacional de Educación a Distancia

Probabilistic Graphical Models for Decision Making in Medicine

by

Manuel Luque Gallego

Dissertation submitted to the Department of Artificial Intelligence
of the Universidad Nacional de Educación a Distancia as partial fulfilment
of the requirements for the European PhD degree in Computer Science

Madrid, September 2009

DEPARTMENT OF
ARTIFICIAL INTELLIGENCE

Escuela Técnica Superior de Ingeniería Informática

Universidad Nacional de Educación a Distancia

PhD Thesis Dissertation:

Probabilistic Graphical Models for Decision Making in Medicine

by

Manuel Luque Gallego

Computer Engineer

Supervised by

Dr. Francisco Javier Díez Vegas

Madrid, September 2009

Index

List of tables

List of figures

List of algorithms

Part I

INTRODUCTION

Chapter 1

Introduction

1.1 Motivation

Probabilistic Graphical Models (PGMs), in particular Bayesian networks and influence diagrams (IDs), were developed in the 1980's by researchers in Artificial Intelligence, Mathematics and Economy with the purpose of solving problems whose complexity exceeded the capacity of the methods existing so far. Nowadays, PGMs are applied to many areas and there exists an increasing interest in the academic field as well as in the business world. Probabilistic graphical models (PGMs) allow to deal with problems that could not be addressed with traditional probabilistic methods or other artificial intelligence techniques.

Several Spanish research groups interested on PGMs arose independently in different universities. The work on PGMs at UNED started in 1990 with Díez (1994)'s doctoral thesis, which consisted of the construction of the expert system DIAVAL, a Bayesian network for the diagnosis of heart diseases by echocardiography.

Some years later, Dr. Carlos Disdier, a pneumologist at the Hospital San Pedro de Alcántara, in Cáceres (Spain), and Javier Díez, at UNED, began the construction of an influence diagram for the mediastinal staging of non-small cell lung cancer. When the author of this thesis joined the group, at UNED, in 2003, he was assigned as a research topic the completion of that influence diagram, which was then in an incipient state.

The research of this group has always been led by concrete medical problems: the needs that have arisen when building them has motivated the development of new models, algorithms, and software tools, which have been later applied to other problems, not only in medicine. It has also been the case in the present thesis.

Firstly, the form of the function that combines utilities (i.e. the decision's maker preferences, in our case, the quantity and quality of life for lung cancer patients) led us to a structure of utility nodes in our influence diagram. This led us to develop a

new evaluation algorithm that could improve the only algorithm existing so far for these influence diagrams.

Secondly, during the interaction with the expert we felt the need of explanation capabilities for influence diagrams, which would help us in the construction of the model in its debugging, and when trying to convince the expert of the results. For this reason we implemented new explanation methods in Elvira.

Third, due to the uncertainty in the parameters of the influence diagram, assessed by the expert based on the literature and on his own date, we implemented some sensitivity analysis techniques.

And forth, due to the discussion in the medical literature about the optimal order in which the tests for the staging of non-small cell lung cancer should be performed, and given that influence diagrams require a total of decisions, we explored the use of unconstrained influence diagrams, a representation that allows a partial order and developed a new anytime algorithm that provides a recommendation for the first decisions when finding the optimal strategy would require an excessive amount of time.

In this report, we describe these algorithms and methods, which are not specific for medicine, and also the medical decision-support system for the mediastinal staging of non-small cell lung cancer, called MEDIASTINET.

1.2 Objectives

Because of the needs described in the previous section, the objectives of this research can be summarized as follows:

- To develop a variable-elimination algorithm for influence diagrams (IDs) with super-value (SV) nodes, and to compare it with the arc-reversal algorithm by Tatman and Shachter (1990).

- To have explanation capabilities and sensitivity analysis tools for IDs with SV nodes.

- To develop an anytime algorithm for unconstrained influence diagrams.

- To build and evaluate a decision-support system for the mediastinal staging of non-small cell lung cancer, which we have called MEDISTINET.

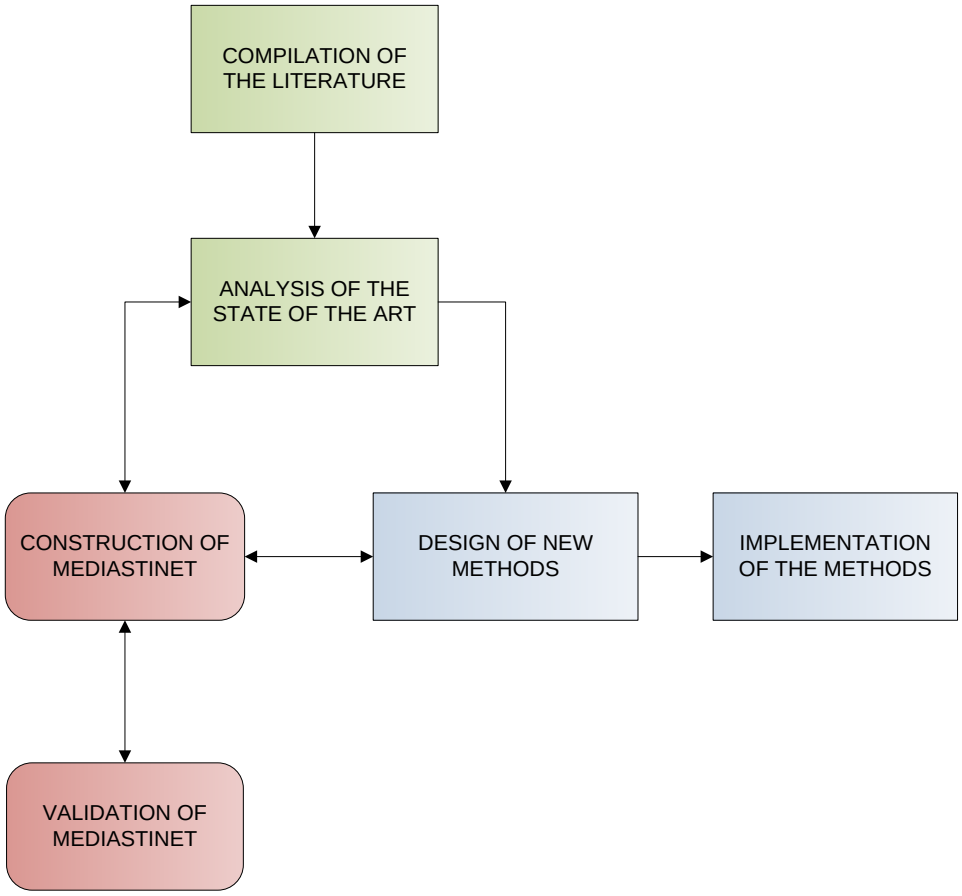

Figure 1.1: Phases in the development of this doctoral thesis.

1.3 Methodology

The methodology followed for achieving the objectives corresponds to several phases, as shown in Figure 1.1.

The first phase consisted of compiling the literature in probabilistic graphical models applied to solving decision problems. The next phase was the study and analysis of the state of the art. The next stage consisted of starting to build the influence diagram of MEDIASTINET with the help of the pneumologist Dr. Carlos Disdier. The construction of the influence diagram made us develop new methods and implement them as software tools. The last phase was the validation of the system, which led to the modification of the model with the expert's help in an iterative process.

It is important to notice that, after the initial phases (colored in green in Figure 1.1) corresponding to the compilation and analysis of the bibliography, the work was guided by the needs that appeared during the construction of MEDIASTINET. Thus, feedback between the different phases of the research has been essential.

1.4 Organization of the thesis

We have structured the thesis in four parts.

Part I explains the motivation, objectives, and methodology of this thesis.

Part II reviews the state of the art and presents the mathematical foundations for the rest of the dissertation.

Part III contains the methodological advances of the thesis. Chapter 3 describes a variable algorithm for influence diagrams with super-value nodes. Chapter 4 presents the explanation capabilities and sensitivity analysis techniques implemented for influence diagrams with super-value nodes. Chapter 4 explains an anytime algorithm for evaluating unconstrained influence diagrams.

Part IV presents a decision support system for the mediastinal staging of non-small cell lung cancer.

Chapters in Part III and IV have been written by following the *hierarchy of levels*, introduced by Marr (1982). In Computer Science, Marr proposed the next hierarchy of levels for any computational problem:

- Computational **Theory**, which indicates the goal of the computation, its justification and the theory that is necessary for its implementation. In our case, it corresponds basically to the study of probabilistic graphical models.
- **Algorithms**, which consists in establishing which the inputs and outputs of each computational task, and which is the algorithm that relate them.
- **Implementation**, which translates the algorithms into a program that can be executed on a computer.

Finally, Part **??** presents the conclusions and future work.

Part II

FUNDAMENTALS

Chapter 2

State of the art: Probabilistic graphical models

In this chapter we present the basic concepts about probabilistic graphical models and the state of the art about explanation in Bayesian networks.

2.1 Basic concepts about graphs

A *graph* $G = (\mathbf{V}, \mathbf{E})$ consists of a finite set of nodes $\mathbf{V}$ and a finite set of edges $\mathbf{E}$. An edge is a pair of nodes (X, Y), where $X, Y \in \mathbf{V}$ and $X \neq Y$; if X and Y are ordered in the edge (X, Y) then it is said to be *directed*; otherwise it is *undirected*. A directed edge will be referred to as an *arc*. If every arc in $\mathbf{E}$ is directed then G is a *directed graph*. On the other hand, if every arc is undirected then G is said to be an *undirected graph*.

A *path* from a node X to a node Y in a graph $G = (\mathbf{V}, \mathbf{E})$ is a sequence $X = X_0, X_1, ..., X_n = Y$ of distinct nodes such that (X_{i-1}, X_i) is an edge in $\mathbf{E}$ for each i such that $1 \leq i \leq n$. The path is a *directed path* if (X_i, X_{i-1}) is a directed arc from X_{i-1} to X_i, for each i such that $1 \leq i \leq n$. A graph is said to be a *tree* if each pair of distinct nodes is connected by exactly one path.

A *cycle* is a path with the exception that $X_0 = X_n$, and a *directed cycle* is a directed path with $X_0 = X_n$. A directed graph with no directed cycles is said to be an *acyclic directed graph* (ADG).

Given an arc (X, Y) from X to Y, the node X is said to be a *parent* of Y and Y is a *child* of X. The set of parents for a node Y is denoted by $Pa(X)$, and the set of children for a node X is denoted by $Ch(X)$. The set of nodes from which there exists a directed path from X is named the *ancestors* of X (denoted by $an(X)$). Similarly, the set of nodes to which there exists a directed path from X is termed the *descendants* of X (denoted by $de(X)$).

2.2 Bayesian networks

A *Bayesian network* (Pearl, 1988)$B = (G, P)$ consists of two elements: an ADG $G = (\mathbf{V}, \mathbf{E})$ in which each node $X \in \mathbf{V}$ (named *chance node*) is drawn as a circle and corresponds to a *chance variable* X; and a probability distribution over $\mathbf{V}$, $P(\mathbf{V})$, which can be factored as:

$$P(\mathbf{v}) = \prod_{X \in \mathbf{V}} P(x|pa(X)), \tag{2.1}$$

where $pa(X)$ denotes a configuration of the parents of X.

Since there is a bijection between a variable in a Bayesian network $B = (G, R)$ and a node in G, the terms node and variable will be used indifferently.

The quantitative information of a Bayesian network $B = (G, R)$ is given by assigning to each node $X \in \mathbf{V}$ a conditional probability distribution $P(X|Pa(X))$. Conditional probability distributions are also referred to as *potentials.* A *potential* is a real-valued function over a domain of finite variables. The *domain* of a potential $\phi = P(X|Pa(X))$ is $dom(\phi) = \{X\} \cup Pa(X)$.

We will assume in the dissertation that Bayesian network $B = (G, P)$ is *discrete*, which means each chance node $X \in \mathbf{V}$ corresponds to a discrete *chance variable* X with a finite set of mutually exclusive and exhaustive states; the *domain* of a variable X is denoted by $dom(X) = (x_1, x_2, ..., x_l)$.

2.3 Influence diagrams

An *influence diagram* is basically a Bayesian network augmented with decision nodes and value nodes. Thus, an influence diagram consists of an ADG $G = (\mathbf{V}, \mathbf{E})$, where the set $\mathbf{V}$ has three types of nodes: *chance nodes* $\mathbf{V}_C$, *decision nodes* $\mathbf{V}_D$ and *utility nodes* $\mathbf{V}_U$.

As in Bayesian networks, chance nodes (drawn as circles) represent chance variables, i.e., events which are not under the direct control of the decision maker. Decision nodes (drawn as rectangles) correspond to actions under the direct control of the decision maker. Utility nodes (drawn as diamonds) represent the expected benefit or loss, or more generally, the preferences of the decision maker. Utility nodes can not be parents of chance or decision nodes.

Tatman and Shachter (1990) proposed an extended framework of IDs with SVNs. They distinguished two types of utility nodes: *ordinary utility nodes*, whose parents are decision and/or chance nodes, and *super-value nodes* (SVNs), whose parents are utility

nodes. We assume that there is a utility node U_0 that is a descendant of all the other utility nodes, and therefore has no children.[1]

There are three types of arcs in an ID, corresponding to the type of node they go into. Arcs into chance nodes represent probabilistic dependency. Arcs into decision nodes, named *informational arcs*, represent availability of information; i.e., if there is an arc from a node X to a decision node D then the state of X is known when decision D is made. Arcs into utility nodes represent functional dependency: arcs into ordinary utility nodes indicate the domain of the associated utility function; arcs into a SVN U indicate that the associated utility function is a combination (generally a sum or a product) of the utility functions of the parents of U.

We assume that there is a path in the ID that includes all the decision nodes, which induces a total order among the n decisions $\{D_1, \ldots, D_n\}$ and indicates the order in which the decisions are made. Such order originates a partitioning of $\mathbf{V}_C$ into a collection of disjoint subsets $\mathbf{C}_0, \mathbf{C}_1, ..., \mathbf{C}_n$, where $\mathbf{C}_i$ contains every chance variable C such that there is an arc $C \rightarrow D_i$ but there is not an arc $C \rightarrow D_j$, $j < i$; i.e., $\mathbf{C}_i$ is the subset of chance variables known for D_i but unknown for any previous decision. This induces a *partial order* $\prec$ in $\mathbf{V}_C \cup \mathbf{V}_D$:

$$\mathbf{C}_0 \prec D_0 \prec \mathbf{C}_1 \prec ... \prec D_n \prec \mathbf{C}_n \ . \tag{2.2}$$

The set of variables known to the decision maker when deciding on D_j is termed the *informational predecessors* of D_j and is denoted $iPred(D_j)$. By assuming the *no-forgetting* hypothesis, which states that the decision maker remembers all previous decisions and observations, we have $iPred(D_i) \subseteq iPred(D_j)$ (for $i \leq j$). In particular, $iPred(D_j)$ is the set of chance variables that occurs before D_j under $\prec$, i.e., $iPred(D_j) = \mathbf{C}_0 \cup \{D_0\} \cup \mathbf{C}_1 \cup \ldots \cup \{D_{i-1}\} \cup \mathbf{C}_i$. If we have a chance or decision variable X, two decisions D_i and D_j such that $i < j$, and two arcs $X \rightarrow D_i$ and $X \rightarrow D_j$, then the latter is said to be a *no-forgetting arc*.

The quantitative information that defines an ID is given by (1) assigning to each chance node C a conditional probability potential $p(C|pa(C))$ for each configuration of its parents, $pa(C)$[2], (2) assigning to each ordinary utility node U a potential $\psi_U(pa(U))$ that maps each configuration of its parents onto a real number, and (3) assigning a utility-combination function to each SVN. Every utility function ψ_U of a utility node U

[1] Clearly, an ID having only one utility node satisfies this condition by identifying such a node with U_0. An ID having several utility nodes assumes that the global utility is their sum, and can be modified to fulfill that condition by adding a new node U_0, of type sum, whose parents are the original utility nodes. Therefore, this assumption does not restrict the types of IDs that our algorithm can solve.

[2] We denote with $Pa(X)$ the set of parents of X, and with $pa(X)$ a configuration of the parents of X.

can finally be expressed as a function of chance and decision nodes, termed the *functional predecessors* of U and denoted by $fPred(U)$. Thus, the functional predecessors of an ordinary utility node are its parents, $fPred(U) = Pa(U)$, and the functional predecessors of a SVN are all the functional predecessors of its parents: $fPred(U) = \cup\{fPred(U') \mid U' \in Pa(U)\}$. The algorithms described in this thesis assume that all the SNs in the ID are either of type sum or product.[3] In analogy with the terms of variable and node, we will use the terms utility function and utility node interchangeably.

For example, in the ID of Figure 2.1 we have $fPred(U_1) = \{X, D\}$, $fPred(U_2) = \{T\}$ y $fPred(U_0) = \{X, D, T\}$. Similarly, considering that U_0 is a sum node, we have $\psi_{U_0}(T, X, D) = \psi_{U_1}(T) + \psi_{U_2}(X, D)$.

In order to simplify the notation, we shall sometimes assume without loss of generality that $fPred(U) = \mathbf{V}_C \cup \mathbf{V}_D$ for every utility node U, i.e., U depends on all the chance variables and decisions.

For each configuration $\mathbf{v}_D$ of the decision variables in $\mathbf{V}_D$ we have a joint probability distribution defined over the set of random variables $\mathbf{V}_C$:

$$P(\mathbf{v}_C : \mathbf{v}_D) = \prod_{X \in \mathbf{V}_C} P(x|pa_C(X) : pa_D(X)) = \prod_{X \in \mathbf{V}_C} P(x|pa(X)) \ , \tag{2.3}$$

where $Pa_C(X)$ and $Pa_D(X)$ denotes the parents of X that are chance and decision variables respectively, i.e., $Pa_C(X) = Pa(X) \cap \mathbf{V}_C$, and $Pa_D(X) = Pa(X) \cap \mathbf{V}_D$. Equation 2.28 represents the probability of configuration $\mathbf{v}_C$ when the decision variables are externally set to the values given by $\mathbf{v}_D$. This notation, introduced by Cowell et al. (1999), is equivalent to the notation used by Pearl (1994, 2000), $P(\mathbf{v}_C|do(\mathbf{v}_D))$.[4]

A very simple example of influence diagram is the **test problem**.

Example 2.3.1 (Test problem) *A physician has to decide whether to treat or not a patient, who may suffer from a disease (X). Before deciding if to treat the patients (D), the physician can decide to perform a test (T). This test will produce the test result (Y), which would help to determine whether the patient suffers from the disease.*

[3] A super-value node U_i representing a combination function other than the sum or the product can be transformed into an ordinary utility node as follows: if U_j is a parent of U_i, we remove U_j from the ID and add its parents as new parents of U_i, and proceed recursively until no parent of U_i is a utility node. The new utility function for U_i derives from the original utility function of U_i and from those of its utility ancestors in the original ID. This transformation is necessary for both our algorithm and arc reversal (Tatman and Shachter, 1990).

[4] $P(\mathbf{v}_C : \mathbf{v}_D)$ should not be confused with $P_\Delta(\mathbf{v}_C|\mathbf{v}_D)$, which we will present further and that is directly derived from the joint probability distribution $P_\Delta(\mathbf{v}_C, \mathbf{v}_D)$ by using Eq. 2.4 and which only makes sense after selecting a strategy Δ. On the contrary, $P(\mathbf{v}_C : \mathbf{v}_D)$ represents the probability of $\mathbf{v}_C$ if the actions given by $\mathbf{v}_D$ are externally set, independently of the values of the variables observed when making each decision.

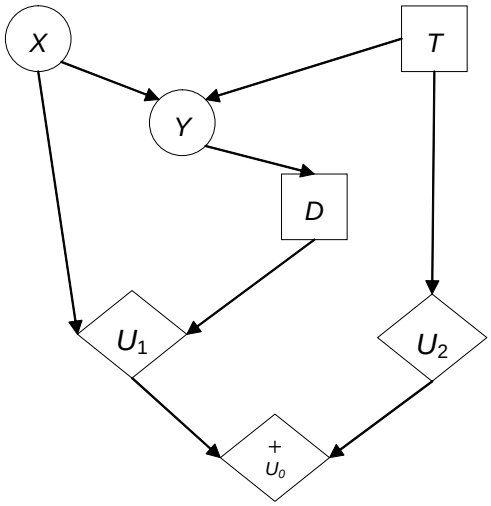

Figure 2.1: An influence diagram representation of the test problem (Example 2.3.1).

An ID representation of this decision problem is given in Figure 2.1. The decision node T designates the decision about whether or not to perform the test to the patient. The chance node Y represents the result of the test (if the test is performed). The utility function associated with the utility node U_1 encodes the cost of performing the test. The decision D is the decision whether or not to treat for the disease. The chance node X corresponds to the diagnosis of the patient. The utility function associated with the utility node U_2 specifies the health state of the patient as a function of the treatment and the diagnosis. The SVN U_0 represents the health state as a sum of the cost of the test (U_2) and the health state of the patient after treating him (U_1).

The directed path from T to D indicates that the physician decides on T before deciding on D. The informational arc from Y to D specifies that the test result is known before deciding on D. On the other hand, as there is no informational arc from X to either of the decision nodes, the state of X is observed (possibly never) after deciding on D. The nodes U_1 and U_2 have as parents to the variables in the domain of their respective utility functions. Thus, the SVN U_0 has as parents to U_1 and U_2, which are combined with sum. The utility function of U_0 can therefore be expressed in terms of $pa(U_1) \cup pa(U_2) = \{T, X, D\}$.

Finally, with respect to the states of the variables in the ID of Figure 2.1, decision test T has two states $+t$ and $\neg t$, and the result of the test (Y) has three states: $+y$, $\neg y$ and *no-result*. Thus, the probability distribution of Y has to reflect that the result of the test is only available if the physician decides to perform it.

2.3.1 Policies and strategies

A *stochastic policy* for a decision D is a probability distribution defined over D and conditioned on the set of its informational predecessors, $P_D(d|iPred(D))$. If P_D is degenerate (consisting of ones and zeros only) then we say that the policy is *deterministic*.

A *strategy* Δ for an ID is a set of policies, one for each decision, $\{P_D \mid D \in \mathbf{V}_D\}$. If every policy in the strategy Δ is deterministic, then Δ is said to be *deterministic*; otherwise Δ is *stochastic*. A strategy Δ *induces* a joint probability distribution over $\mathbf{V}_C \cup \mathbf{V}_D$ defined as follows:

$$P_\Delta(\mathbf{v}_C, \mathbf{v}_D) = P(\mathbf{v}_C : \mathbf{v}_D) \prod_{D \in \mathbf{V}_D} P_D(d|iPred(D)) = \prod_{C \in \mathbf{V}_C} P(c|pa(C)) \prod_{D \in \mathbf{V}_D} P_D(d|iPred(D)) \ . \tag{2.4}$$

Let I be an ID, Δ a strategy for I and $\mathbf{r}$ a configuration defined over a set of variables $\mathbf{R} \subseteq \mathbf{V}_C \cup \mathbf{V}_D$ such that $P_\Delta(\mathbf{r}) \neq 0$. The conditional probability distribution *induced by the strategy* Δ *given the configuration* $\mathbf{r}$, defined over $\mathbf{R}' = (\mathbf{V}_C \cup \mathbf{V}_D) \setminus \mathbf{R}$, is given by:

$$P_\Delta(\mathbf{r}'|\mathbf{r}) = \frac{P_\Delta(\mathbf{r}, \mathbf{r}')}{P_\Delta(\mathbf{r})}. \tag{2.5}$$

For example, in the ID of Figure 2.1 we have:

$$P_\Delta(\mathbf{v}_C, \mathbf{v}_D) = P(x) \cdot P(y|t, x) \cdot P_T(t) \cdot P_D(d|t, y),$$

where P_T and P_D are the policies contained in strategy Δ. If we have $\mathbf{R} = \{T, Y\}$, then the conditional probability distribution induced $P_\Delta(\mathbf{r}'|\mathbf{r})$ is:

$$P_\Delta(\mathbf{r}'|\mathbf{r}) = P_\Delta(x, d|t, y) = \frac{P_\Delta(x, d, t, y)}{P_\Delta(t, y)}. \tag{2.6}$$

Thus, we have in Equation 2.6 the conditional probability distribution $P_\Delta(x, d|t, y)$, which represents the posteriori probability of x and d, given the values t and y.

Using the distribution $P_\Delta(\mathbf{r}'|\mathbf{r})$ defined in Equation 2.5 we can compute the *expected utility of* U *under the strategy* Δ *given the configuration* $\mathbf{r}$ as:

$$EU_U(\Delta, \mathbf{r}) = \sum_{\mathbf{r}'} P_\Delta(\mathbf{r}'|\mathbf{r})\, \psi_U(\mathbf{r}, \mathbf{r}'). \tag{2.7}$$

For the terminal utility node U_0, $\mathrm{EU}_{U_0}(\Delta, \mathbf{r})$ is said to be the *expected utility of the strategy* Δ *given the configuration* $\mathbf{r}$, and denoted by $\mathrm{EU}(\Delta, \mathbf{r})$. For example, in the ID

of Figure 2.1, if we have $\mathbf{R} = \{T, Y\}$, then the expected utility the strategy Δ given the configuration $\mathbf{r} = \{t, y\}$ is as follows:

$$\mathrm{EU}_U(\Delta, \mathbf{r}) = \sum_x \sum_d P_\Delta(x, d|t, y) \cdot \psi_{U_0}(x, y, d, t) = \sum_x \sum_d \frac{P_\Delta(x, d, t, y)}{P_\Delta(t, y)}(U_1(x, d) + U_2(t)), \tag{2.8}$$

by using Equation 2.6.

We define the *expected utility of U under the strategy Δ* as $\mathrm{EU}_U(\Delta) = \mathrm{EU}_U(\Delta, \blacklozenge)$, where $\blacklozenge$ is the empty configuration. We have that

$$\mathrm{EU}_U(\Delta) = \sum_{\mathbf{v}_C} \sum_{\mathbf{v}_D} P(\mathbf{v}_C, \mathbf{v}_D)\, \psi_U(\mathbf{v}_C, \mathbf{v}_D). \tag{2.9}$$

We also define the *expected utility of the strategy Δ* as $\mathrm{EU}(\Delta) = \mathrm{EU}_{U_0}(\Delta)$.

An *optimal strategy* is a strategy Δ_{opt} that maximizes the expected utility:

$$\Delta_{opt} = \arg\max_{\Delta \in \Delta^*} \mathrm{EU}(\Delta), \tag{2.10}$$

where Δ^* is the set of all the strategies for I. Each policy in an optimal strategy is said to be an *optimal policy*. The *maximum expected utility* (MEU) is

$$MEU = EU(\Delta_{opt}) = \max_{\Delta \in \Delta^*} EU(\Delta). \tag{2.11}$$

The evaluation of an ID consists in finding the MEU and an optimal strategy, composed by an optimal policy for each decision. It can be proved (Cowell et al., 1999) that

$$MEU = \sum_{\mathbf{c}_0} \max_{d_0} \ldots \sum_{\mathbf{c}_{n-1}} \max_{d_{n-1}} \sum_{\mathbf{c}_n} P(\mathbf{v}_C : \mathbf{v}_D)\, \psi_{U_0}(\mathbf{v}_C, \mathbf{v}_D). \tag{2.12}$$

An *optimal policy* δ_{D_i} is therefore a function that maps each configuration of the variables in $\mathrm{iPred}(D_{i-1})$, i.e., those at the left of $\max_{D_i}$ in the above expression, onto the value d_i of D_i that maximizes the expression at the right of D_i (in the case of a tie, any of the values of D_i that maximize that expression can be chosen arbitrarily):

$$\delta_{D_i}(\mathrm{iPred}(D_i)) = \arg\max_{d_i \in D_i} \sum_{\mathbf{c}_i} \max_{d_{i+1}} \ldots \sum_{\mathbf{c}_{n-1}} \max_{d_n} \sum_{\mathbf{c}_n} P(\mathbf{v}_C : \mathbf{v}_D)\, \psi_{U_0}(\mathbf{v}_C, \mathbf{v}_D)\ . \tag{2.13}$$

For instance, the *MEU* for the ID in Fig. 2.1 is

$$MEU = \max_t \sum_y \max_d \sum_x P(x) \cdot P(y|t,x) \cdot (U_1(x,d) + U_2(t)) \tag{2.14}$$

and an optimal policy δ_D is

$$\delta_D(b) = \arg\max_{d \in D} \sum_x P(x) \cdot P(y|t,x) \cdot (U_1(x,d) + U_2(t)) \tag{2.15}$$

There can be more than one optimal strategy for an ID. However, we can always find an optimal strategy that is deterministic. The literature about IDs usually assume that the strategies in IDs are deterministic. We will also assume in the dissertation that the strategies are deterministic, except when we point out that they can also be stochastic.

2.3.2 Evaluation algorithms

Several algorithms have been proposed for evaluating IDs without SVNs (Shachter, 1986; Jensen et al., 1994; Dechter, 1996; Cowell et al., 1999). However, there is only one algorithm for IDs with SVNs (Tatman and Shachter, 1990), which is described in this section.

Algorithm of Tatman and Shachter Tatman and Shachter (1990) proposed the first algorithm for evaluating IDs with SVNs. They employed a similar scheme to the arc reversal (AR) algorithm (Olmsted, 1983; Shachter, 1986) used for IDs without SVNs. It was adapted for ID with SVNs by maintaining the separability of the utility function ψ as long as possible during the evaluation of the ID.

The algorithm accomplishes successive transformations on the ID. The new ID obtained after each transformation preserve the *MEU* and the optimal policies of the original ID. The transformations operate over the probability potentials of the chance nodes and over the utility functions, and compute the optimal policies of the decisions.

We explain the five basic transformations performed by algorithm of Tatman and Shachter: *reduce*, *merge*, *eliminateChance*, *eliminateDecision* and *reverseArc*.

reduce The most basic method is *reduce* (see Algorithm 2.1), which converts a SVN U of the ID in other utility node with same utility function ψ_U, but whose parents are $fPred(U)$. The utility nodes that are ancestors of U and their incoming and outgoing links are removed from the ID[5].

[5]The utility nodes ancestors of U can be removed from the ID because Tatman and Shachter's algorithm assume that for each utility node there is an only path connecting it to U_0. Otherwise, those utility

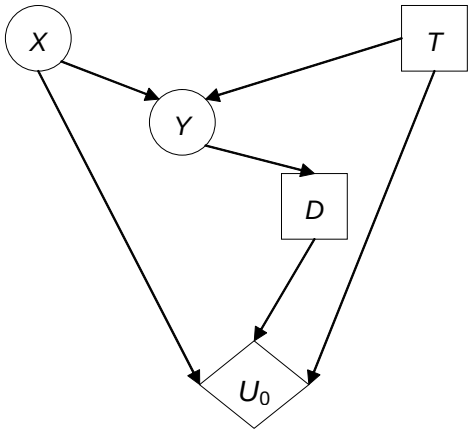

Figure 2.2: ID obtained after reducing SV node U_0 in ID of Figure 2.1.

For, example, SVN U_0 can be reduced in ID of Figure 2.1, which would convert the ID into the new ID of Figure 2.2. Utility nodes U_1 and U_2 and their links have been removed. Moreover, while U_0 was a SVN in the original ID, it has been converted into an ordinary utility node. The parents of U_0 in the new ID are all the parents of U_1 and U_2 in the original ID. The utility function of U_0 has not changed. However, it is not decomposed with a structure of SVNs. On the contrary, the operation of reduction has calculated the values of the utility function of U_0 as:

$$U_0(x, t, d) = U_1(x, d) + U_2(t).$$

merge Another basic method is *merge* (see Algorithm 2.2). It combines several utility nodes that have a common child U into a new SVN U' and then reduces U'. If the set of utility nodes to be merged is the entire set of parents of U, then merging them simply reduces U.

For example, let us consider an example with the ID in Figure 2.3. Utility nodes U_2 and U_3 can be merged. That operation would create a SVN U_4 that combines with a sum the nodes U_2 and U_3, as we can see in the ID of Figure 2.4. U_4 is then reduced, obtaining the new ID of Figure 2.5.

eliminateChance The third operation of the algorithm is given by method *eliminateChance*, described by Algorithm 2.3, which eliminates a chance node.

The next definition and theorem establishes the conditions that a node must satisfy

nodes should not necessarily be removed from the ID.

Algorithm 2.1 reduce

Input: U : a utility node of an ID I;

Effects: the node U is converted in an ordinary utility node, the utility nodes ancestors of U are eliminated from I, the new parents of U are $fPred(U)$ and the utility function ψ_U is computed.

1. **if** $U \notin \mathbf{V}_{OU}$ **then**
2. **for all** $U' \in pa(U)$ **do**
3. $reduce(U')$;
4. **end for**
5. $\Phi_U := \{\psi_{U'} \mid U' \in \text{pa}(U)\}$;
6. **if** U is a sum node **then**
7. $\psi_U := \sum\{\psi \mid \psi \in \Phi_U\}$;
8. **else**
9. $\psi_U := \prod\{\psi \mid \psi \in \Phi_U\}$;
10. **end if**
11. $fPred := \cup\{pa(U') \mid U' \in pa(U)\}$;
12. **for all** $U' \in pa(U)$ **do**
13. delete U' from I;
14. **end for**
15. **for all** $X \in fPred$ **do**
16. add the arc $X \rightarrow U$;
17. **end for**
18. **end if**

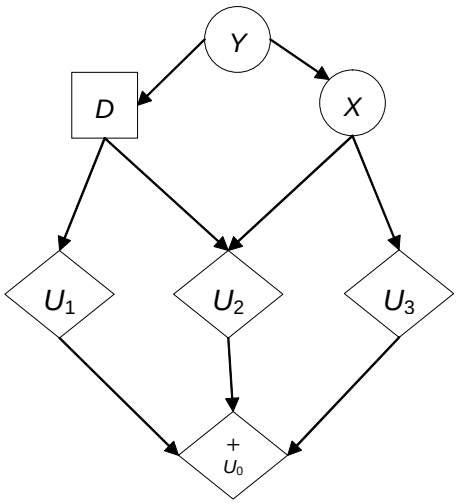

Figure 2.3: Example of ID where nodes U_2 and U_3 are going to be merged.

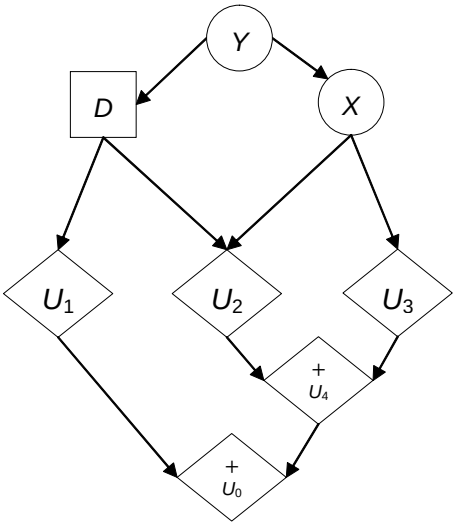

Figure 2.4: ID obtained from the ID of Figure 2.3 when merging utility nodes U_2 and U_3. SV node U_4 has been added to the ID and will be reduced.

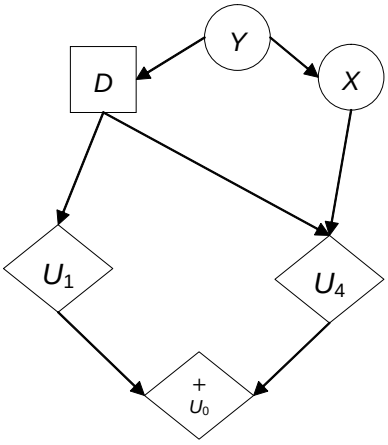

Figure 2.5: ID after reducing U_4 in ID of Figure 2.4.

Algorithm 2.2 merge

Input: **U** : a set of utility nodes of an ID I, parents of the same SVN U';
Effects: the nodes in **U** are merged.

1. **if** $pa(U') = \mathbf{U}$ **then**
2. $\quad reduce(U')$;
3. **else**
4. $\quad$ **for all** $U \in \mathbf{U}$ **do**
5. $\quad\quad$ eliminate the arc $U \rightarrow U'$;
6. $\quad$ **end for**
7. $\quad$ add a new SVN U'' of the same type that U';
8. $\quad$ add the arc $U'' \rightarrow U'$;
9. $\quad$ **for all** $U \in \mathbf{U}$ **do**
10. $\quad\quad$ add the arc $U \rightarrow U''$;
11. $\quad$ **end for**
12. $\quad reduce(U'')$;
13. **end if**

Algorithm 2.3 eliminateChance

Input: I : an ID with SVNs; A: chance node to be eliminated, whose only successor is the utility node U;
Effects: the chance node A is eliminated from I.

1. $\psi'_U := \sum_a P(a|pa(A))\,\psi_U(pa(U))$;
2. **for all** $X \in pa(A)$ **do**
3. $\quad$ add the arc $X \rightarrow U$;
4. **end for**
5. replace ψ_U with ψ'_U as utility function of U;
6. delete the node A from I;

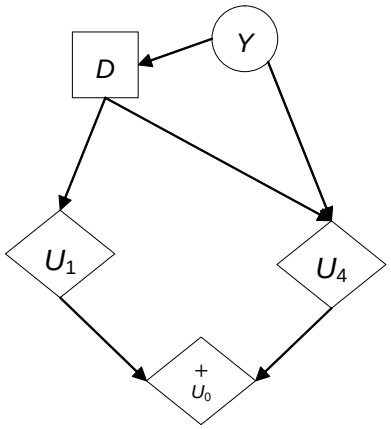

Figure 2.6: ID obtained by eliminating chance node X from the ID of Figure 2.5.

before it can be eliminated.

Definition 2.3.1 *A chance node C is said to be removable if all its children are utility nodes.*

Theorem 2.3.1 *Let C be a removable chance node. Then C can be eliminated from the ID by using Algorithm 2.3, after possibly previous reductions and/or merges of utility nodes to leave it with an only child.*

The proof of the correctness of the theorem can be found in (Tatman and Shachter, 1990).

When a chance node A is going to be eliminated its only child is a utility node U. The elimination basically consists in updating the utility function of U according to:

$$\psi'_U := \sum_a P(a|pa(A))\,\psi_U(pa(U)).$$

The parents of A in the original ID are parents of U in the new ID. Finally, node A and its links are removed from the ID.

Let us see the ID of Figure 2.3. Chance node X satisfies the conditions of Theorem 2.3.1 and can therefore be eliminated, after previous reductions and/or merges of utility nodes. The operations required for leaving X with an only child have been described when explaining the method *merge*, i.e, merging U_2 and U_2 becomes X in removable (see Figure 2.5). Finally, chance node X can be eliminated from the ID of Figure 2.5, obtaining the ID of Figure 2.6.

eliminateDecision Another method of the algorithm is *eliminateDecision*, described by Algorithm 2.4, which eliminates a decision node.

Algorithm 2.4 eliminateDecision

Input: D: decision node to be eliminated, whose only successor is the utility node U;
Effects: the decision node D is eliminated from I;
Output: an optimal policy δ_D for D.

1.
2. $\psi'_U := \max_D \psi_U(Pa(U))$;
3. $\delta_D := \arg\max_D \psi_U(Pa(U))$;
4. replace ψ_U with ψ'_U as utility function of U;
5. delete the node D from I;
6. **return** the optimal policy δ_D;

The conditions to be satisfied by a decision node before eliminating it are described by the next theorem.

Definition 2.3.2 *A decision node is said to be removable if all utility nodes take on only non-negative values and all its children are utility nodes.*

Theorem 2.3.2 *Let D be a removable decision node. Then D can be eliminated from the ID by using Algorithm 2.4, after possibly previous reductions and/or merges of utility nodes to leave it with an only child whose parents are D or any of the parents of D.*

The proof of the correctness of the theorem can be found in (Tatman and Shachter, 1990).

When a decision node D is going to be eliminated its only child is a utility node U. The elimination basically consists in updating the utility function of U according to:

$$\psi'_U := \max_D \psi_U(Pa(U)).$$

Finally, node D and its links are removed from the ID. The removal of D can leave some nodes without descendants. Such nodes are said to be *barren*. Barren nodes can be removed directly from the ID because they do not influence the optimal policy of any decision nor the MEU.

For example, let us focus on the ID of Figure 2.7. Decision D is removable, because all its children are utility nodes. If we reduce U_2 we obtain the ID of Figure 2.8, where D appears with an only child, U_2, and can be eventually eliminated. The ID obtained after eliminating D is given by Figure 2.9.

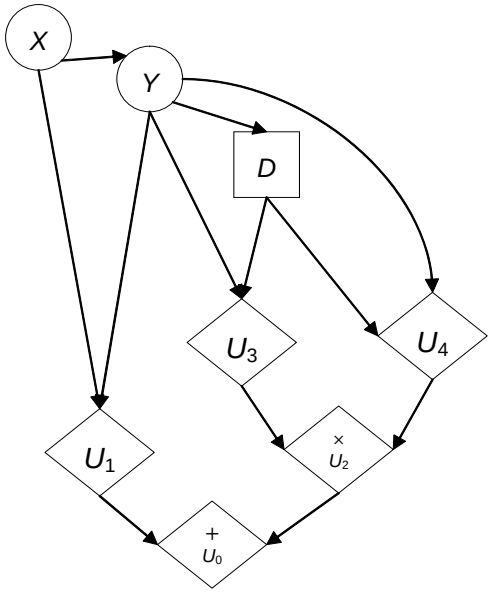

Figure 2.7: ID in which decision D is removable.

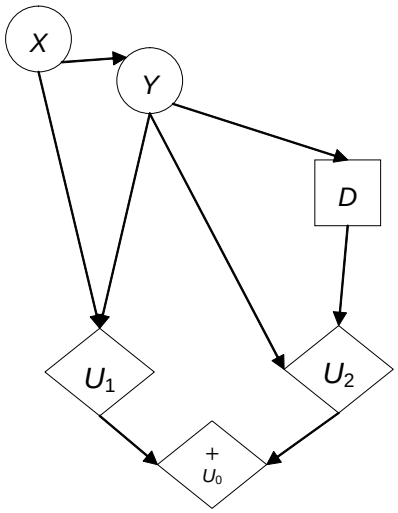

Figure 2.8: ID after reducing U_2 in ID of Figure 2.7.

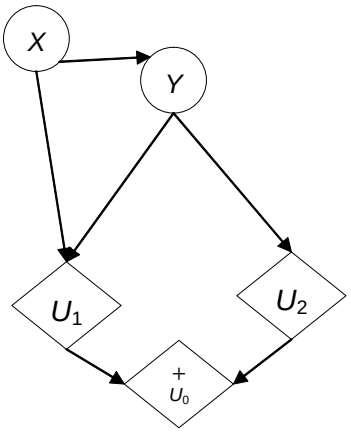

Figure 2.9: ID obtained by eliminating D from the ID of Figure 2.8.

reverseArc Algorithm 2.5 describes the fifth operation to perform on the ID, named *reverseArc*, which reverses an arc in the ID by applying Bayes' rule.

Algorithm 2.5 reverseArc

Input: $X \rightarrow Y$: arc to be reversed in the ID;
Effects: the arc $X \rightarrow Y$ is reversed.

1. $Pa'_Y := (Pa(Y) \setminus \{X\}) \cup Pa(X)$;
2. $Pa'_X := (Pa(X) \cup \{Y\}) \cup Pa(Y)$;
3. $P'(Y|Pa'_Y) := \sum_x P(x|Pa(X))\, P(Y|Pa(Y))$;
4. $P'(X|Pa'_X) := \dfrac{P(X|Pa(X))P(Y|Pa(Y))}{P'(Y|Pa'_Y)}$;
5. eliminate the arc $X \rightarrow Y$;
6. **for all** $Z \in (Pa'_X \setminus Pa(X))$ **do**
7. add the arc $Z \rightarrow X$;
8. **end for**
9. assign to X the conditional probability distribution $P'(X|Pa'_X)$;
10. **for all** $Z \in (Pa'_Y \setminus Pa(Y))$ **do**
11. add the arc $Z \rightarrow Y$;
12. **end for**
13. assign to Y the conditional probability distribution $P'(Y|Pa'_Y)$;

The next theorem states that, under certain conditions, a chance node is removable after the appropriate arc reversals.

Theorem 2.3.3 *Let C be a chance node. If C is a functional predecessor of U_0 and none of the decision nodes is child of C, then, after reversing the arcs in the set $\{C \rightarrow C' \mid C'$ is chance node and child of $C\}$, the node C is removable. We will say in this case that the chance node C is removable after arc reversals.*

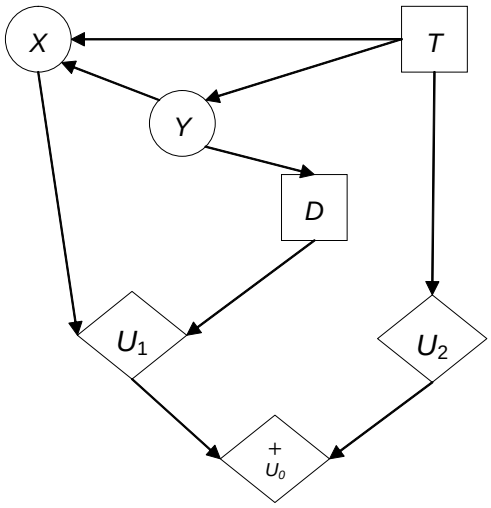

Figure 2.10: ID after reversing arc $X \rightarrow Y$ in ID of Figure 2.1.

Proof. After the arc reversals, there can only be utility nodes as children of C. Thus, the node C is removable. ■

Let us consider the ID of the test problem, displayed in Figure 2.1. There are two chance nodes in that ID: X and Y. Node Y is not removable because decision D is child of Y. However, children of X are utility node U_1 and chance node Y. Thus, X is removable after arc reversals. The only arc to be reversed is $X \rightarrow Y$.

After reversing the arc $X \rightarrow Y$ the only parent of Y is T, and the new parents of X are Y and T. The new conditional probability distribution of X and Y, $P'(x|y,t)$, is and $P'(y|t)$ respectively, are:

$$P'(y|t) := \sum_x P(x|y,t)$$

$$P'(x|y,t) := \frac{P(x)P(y|x,t)}{P'(y|t)} \ .$$

The ID obtained after reversing the arc $X \rightarrow Y$ of the ID of the test problem is illustrated by Figure 2.10. Node X is removable on that ID.

Additionally, Tatman and Shachter (1990) proposed a heuristic for reducing the storage size of the utility functions during the evaluation, termed *subset rule*. They explain that, if two utility nodes U_1 and U_2 have the same successor, a SVN U, and $Pa(U_2) \subseteq Pa(U_1)$, then eliminating U_1 and U_2 will not increase the size of any operation necessary for solving the ID, and should therefore be merged.

The algorithm of Tatman and Shachter is shown as Algorithm 2.6. We will name it *Arc Reversal* and we will refer to it as AR in the rest of the dissertation.

Algorithm 2.6 Algorithm AR (Arc Reversal)

Input: I : an ID with SVNs;
Output: the MEU of the ID and an optimal policy δ_D for each decision variable D.

```
1. delete barren nodes from I;
2. while pa(U_0) ≠ ∅ do
3.    if the subset rule can be applied to some set of utility nodes U then
4.       merge(U);
5.    else if there exists a chance node C that is removable then
6.       merge and reduce the necessary utility nodes for eliminating C;
7.       eliminateChance(C);
8.    else if there exists a decision node D that is removable then
9.       merge and reduce the necessary utility nodes for eliminating D;
10.      eliminateDecision(D);
11.      delete barren nodes from I;
12.   else if there exists a chance node C that is removable after arc reversals then
13.      for all X ∈ Children(C) do
14.         reverseArc(X → C);
15.      end for
16.   end if
17. end while
```

2.4 Unconstrained influence diagrams

Unconstrained influence diagrams (UIDs) were proposed by Jensen and Vomlelova (2002) to represent and solve decision problems in which the order of some decisions is unspecified and the decision maker is interested in knowing the best ordering. Let us see the limitations of IDs with partially ordered decisions through a simplified medical example: the **diabetes diagnosis problem** (Demirer and Shenoy, 2001). After that, we will review the UIDs framework.

2.4.1 Limitations of IDs for partially ordered decisions

Example 2.4.1 *[Diabetes diagnosis problem]A physician is trying to find a policy for treating a patient who may suffer from diabetes. After an initial examination of his symptoms, the physician has to diagnose whether he suffers from diabetes. Diabetes has two symptoms, glucose in urine and glucose in blood. The physician can decide to perform a urine test and/or a blood test. Their corresponding test results are observed before deciding on whether to treat the patient. The order in which the tests are performed is not specified. The physician can decide to not perform any test.*

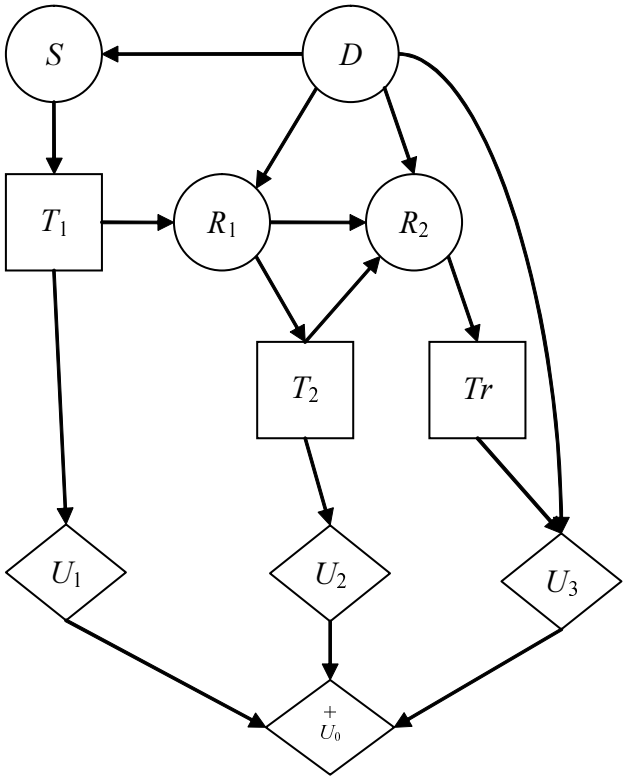

Figure 2.11: ID for modeling the diabetes diagnosis problem.

IDs are not suitable for a problem like this in which decisions are not totally ordered. We would need to circumvent this limitation by representing the unspecified order of the tests as a linear ordering of decisions. To model the diabetes diagnosis problem it can be done by introducing two decision variables T_1 and T_2, as shown in Figure 2.11; these two variables model the first test and the second one, respectively. The arc $R_1 \rightarrow R_2$ indicates that a repetition of a test would give an identical result.

With respect to the states of the variables, the test decisions (T_1 and T_2) have three states, *bt*, *ut* and *no-test*. The result nodes (R_1 and R_2) have five states, $+t$, $\neg t$, $+u$, $\neg u$, and *no-result*. It has been necessary to add the state *no-result* to reflect the fact that the result of the test is unknown if the physician decides not to perform it.

Unfortunately, the structure of the decision problem is not apparent from the model and for large decision problems this technique would be prohibitive, as all possible scenarios should be explicitly encoded in the model.

As an alternative model, Jensen and Vomlelova (2002) introduced unconstrained influence diagrams to represent this kind of decision problems. In the UID framework, the combinatorial problem of representing non-sequential decision problems is postponed to the solution phase.

2.4.2 UID representation

An *unconstrained influence diagram* (UID) is an ADG over three sets of nodes: decision nodes $\mathbf{V}_D$, chance nodes $\mathbf{V}_C$, and utility nodes $\mathbf{V}_U$. The semantics of these sets of nodes is identical to the case of IDs. Nevertheless, the set of chance nodes nodes $\mathbf{V}_C$ is partitioned into two subsets: *observable* nodes, $\mathbf{V}_O$ (drawn as double circles), and *non-observable* nodes, $\mathbf{V}_N$ (single circles).

As in IDs, the quantitative information associated with a UID consists of probability distributions and utility functions. However, UIDs do not have super-value nodes and assume that the utility functions combine additively into a joint utility function, ψ. Moreover, a convention in UIDs establishes that each decision variable D has a cost. If this cost only depends on D, it is not represented graphically, and the cost function is attached to D.

The semantics of the links is equivalent to the case of IDs, and the no-forgetting hypothesis is also assumed. However, a total ordering of the decision nodes is not required. While non-observable variables are variables that will never be observed, an observable variable will be observed when all its antecedent decisions have been made, i.e., an option has been chosen for each decision.

The structural specification of a UID yields a partial temporal order. If a partial order is extended to a total order we get an influence diagram. Such an extended order is called an *admissible order*.

For example, a UID for the diabetes diagnosis problem is shown in Figure 2.11. Decision nodes BT and UT model the decisions of performing the blood test and the urine test respectively. The results of these tests are represented by the observable chance variables B and U. The chance variable D models the presence or absence of diabetes, and the symptom is presented in the UID by variable S. The decision node Tr designates the decision about whether to treat the patient. The utility node V represents the health state of the patient as a function of the treatment and the diagnosis. Contrary to the case of the ID in Figure 2.11, the UID of Figure 2.12 does not require to represent the costs of the urine and the blood tests graphically because their utility functions only depend on the test decision.

With respect to when the variables are observed, for instance, in Figure 2.12 S is observed before the first decision, since it has no antecedent decision variables, and B is observed after deciding on BT.

The directed paths from BT and UT to D indicate that the physician decide on BT and UT before deciding on D. The arc from BT to B specifies that the result of the

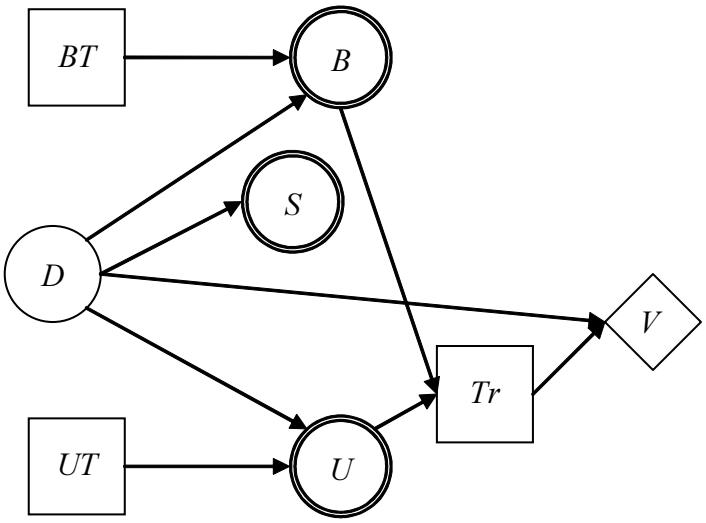

Figure 2.12: An unconstrained influence diagram representation of the diabetes diagnosis problem.

blood test B can be observed after deciding on BT. Similarly, the result of the urine test U is available after making decision UT. Variable S can be observed before making any decision, since it has no antecedent decision variables. On the other hand, as variable D is non-observable, it cannot be observed before deciding on Tr. Let us note that the UID of Figure 2.12 does not specify the order of decisions BT and UT.

Finally, with respect to the states of the variables in the UID of Figure 2.12, the test decisions (BT and UT) have two states $+bt$ and $\neg bt$, and $+ut$ and $\neg ut$, respectively. The result of the test B has three states: $+b$, $\neg b$ and *no-result.* Similarly, the states of U are $+u$, $\neg u$ and *no-result.* The fact that the result of a test is only available if the physician decides to perform it is encoded in the probability distributions of B and U; for instance, $P(+b|\neg bt) = P(+b|\neg bt) = 0$, and $P(\textit{no-result}|\neg bt) = 0$.

2.4.3 Solving a UID

Solving a UID means calculating an optimal strategy. However, the concept of strategy is more complex than in IDs.

An *S-DAG* is a DAG G whose nodes contain variables from $\mathbf{V}_D \cup \mathbf{V}_O$. The set of variables contained in a node N of G is called *labels* of N, and denoted by $\textit{labels}(N)$. Each maximal directed path in G represents an admissible ordering of the variables in $\mathbf{V}_D \cup \mathbf{V}_O$. A *GS-DAG* is a minimal GS-DAG containing all admissible orderings for computing an

optimal strategy.

Let N be a node in the S-DAG G. The *history* of N, denoted by $hst(N)$, contains $labels(N)$ and the labels of its ancestors. We distinguish two kinds of *policies*: *step-policy* and *decision-policy*. A *step-policy* for a node N in an S-DAG is a conditional probability potential $P_N(ch(N)|hst(N))$ that maps each configuration of the history of N into a probability distribution over the children of N. A *decision-policy* for a decision D of a node N in a S-DAG is a probability distribution $P_{N,D}(d|hst(N))$ that assigns to each configuration of $hst(N)$ a distribution over the alternatives for D. If a policy is degenerate (consisting of ones and zeros only) then we say that the policy is *deterministic*.[6]

Definition 2.4.1 *Let I be a UID and let G be a S-DAG for I. A* strategy *for I is a pair $\Delta = \{\Delta_S, \Delta_D\}$, where:*

- $\Delta_S = \{P_N \mid N$ is a node in G and P_N is a step-policy for $N\}$, and
- $\Delta_D = \{P_{N,D} \mid N$ is a node in G, D is a decision node in $labels(N)$, and $P_{N,D}$ is a decision-policy for decision D of $N\}$.

Let us see the example of UID in Figure 2.13 for clarifying the concepts defined for solving a UID.

According to the semantics of UIDs, decisions T_1 and T_2 are unordered in the UID, but both precede T_3. Finally, T_4 is the last decision to be made. The partial order of decisions specified by the UID is used by Jensen and Vomlelova (2002) for building an S-DAG where the UID is solved. An S-DAG for the UID of Figure 2.13 is presented in Figure 2.14. It is also a GS-DAG because it is the minimal S-DAG containing all admissible orderings for solving the UID.

Following the S-DAG of Figure 2.14, a strategy for the UID of Figure 2.13 has to contain:

- A step-policy for the branch point in node S, $P(ch(S)|s)$, which assigns a probability of selection of the upper and lower branches created by the outgoing arcs from S.
- Six decision-policies: One for each decision variable appearing in the S-DAG. With regard to the domain of the decision-policies, the policy $P_{\{T_3\}}$ required in the node appearing T_3 is $P_{\{T_3\}}(t_3|s, t_1, r_1, r_2, t_2, r_3)$. A policy $P_{\{T_1\}^u}$ in the node appearing T_1 in the upper branch of the S-DAG has the form $P_{\{T_1\}^u}(t_1|s)$, which is different from the policy $P_{\{T_1\}^l}$ for the node with T_1 in the lower branch: $P_{\{T_1\}^l}(t_1|t_2, r_3, t_1, r_1, r_2)$.

[6] Our definition of policy differs from that Jensen and Vomlelova (2002) proposed in the original formulation of UIDs because we allow policies to be not only deterministic functions but also probability distributions.

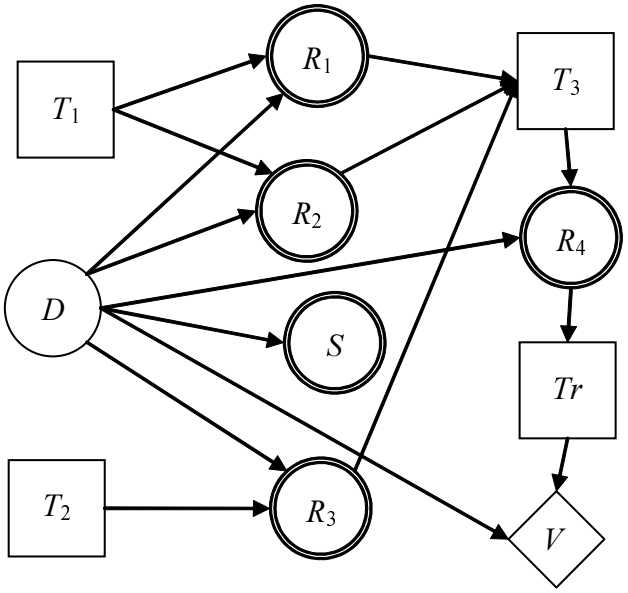

Figure 2.13: Example of UID.

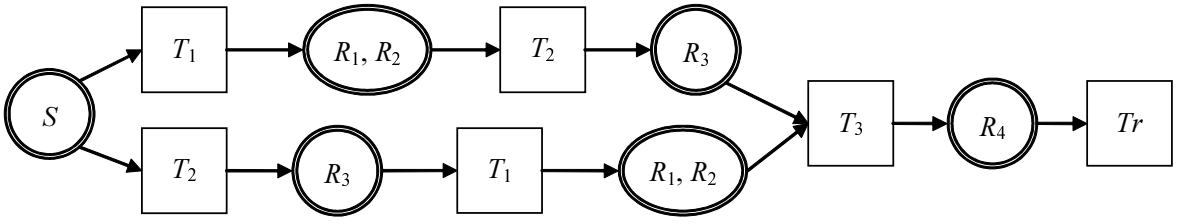

Figure 2.14: SDAG for the UID of Figure 2.13.

A strategy Δ induces a joint probability distribution over $\mathbf{V}_C \cup \mathbf{V}_D$ defined by:

$$P_\Delta(\mathbf{v}_C, \mathbf{v}_D) = \sum_N \prod_N P_N(ch(N)|hst(N)) \prod_{C \in \mathbf{V}_C} P(c|pa(C)) \tag{2.16}$$

$$= \prod_{D \in (\mathbf{V}_D \cap var(N))} P_{N,D}(d|hst(N)) \tag{2.17}$$

Definition 2.4.2 *Let N be a node of the S-DAG G and let $\mathbf{X}$ and $\mathbf{Y}$ be two subsets of variables in N such that $\mathbf{X} \subseteq var(N)$ and $\mathbf{Y} = var(N) \setminus \mathbf{X}$. The* expected utility *of Δ for a node N and a configuration $\mathbf{e}$ of $\mathbf{E} \subseteq hst(N)$ is denoted by $EU_\Delta(N, \mathbf{e})$. If $\mathbf{Y} \neq \varnothing$, then $EU_\Delta(N, \mathbf{e})$ is given by:*

$$EU_\Delta(N, \mathbf{e}) = \sum_y P_\Delta(y|\mathbf{e}) EU_\Delta(N, \{\mathbf{e}, y\}) \tag{2.18}$$

where $Y \in \mathbf{Y}$ and $P_\Delta(\mathbf{y}|\mathbf{e}) = P_\Delta(y, \mathbf{e})/P_\Delta(\mathbf{e})$. If $\mathbf{Y} = \varnothing$, then $EU_\Delta(N, \mathbf{e})$ is defined by:

$$EU_\Delta(N, \mathbf{e}) = \begin{cases} \sum_{N' \in ch(N)} P_N(N'|\mathbf{e}) EU_\Delta(N', \mathbf{e}) & \text{if } ch(N) \neq \varnothing \\ U(\mathbf{e}) & \text{if } ch(N) = \varnothing \end{cases} \tag{2.19}$$

We define the expected utility of strategy Δ *as $EU_\Delta = EU_\Delta(N_0, \blacklozenge)$, where N_0 is the root node in G and $\blacklozenge$ is the empty configuration.*

An *optimal strategy* is a strategy Δ_{opt} that maximizes the expected utility. Similarly to IDs, we can always find an optimal strategy that is deterministic.

Jensen and Vomlelova (2002) proposed to solve UIDs by constructing a GS-DAG G and solving G through dynamic programming in way similarly to solving influence diagrams (i.e., eliminating the variables in variables in reverse temporal order). Initially, we have a set of probability and utility potentials. The non-observable variables are eliminated first. Then chance and decision variables are eliminated as in variable-elimination for IDs. When a bifurcation point is met, the elimination branches out and the potentials are transferred to the different branches. When several branches meet, the probability tables will be identical, and the utility potentials are unified through maximization. The latter also generates the step-policy for the branching node.

Let us see how the evaluation of the diabetes diagnosis problem is performed by the algorithm proposed by Jensen and Vomlelova (2002). A GS-DAG for the diabetes diagnosis problem appears in Figure 2.15.

The solution of the Diabetes diagnosis problem through the GS-DAG is as follows.

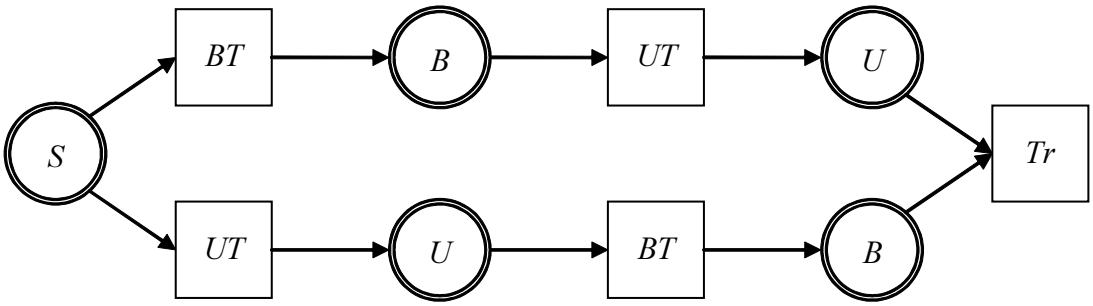

Figure 2.15: GS-DAG of the UID of diabetes problem (see Figure 2.12.

We start with the sets of potentials:

$$\Phi = \{\phi_1(B, BT, D), \phi_2(U, UT, D), \phi_3(S, D), \phi_4(S, D)\}$$
$$\Psi = \{\psi_1(BT), \psi_2(UT), \psi_3(D, Tr)\} ,$$

where:

$$\begin{aligned}
\phi_1(B, BT, D) &= P(B|BT, D) \\
\phi_2(U, UT, D) &= P(U|UT, D) \\
\phi_3(S, D) &= P(S|D) \\
\phi_4(S, D) &= P(D) \\
\psi_1(BT) &= \psi_{BT}(BT) \\
\psi_2(UT) &= \psi_{UT}(UT) \\
\psi_3(D, Tr) &= \psi_V(D, Tr) .
\end{aligned}$$

Let us note that ψ_{BT} and ψ_{UT} are the costs of the urine and blood tests, not represented graphically in the UID.

First the non-observable variable D is eliminated. We therefore obtain the sets:

$$\Phi' = \{\phi_5(S, B, U, BT, UT)\}$$
$$\Psi' = \{\psi_1(BT), \psi_2(UT), \psi_4(S, BT, B, UT, U, Tr)\} ,$$

where

$$\phi_5(S,B,U,BT,UT) = \sum_D \phi_1(B,BT,D)\,\phi_1(B,BT,D)\,\phi_3(S,D)\,\phi_4(S,D)$$
$$\psi_4(S,BT,B,UT,U,Tr) =$$
$$= \frac{1}{\phi_5(S,B,U,BT,UT)} \sum_D \phi_1(B,BT,D)\,\phi_1(B,BT,D)\,\phi_3(S,D)\,\phi_4(S,D)\,\psi_3(D,Tr)\ .$$

Next the variable Tr is eliminated. Then, we get the next potentials:

$$\Phi' = \{\phi_5(S,B,U,BT,UT)\}$$
$$\Psi' = \{\psi_1(BT), \psi_2(UT), \psi_5(S,BT,B,UT,U)\}\ ,$$

where

$$\psi_5(S,BT,B,UT,U) = \max_{Tr} \psi_4(S,BT,B,UT,U,Tr)\ .$$

After that a bifurcation point is met, so we branch. In the upper branch, after eliminating U we obtain the next potentials:

$$\Phi^U = \{\phi_6^U(S,B,BT,UT)\}$$
$$\Psi^U = \{\psi_1(BT), \psi_2(UT), \psi_6^U(S,BT,B,UT)\}\ ,$$

where

$$\phi_6^U(S,B,BT,UT) = \sum_U \phi_5(S,B,U,BT,UT)$$
$$\psi_6^U(S,BT,B,UT) = \frac{1}{\phi_6^U(S,B,BT,UT)} \sum_U \phi_5(S,B,U,BT,UT)\,\psi_5(S,BT,B,UT,U)\ .$$

In the lower branch, after eliminating B we obtain the next potentials:

$$\Phi^B = \{\phi_6^B(S,U,BT,UT)\}$$
$$\Psi^B = \{\psi_1(BT), \psi_2(UT), \psi_6^B(S,BT,B,UT)\}\ ,$$

where

$$\phi_6^B(S,U,BT,UT) = \sum_B \phi_5(S,B,U,BT,UT)$$
$$\psi_6^B(S,BT,UT,U) = \frac{1}{\phi_6^B(S,U,BT,UT)} \sum_B \phi_5(S,B,U,BT,UT)\,\psi_5(S,BT,B,UT,U)\ .$$

When BT has been eliminated in the upper branch, and UT is eliminated in the lower branch, we have two sets of potentials:

$$\Phi'^U = \{\phi_7^U(S)\}$$
$$\Psi'^U = \{\psi_7^U(S)\}$$
$$\Phi'^B = \{\phi_7^B(S)\}$$
$$\Psi'^B = \{\psi_7^B(S)\}\ .$$

The probability potentials of the two branches are identical. The utility potentials must be unified through maximization, thus the utility potential obtained is:

$$\Psi'' = \{\max(\psi_7^U(S), \psi_7^B(S)\}$$

and the step-policy generated is:

$$\sigma(s) = \begin{cases} BT & \text{if } U^B(s) \geq U^U(s) \\ UT & \text{otherwise} \end{cases}$$

A measure of the complexity of solving a S-DAG is the number of eliminations performed. The evaluation requires one elimination per each variable in a node of the S-DAG. Thus, methods are devised to construct small S-DAGs. We shall, however, not deal with this issue in the current dissertation.

2.5 Explanation in expert systems

Bayesian networks and influence diagrams are probabilistic graphical models widely used for building diagnosis- and decision-support expert systems. Explanation of both the model and the reasoning is important for debugging these models, for alleviating users' reluctance to accept their advice, and for using them as tutoring systems. We describe in this section the importance of explanation of expert systems and its main features.

2.5.1 Importance of explanation in expert systems

In the context of expert systems, either probabilistic or heuristic, the development of explanation facilities is important for three main reasons (Lacave and Díez, 2002; Lacave, 2003). First, because the construction of those systems with the help of human experts is a difficult and time-consuming task, prone to errors and omissions. An explanation tool can help the experts and the knowledge engineers taking part in the project to *debug* the system when it does not yield the expected results and even before a malfunction occurs. Second, because human beings are reluctant to *accept* the advice offered by a machine if they are not able to understand how the system arrived at those recommendations; this reluctancy is especially clear in medicine (Wallis and Shortliffe, 1984). And third, because an expert system used as an intelligent *tutor* must be able to communicate the apprentice the knowledge it contains, the way in which the knowledge has been applied for arriving at a conclusion, and what would have happened if the user had introduced different pieces of evidence (what-if reasoning).

These reasons are especially relevant in the case of probabilistic expert systems, because the elicitation of probabilities is more difficult than the assessment of uncertainty in heuristic expert systems and because, even though probabilistic reasoning is just a formalization of (a part of) common-sense reasoning, the algorithms for the computation of probabilities and utilities are very different from the way a human being would draw conclusions from a probabilistic model.

Unfortunately, the explanation methods proposed so far are still unsatisfactory, as shown by the fact that most expert systems and commercial tools available today, either heuristic or probabilistic, have virtually no explanation capability (Lacave and Díez, 2002, 2004). Despite the practical interest of this issue, very little research is currently carried out about explanation in probabilistic graphical models. As an attempt to palliate this shortcoming, in this paper we describe some methods for explaining both the model and the reasoning of probabilistic expert systems as influence diagrams, which have been implemented in Elvira, a public software tool developed as a joint project of several Spanish universities. We also discuss how such methods respond to the needs that we have detected when building and debugging medical expert systems (Díez et al., 1997; Lacave and Díez, 2003; Luque et al., 2005) and when teaching probabilistic graphical models to pre- and postgraduate students of computer science and medicine Díez (2004).

2.5.2 Features of explanation in expert systems

Explanation methods are characterized by several properties, corresponding to the main concepts on which an explanation is based (Lacave and Díez, 2002, 2004): content, communication and adaptation. The content of an explanation deals with the model, the reasoning, or the available evidence. Explanation of the model, also known as static explanation (Henrion and Druzdzel, 1990), consists in showing the information represented by the knowledge base of the expert system in a way that can be easily understood by the user. Explanation of the reasoning, or dynamic explanation, describes how and why the system has obtained certain results. Explanation of evidence usually consists in finding the most probable configuration that justify the evidence (Pearl, 1988), which is also known as abduction. Dynamic explanations can be generated at the micro or the macro level (Sember and Zukerman, 1989): micro-level explanations try to justify why the probability of a certain variable has varied, why the belief on a certain hypothesis has changed, or why a rule has fired as a consequence of the variations in its neighbor variables or rules; on the contrary, macro-level explanations analyze the main lines of reasoning (the paths in the Bayesian network, the chains of rules, etc.) that led from the evidence to a certain conclusion. The second main aspect of explanation, namely communication, is related to the way of interacting with the user and the way of presenting the explanations, either textually or graphically or by a combination of both. Finally, adaptation refers to the ability to modify the explanations and the interaction depending on the user's expertise and needs. See (Lacave and Díez, 2002, 2004) for a more detailed analysis of these features and for a detailed review of the most relevant methods and systems offering some kind for explanation, both for Bayesian networks (Lacave and Díez, 2002) and for heuristic expert systems (Lacave and Díez, 2004).

2.6 Cost-effectiveness analysis in medicine

Cost-effectiveness analysis (CEA) in medicine is a particular case of multicriteria decision making with two objective functions: the health benefit, measured in clinical units (effectiveness), which we want to maximize, and the economic cost, measured in monetary units, which we want to minimize. In this section we describe the methods of CEA.

2.6.1 Net benefit and incremental cost-effectiveness ratio

In multicriteria decision making, the most usual way of combining the objectives is a weighted linear sum (Steuer, 1986). In the case of CEA, the global utility is called net benefit, and defined by:

$$NB = \lambda E - C, \tag{2.20}$$

where E is the effectiveness, C is the cost, and λ, sometimes called *willingness to pay*, is used here to convert the effectiveness into a monetary scale. Its value depends on each decision maker. NB is the *net benefit*, and can be seen as the effectiveness (converted into monetary value) minus the associated economic costs.

In health-related CEA, the effectiveness is measured in clinical units, such as the number of deaths avoided. CEA assumes parameter λ is positive but unknown.

Cost-utility analysis is a particular case of CEA in which effectiveness is identified with the *quality-adjusted life expectancy* (QALE) (Drummond et al., 2005) whose unit is the *quality-adjusted life year* (QALY)[7]. When the quality of life varies along the time, the QALE corresponding to the interval of time $[t_1, t_2]$ is given by:

$$QALE = \int_{t_1}^{t_2} Q(t)dt, \tag{2.21}$$

where $Q(t)$ represents the quality of life. When the quality of life is constant in the interval, being Q its value, the QALE can be calculated as:

$$QALE = Q \cdot [t_2 - t_1]. \tag{2.22}$$

The quality of life, Q, is measured in a scale where value 1 represents the best possible health state, 0 represents death, and negative values indicate health states that people deem than being dead. Thus, 1 QALY represents one year of life at the best possible health state, or 2 years with a quality of life of 0.5, etc.

[7]Some authors distinguish between cost-effectiveness analysis and cost-utility analysis, while others use the term cost-effectiveness for both. In general, the British and Canadians tend to make such a distinction; for example, the famous book by Drummond et al. (2005), written by researchers from the Universities of York (United Kingdom) and McMaster (Ontario, Canada) devotes a chapter to each type of analysis, explaining the differences between them. In contrast, in the United States it is frequent to refer to both types of studies as cost-effectiveness analysis. Thus, the book of Gold (1996), although it is titled *Cost-Effectiveness in Health and Medicine*, focuses on cost-utility analysis. In Spain we have a mixed situation: there are those who distinguish between cost-effectiveness and cost-utility, while others use the term cost-effectiveness for both types of studies.

2.6.2 Deterministic CEA

Deterministic CEA assumes that we have a set of interventions, each one with known effectiveness and cost. An intervention can be very simple. For example, the interventions I_1 can be not to perform a test, while I_2 can be not to do it; I_3 can be not to apply any treatment, I_4 can be to apply chemotherapy, and I_5 can be to apply supportive care. We can also have complex interventions. For example, an intervention can be "do the test 1; if the result is positive then apply treatment T_1; otherwise do the test 2, and then...".

The interventions can be represented in a bidimensional plot, where the effectiveness is in the X-axis, and the economic cost is in the Y-axis (see Figure 2.16).

Having λ unknown, the purpose of CEA is investigating which is the best intervention by taking into account all the possible positive values for λ. (We assume that λ is positive because everyone is willing to pay a certain amount of money, perhaps very small, for obtaining some benefit).

Comparison of interventions

We are going to see how to compare interventions, which is the basis of CEA.

Comparison of two interventions When comparing two interventions there are three possible cases: coincidence, dominance and non-dominance. The first case consists in having two interventions with identical effectiveness and cost. According with the utility theory by von Neumann and Morgenstern (1944) in this case the decision maker will have no preference independently of the value of λ. Cases of dominance and non-dominance are explained below.

Dominance We introduce below the definition of dominance and a corollary for analyzing this case.

Definition 2.6.1 *When A and B are two possible interventions, it is said that B* dominates *A if:*

- $E_B \geq E_A$ *and* $C_B \leq C_A$, *and*
- $E_B > E_A$ *or* $C_B < C_A$.

Corollary 2.6.1 *If B dominates A, then $NB_B > NB_A$ for any positive value of λ.*

Therefore, when B dominates A the decision maker will prefer B independently of the value of λ.

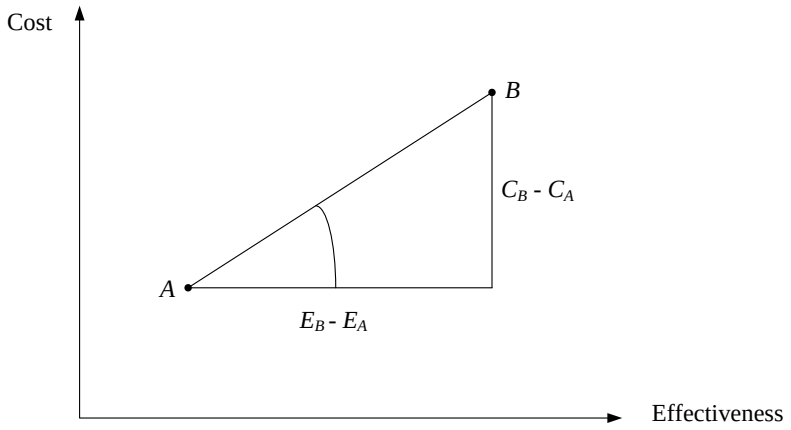

Figure 2.16: Comparison of interventions A and B. The scope of line through A and B is the incremental cost-effectiveness ratio (ICER) of A and B.

Non-dominance When no intervention dominates the other, it implies one, say B, is more effective but more costly: $C_B > C_A$ and $E_B > E_A$.

Definition 2.6.2 *Let A and B be two interventions, such that $C_B > C_A$ and $E_B > E_A$. The* incremental cost-effectiveness ratio *(ICER) of A and B is defined by:*

$$ICER(A, B) = \frac{(C_B - C_A)}{(E_B - E_A)}.$$

The ICER can graphically be interpreted as the slope of the line through the points A and B, as can be seen in Figure 2.16.

From Equation 2.23 we can deduce:

$$NB_B > NB_A \iff \lambda E_B - C_B > \lambda E_A - C_A \iff \lambda(E_B - E_A) > (C_B - C_A) \tag{2.23}$$

$$\iff \lambda > \frac{(C_B - C_A)}{(E_B - E_A)}. \tag{2.24}$$

Therefore:

$$ICER(A, B) > \lambda \Rightarrow NB_A > NB_B \tag{2.25}$$

$$ICER(A, B) = \lambda \Rightarrow NB_A = NB_B \tag{2.26}$$

$$ICER(A, B) < \lambda \Rightarrow NB_A < NB_B \tag{2.27}$$

Put another way, the maximization of the NB, which is the objective of CEA, can be performed by comparing the ICERs. Even though this is the most common approach, in

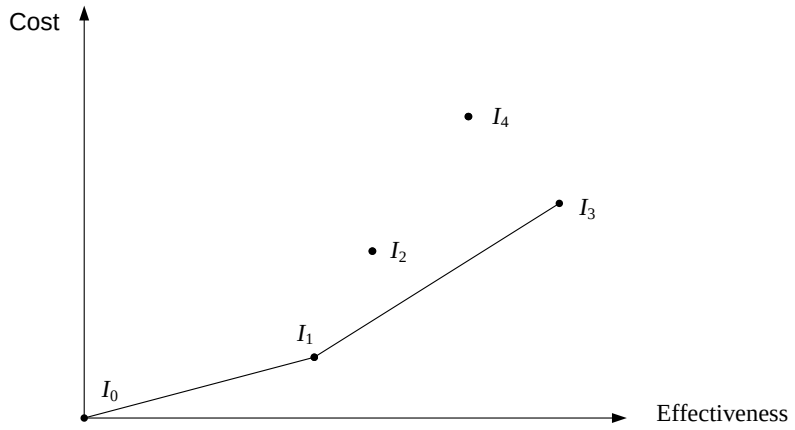

Figure 2.17: Example of CEA with more than two interventions. The set $\mathbf{S} = \{I_0, I_1, I_3\}$ is the efficient set of interventions. For any positive value of λ the most beneficial intervention will always belong to $\mathbf{S}$.

the medical application described in Chapter 6, we have performed a CEA based directly on the NB (see Section 6.4.3).

Comparison of several interventions When having more than two interventions the situation is a bit more complicated. Let $\mathbf{I}$ be the set of possible interventions. The subset of interventions $\mathbf{S} \subseteq \mathbf{I}$ is said to be *efficient* if it is a minimal set satisfying that for any positive value of λ the most beneficial intervention of $\mathbf{I}$ belongs to $\mathbf{S}$. Given that an efficient set is minimal, an intervention dominated by another one can not belong to an efficient set (Steuer, 1986).

For example, let us see Figure 2.17, where we have the set of interventions $\mathbf{I} = \{I_0, I_1, I_2, I_3, I_4\}$. The subset $\{I_0, I_1, I_3\}$ constitutes the only efficient set. We can discard interventions I_2 and I_4 because for any positive value of λ the most beneficial action will be I_0, I_1 or I_3.

A possible algorithm for finding the efficient set of interventions is described in (Arias, 2009). It orders by cost the interventions of the efficient set, and by using the ICER for consecutive interventions it establishes a partitioning of the interval $(0, +\infty)$ for the parameter λ. Each subinterval is mapped into the intervention of the efficient set that maximizes the net benefit in the subinterval. For example, for the Figure 2.17 we would have the subintervals $(0, l_{0,1})$, $(l_{0,1}, l_{1,3})$, and $(l_{1,3}, +\infty)$, where $l_{i,j} = ICER(A_i, A_j)$. Each of these subintervals for λ is assigned to A_0, A_1, and A_3, respectively.

2.6.3 Probabilistic CEA

Let us see a more general situation, where we have a set of interventions and we can also have uncertainty due to the presence of chance variables.

CEA with decision trees(Arias, 2009)

When the cost and effectiveness is known for every intervention, we can apply the standard method of CEA. It assumes that we have a decision tree representation of the problem (Raiffa and Schlaifer, 1961), with an only decision, denoted by D, which is placed in the root of the tree. Leaves of the tree represent the scenarios of the problem. Each scenario is labeled with an economic cost and an effectiveness.

Having the parameter λ as unknown to the decision maker, the objective of performing a CEA of the DT with an only decision, placed in the root, is to obtain a partitioning of the interval $(0, +\infty)$ for λ, where each interval of the partition is mapped into an option of D that maximizes the net benefit in that interval of λ.

The CEA of the DT is performed from the leaves towards the root as follows:

- When a chance node is reached the cost and the effectiveness are separately calculated by expectation by using the corresponding cost and effectiveness of each child.
- When the root of the tree is reached we perform CEA as described in in each interval as described in Section 2.6.1, which can split some intervals.

A clear limitation of the standard method is that there can only be one decision in the DT, placed in the root. This is very restrictive constraint for many real problems. There are several possibilities for solving this limitation: (1) represent in the root of the DT a decision with one intervention per each possible strategy, (2) evaluate the decision tree assuming λ as known, or (3) repeat the evaluation of the DT for all the possible values of λ. However, the first two possible solutions are unsatisfactory (Arias, 2009), while the third is impossible because the there are infinite values of λ.

Given the limitations of standard method of CEA, Arias (2009) propose a method for performing CEA of a decision problem represented in a DT that can contain several decisions located in any place of the tree. Having the parameter λ as unknown to the decision maker, the objective of performing a CEA of a DT is now to assign to each decision node D in the DT a partitioning of the interval $(0, +\infty)$ for λ, where each interval of the partition is mapped into an option of D that maximizes the net benefit in that interval of λ.

The CEA of the DT is performed from the leaves towards the root. Initially, each leaf is labeled with an economic cost, an effectiveness, and its initial set of intervals for λ is $\{(0, +\infty)\}$.

When a chance node C is reached the interval $(0, +\infty)$ is partitioned by joining all the points that determine the intervals of λ in the children. For each interval obtained in C the costs and the effectiveness are separately calculated by expectation by using the corresponding cost and effectiveness of each child.

When a decision node D is reached the interval $(0, +\infty)$ is partitioned as in the case of chance nodes. A CEA is performed in each interval as described in Section 2.6.1, which can split some intervals. Two consecutive intervals in D could eventually be merged if they have identical cost, effectiveness and optimal decision option.

CEA with influence diagrams

One possibility to cope with the multiobjective problem of CEA in the framework of IDs would be to apply the extension proposed by Nielsen et al. (2007). They represented the different objectives as a set of ordinary utility nodes and developed an algorithm for analyzing the solution of this kind of IDs. The method allow to solve influence diagrams with any number of objective functions, n. Decision regions in method by Nielsen et al. (2007) are hyperplanes of the space $\mathbb{R}^n$, each one with dimension $n-1$. In the case of CEA, where $n = 2$, $\mathbb{R}^2$ is a plane, and the hyperplane is a straight line. The method is thus perfectly applicable to CEA and would give to the decision maker a solution highly satisfactory.

Despite of the possibility of applying multi-currency influence diagrams Nielsen et al. (2007)for performing CEA in influence diagrams, Arias (2009) adapted the method proposed by Nielsen et al. (2007) to the particular case of CEA, where there are only two objective functions. Arias (2009) constructs an influence diagram with two utility nodes: one for the cost, and other for the effectiveness. He proposes a variable elimination-based algorithm for performing the CEA. The output of the analysis is a *cost-effectiveness policy* (CEP) for each decision D of the diagram, which assigns to each configuration of D a partitioning of the interval $(0, +\infty)$ for λ, where each subinterval of the partition is mapped into an decision option of D that maximizes the net benefit in that interval of λ. The basic schema of the method is proposed in Algorithm 2.7.

Similarly to variable elimination algorithms, it utilizes a set of probability potentials. It also uses a *cost-effectiveness partition table* (CEPT), which assigns to each configuration of its domain a partitioning of the interval $(0, +\infty)$,where each subinterval is assigned to

two values: a cost and an effectiveness.

There are only three operations in the variable-elimination scheme of Algorithm 2.7 that are specific of CEA:

- The CEPT is initialized by combining the utility nodes of cost and effectiveness into an only table that keep its values separately, and assigns the interval $(0, +\infty)$ to each configuration.
- The summation of a CEPT over a chance variable combines, for each configuration, the partitions corresponding to the values of X, similarly to the case of DTs (see Section 2.6.3).
- The CEA when eliminating a decision D do the next: for each configuration $\mathbf{y} \in dom(CEPT) \setminus \{D\}$, it modifies the CEPT to an equivalent table where we have the same interval for $\mathbf{y}$ for any decision option of D. This allow to eliminate D from the CEPT by maximization, which originates a CEP for D.

Algorithm 2.7 CEA in IDs using the method byArias (2009).

Input: I: an ID that may contain SVNs;
O: a legal sequence elimination for the variables in $\mathbf{V}_C \cup \mathbf{V}_D$;
Output: a CEP for each decision variable D.

1. initially the CEPT of the diagram;
2. **for all** $V \in O$ **do**
3. $\Phi_X := \{\phi \in \Phi | V \in dom(\phi)\}$;
4. $\phi_X := \prod_{\phi \in \Phi_X} \phi$;
5. **if** $V \in \mathbf{V}_C$ **then**
6. $\phi'_X := \sum_X \phi_X$;
7. $\phi''_X := \frac{\phi_X}{\phi'_X}$;
8. multiply the CEPT by ϕ''_X and calculate the summation over X;
9. **else**
10. $//V \in \mathbf{V_D}$
11. $\phi''_X :=$project ϕ_X over X;
12. peform CEA over the CEPT, which originates a CEP for V;
13. **end if**
14. $\Phi := (\Phi \setminus \Phi_X) \cup \{\phi'_X\}$;
15. **end for**

Examples and details of the algorithm can be found in (Arias, 2009). The main advantage of their method compared to the proposed by Nielsen et al. (2007) is that the use of a parameter during the method instead of two weights used by Nielsen et al. (2007) when combining the two objectives functions simplify the evaluation and provide with

a representation, the CEP, which is more understandable for a human expert. Authors have also implemented the method as a plugin available in the free software tool Carmen (Arias and Díez, 2008), which is very

2.7 Sensitivity analysis in probabilistic decision problems

The object of decision analysis on a probabilistic decision problem, represented for example in a decision tree, an influence diagram or an unconstrained influence diagram, is twofold: to determine an optimal strategy, consisting of an optimal policy for each decision, and on the other hand, to compute the maximum expected utility (MEU). In general it is usual to computes first the optimal policies and the MEU for a particular model, called the *reference* case, in which all the parameters are assumed to be known with certainty, and in a posterior phase, the decision analyst investigates whether these results depend on (are sensitive) to the uncertainty about the model. This post-hoc investigation is called *sensitivity analysis.* The optimal policies and the MEU are sensitive to variations in both the qualitative part of the ID (arcs and nodes) and the quantitative part (the utilities and the probabilities).

2.7.1 Basic concepts

Sensitivity analysis (SA) consists in determining whether the conclusions obtained for the reference case (optimal strategy and MEU) hold in spite of the uncertainty about the accuracy of the model itself.

There exist several types of SA. Depending on the part of the model studied, SA can be:

- **qualitative**, also referred to as **structural**, which examines how variations of the structure of the model can affect the conclusions;
- **quantitative**, which explores the effect of the variations in the probabilities and utilities.

Depending on the types of conclusions studied, we can distinguish two types of SA:

- **value sensitivity analysis**, which measures variations in the expected utility;
- **decision sensitivity analysis**, which explores the changes in the optimal strategy.

Parameters	Distribution	Details
Probabilities	Beta	Between 0 and 1
Costs	Log-normal Gamma	Ranging from 0 to ∞
Utilities	Beta Gamma $(1-U)$	$-\infty$ to 1
Relative risks	Log-normal	Ratios Additive in log scale

Table 2.1: Commonly used distributions in SA in medical decision making. [Taken from `http://www.york.ac.uk/inst/che/pdf/teehtacosteff04.pdf`]

Quantitative SA can furthermore be characterized as:

- **interval-based SA**: each parameter to be within a certain interval; for example, considering that the prevalence of sensitivity of *N2_N3* is between 0.26 and 0.29, but we do not know its value with certainty.
- **probabilistic SA**: it assigns a probability distribution to each parameter; for example, by assuming that our estimation of the prevalence of *N2_N3* is given by a Gaussian distribution with a mean of 0.14 and a standard deviation of 0.03. For example, a particular type of probabilistic SA consists of computing the probability that the optimal strategy be different from that obtained for the reference case (Doubilet et al., 1985).
- **policy change thresholds SA**: it investigates the admissible values a parameter (or a set of parameters) can be assigned without changing of the optimal strategy of the reference case. For example, let us assume that the prevalence of *N2_N3* is 0.28 in the reference case. A threshold of 0.26 and 0.29 would mean that if the prevalence were less than 0.26 or greater than 0.29, then the optimal strategy would be different.

The distributions commonly used in probabilistic SA when applied to medical decision making are shown in Table 2.1.

Quantitative SA can be classified into three types (Díez, 2007):

- **one-way**: it is concentrated on just one parameter; for example, the prevalence;
- ***n*-way independent analysis**, which consists in considering the consequences of individual variations of each of n parameters;
- ***n*-way joint analysis**: it analyzes the joint variation of a set of n parameters.

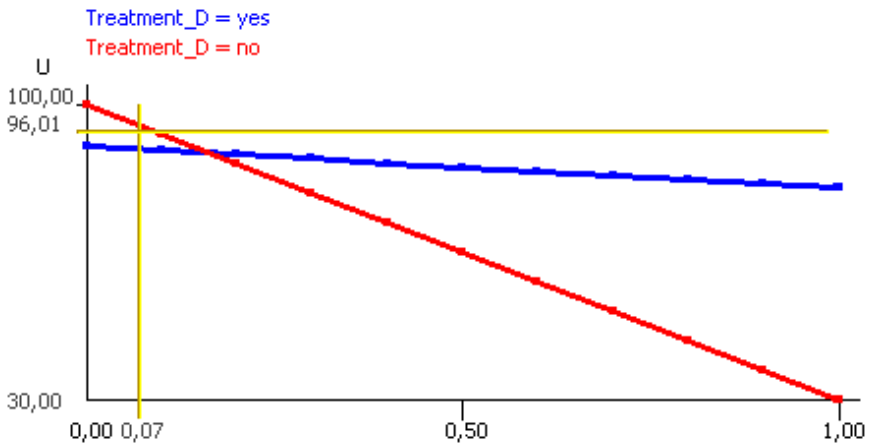

Figure 2.18: Utility plot on the prevalence of the disease, which is represented in the the x-axis. The y-axis represents the expected utility. The treatment threshold is 0.17.

There are three graphical representations very popular in SA of Bayesian decision problems (Díez, 2007; Clemen and Reilly, 2001):

- **Utility plots**: They can be used for finding treatment thresholds in different scenarios (van der Gaag and Coupe, 2000) and, in consequence, to explain the optimal policies. For instance, in Fig. 2.18, which shows the results of one-way sensitivity analysis on the prevalence of X for the ID given in Fig. 4.1. This graph is obtained by evaluating several instances of the ID, each having a different value of $P(+x)$. We can see that the treatment threshold is approximately 0.17, i.e., when $P(+x) < 0.17$ the best option is not to treat the patient, and when $P(+x) > 0.17$ it is better to treat. This way, utility plots show graphically the policy changes thresholds and why they emerge.

- **Tornado diagrams**: They are a form of interval-based n-way independent analysis. They show graphically which parameters in the model have the greatest influence on the expected utility. Each parameter is assigned to a bar whose length indicates the variation of the expected utility. The graph is laid out so that the most sensitive parameter (the one with the longest bar) is at the top, and the least sensitive is at the bottom, with the bars arranged in this order. An example of tornado diagram is presented in Figure 2.19. The vertical bar represents the MEU for the reference case.

- **Spider diagrams**: The analysis is identical to the tornado diagram. The only difference is in how the results are presented. In a spider diagram, the utility is not

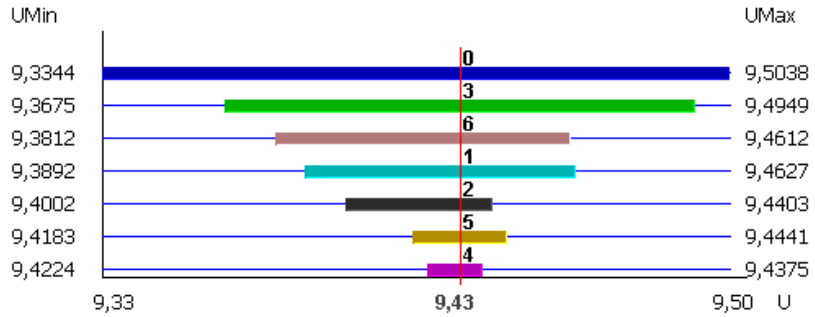

Figure 2.19: A tornado diagram. [Image taken from (Díez, 2007)].

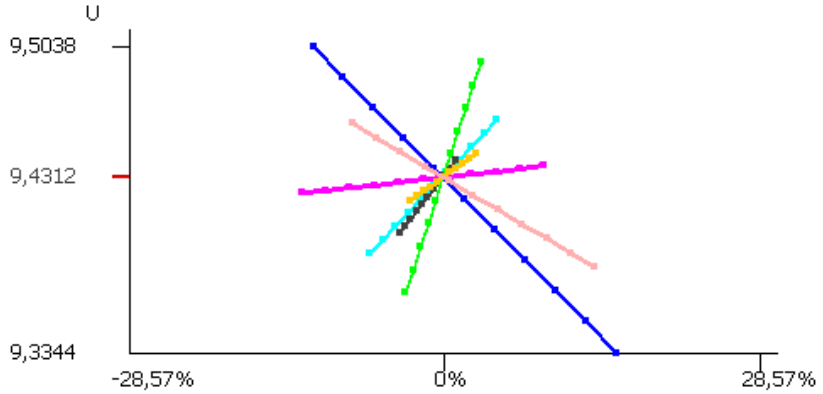

Figure 2.20: A spider diagram. [Image taken from (Díez, 2007)].

represented on the horizontal axis but on the vertical one; the percentage variation of each parameter over its reference value is represented on the horizontal axis. An example of spider diagram is presented in Figure 2.20.

2.7.2 Sensitivity analysis in influence diagrams

One of the initial steps towards a coherent approach for SA in IDs was proposed by Bielza et al. (1996). They consider the situation where some of the utilities and probabilities have only partially assessed. Uncertain values are represented by parameters, and a set of non-dominated strategies is computed based on that parametric model.

A method based on value sensitivity was proposed in (Felli and Hazen, 1998). This method uses the expected value of perfect information, and it requires that a probability distribution is assigned to each parameter under investigation. Formally, let t be an uncertain parameter, and let Δ_0 be the optimal strategy found with the initial values of the parameters $\mathbf{t}$, denoted by $\mathbf{t}_0$. Let $EU(\Delta, \mathbf{t})$ denote the expected utility of the ID under strategy Δ and values of parameters $\mathbf{t}$. Then, the *expected value of perfect information*

(EVPI) is given by:

$$EVPI = E_{\mathbf{t}}[(max_{\Delta} EU(\Delta, \mathbf{t})) - EU(\Delta_0, \mathbf{t})], \tag{2.28}$$

where $E_{\mathbf{t}}$ denotes the expected value with respect to probability distribution of parameters $\mathbf{t}$. Monte Carlo methods are usually applied to sample values for the parameters in order to calculate *EVPI*. The expectation $E_{\mathbf{t}}$ in Equation is then approximated by calculating the mean over the set of generated samples.

Nielsen and Jensen (2003) proposed a method that performs decision SA in IDs based on threshold-proximity. They proposed very efficient algorithms for performing one-way and n-way SA. Their method is based on an explicit representation of the parameters in question, and the calculations are performed in the underlying junction tree representation of the ID.

2.8 Elvira

Elvira[8] is a tool for building and evaluating graphical probabilistic models (Elvira Consortium, 2002). It resulted from a joint research project of several Spanish universities. It is implemented in Java, so that it can run on different platforms. It contains a graphical interface for editing networks, with specific options for canonical models (e.g., OR, AND, MAX...), exact and approximate algorithms for discrete and continuous variables, explanation facilities, learning methods for building networks from databases, algorithms for fusing networks, etc. Although some of the algorithms work with both discrete and continuous variables, the explanation capabilities (see Chapter 4) assume that all the variables are discrete.

Architecture of Elvira

Elvira is structured in four main modules:

- Data representation, which contains the definition of the data structures needed for managing BNs and IDs in Java.
- Data acquisition, including the necessary classes for saving and loading a network both from a file and from a data base, the parser, etc. It also contains classes for exporting and importing the networks in several formats.

[8] At `http://www.ia.uned.es/~elvira` it is possible to obtain the source code and several technical documents about Elvira.

- Processing. This module implements the algorithms for processing and evaluating the models. It is organized in several submodules, one for each task: inference, learning, fusion, decision trees, sensitivity analysis...

- Visualization, which mainly defines the Elvira GUI and which, obviously, makes use of the classes included in the previous modules. This module contains the classes for generating explanations and for the *internationalization* of the whole program, i.e., for showing the dialogs in a certain language. At this moment, only Spanish and English are supported, but thanks to this module, it can be easily extended to other languages.

The main advantages of this modular design is that the groups working at different universities can focus on different tasks and that the program can be easily extended with new functionality or adapted to different processing needs.

Working with the Elvira GUI

In addition to invoking Elvira's classes from the command line and using it as an API, it is possible to interact with Elvira by means of its GUI, which has two working modes:

- edit, for graphically editing BNs and IDs. This is possible by means of several windows which help the user to build or to modify the model manually, by requesting all the data associated to the nodes, the arcs and the properties of the whole BN or ID. Alternatively, BNs can be built from data bases by applying some of the many learning algorithms implemented in Elvira; and

- inference, for propagating evidence and explaining the results. The introduction of evidence can be done by clicking on the node, as in other software tools, or by means of an editor of cases (Lacave et al., 2000), which provides a list of the variables in the model. With respect to the inference process, the user can choose one of several algorithms, with many variations, and in the case of a BN, she can select either evidence propagation or abduction[9], and whether the evidence is propagated automatically (i.e., just after the user introduces or removes a finding) or manually (under demand). Most of the explanation capabilities provided by Elvira are offered in the inference mode.

[9]In the context of BNs, *evidence propagation* usually refers to computing the posterior probability of each single variable given the available evidence, while *abduction* consists in computing the joint probability of a set of variables of interest given the evidence, what is also called *most probable explanation (MPE)* (Pearl, 1988).

How to obtain the software of Elvira

Elvira has been implemented in Java 6.0. Java is a programming language originally developed by Sun Microsystems. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. There is a version of a JVM for each specific architecture available on the website http://java.sun.com/. A user interested in running Elvira only needs to download a Java Runtime Environment (JRE). However, Sun also distributes a superset of the JRE called the Java SDK (more commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar and debugger. A user interested in programming by using the API of Elvira would need a JDK.

Having obtained the corresponding JVM, the user has mainly two options for obtaining Elvira software:

1. The source code, a user manual, and other documents can be downloaded from http://www.ia.uned.es/ elvira. This web page has a hyperlink to the source code, which is compressed periodically in a file in format tar.gz and made available through the web page. This method does not guarantee to the user to have the last version of the code of Elvira. It is a very simple method for obtaining the source code of Elvira.

2. The last version of Elvira can be obtained directly from the CVS repository of Elvira. There are instructions for downloading the source code logging in the CVS server through the link http://leo.ugr.es/elvira/. We recommend this method for advanced users.

Appendix B presents the software developed in Elvira by the author of this thesis.

Part III

METHODOLOGICAL ADVANCES

Chapter 3

Variable elimination for influence diagrams with super value nodes

In the original formulation of influence diagrams (IDs), each model contained exactly one utility node. Tatman and Shachter (1990), introduced the possibility of having super value nodes that represent a combination of their parents' utility functions. They also proposed an arc reversal algorithm for IDs with super value nodes. In this paper we propose a variable-elimination algorithm for influence diagrams with super value nodes which is faster in most cases, requires less memory in general, introduces much fewer redundant (i.e., unnecessary) variables in the resulting policies, may simplify sensitivity analysis, and can speed up inference in IDs containing canonical models, such as the noisy OR.

The content of this chapter is based on (Luque and Díez, 2008).

3.1 Introduction

3.1.1 Influence diagrams

An influence diagram (Howard and Matheson, 1984) is a probabilistic graphical model for decision analysis, having three kinds of nodes: chance, decision, and utility—see Sections 2.3 and 3.1.2 for a formal definition. The goal of evaluating an ID is to obtain the expected utility and an optimal strategy, which consists of a policy for each decision. The first algorithm for evaluating IDs proceeded by expanding and evaluating an equivalent decision tree (Howard and Matheson, 1984). Later, Olmsted (1983) proposed the arc reversal (AR) algorithm, which evaluates the ID recursively by eliminating its nodes and inverting arcs when necessary—see also (Shachter, 1986).

In the original proposal (Howard and Matheson, 1984), each influence diagram (ID)

had only one utility node. A node like this, whose parents are chance nodes or decision nodes, is nowadays called an *ordinary utility node*, in contrast with *super value nodes* (SVNs), whose parents are other utility nodes; an SVN represents a utility that is a combination of the utilities of its parents. SVNs were introduced in 1990 by Tatman and Shachter (1990), who also extended the AR algorithm to deal with SVNs of type sum and product. A description of the algorithm proposed by Tatman and Shachter (1990) is described in Section 2.3.2.

In the next decade, several variable-elimination algorithms were proposed for IDs, some of them combined with clustering algorithms (Cowell et al., 1999; Dechter, 1996; Jensen et al., 1994; Jensen and Nielsen, 2007; Ndilikilikesha, 1994; Shenoy, 1992). They allow the ID to contain several ordinary utility nodes, under the assumption that the global utility is the sum of all of them, but none of those algorithms can deal with SVNs.

Our interest in SVNs arose during the construction of a decision-support system for the mediastinal staging of non-small cell lung cancer (Luque et al., 2005), whose utilities combine additively and multiplicatively, as shown in Figure 3.1. In order to evaluate this ID, we wanted to have an algorithm for IDs with SVNs, such that it:[1]

1. were faster than AR;
2. required less memory;
3. avoided redundant variables;
4. simplified sensitivity analysis;
5. could be integrated with state-of-the-art algorithms for inference in IDs containing canonical models.

The first two objectives are obvious. We conjectured that a variable elimination algorithm for IDs with SVNs might fulfill them because, unlike AR, it does not need to divide potentials and the number of potentials stored in the working memory is smaller.

The third objective refers to redundant variables, i.e., those whose value is known when making a decision but that do not affect the optimal policy—see Section 3.1.3. As the complexity of a policy grows exponentially with the number of variables in its domain, it is desirable to remove as many redundant variables as possible, not only to reduce the storage space but, more importantly, to communicate the policy to a human being. In fact, the explanation of reasoning is a crucial issue for building and deploying decision-support

[1] In fact, the fourth objective was not set at the beginning of our study, but emerged as a possibility during the design of the algorithm.

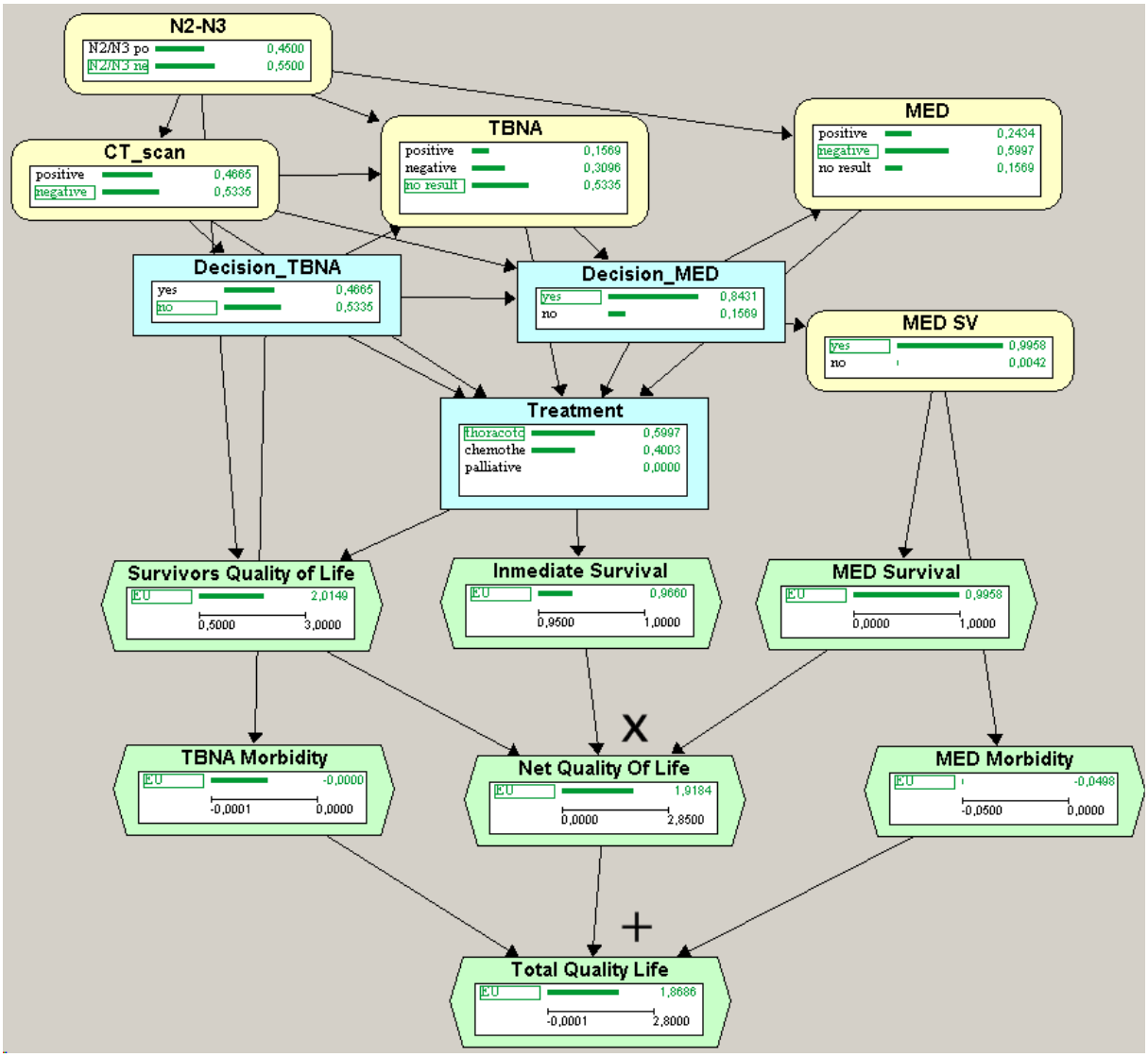

Figure 3.1: Decision-support system for the mediastinal staging of non-small lung cancer.

systems because it helps to debug the model and to convince the user that the results are correct, building intelligent tutoring systems (Lacave and Díez, 2002; Lacave et al., 2007, 2006). Policies containing redundant variables are more difficult to understand and debug. Even worse, the inclusion of structurally-redundant variables (cf. Sec. 3.1.3), i.e., those that cannot affect the policy due to the nature of the causal relations involved in the problem, undermines the user's confidence in the policies recommended by the expert system. Several algorithms have been proposed in the literature (Faguiouli and Zaffalon, 1998; Shachter, 1998; Nielsen and Jensen, 1999; Nilsson and Lauritzen, 2000; Vomlelova and Jensen, 2004) for detecting structurally redundant variables efficiently by analyzing the graph in IDs, but none of them can analyze IDs with super value nodes.

The fourth objective refers to sensitivity analysis, which consists of studying how the

expected utility and the optimal strategy vary as a consequence of changes in the model (Clemen and Reilly, 2001; Nielsen and Jensen, 2003). Parametric sensitivity analysis in IDs is performed by assigning a range of variation or a probability distribution to (some of) the utilities and conditional probabilities that define the ID, or by detecting the thresholds that determine a change in the optimal policies. However, in some cases the structure of the graph that defines an ID implies that the values of a certain potential (namely, a conditional probability table or a utility function) do not affect the expected utility or the optimal policies of some decisions. In that case, it is not necessary to analyze such parameters, thus saving computations and simplifying the reporting of the results. Section 3.2.2 shows an example of this.

Finally, the fifth objective is concerned with canonical models, which are probabilistic relations defined by some constraints usually stemming from casual assumptions (Díez and Druzdzel, 2006). They are called "canonical" because they can be used as elementary blocks that combine to build up more sophisticated probabilistic models (Pearl, 1988). In particular, the relation between a node and its parents in a Bayesian network (or a chance node and its parents in an ID), which in the case of discrete variables takes the form of a conditional probability table (CPT), can sometimes be represented by a particular canonical model, while other CPTs in the same network might be based on different models. The canonical models that appear more often in practice are the noisy OR and its extension, the noisy MAX. Canonical models do not only simplify the process of building CPTs, but may also lead to drastic computational savings in both memory and time. For instance, CPCS (Pradhan et al., 1994) is a large medical Bayesian network that for many years was resistant to exact inference algorithms, because all of them ran out of memory when trying to compute some marginal queries—see the references in (Takikawa and D'Ambrosio, 1999). Even most of the approximate algorithms converged very slowly when the evidence introduced was very unlikely (Cheng and Druzdzel, 2000). However, Takikawa and D'Ambrosio (1999) proposed a new factorization of the noisy MAX that was able to do exact inference on that network in less than a second, and the factorization by Díez and Galán (2003) further reduced that time to 0.05 seconds. Those factorizations can be integrated with both variable elimination and clustering algorithms, but not with arc reversal. That was an additional reason for developing a variable elimination algorithm for IDs with SVNs, in order to obtain savings similar to those of Bayesian networks.

Our algorithm is an extension of variable-elimination algorithms for IDs (Cowell et al., 1999; Dechter, 1996; Jensen et al., 1994; Jensen and Nielsen, 2007; Shenoy, 1992); in fact, when the ID has no SVN, it performs essentially the same operations as them. The main

difference is that it represents the utility function of the ID in the form of a tree, and in a refined version of the algorithm, in the form of an acyclic directed graph (ADG). The algorithm usually transforms that tree or ADG before eliminating each variable, trying to preserve its separability as long as possible, in order to reduce the space complexity and to avoid redundant variables in the policies.

The remainder of this chapter is structured as follows. Section 3.1.2 presents the basic definitions for IDs and Section 3.1.3 analyzes the problem of redundant variables. Section 3.2 presents a new algorithm for eliminating chance variables (Sec. 3.2.1) and decision variables (Sec. 3.2.2) from a tree of potentials (ToP) (and also exposes a variable-elimination algorithm for IDs with SVNs on a ToP). Section 3.3 improves the previous algorithm by using an acyclic directed graph of potentials (ADGoP) instead of a ToP. Section 3.4 proposes three variations of that algorithm that in some cases may lead to more efficient computations. Section 3.5 describes the empirical evaluation of different versions of our algorithm, comparing them with arc-reversal. We discuss related work, conclusions and future research lines in Section 3.6.

3.1.2 Basic definitions

Influence diagrams with super value nodes framework was exposed and detailed in Section 2.3. We encourage the reader to have a quick look on the definitions given in that section, because most of them will be used in this chapter.

A concept not defined there and that is crucial in this chapter is the term *matrix*. The *matrix* of an ID ψ, is defined by

$$\psi(\mathbf{V}_C, \mathbf{V}_D) = P(\mathbf{v}_C, \mathbf{v}_D)\, \psi_{U_0}(\mathbf{v}_C, \mathbf{v}_D)\ . \tag{3.1}$$

Then, by using the definition of matrix, the expression of the *maximum expected utility* (*MEU*) of an ID, already defined in Equation 2.12, is

$$MEU = \sum_{\mathbf{c}_0} \max_{d_1} \sum_{\mathbf{c}_1} \ldots \sum_{\mathbf{c}_{n-1}} \max_{d_n} \sum_{\mathbf{c}_n} \psi(\mathbf{v}_C, \mathbf{v}_D)\ , \tag{3.2}$$

and an *optimal policy* δ_{D_i}, defined in Equation 2.13, is:

$$\delta_{D_i}(iPred(D_i)) = \arg\max_{d_i \in D_i} \sum_{\mathbf{c}_i} \max_{d_{i+1}} \ldots \sum_{\mathbf{c}_{n-1}} \max_{d_n} \sum_{\mathbf{c}_n} \psi(\mathbf{v}_C, \mathbf{v}_D)\ . \tag{3.3}$$

3.1.3 Redundant variables

According to Equation 2.13, in principle, the domain of a policy consists of all the variables whose value is known when making that decision: $dom(\delta_{D_i}) = iPred(D_i)$. However, in some cases the policy δ_{D_i} does not depend on a particular variable X of $iPred(D_i)$; we then say that X is redundant. The formal definition is as follows.

Definition 3.1.1 *Let D be a decision variable in an ID and X an informational predecessor of D: $X \in iPred(D)$. Variable X is said to be* redundant *for D if and only if*

$$\forall x, \forall x', \forall \mathbf{y}, \;\; \delta_D(x, \mathbf{y}) = \delta_D(x', \mathbf{y})$$

where x and x' are values of X and $\mathbf{y}$ is a configuration of the other informational predecessors of D: $\mathbf{Y} = iPred(D) \setminus \{X\}$.

Shachter (1998) distinguished two types of redundant variables, under the names of "irrelevant" and "probabilistically irrelevant". Following partially the terminology of (Faguiouli and Zaffalon, 1998), we prefer to use the terms "structurally redundant" and "numerically redundant", which are defined as follows.

Definition 3.1.2 *A redundant variable for decision D in an ID I is* structurally redundant *if and only if it is redundant for all the IDs having the same graph as I. Otherwise, it is* numerically redundant.

Therefore, the structural redundancy only depends on the graph of the ID, while numerical redundancy depends on the assignment of probability and utility potentials. Several algorithms have been proposed in the literature for detecting structurally redundant variables by analyzing the graph (Faguiouli and Zaffalon, 1998; Nielsen and Jensen, 1999; Nilsson and Lauritzen, 2000; Shachter, 1998; Vomlelova and Jensen, 2002), but none of them can cope with SVNs. In a future paper we will propose a new algorithm that solves this problem, but in our opinion this is a not crucial issue, as the variable elimination algorithm that we describe in this dissertation rarely includes structurally redundant variables—see the experiments in Section 3.5. However, we could always apply the redundancy-detection algorithm if redundant variables were a relevant problem in our domain of application.

An additional advantage of our algorithm, related to this topic, is that it usually avoids the introduction of quasi-structurally redundant variables, which we define as follows:

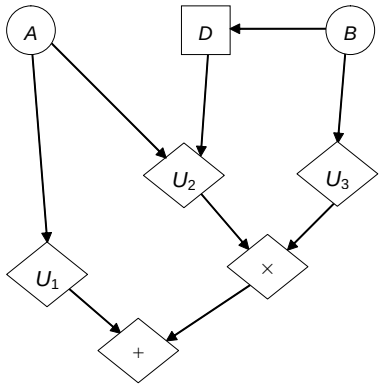

Figure 3.2: Graph of a small ID containing two super-value nodes: one of them is of type product, and the other of type sum.

Definition 3.1.3 *An ordinary utility node in an ID is* monotonic *if its utility function contains only non-positive or non-negative values.*

Definition 3.1.4 *A variable X in an ID is* quasi-structurally redundant *for a decision D with respect to a subset of utility nodes if the monotonicity of all those nodes implies that X is redundant for D.*

Please note that quasi-structural redundancy is related with numerical redundancy, because it depends on the values of some parameters in the ID, but on the other hand, such a variable will be redundant in all the IDs having the same graph and satisfying that condition, which implies that quasi-structural redundancy is a property of the graph, not of a particular ID—hence the name "quasi-structural".

For instance, for the graph given in Figure 3.2 the optimal policy for decision D, given by Equation 2.15, depends on B. That equation can be rewritten as

$$\delta_D(b) = \arg\max_{d \in D} P(b) \cdot [u'_1 + U'_2(d) * U_3(b)] \tag{3.4}$$

where $u'_1 = \sum_a P(a) \cdot U_1(a)$ and $U'_2(d) = \sum_a P(a) \cdot U_2(a, d)$. Given that $P(b)$ is always non-negative and u'_1 is a constant,

$$\delta_D(b) = \arg\max_{d \in D} U'_2(d) * U_3(b) \tag{3.5}$$

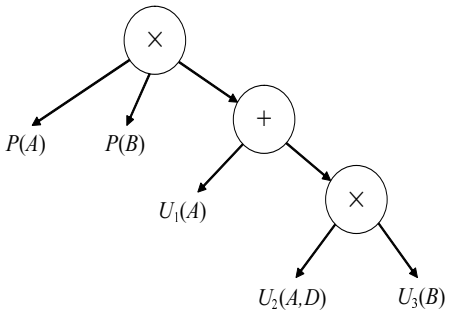

Figure 3.3: Tree of potentials (ToP) for the ID of Figure 3.2.

If the utility node U_3 is monotonic, then its values are either all non-negative or all non-positive. In the former case,

$$\forall b,\ U_3(b) \geq 0 \implies \forall b,\ \max_{d \in D} U_2'(d) * U_3(b) = U_3(b) * \max_{d \in D} U_2'(d) \tag{3.6}$$

$$\implies \forall b,\ \delta_D(b) = \arg\max_{d \in D} U_2'(d) \tag{3.7}$$

and in the latter

$$\forall b,\ U_3(b) \leq 0 \implies \forall b,\ \max_{d \in D} U_2'(d) * U_3(b) = U_3(b) * \min_{d \in D} U_2'(d) \tag{3.8}$$

$$\implies \forall b,\ \delta_D(b) = \arg\max_{d \in D} -U_2'(d) \tag{3.9}$$

In both cases $\delta_D(b)$ is independent of B, i.e., B is quasi-structurally redundant for decision D with respect to the subset of utility nodes $\{U_3\}$.

3.2 Variable-elimination on a tree of potentials

The basic idea of our algorithm consists of representing the matrix of an influence diagram, defined in Equation 3.1, as a tree of potentials (ToP), whose leaves (also called *terminal nodes*) represent probability potentials ϕ_i or utility potentials ψ_j, and each non-terminal node indicates either the sum or the product of the potentials represented by its children.

For instance, Figure 3.3 shows the ToP for the ID in Figure 3.2, whose matrix is $P(a) \cdot P(b) \cdot [U_1(a) + U_2(a, d) \cdot U_3(b)]$.

The construction of the ToP proceeds as follows. The root will always be a non-terminal node of type product. Each probability potential of the ID is added as a child of the root. If the bottom node of the ID, U_0, is an ordinary utility node or a super value node of type sum, it is also added as a child of the root. On the other hand, if U_0 is a

super value node of type product, its parents in the ID are added as children of the root in the ToP. All the other utility nodes in the ID must be added in the same way. As a result, the ToP represents the matrix of the ID, i.e., the tree of utility nodes in the ID, although upside down, together with the probability potentials.

A non-terminal node in a ToP is said to be *duplicated* if it is of the same type as its parent. A duplicated node in a ToP can be removed by transferring its children to its parent.

We describe in the next two subsections how the elimination of chance and decision variables in an ID can be handled by applying the sum and max operators, respectively, to the ToP. We will assume that the ToP does not contain duplicated nodes.

3.2.1 Elimination of a chance variable on a ToP

The elimination of a chance variable A consists of applying the operator $\sum_A$ to the ToP. We divide this process into two phases: we first unfork the ToP, and then eliminate A in the leaves of the new ToP, according to the following definitions.

Definition 3.2.1 *A variable X* appears *in a ToP t if it belongs to its domain, i.e., if it belongs to the domain of some of the terminal nodes of t.*

Definition 3.2.2 *A node n of type product* is forked with respect to (wrt) a variable A *if A appears in more than one of the branches of n.*

Definition 3.2.3 *A ToP* is forked wrt A *if at least one of its product nodes is forked wrt A. Otherwise, it is* non-forked.

For example, variable A appears in the ToP in Figure 3.3. The root node is forked wrt A because A appears in two of its three branches. Consequently, the ToP in that figure is forked wrt variable A. In contrast, the subtree rooted at the sum node is not forked wrt A because it only contains one product node, which is not forked.

Algorithm for eliminating forked nodes

Using an object-oriented programming representation, each node in a ToP may be implemented as an object of class *ToP-node* having three properties:

- *dependsOnVariable,* a Boolean value that indicates whether the node depends on the variable A to be eliminated;

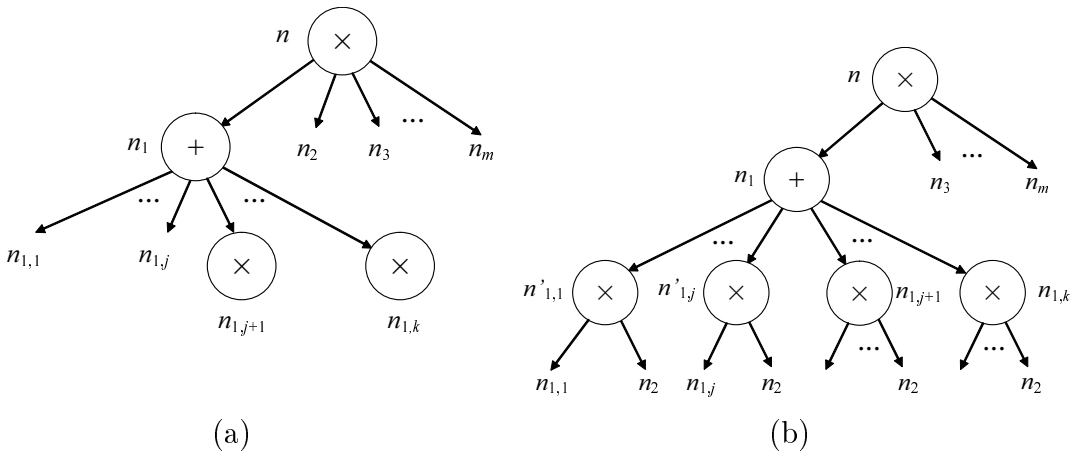

Figure 3.4: (a) A ToP, where both n_1 and n_2 depend on the chance variable to be eliminated, A. (b) A ToP equivalent to (a), in which n_2 has been distributed with respect to n_1.

- *dependentChildren,* a list of the children of the node that depend on variable A;
- *mayBeForked*, a Boolean value used by the method *unfork* to avoid visiting each subtree several times; it is initialized to `true` for all nodes and is also set to `true` when the method *unfork* has to visit the same subtree again for a different variable.

The class *ToP-node* has a main method, *unfork*, which uses two auxiliary methods: *distribute* and *compact*. The method *distribute* transforms the tree in Figure 3.4.a, in which both siblings n_1 and n_2 depend on A, into the tree in Figure 3.4.b. Nodes n_1 and n_2 are children of a product node forked wrt A. The procedure *distribute* is described by Algorithm 3.1—see also Figure 3.4.

Under the conditions of the method *distribute*, illustrated in Figure 3.4, it is clear that the potential obtained after distributing n_2 is equivalent to the original potential, because

$$\psi_2 \times \sum_{l=1}^{k} \psi_{1,l} = \sum_{l=1}^{k} \psi_2 \times \psi_{1,l} \ . \tag{3.10}$$

For instance, in the example in Figure 3.3, whose potential was $P(a) \cdot P(b) \cdot [U_1(a) + U_2(a,d) \cdot U_3(b)]$, after distributing $P(a)$ with respect to the sum node, the new potential will be $P(b) \cdot [P(a) \cdot U_1(a) + P(a) \cdot U_2(a,d) \cdot U_3(b)]$ (see Figure 3.5.a).

Please note that the product node that represents $P(a) \cdot U_1(a)$ in Figure 3.5.a is forked, but since $P(a)$ and $U_1(a)$ are terminal nodes, it can be unforked by multiplying its children. This process is performed by the method *compact*, shown as Algorithm 3.2.

Algorithm 3.1 distribute (for a ToP)

Input: n_1: a sum node depending on A, child of a product node n forked wrt A;
n_2: another node depending on A, sibling of n_1;
A: a chance variable.
Effects: the subtrees under n_1 are multiplied by n_2.

1. **for all** $n_{1,i} \in children(n_1)$ **do**
2. **if** $n_{1,i}$ is a terminal node **then**
3. remove $n_{1,i}$ as a child of n_1;
4. add a new product node $n'_{1,i}$ as a child of n_1;
5. add both $n_{1,i}$ and n_2 as children of $n'_{1,i}$;
6. $n'_{1,i}.mayBeForked$:= `true`;
7. **else**
8. // $n_{1,i}$ is a product node
9. add n_2 as a child of $n_{1,i}$;
10. $n_{1,i}.mayBeForked$:= `true`;
11. **end if**
12. **end for**
13. $n_1.mayBeForked$:= `true`;

Algorithm 3.2 Compact

Input: n: a product node, whose children depending on A are all terminal nodes.
Effects: the children of n that depend on A are replaced by their product.

1. remove from $n.dependentChildren$ and $n.children$ all the leaves that depend on A and remove them as children of n;
2. add the product of all to $n.children$ and to $n.dependentChildren$;
3. **if** n has only one child (say n_1) **then**
4. replace n with n_1 in the tree;
5. **end if**

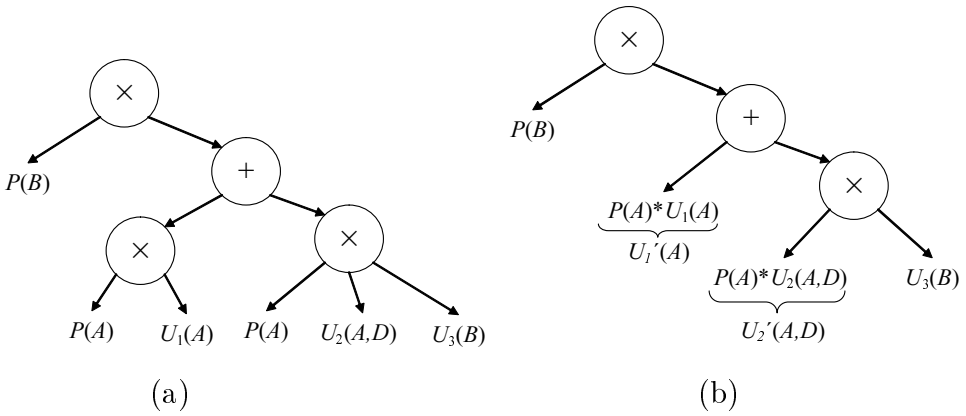

Figure 3.5: (a) ToP equivalent to that in Figure 3.2.b, in which $P(A)$ has been distributed with respect to the sum node. (b) ToP equivalent to (a), in which the leaves dependent on A have been compacted and replaced by two new potentials, $U_1'(A)$ and $U_2'(A, D)$.

Figure 3.5.b displays the result of applying the method *compact* to the product nodes of the ToP of Figure 3.5.a, when A is the variable to be eliminated.

Finally, the method *unfork*, invoked as $n.unfork(A)$, transforms the subtree under node n into a new subtree representing an equivalent potential, but unforked wrt A. The method *unfork* is described by Algorithm 3.3.

Please note that the while loop in this algorithm executes only when at least two children of n depend on A, and at least one of them—which we have called n_1—is of type sum, because after executing $compact(n)$ node n cannot have two leaf children depending on A.

Theorem 3.2.1 *For every ToP, the algorithm* unfork *terminates in a finite number of steps, returning a non-forked ToP.*

The proof can be found in Appendix A.0.1.

Elimination of a chance variable from a non-forked tree

When a tree is non-forked, the process of eliminating a chance variable A can be understood as "transferring" the $\sum_A$ operator from the root of the ToP down to the leaves that depend on A, according to the following theorem.

Theorem 3.2.2 *Let t be a ToP, non-forked wrt A, representing the potential ψ. The potential $\sum_A \psi$ is equivalent to the potential represented by the ToP t' obtained by replacing in t each terminal node ψ_i depending on A with the potential $\sum_A \psi_i$.*

Algorithm 3.3 Unfork

Input: n (object receiving the message): a node in the ToP;
A: a chance variable.
Effects: n is unforked wrt A, its attribute *mayBeForked* is set to `false` and the attribute *dependsOnVariable* indicates whether n depends on A.

1. **if** $n.mayBeForked$ = `true` **then**
2. **if** n is a terminal node **then**
3. **if** the potential of n depends on A **then**
4. $n.dependsOnVariable$:= `true`;
5. **else**
6. $n.dependsOnVariable$:= `false`;
7. **end if**
8. **else**
9. // n is a non-terminal node
10. **for all** $n_i \in children(n)$ **do**
11. $n_i.unfork(A)$;
12. **end for**
13. $dependentChildren$:= children n_i of n such that $n_i.dependsOnVariable$ = `true`;
14. **if** $(size(dependentChildren) > 0)$ **then**
15. $n.dependsOnVariable$:= `true`;
16. **else**
17. $n.dependsOnVariable$:= `false`;
18. **end if**
19. **if** n is of type product **then**
20. $compact(n)$;
21. **while** $(size(dependentChildren) > 1)$ **do**
22. n_1 := a sum node in $dependentChildren$;
23. n_2 := other node in $dependentChildren$;
24. $distribute(n_1, n_2)$;
25. $n_1.unfork(A)$;
26. **end while**
27. **end if**
28. **end if**
29. $n_1.mayBeForked$:= `false`;
30. **end if**

The proof can be found in Appendix A.0.2.

Going back to the example in Figure 3.2, the potential $P(b) \cdot [U_1'(a) + U_2'(a,d) \cdot U_3(b)]$ was represented by the tree in Figure 3.5.b, which is non-forked wrt A. The elimination of chance variable A is performed by replacing $U_1'(a)$ with the constant $u_1' = \sum_a U_1'(a)$, and replacing $U_2'(a,d)$ with $U_2'(d) = \sum_a U_2'(a,d)$. The result is $\psi = P(b) \cdot [u_1' + U_2'(d) \cdot U_3(b)]$.

3.2.2 Elimination of a decision variable on a ToP

The elimination of a decision variable D from a potential ψ that does not depend on D is trivial, because $\max_D \psi = \psi$. The elimination from a terminal potential that depends on D is also immediate. Let us assume that ψ is represented by a ToP, whose root node r is not terminal, and ψ_i is the potential represented by the i-th child of r.

We analyze first the case in which r is of type sum. If more than one of the ψ_is depend on D, it is not correct to eliminate D by replacing each ψ_i with $\max_D \psi_i$, because $\max_D(\psi_i + \psi_{i'})$ may be different from $\max_D \psi_i + \max_D \psi_{i'}$—please note the contrast with Theorem 3.2.2. The correct procedure when r is of type sum is to add all the potentials that depend on D before eliminating D; the rest of the potentials are not modified. Formally, if J is the set of subindices such that ψ_j does not depend on D, and K contains the other subindices, then:

$$\max_D \psi = \max_D \sum_i \psi_i = \sum_{j \in J} \psi_j + \max_D \underbrace{\sum_{k \in K} \psi_k}_{\psi_D} . \tag{3.11}$$

If r is of type product, we define J as the set of subindices such that ψ_j is monotonic and does not depend on D, and K as its complementary. We also define m as the number of indices in J such that ψ_j has at least one negative value (the other values of ψ_j must be either negative or null, because ψ_j is monotonic). If m is even, then $\prod_{j \in J} \psi_j$ is non-negative, and consequently,

$$\max_D \psi = \max_D \prod_i \psi_i = \prod_{j \in J} \psi_j \cdot \max_D \underbrace{\prod_{k \in K} \psi_k}_{\psi_D} . \tag{3.12}$$

If m is odd, then $-\prod_{j \in J} \psi_j$ is non-negative and

$$\max_D \psi = (-1) \cdot \prod_{j \in J} \psi_j \cdot \max_D \underbrace{-\prod_{k \in K} \psi_k}_{\psi_D} . \tag{3.13}$$

This analysis leads to Algorithm 3.4, which corresponds to the elimination of a decision variable D on a ToP. The method for changing the sign of a subtree rooted at n, used in step 23, is given by Algorithm 3.5. Please note that when only one of the children of n depends on D, namely $n_{k'}$, then the algorithm is invoked recursively on $n_{k'}$, preserving the structure of that subtree (steps 11 and 25). On the contrary, when more than one children depend on D, those branches collapse into a potential ψ_D (steps 13 and 3.12), which after maximizing on D, is reinserted as a terminal node (step 36).

For example, we have already shown that, for the ID in Figure 3.2 (see also Fig. 3.5), after eliminating A the matrix is $\psi = P(b) \cdot [u'_1 + U'_2(d) \cdot U_3(b)]$. When eliminating D, we have $\max_d \psi = P(b) \cdot [u'_1 + \max_d(U'_2(d) \cdot U_3(b))]$. In general,

$$\delta_D(b) = \arg\max_{d \in D} \left(U'_2(d) \cdot U_3(b)\right) . \tag{3.14}$$

However, if U_3 is non-negative, then $\max_d \psi = P(b) \cdot [u'_1 + u'_2 \cdot U_3(b)]$, where $u'_2 = \max_d U'_2(d)$, and the optimal policy is

$$\delta_D = \arg\max_{d \in D} U'_2(d) , \tag{3.15}$$

which does not depend on B. If U_3 is non-positive, then $\max_d \psi = P(b) \cdot [u'_1 + (-1) \cdot u'_2 \cdot U_3(b)]$, where $u'_2 = \max_d(-U'_2(d))$, and the optimal policy is

$$\delta_D = \arg\max_{d \in D} -U'_2(d) , \tag{3.16}$$

which does not depend on B, either. Therefore, when $U_3(b)$ is monotonic, our algorithm does not include in the domain of δ_D the quasi-structurally redundant variable B. In contrast, the arc-reversal algorithm (Tatman and Shachter, 1990) would collapse all the utility nodes into a single node when eliminating A; as the parents of the new utility node are B and D, the elimination of D would always include B in the domain of δ_D.

Sensitivity analysis

Another advantage of our algorithm, closely related to the attempt to avoid redundant variables, is the possibility of simplifying sensitivity analysis. For instance, in the above example the policy for D was given by Equation 3.14, where $U'_2(d) = \sum_a P(a) \cdot U_2(a, d)$. Consequently, if we perform a sensitivity analysis for variable D in order to determine which variations of the parameters of the ID may lead to a different policy, we need only

Algorithm 3.4 eliminateDecision

Input: n: a node in the ToP;
D: a decision variable.
Effects: the decision variable D is eliminated from the tree rooted at n.
Output: an optimal policy δ_D for D.

if n depends on D **then**
  if n is a terminal node with associate potential ψ **then**
    replace ψ with $\max_D \psi$;
    return the optimal policy, $\delta_D := \arg\max_D \psi$;
  else
    // n is non-terminal; ψ_i is the potential represented by n_i, the i-th child of n
    if n is of type sum **then**
      $K :=$ set of subindices such that ψ_k depends on D;
      if $|K| = 1$ **then**
        // K contains only one index, k'
        return *eliminateDecision*($n_{k'}$, D);
      else
        $\psi_D := \sum_{k \in K} \psi_k$; // cf. Equation 3.11
      end if
    else
      // n is of type product
      $J :=$ set of subindices such that ψ_j is monotonic and does not depend on D;
      $K :=$ set of subindices complementary of J;
      $m :=$ number of subindices in J such that ψ_j has at least one negative value;
      if $|K| = 1$ **then**
        if m is odd **then**
          add a node with the constant potential -1 as a child of n;
          changeSign($n_{k'}$);
        end if
        return *eliminateDecision*($n_{k'}$, D);
      else
        // several potentials depend on D
        $\psi_D := \prod_{k \in K} \psi_k$; // cf. Equation 3.12
        if m is odd **then**
          add a node with the constant potential -1 as a child of n;
          $\psi_D := -\psi_D$; // cf. Equation 3.13
        end if
      end if
    end if
    for every $k \in K$, remove the k-th child of n;
    add a node with the potential $\max_D \psi_D$ as a child of n;
    return the optimal policy $\delta_D := \arg\max_D \psi_D$;
  end if
end if

Algorithm 3.5 changeSign

Input: n: a node in the ToP, representing the potential ψ.
Effects: The subtree rooted at n is modified to represent the potential $-\psi$.

```
1. if n is a terminal node then
2.    replace ψ with −ψ;
3. else
4.    // n has several children, {n_i}_{i∈I}
5.    if n is of type sum then
6.       for all i ∈ I do
7.          changeSign(n_i);
8.       end for;
9.    else
10.      // n is of type product;
11.      add a node with the constant potential −1 as a child of n;
12.   end if
13. end if
```

to examine the probabilities in $P(a)$ and the utilities in $U_2(a, d)$ and $U_3(b)$, because δ_D does not depend at all on the other parameters, namely those in $P(b)$ and $U_1(a)$. This may lead to a significant simplification of sensitivity analysis.

Additional simplifications occur when $U_3(b)$ is monotonic. In this case, Equations 3.15 and 3.16 tell us that the policy only depends on the values of $P(a)$ and $U_2(a, d)$, and on the sign of the values of $U_3(b)$. This is important because usually human experts are uncertain about the exact value of a parameter, but not about its sign.

In contrast, in this example arc reversal (Tatman and Shachter, 1990) would eliminate D by maximizing on a potential derived from all the potentials that define the ID, namely $P(a)$, $P(b)$, $U_1(a)$, $U_2(a, d)$, and $U_3(b)$, which seems to indicate that every parameter in ID might affect the policy δ_D.

In summary, our variable-elimination algorithm may simplify sensitivity analysis if we keep track (for instance, by maintaining a set of pointers) of the ID potentials that have been involved in the computation of each potential at the ToP.

Inclusion of redundant variables

Although the distribution of potentials in general avoids the inclusion of redundant variables in the policies, as we have seen in the previous examples (see also the experiments in Section 3.5.2), it may fail to avoid them in some cases. The following example explains why.

Let us assume that we are interested in computing $\max_d \sum_a \psi$, where $\psi = [\psi_1(a) + \psi_2(b)] \cdot [\psi_3(a) + \psi_4(d)]$. When eliminating A, the node that represents ψ is forked wrt A.

If the algorithm takes $[\psi_1(a) + \psi_2(b)]$ as n_1 and $[\psi_3(a) + \psi_4(d)]$ as n_2, the result of the distribution is $\psi = \psi_1(a) \cdot [\psi_3(a) + \psi_4(d)] + \psi_2(b) \cdot [\psi_3(a) + \psi_4(d)]$. The first summand is represented by a product node, which is still forked. A new distribution leads to $\psi = \psi_1(a) \cdot \psi_3(a) + \psi_1(a) \cdot \psi_4(d) + \psi_2(b) \cdot [\psi_3(a) + \psi_4(d)]$ and $\sum_a \psi = \psi_{13} + \psi'_1 \cdot \psi_4(d) + \psi_2(b) \cdot [\psi'_3 + \psi_4(d)]$, where ψ_{13}, ψ'_1, and ψ'_3 are the constant potentials that result from summing out A from the terminal leaves of the previous potential. Then, the elimination of D will explicitly compute $\psi'_1 \cdot \psi_4(d) + \psi_2(b) \cdot [\psi'_3 + \psi_4(d)]$, which yields a terminal potential that depends on both B and D:

$$\max_d \sum_a \psi = \max_d \underbrace{\psi'_1 \cdot \psi_4(d) + \psi_2(b) \cdot [\psi'_3 + \psi_4(d)]}_{\psi(b,d)} .$$

Therefore the algorithm will include B in the policy δ_D.

However, the algorithm would have been able to detect that B is quasi-structurally redundant (wrt ψ'_1 and ψ_2) if it had distributed the top factors in a different way:

$$\begin{aligned}
\max_d \sum_a \psi &= \max_d \sum_a \underbrace{[\psi_1(a) + \psi_2(b)]}_{n_2} \cdot \underbrace{[\psi_3(a) + \psi_4(d)]}_{n_1} \\
&= \max_d \sum_a \{[\psi_1(a) + \psi_2(b)] \cdot \psi_3(a) + [\psi_1(a) + \psi_2(b)] \cdot \psi_4(d)\} \\
&= \max_d \{\psi_{13} + \psi_2(b) \cdot \psi'_3 + [\psi'_1 + \psi_2(b)] \cdot \psi_4(d)\} \\
&= \psi_{13} + \psi_2(b) \cdot \psi'_3 + [\psi'_1 + \psi_2(b)] \cdot \max_d \psi_4(d) .
\end{aligned}$$

As we have seen, the first distribution performed in this example failed to detect that ψ_4 is a common factor for ψ'_1 and ψ_2.

This example underlies the importance of deciding which candidates for a distribution (i.e., those factors of type sum depending on the variable to be eliminated) should be chosen as n_1 and n_2. The problem is that our algorithm performs myopically, in the sense that when eliminating a variable it does not take into account the effect that it will have on the subsequent elimination of other variables. The refinement of our algorithm in order to avoid redundant variables is an open problem, as mentioned in Section 3.6.[2]

[2]It is worthy of note that in this example our algorithm can obtain the correct domain for decision D if it selects the right distribution of potentials, while arc reversal (Tatman and Shachter, 1990) would always include the redundant variable B in the policy of D. However, the experiments in Section 3.5 show that in some exceptional cases arc reversal may include fewer redundant variables than ours—an issue that deserves further investigation.

3.2.3 Summary: Variable elimination algorithm using a ToP

Finally, Algorithm 3.6 integrates all the steps described so far. Please note that the input of this algorithm is not only an ID, but also an elimination order for the variables in $\mathbf{V}_C \cup \mathbf{V}_D$. This order has to be a *legal elimination sequence* (Nielsen and Jensen, 1999), which means that it must eliminate first the variables in $\mathbf{C}_n$, then D_n, then those in $\mathbf{C}_{n-1}$, and so on (see Equation 3.2). However, this condition only imposes a partial ordering on the variables in $\mathbf{V}_C \cup \mathbf{V}_D$: it is still necessary to order the variables inside each $\mathbf{C}_i$. The similarity of this problem with others in Bayesian networks make us conjecture that finding an optimal elimination sequence for our algorithms is NP-complete. Finding near-optimal orderings is an open issue, that we will discuss in Section 3.6.

Algorithm 3.6 Variable elimination for IDs with SVNs on a ToP

Input: I: an ID that may contain SVNs;
O: a legal sequence elimination for the variables in $\mathbf{V}_C \cup \mathbf{V}_D$;
Output: the *MEU* of the ID and an optimal policy δ_D for each decision variable D.

1. construct the ToP t of I;
2. remove the duplicate nodes from t; // see Section 3.2
3. **for all** $V \in O$ **do**
4. **if** $V \in \mathbf{V}_C$ **then**
5. unfork t wrt V; // Algorithm 3.3;
6. for each terminal node n_i in t depending on V, replace that node with $\sum_v \psi_i$;
7. **else**
8. // $V \in \mathbf{V}_D$
9. *eliminateDecision*(r, V); // Algorithm 3.4;
10. **end if**
11. **end for**
12. *MEU* := numerical value of the potential at t (a constant);

The correctness of Algorithm 3.6 is ensured given that the elimination of chance and decisions variables are performed according to a legal elimination sequence, and each transformation of the ToP preserves the *MEU* and the optimal policies.

3.3 Variable elimination on an ADG of potentials

In Section 3.2, the matrix of an ID was represented as a ToP. However, the matrix can also be represented as an acyclic directed graph of potentials (ADGoP). The main advantage of this representation is twofold: the saving of space in memory when a subtree appears more than once in a tree, and the saving of computational time when distributing a potential

and when eliminating a variable. Another advantage is the ability to cope with IDs in which a utility node can have two or more super-value children, as shown in Figure 3.6.

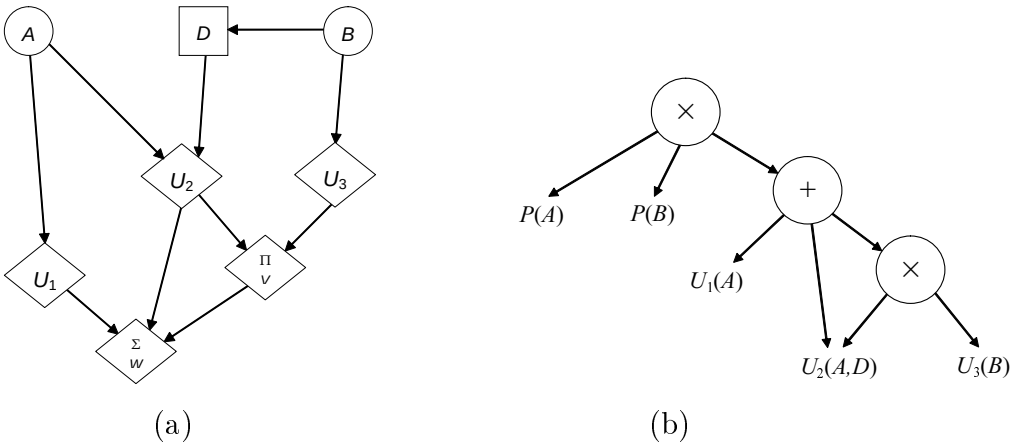

Figure 3.6: (a) Graph of an ID with super value nodes, in which U_2 has two children. The global utility potential is $\psi = U_1 + U_2 + (U_2 \cdot U_3)$. (b) Acyclic directed graph of potentials (ADGoP) for this ID.

The construction of the ADGoP from an ID is very similar to that of the ToP. The next two subsections explain how to eliminate chance and decision variables from an ADGoP, under the assumption that redundant nodes have already been removed—see Section 3.2.

3.3.1 Elimination of a chance variable on an ADGoP

The elimination of a chance variable A consists in applying the operator $\sum_a$ to the ADGoP. This process is divided in two phases: unforking the ADGoP, and eliminating A from the leaves of the new ADGoP.

Algorithm for eliminating forked nodes

The process of unforking the ADGoP is similar to that of the ToP. This way, each node in a ADGoP may be implemented as an object of class *ADGoP-node*, which has the same properties and methods as *ToP-node* (see Sec. 3.2.1). The method *unfork* is identical, but *distribute* and *compact* are slightly different in both classes, because when a node compacts its leaves, their children having other parents can not be removed from the ADGoP.

For instance, in Figure 3.7.a, the node n_2 is forked wrt A. When n_2 compacts its leaves $\phi_1(A)$ and $\phi_2(A)$, the leaf $\phi_1(A)$ can not be removed from the ADGoP because it is also a child of n_3. The link $n_2 \rightarrow \phi_1(A)$ will be removed and $\phi_1(A)$ will be multiplied by $\phi_2(A)$, but link $n_3 \rightarrow \phi_1(A)$ will remain, as shown in Figure 3.7.b.

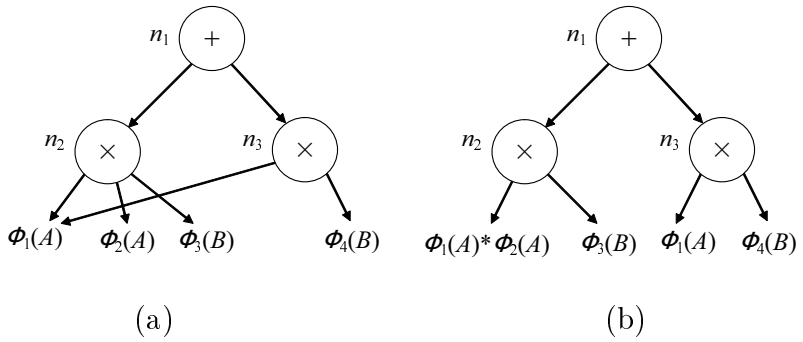

Figure 3.7: (a) An ADGoP, in which n_2 is forked wrt A. (b) ADGoP after the node n_2 in (a) compacts its leaves dependent on A.

In turn, the method *distribute* differs in that instead of creating several copies of n_2, as we did in the case of a ToP (see Fig. 3.4.b), we will draw several links from the children of n_1 to n_2 (see Fig. 3.8).

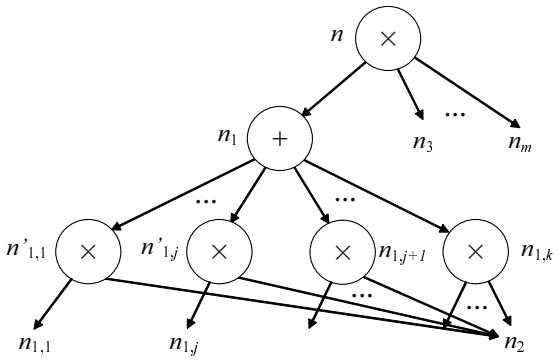

Figure 3.8: An ADGoP equivalent to the potential in Figure 3.4.a, in which n_2 has been distributed with respect to n_1.

Elimination of a chance variable from a non-forked ADGoP

When the ADGoP is non-forked, the process of eliminating a chance variable A is performed as in a ToP, i.e., the $\sum_a$ operator is "transferred down" from the root of the ADGoP to the leaves that depend on A. Even if a leaf has several parents, the $\sum_a$ operator is applied to it only once, thus saving time with respect to the case of a ToP.

3.3.2 Elimination of a decision variable on an ADGoP

The elimination of a decision variable D from an ADGoP is similar to its elimination from a ToP (cf. Algorithm 3.4, in Section 3.2.2): Algorithm 3.4 can also be applied when eliminating a decision variable in an ADGoP, but when a node compacts its leaves, as required by some steps of Algorithm 3.4, their children having other parents can not be removed from the ADGoP. Then, if a child n_1 of n has other parents, the link $n \rightarrow n_1$ must be removed, but n_1 cannot be eliminated from the graph. This is the same situation that appears when compacting the leaves of a node in an ADGoP before eliminating a chance variable.

3.3.3 Summary: algorithm VE

The variable-elimination algorithm on an ADGoP is performed as in a ToP (see Algorithm 3.6), with the only difference in step 1 we construct an ADGoP instead of a ToP, and then we perform all the operations of Algorithm 3.6 in the ADGoP as we have described in this section, i.e., by adapting the corresponding algorithms.

3.4 Variations of the algorithm

The previous section has presented the basic algorithm of variable elimination (VE) on an ADGoP. We discuss now three variations of that algorithm that may lead to more efficient computations. In Section 3.5 we will compare empirically these versions with the standard VE.

3.4.1 Division of potentials (algorithm VE-D)

The VE algorithm presented above does not distinguish between probability and utility potentials. In this respect, it is similar to some variable-elimination algorithms for IDs without SVNs (Cowell et al., 1999; Dechter, 1996; Shenoy, 1992). On the contrary, the variable-elimination algorithm proposed in (Jensen and Nielsen, 2007) for IDs without super-value nodes differentiates both types of potentials and, when eliminating a chance variable, normalizes the probability potentials by means of a *division*. The main advantage of this process is that the utility potentials obtained after multiplication by the normalized potentials represent the utilities associated with different scenarios, which may be useful for explaining the decision process to the user (Lacave et al., 2007).

Similarly, it is possible to design a new version of our VE algorithm for ID with SVNs, called VE-D (where "D" stands for divisions), which instead of storing all the potentials in the same ADGoP, manages a *list of probability potentials* (LoPP) and an *ADG of utility potentials* (ADGoUP). Their product represents the matrix of the ID. The construction of the ADGoUP for an ID is identical to that of the ADGoP (cf. Secs. 3.2 and 3.3), with the only difference that it does not include the probability potentials.

The procedure of VE-D is described by Algorithm 3.7. This algorithm substitutes the variable elimination algorithm on an ADGoP (Algorithm 3.6 adapted for using an ADGoP instead of a ToP). The main difference between both algorithms is that Algorithm 3.7 keeps the probability potentials in a LoPP, separated from the utility potentials contained in the ADGoP. The LoPP and the ADGoP are modified when a variable V is eliminated as follows. The LoPP is updated by removing the probability potentials dependent of V and adding to it their marginalization. The ADGoP is prepared to proceed to the elimination of V, which is performed in the ADGoP as described in Sections 3.3.1 and 3.3.2.

In Algorithm 3.7, the operator project_V in step 16 only makes sense when applied to a potential that does not depend on V, i.e., a potential whose value is the same for all the configurations having the same value of V. For instance, given a potential $\phi(v_1, v_2)$ such that $\phi(+v_1, +v_2) = \phi(\neg v_1, +v_2) = 0.9$ and $\phi(+v_1, \neg v_2) = \phi(\neg v_1, \neg v_2) = 0.4$, which does not depend on V_1, the operator project_{V_1} gives a new potential $\phi'(v_2) = \text{project}_{V_1}\phi(v_1, v_2)$ such that $\phi'(+v_2) = 0.9$ and $\phi'(\neg v_2) = 0.4$.

In Appendix A.0.3 we prove the correctness of VE-D, including the fact that when Algorithm 3.7 applies the operator $\text{project}_V\ \phi_V$, this potential does not depend on V.

Example 3.4.1 *For the ID in Figure 3.9, we have*

$$MEU = \sum_b \max_{d_1} \sum_c \max_{d_2} \sum_a P(a) \cdot P(b|a) \cdot P(c|a, d_1) \cdot \psi(a, d_2)\ . \tag{3.17}$$

When eliminating A we have $\phi_A(a, b, c, d_1) = P(a) \cdot P(b|a) \cdot P(c|a, d_1)$, $\phi^*_A(b, c, d_1) = \sum_a \phi_A(a, b, c, d_1)$, and

$$MEU = \sum_b \max_{d_1} \sum_c \max_{d_2}\ \phi^*_A(b, c, d_1) \underbrace{\sum_a \frac{\phi_A(a, b, c, d_1)}{\phi^*_A(b, c, d_1)} \cdot \psi(a, d_2)}_{\psi(b, c, d_1, d_2)}\ , \tag{3.18}$$

When eliminating D_2 there is no probability potential depending on D_2. Therefore, it is

Algorithm 3.7 Algorithm VE-D (variable elimination with divisions)

Input: I: an ID that may contain SVNs;
O: a legal sequence elimination for the variables in $\mathbf{V}_C \cup \mathbf{V}_D$;
Output: the *MEU* of the ID and an optimal policy δ_D for each decision variable D.

1. construct the LoPP l and the ADGoP g of I;
2. remove the duplicate nodes from g;
3. **for all** $V \in O$ **do**
4. remove from l all the potentials that depend on V;
5. let ϕ_V be the product of all of them;
6. **if** $V \in \mathbf{V}_C$ **then**
7. $\phi_V^* := \sum_V \phi_V$;
8. // multiply g by ϕ_V / ϕ_V^*
9. **if** the root of g (say r) is of type product **then**
10. add ϕ_V / ϕ_V^* to the children of r;
11. **else**
12. replace r by a product node r', and add r and ϕ_V / ϕ_V^* as children of r';
13. **end if**
14. **else**
15. // V is a decision
16. $\phi_V^* := \text{project}_V\ \phi_V$;
17. **end if**
18. add ϕ_V^* to l;
19. eliminate V from g, as explained in Sections 3.3.1 and 3.3.2;
20. **end for**
21. *MEU* := numerical value of the potential at g (a constant);

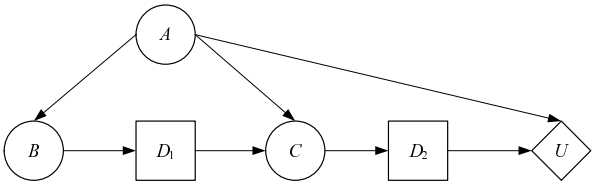

Figure 3.9: Influence diagram whose variable-elimination evaluation is detailed in Example 3.4.1.

not necessary to perform any multiplication nor any projection:

$$MEU = \sum_b \max_{d_1} \sum_c \phi_A^*(b,c,d_1) \cdot \underbrace{\max_{d_2} \psi(b,c,d_1,d_2)}_{\psi(b,c,d_1)} . \tag{3.19}$$

When eliminating C only one probability potential depends on this variable, namely ϕ_A^*. Therefore, $\phi_C(b,c,d_1) = \phi_A^*(b,c,d_1)$, $\phi_C^*(b,d_1) = \sum_c \phi_C(b,c,d_1)$, and

$$MEU = \sum_b \max_{d_1} \phi_C^*(b,d_1) \underbrace{\sum_c \frac{\phi_C(b,c,d_1)}{\phi_C^*(b,d_1)} \cdot \psi(b,c,d_1)}_{\psi(b,d_1)} . \tag{3.20}$$

When eliminating D_1 only one probability potential depends (apparently) on this variable: $\phi_{D_1}(b,d_1) = \phi_C^*(b,d_1)$. However, Lemma A.0.2 (cf. Appendix A.0.3) states that $\phi_{D_1}(b,d_1)$ does not depend on d_1. We then have $\phi_{D_1}^*(b) = \text{project}_{D_1}\ \phi_{D_1}(b,d_1)$. Then,

$$MEU = \sum_b \phi_{D_1}^*(b) \cdot \underbrace{\max_{d_1} \psi(b,d_1)}_{\psi(b)} . \tag{3.21}$$

Finally, when eliminating B we have $\phi_B(b) = \phi_{D_1}^*(b)$, $\phi_B^* = \sum_b \phi_B(b)$, and

$$MEU = \phi_B^* \sum_b \phi_B(b) \cdot \psi(b) . \tag{3.22}$$

When the algorithm VE-D eliminates a decision D_i, the ADPoUP represents a potential that depends on the informational predecessors of D_i: $\psi_i(\mathbf{c}_0, d_1, \ldots, d_{i-1}, \mathbf{c}_{i-1}, d_i)$. It is possible to show that the value of ψ_i is the utility of the scenario in which (1) the variables in $\mathbf{C}_j$ $(0 \leq j \leq i)$ take the values dictated by the configuration $\mathbf{c}_j$, (2) the decision maker chooses option d_k for each decision D_k $(1 \leq k \leq i)$ and (3) chooses the best option for the decisions after D_i.[3] This is the main reason for dividing the probability potentials:

[3]The proof is similar to that offered in (Jensen and Nielsen, 2007) for the variable-elimination algorithm

when a user of the decision-support system is often interested in knowing why the system recommends option d_i for scenario $(\mathbf{c}_0, d_1, \ldots, d_{i-1}, \mathbf{c}_{i-1})$ rather than option d'_i, it is useful to display the values $\psi_i(\mathbf{c}_0, d_1, \ldots, d_{i-1}, \mathbf{c}_{i-1}, d_i)$ and $\psi_i(\mathbf{c}_0, d_1, \ldots, d_{i-1}, \mathbf{c}_{i-1}, d'_i)$, i.e., the utilities of the two options (Luque and Díez, 2006). When evaluating an ID with the arc-reversal algorithm, these values can be read directly from the utility table of the ID before eliminating D_i. However, in general VE (variable elimination without divisions) can not show these utilities, and this is the main reason for using VE-D instead of VE.

3.4.2 Subset rule

Tatman and Shachter (1990) proposed a heuristic, called the *subset rule*, for reducing the storage space required by their arc reversal algorithm: if two utility nodes U_1 and U_2 have the same successor U, being a super-value node of type sum/product, and $Pa(U_2) \subseteq Pa(U_1)$, it is possible to replace them by a new node U', such that [1] $Pa(U') = Pa(U_1)$, [2] U' is a parent of U, and [3] $U' = U_1 + U_2$ or $U' = U_1 \times U_2$, respectively. This replacement seems to be advantageous in general because it does not increase the size of any operation necessary to solve the ID, and may simplify subsequent combinations of potentials.

The subset rule can also be introduced in the algorithms VE and VE-D: when two leaves n_1 and n_2 representing potentials ϕ_1 and ϕ_2, respectively, have the same parent in the ToP of in the AGDoUP and $dom(\phi_1) \subseteq dom(\phi_2)$, then compacting n_1 and n_2 liberates storage space and may simplify the next operations.

However, the application of the subset rule needs to check if $dom(\phi_1) \subseteq dom(\phi_2)$ for every pair of children of each potential, which has a certain computational cost, thus this rule is counterproductive in some cases. In Section 3.5.2 we analyze empirically the changes in the time and space required when the subset rule is applied.

3.4.3 Unity potentials

The efficiency of the algorithms VE and VE-D can be improved by avoiding certain computations. For example, when eliminating a barren node X_i,[4] the ADGoP (for VE) or the LoPP (for VE-D) contains a potential $P(X_i|pa(X_i))$. The elimination of X_i implies computing $\sum_{x_i} P(x_i|pa(x_i))$, which is 1. Similarly, if $\phi_X = P(x|pa(X))$ and X is a chance variable, the VE-D algorithm will compute $\phi'_X = \sum_x \phi_X$, which is also 1. In other cases,

for IDs without super value nodes: the proof remains valid if ψ is given by an ADGoUP instead of a sum of utility potentials.

[4] According with (Shachter, 1986), a chance or decision node without descendants is said to be barren.

it will be necessary to perform the marginalization $\sum_x [\phi_X / \phi_X^*]$, which is equal to 1 even if ϕ_X was not a conditional probability, because $\phi_X^* = \sum_x \phi_X$—see Algorithm 3.7, Step 7.

If the algorithm recognizes these situations, it can save the computational cost of summing out X. More importantly, the algorithm will be able to replace $\sum_x P(x|pa(X))$ with 1—a constant, while the computation of $\sum_x P(x|pa(x))$ may return a potential whose domain is apparently $pa(X)$, and this may lead to including redundant variables in the policies.

An open line for future research is how to integrate in our variable elimination algorithm the recent improvements for the lazy evaluation of IDs (Madsen and Jensen, 1999; Vomlelova and Jensen, 2002), whose purpose is to avoid the operations that yield unity potentials.

3.5 Empirical evaluation

We have performed a series of experiments for assessing the efficiency of the variable elimination algorithm proposed in this paper. The first problem we faced is that we only have a few real-world examples of IDs with SVNs, and these are not complex enough for comparing the different versions of our algorithm between themselves and with the arc reversal (AR) algorithm of Tatman and Shachter. The repositories of graphical probabilistic models available on Internet do not contain IDs with SVNs. For this reason, we have run the experiments on randomly generated IDs.

3.5.1 Algorithm for generating IDs randomly

Vomlelova (2003) proposed an algorithm for randomly generating IDs with several (ordinary) utility nodes. We have adapted and extended it in order to generate IDs with SVNs. The parameters of the algorithm are: *nNodes,* the total number of chance and decision nodes; *decisionRatio,* the probability that a node is a decision (otherwise, it is a chance node); N, the number of iterations, each adding or deleting an arc; *nParents*, the maximum number of parents for a chance or decision node; *nUtil,* the number of ordinary utility nodes; and *nParentsUtil*, the number of parents per utility node. The procedure for generating an ID is described by Algorithm 3.8.

We must note in step 13 that the fact that i and j are not ordered (step 12) implies that there is no path from one node to the other. Therefore, drawing a link between them (step 13) can not create a loop nor a cycle.

Algorithm 3.8 Generate a random ID

Input: *nNodes*, *decisionRatio*, *N*, *nParents*, *nUtil* and *nParentsUtil* (cf. Sec. 3.5.1).
Output: an influence diagram.

1. create a tree having *nNodes* nodes;
2. for each node, randomly decide whether it is a decision (with probability *decisionRatio*) or a chance node;
3. **for** $k = 1$ to N **do**
4. select randomly a pair of distinct nodes i and j;
5. **if** the arc $i \rightarrow j$ exists in the graph **then**
6. if the graph remains connected, delete this arc;
7. **else**
8. if the graph remains acyclic and the number of parents of j does not exceed *nParents*, add the arc*;*
9. **end if**
10. **end for**
11. **while** there are at least two decisions i and j such that $i \notin ancestors(j)$ and $j \notin ancestors(i)$ **do**
12. select two distinct decision nodes i and j such that $i \notin ancestors(j)$ and $j \notin ancestors(i)$;
13. randomly decide whether the arc $i \rightarrow j$ or $i \rightarrow j$ is added to the graph (with probability 0.5);
14. **end while**
15. generate *nUtil* ordinary utility nodes, each with *nParentsUtil* parents randomly selected among the decision and chance nodes;
16. **while** there are several utility nodes without descendants **do**
17. **if** the number of utility nodes without descendants is greater than nParentsUtil **then**
18. randomly select *nParentsUtil* utility nodes without descendants and add arcs from them to a new super-value node;
19. **else**
20. draw arcs from them to a new super-value node;
21. **end if**
22. randomly decide if the new super-value node is sum (with probability 0.5) or product;
23. **end while**
24. generate a probability table for each chance node;
25. generate a utility table for each ordinary utility node;

We have assigned random non-negative values to the potentials, with the only restriction that probabilities must be normalized.

3.5.2 Experimental results

We executed the above algorithm with $decisionRatio = 0.3$, $N = 300$ (additions or removals of arcs), $nParents = 3$, $nUtil = 7$, and $nParentsUtil = 7$. All the variables were binary. The number of nodes, $nNodes$, varied from 5 to 24. We generated 100 influence diagrams for each number of nodes, which amounts to a total of 2,000 IDs. Figure 3.10 displays the graph of one of the influence diagrams generated.

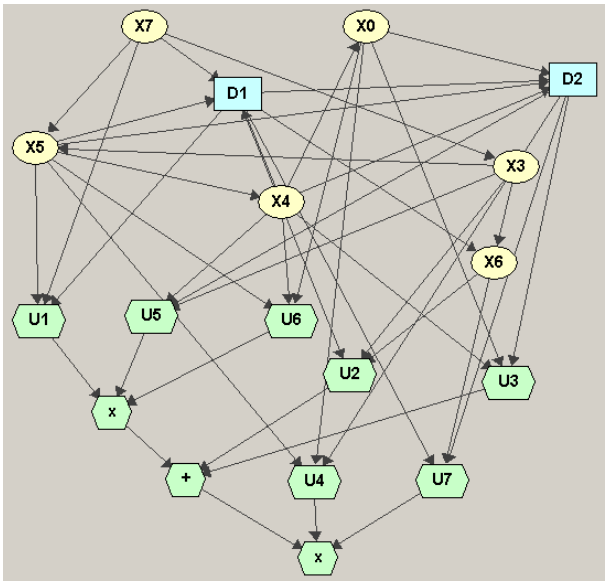

Figure 3.10: A random influence diagram generated by Algorithm 3.8, with $nNodes = 8$.

Each ID was evaluated with three algorithms: Tatman and Shachter's arc reversal (AR), variable-elimination without divisions (VE) and with divisions (VE-D). We only calculated the global utility of the ID and the optimal policy for each decision because VE cannot compute the expected utility of each option.

All the algorithms employed the same elimination order of variables when evaluating each ID in order to compare them in the same conditions. The elimination order was previously established by evaluating the ID qualitatively with Tatman and Shachter's algorithm.[5]

[5]The time necessary to evaluate an ID qualitatively with AR is negligible compared with the time required by a full evaluation. Therefore, the fact that VE and VE-D would need an additional amount of time to obtain the elimination order does not affect the results of our experiments—see also Section 3.6.

The algorithms were implemented in Java 6.0 with the Elvira software package.[6] The tests were run on an Intel Core 2 computer (2.4 GHz) with 2 GB of memory under Windows XP.

Comparison of AR, VE, and VE-D (without the subset rule)

Time and space efficiency First, we computed the ratio of the times required by AR and VE for each ID. Table 3.1 summarizes the results, grouped by the number of nodes. Given that the distribution is very skewed, we show both the median and the mean along with some other percentiles in this table.[7]

By observing the means (second column), we can see that on average VE is around 10 times faster than AR. The last column in this table tells us that, for one ID, AR was 339 times slower than VE—see also Figure 3.11.[8] The 95th percentile column shows that in around 5% of cases VE is at least 30 times faster than AR. On the contrary, the cases in which AR is faster than VE (those in which the ratio is smaller than 1) are infrequent, as shown by the 5th percentile column.

The minimum displayed in the table means that in the most favorable case for AR, it was only 5 times faster than VE, and this difference occurred for an ID having only 6 nodes, for which the time spent by both algorithms is negligible. For bigger diagrams, AR could never be twice faster than VE. In contrast, Figure 3.11 shows that for several influence diagrams VE was over 100 times faster than AR, with a maximum of around 340 times.

When comparing the storage space required by these algorithms, measured as the total size of the numerical tables, we observe that in general VE needs less memory than AR—see the "means" column in Table 3.2. In the case of large IDs, in which the limit of memory is a critical issue, AR requires on average around 3 or 4 times more space than VE, with a median ratio of almost 2. The 5th and 95th percentile columns in Table 3.2 also show the superiority of VE over AR in the case of large IDs: the former rarely needs twice more space than the latter (the maximum shown in the "min"column is $1/0.14 = 7$ times), while in 5% of the cases AR needs at least 10 times more space than VE (the maximum being almost 60). Correspondingly, in Figure 3.12 we can see that for several

[6]The Elvira program was developed by several Spanish universities (Elvira Consortium, 2002). The source code, a user manual, and other documents can be downloaded from `www.ia.uned.es/~elvira`.

[7]The minimum, the median, and the maximum are the 0th, 50th, and 100th percentiles, respectively.

[8]In this figure we have used *boxplots*, which provide a summary of a set of data in graphical terms. The top and bottom of each box represent the upper and lower quartiles of each group of data, and the line in the middle represents the median. The extremes of the Whiskers that extend from each end of the box are 1.5 times the interquantile range from the ends of the box. Outliers, represented by red circles, are individuals whose value is higher or lower than the extremes of the whiskers.

Nodes	Mean	Min	5th perc.	Median	95th perc.	Max
5	13.20	2.91	7.58	13.06	17.72	37.93
6	11.78	0.20	6.50	11.62	17.00	26.39
7	10.81	3.70	4.88	11.10	16.27	20.26
8	11.21	2.59	6.40	10.78	19.51	22.77
9	9.19	2.23	4.30	8.49	15.66	21.04
10	9.90	3.14	3.80	9.34	18.27	28.33
11	9.81	1.96	2.65	9.09	19.97	24.85
12	11.12	1.50	3.21	10.11	22.76	34.25
13	9.36	0.76	2.42	6.82	17.89	64.99
14	11.41	0.89	2.06	8.07	35.51	62.27
15	12.22	0.74	2.18	9.60	35.28	45.46
16	12.07	1.08	2.42	8.15	29.26	101.98
17	16.84	0.97	2.14	8.04	79.22	112.67
18	13.58	0.98	1.80	7.48	45.85	96.57
19	15.08	1.29	2.10	6.13	57.53	122.20
20	11.37	0.54	1.48	7.43	40.54	79.38
21	15.80	0.88	1.64	7.12	46.42	338.84
22	14.03	1.24	1.72	6.54	39.77	219.88
23	12.62	0.74	1.42	5.87	53.02	90.36
24	12.29	0.82	1.54	6.64	39.21	157.80
Total	**12.18**	**0.20**	**2.16**	**9.15**	**30.76**	**338.84**

Table 3.1: Ratio of the times required by AR and VE.

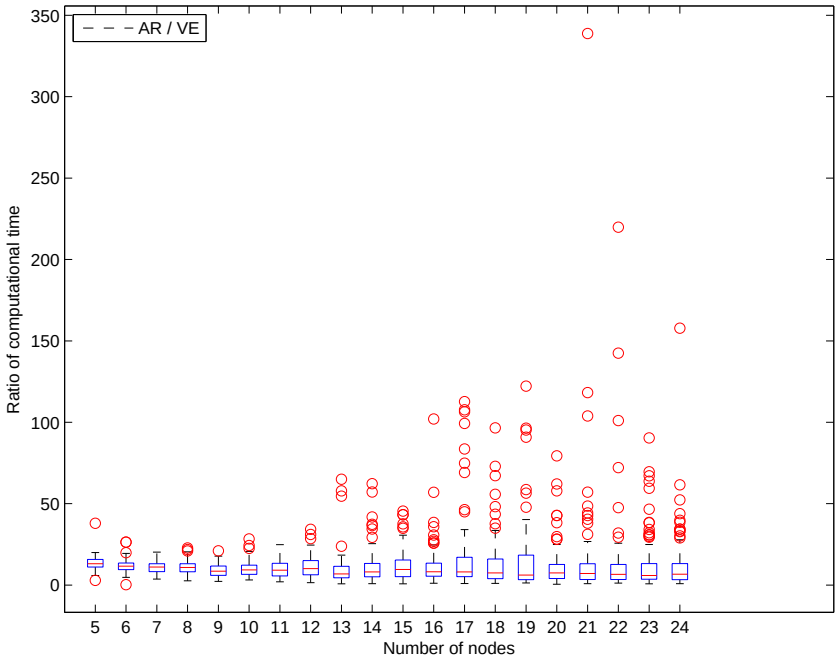

Figure 3.11: Ratio of the times required by AR and VE.

Nodes	Mean	Min	5th perc.	Median	95th perc.	Max
5	1.00	0.23	0.67	0.99	1.39	4.07
6	0.96	0.02	0.79	0.97	1.19	1.41
7	1.00	0.67	0.82	0.98	1.16	2.48
8	1.01	0.69	0.86	1.00	1.24	1.39
9	1.01	0.39	0.81	1.00	1.19	1.31
10	1.03	0.79	0.88	1.01	1.21	1.35
11	1.03	0.42	0.80	1.00	1.33	1.51
12	1.08	0.58	0.91	1.03	1.44	1.94
13	1.06	0.43	0.83	1.02	1.44	1.77
14	1.06	0.74	0.85	1.01	1.42	1.71
15	1.03	0.27	0.77	1.02	1.33	1.42
16	1.08	0.63	0.73	1.06	1.45	1.64
17	1.16	0.51	0.80	1.03	1.52	7.90
18	1.12	0.50	0.86	1.08	1.50	1.90
19	1.34	0.69	0.81	1.11	1.70	20.82
20	1.12	0.32	0.80	1.05	1.69	2.05
21	1.15	0.28	0.87	1.10	1.52	1.86
22	1.15	0.48	0.82	1.11	1.58	2.42
23	1.17	0.19	0.77	1.11	1.68	3.46
24	1.13	0.10	0.64	1.11	1.63	1.87
Total	**1.09**	**0.02**	**0.80**	**1.03**	**1.44**	**20.82**

Table 3.5: Ratio of the times required by VE-D and VE.

IDs, AR needed 10, 20, and even 60 times more space than VE, which is a significant difference.

We obtained very similar results when comparing AR and VE-D, both with respect to time (Table 3.3 and Figure 3.13) and space (Table 3.4 and Figure 3.14). This result is coherent with the experimental evidence that the performances of VE and VE-D are very close, both in terms of time (Table 3.5) and space (Table 3.6). The cases in which VE is significantly more efficient than VE-D, or vice versa, are very rare. If time or space is a critical issue for a decision-support system based on an ID, it would be necessary to perform an ad-hoc comparison for that problem.

Redundant variables We have also recorded the cases in which one algorithm included more redundant variables than the other: Table 3.7 shows that in 22.4% of cases (448 out of 2,000) VE included fewer redundant variables than AR, while AR outperformed VE in only 8 cases, i.e., 0.4%. It means that for each case in which AR was superior, there were over 50 cases in which the reverse was the case.

Nodes	Mean	Min	5th perc.	Median	95th perc.	Max
5	0.96	0.51	0.69	1.00	1.05	1.16
6	0.91	0.37	0.61	1.00	1.00	1.00
7	0.88	0.31	0.41	0.99	1.17	1.27
8	1.10	0.34	0.67	1.00	1.72	2.00
9	1.24	0.35	0.56	1.14	2.22	3.97
10	1.43	0.43	0.51	1.14	3.25	4.76
11	1.67	0.39	0.50	1.23	4.27	6.28
12	1.99	0.39	0.56	1.67	4.56	8.13
13	1.76	0.14	0.56	1.32	3.71	11.81
14	2.31	0.27	0.47	1.59	7.00	16.19
15	2.80	0.22	0.45	1.88	8.96	19.49
16	3.14	0.24	0.67	1.97	8.85	33.68
17	3.26	0.27	0.64	1.94	10.49	22.42
18	2.81	0.17	0.39	1.69	8.26	12.03
19	4.02	0.34	0.53	1.91	13.84	51.36
20	3.28	0.25	0.52	1.92	10.14	25.03
21	4.18	0.42	0.68	1.96	14.41	58.86
22	3.74	0.24	0.61	1.88	12.18	58.74
23	4.28	0.33	0.41	1.98	16.25	59.89
24	3.26	0.33	0.56	1.88	11.58	16.01
Total	**2.45**	**0.14**	**0.55**	**1.29**	**7.75**	**59.89**

Table 3.2: Ratio of the maximum storage space required by AR and VE.

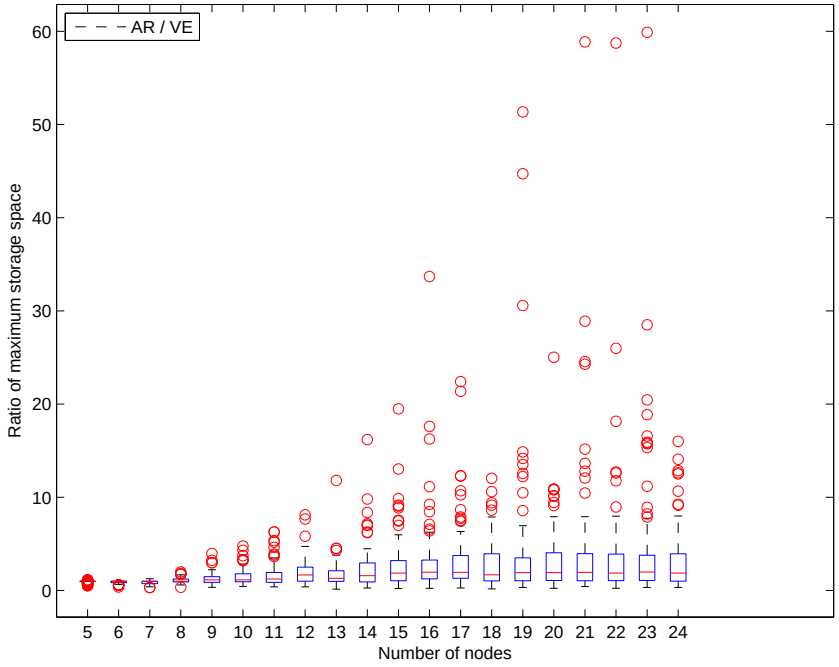

Figure 3.12: Ratio of the maximum storage space required by AR and VE.

Nodes	Mean	Min	5th perc.	Median	95th perc.	Max
5	13.82	3.79	7.16	14.11	19.37	38.86
6	12.35	4.83	7.06	12.35	18.97	28.98
7	11.05	3.21	4.51	11.54	16.80	21.66
8	11.15	2.46	6.20	10.66	19.77	22.69
9	9.15	1.97	4.42	8.63	15.96	21.76
10	9.76	2.90	3.75	9.33	19.19	26.14
11	9.57	1.71	2.69	8.98	19.68	25.46
12	10.40	1.24	2.84	9.72	23.04	29.81
13	9.13	0.76	2.25	6.40	18.25	62.42
14	10.97	0.94	1.86	8.02	30.74	63.37
15	12.09	0.73	2.07	9.70	32.85	53.59
16	11.40	1.48	2.29	7.76	27.46	94.86
17	17.11	0.78	2.11	7.05	89.70	146.80
18	12.72	0.82	1.81	7.30	46.44	97.43
19	13.70	0.78	1.73	5.09	64.95	109.88
20	11.21	0.54	1.33	6.47	44.98	71.09
21	14.47	0.82	1.39	6.09	43.20	339.83
22	12.54	0.94	1.43	5.80	39.20	153.88
23	13.14	0.70	1.21	4.93	59.17	151.10
24	14.19	0.74	1.37	5.79	39.36	277.02
Total	**12.00**	**0.54**	**1.96**	**8.79**	**29.44**	**339.83**

Table 3.3: Ratio of the times required by AR and VE-D.

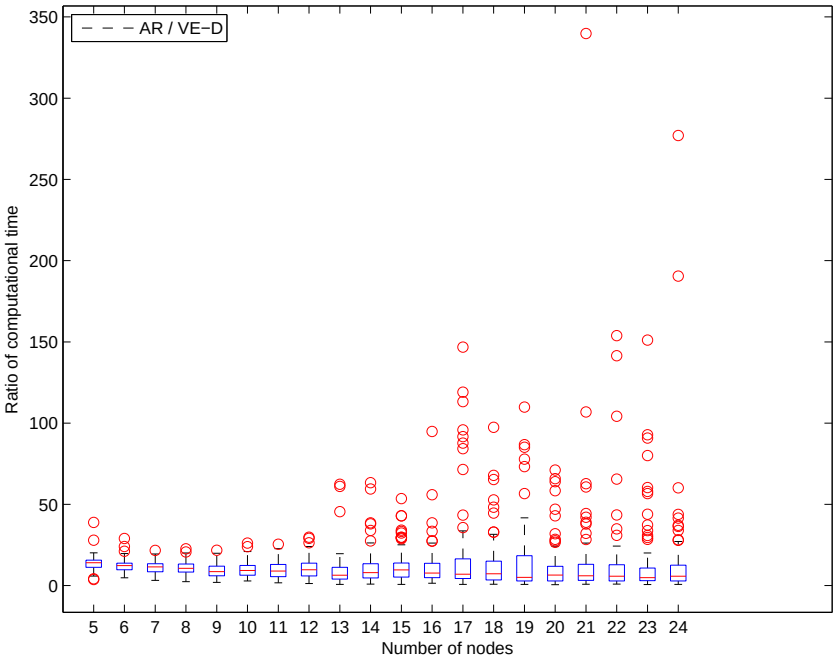

Figure 3.13: Ratio of the times required by AR and VE-D.

Nodes	Mean	Min	5th perc.	Median	95th perc.	Max
5	0.96	0.49	0.70	1.00	1.05	1.16
6	0.91	0.36	0.61	1.00	1.00	1.00
7	0.88	0.30	0.41	1.00	1.16	1.26
8	1.09	0.34	0.66	1.00	1.68	2.03
9	1.23	0.35	0.56	1.13	2.22	3.97
10	1.43	0.44	0.54	1.14	3.26	4.80
11	1.70	0.39	0.50	1.23	4.56	6.28
12	1.95	0.39	0.53	1.60	4.57	8.13
13	1.78	0.14	0.56	1.34	3.70	11.30
14	2.32	0.26	0.47	1.65	6.91	16.25
15	2.85	0.22	0.51	1.92	8.98	19.58
16	3.18	0.27	0.67	1.97	9.81	33.68
17	3.26	0.26	0.57	1.93	10.49	22.48
18	2.84	0.17	0.39	1.71	8.29	12.03
19	3.96	0.08	0.46	1.79	14.22	50.10
20	3.53	0.25	0.51	1.91	11.11	31.48
21	4.00	0.43	0.66	1.95	13.91	40.44
22	3.77	0.23	0.62	1.85	11.68	58.74
23	4.68	0.33	0.41	1.97	17.72	58.20
24	3.66	0.33	0.57	1.87	12.47	48.24
Total	**2.50**	**0.08**	**0.54**	**1.28**	**7.75**	**58.74**

Table 3.4: Ratio of the maximum storage space required by AR and VE-D.

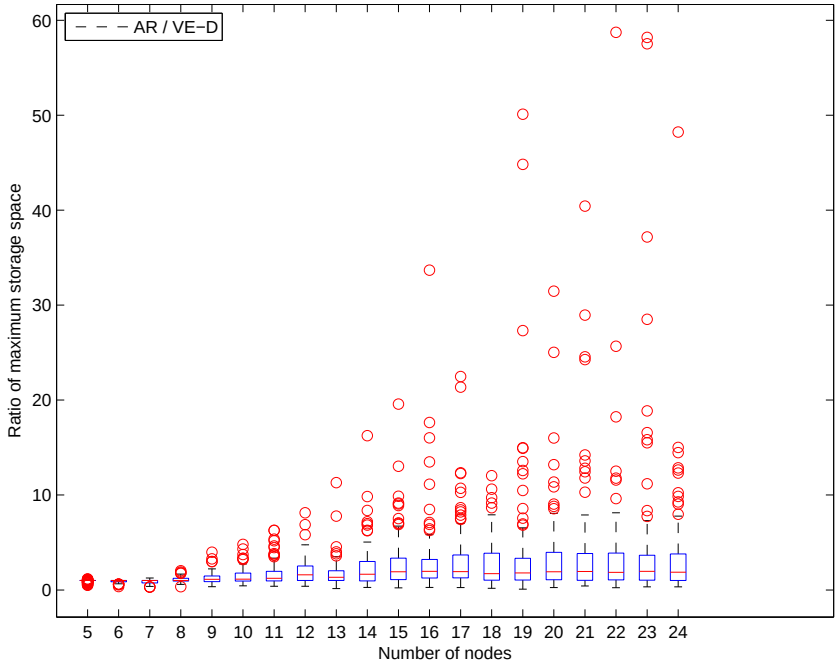

Figure 3.14: Ratio of the maximum storage space required by AR and VE-D.

Nodes	**Mean**	**Min**	**5th perc.**	**Median**	**95th perc.**	**Max**
5	1.00	0.99	0.99	1.00	1.00	1.08
6	1.00	0.99	0.99	1.00	1.00	1.03
7	1.00	0.99	0.99	1.00	1.08	1.11
8	1.01	0.70	0.99	1.00	1.10	1.27
9	1.01	0.99	0.99	1.00	1.10	1.22
10	1.00	0.68	0.96	1.00	1.09	1.19
11	0.99	0.50	0.87	1.00	1.08	1.20
12	1.02	0.49	0.93	1.00	1.22	1.43
13	1.00	0.40	0.74	1.00	1.17	1.44
14	1.00	0.73	0.81	1.00	1.07	1.14
15	0.98	0.30	0.86	1.00	1.08	1.12
16	1.00	0.69	0.76	1.00	1.11	1.21
17	1.05	0.66	0.87	1.00	1.18	4.72
18	0.99	0.34	0.70	1.00	1.18	1.28
19	1.16	0.50	0.86	1.00	1.18	13.65
20	1.01	0.34	0.81	1.00	1.18	1.49
21	1.03	0.49	0.80	1.00	1.15	3.28
22	1.00	0.51	0.76	1.00	1.15	1.48
23	1.07	0.28	0.77	1.00	1.24	5.21
24	1.00	0.16	0.85	1.00	1.17	1.25
Total	**1.02**	**0.16**	**0.89**	**1.00**	**1.12**	**13.65**

Table 3.6: Ratio of the spaces required by VE-D and VE.

	AR	VE	VE-D	*Won*
AR	-	8 (0.4%)	5 (0.25%)	13 (0.33%)
VE	448 (22.4%)	-	10 (0.5%)	458 (11.45%)
VE-D	461 (23.05%)	58 (2.9%)	-	519 (12.98%)
Lost	909 (22.73%)	66 (1.65%)	15 (0.38%)	990 (8.25%)

Table 3.7: Comparison of the number of redundant variables between AR, VE and VE-D. Each cell (i, j) shows how many times the algorithm in the i-th row outperformed the algorithm in the j-th column. For instance, VE returned smaller policies than AR for 448 out of the 2,000 IDs (22.4%), while AR has beaten VE only in 8 cases. The *Won* column indicates how many times each algorithm beat each of the others. The percentages in this column are computed over 2,000×2=4,000 cases, because each algorithm is compared twice for each ID. The interpretation of the *Lost* column is similar. 990 is the number of cases in which there was a winner.

Similarly, VE-D outperformed AR in 461 cases (23%), while the opposite happened only in 5 cases (0.25%); i.e., for each case in which AR returned fewer redundant variables, there were almost 100 cases in which it returned more.

VE-D was also superior to VE: the former performed better in 58 cases (2.9%), while the latter gave better results in 10 cases (0.5%), a difference of almost 6 to 1. Therefore, if avoiding redundant variables is a priority, we should use VE-D or an algorithm for detecting redundant variables. Given that VE and VE-D have a similar efficiency in time and space on average, we recommend VE-D as the default algorithm for evaluating IDs.

Effect of the subset rule

As mentioned in Section 3.4.2, Tatman and Shachter (1990) proposed the *subset rule* (SR) as a heuristic for reducing the storage space required by their arc reversal algorithm, but they did not provide any empirical evidence of such saving of space. On the other hand, the application of the SR entails a computational cost, which might make it counterproductive, as we discussed in Section 3.4.2. For this reason, we thought it would be worthy to test empirically the utility of the SR.

First we analyzed the impact of that rule on the space requirements of AR. (We denote by AR-SR the version of arc reversal that uses the subset rule.) Contrary to the intuition by Tatman and Shachter, in most of the cases the SR did not save any space in general: if we measure the maximum storage space of both algorithms and compute the ratio $s_{\text{AR-SR}}/s_{\text{SR}}$, we see that the mean of ratios is 1.00, up to rounding errors, and the median is exactly 1—cf. Table 3.8. There were cases in which AR-SR saved space, but they were quite infrequent and in general the difference was negligible; only in some exceptional cases the ratio reached values as low as 0.56 or 0.69, which means that the SR saved around half the storage space. More surprisingly, in other cases, also very exceptional, AR-SR required 1.82 or 2.29 times more space than AR. When we compared the computational times, we found slightly bigger differences (cf. Table 3.9): in one case AR-SR was almost twice faster, but in others AR was between 2 and 4 times faster. Given the scarce number of cases in which there was a significant difference and the opposite signs of the differences, the only conclusion that we can draw is that in general the SR has virtually no impact on the performance of the arc reversal algorithm.

We then studied the effect of that rule on our variable elimination algorithm. When we compared VE (without divisions and without the subset rule) with VE-SR (with the subset rule), we found no consistent difference in the maximum storage space (cf. Table 3.10) nor in the computational time (cf. Table 3.11), even thought the differences

Nodes	Mean	Min	5th perc.	Median	95th perc.	Max
5	1.00	0.90	1.00	1.00	1.00	1.00
6	1.00	1.00	1.00	1.00	1.00	1.00
7	1.00	0.90	1.00	1.00	1.00	1.16
8	0.99	0.84	0.94	1.00	1.00	1.00
9	0.99	0.88	0.96	1.00	1.00	1.00
10	1.00	0.95	0.97	1.00	1.00	1.00
11	1.00	0.89	0.98	1.00	1.00	1.00
12	1.00	0.89	0.97	1.00	1.00	1.25
13	1.00	0.96	0.98	1.00	1.00	1.00
14	1.00	0.98	0.99	1.00	1.00	1.00
15	1.00	0.96	0.99	1.00	1.00	1.00
16	1.00	0.98	0.99	1.00	1.00	1.00
17	0.99	0.56	0.99	1.00	1.00	1.00
18	1.00	0.99	1.00	1.00	1.00	1.00
19	1.00	0.69	0.99	1.00	1.00	1.82
20	1.00	1.00	1.00	1.00	1.00	1.00
21	1.00	1.00	1.00	1.00	1.00	1.00
22	1.00	0.99	1.00	1.00	1.00	1.00
23	1.00	0.99	1.00	1.00	1.00	1.00
24	1.01	1.00	1.00	1.00	1.00	2.29
Total	**1.00**	**0.56**	**0.99**	**1.00**	**1.00**	**2.29**

Table 3.8: Ratio of the spaces required by AR-SR and AR.

Nodes	**Mean**	**Min**	**5th perc.**	**Median**	**95th perc.**	**Max**
5	1.03	0.77	0.93	1.00	1.08	3.92
6	1.01	0.45	0.95	1.00	1.09	2.78
7	0.99	0.78	0.92	1.00	1.05	1.24
8	0.99	0.73	0.89	0.99	1.08	1.18
9	1.00	0.87	0.95	1.00	1.07	1.22
10	0.99	0.83	0.93	0.99	1.05	1.15
11	1.00	0.86	0.92	1.00	1.03	2.13
12	1.00	0.88	0.94	1.00	1.04	1.66
13	0.98	0.66	0.91	0.99	1.03	1.14
14	0.99	0.78	0.91	1.00	1.03	1.08
15	0.99	0.87	0.93	1.00	1.02	1.19
16	0.99	0.76	0.87	1.00	1.04	1.34
17	0.98	0.83	0.91	1.00	1.01	1.04
18	0.99	0.78	0.89	1.00	1.04	1.10
19	0.98	0.74	0.90	0.99	1.02	1.29
20	0.98	0.80	0.88	1.00	1.01	1.03
21	0.99	0.83	0.91	1.00	1.00	1.05
22	0.98	0.83	0.91	1.00	1.01	1.05
23	0.98	0.84	0.88	1.00	1.01	1.06
24	0.99	0.75	0.89	1.00	1.01	1.83
Total	**0.99**	**0.45**	**0.91**	**1.00**	**1.03**	**3.92**

Table 3.9: Ratio of the times required by AR-SR and AR.

Nodes	Mean	Min	5th perc.	Median	95th perc.	Max
5	0.97	0.68	0.84	1.00	1.00	1.06
6	0.97	0.73	0.87	1.00	1.00	1.05
7	0.98	0.68	0.89	0.99	1.02	1.28
8	0.97	0.75	0.86	0.98	1.04	1.49
9	0.94	0.50	0.69	0.97	1.00	1.18
10	0.94	0.42	0.70	0.97	1.05	1.19
11	0.94	0.53	0.64	0.98	1.02	1.53
12	0.96	0.54	0.76	0.98	1.02	1.60
13	0.94	0.37	0.67	0.99	1.02	1.61
14	0.97	0.52	0.65	0.99	1.06	3.59
15	0.96	0.39	0.77	0.99	1.00	1.89
16	0.95	0.64	0.68	0.99	1.04	1.92
17	0.96	0.45	0.72	0.99	1.02	1.99
18	0.95	0.40	0.67	0.99	1.10	1.50
19	0.99	0.44	0.61	1.00	1.26	2.30
20	0.95	0.38	0.67	1.00	1.00	1.17
21	0.95	0.39	0.68	1.00	1.00	1.02
22	0.92	0.43	0.53	1.00	1.00	1.59
23	0.96	0.36	0.68	1.00	1.00	1.76
24	0.93	0.52	0.67	1.00	1.00	1.24
Total	**0.95**	**0.36**	**0.68**	**0.99**	**1.01**	**3.59**

Table 3.10: Ratio of the spaces required by VE-SR and VE.

seemed to be slightly higher than in the comparison of AR with AR-SR. We might be tempted to say, after looking at the "mean" columns of those tables, that the subset rule reduces slightly the maximum storage space for large IDs, on average, at the expense of increasing the time of computation, as we expected, but a look at the "median" column makes this conclusion doubtful.

Interestingly, in the case of the VE-D algorithm, the subset rule seems to save both time and space for large IDs, but the average difference is small and the medians show no difference in storage space (Table 3.12) nor in computational time (Table 3.13).

3.6 Discussion

There are several variable elimination algorithms for IDs proposed in the literature (Cowell et al., 1999; Dechter, 1996; Jensen et al., 1994; Shenoy, 1992; Jensen and Nielsen, 2007), but none of them can evaluate IDs with super value nodes. The algorithm that we have presented in this chapter can be seen as an extension of those methods, designed as an

Nodes	Mean	Min	5th perc.	Median	95th perc.	Max
5	0.96	0.52	0.70	0.95	1.26	2.28
6	0.97	0.02	0.77	0.98	1.16	1.54
7	0.97	0.66	0.72	0.99	1.15	1.32
8	1.00	0.70	0.82	0.99	1.20	2.10
9	1.00	0.62	0.71	0.99	1.24	2.91
10	0.99	0.53	0.73	0.99	1.24	1.58
11	1.04	0.35	0.71	1.01	1.52	2.22
12	1.06	0.67	0.81	1.03	1.31	2.36
13	1.00	0.59	0.74	1.00	1.26	1.81
14	1.09	0.53	0.77	1.02	1.33	6.50
15	1.05	0.54	0.74	1.04	1.43	1.94
16	1.04	0.52	0.71	1.02	1.48	2.06
17	1.07	0.64	0.76	1.04	1.48	1.90
18	1.06	0.59	0.78	1.04	1.33	1.85
19	1.09	0.60	0.75	1.04	1.62	2.00
20	1.06	0.39	0.78	1.04	1.40	1.69
21	1.04	0.42	0.79	1.03	1.32	1.73
22	1.05	0.59	0.68	1.04	1.49	2.20
23	1.06	0.43	0.75	1.04	1.34	1.68
24	1.00	0.25	0.71	1.00	1.30	1.80
Total	**1.03**	**0.02**	**0.74**	**1.02**	**1.34**	**6.50**

Table 3.11: Ratio of the times required by VE-SR and VE.

Nodes	**Mean**	**Min**	**5th perc.**	**Median**	**95th perc.**	**Max**
5	0.98	0.69	0.82	1.00	1.00	1.00
6	0.99	0.73	0.91	1.00	1.00	1.00
7	0.98	0.69	0.84	1.00	1.00	1.00
8	0.97	0.75	0.84	1.00	1.00	1.05
9	0.95	0.49	0.71	1.00	1.00	1.00
10	0.94	0.43	0.68	1.00	1.00	1.00
11	0.93	0.51	0.64	1.00	1.00	1.00
12	0.94	0.52	0.72	1.00	1.00	1.00
13	0.93	0.37	0.67	1.00	1.00	1.06
14	0.92	0.42	0.58	0.99	1.00	1.00
15	0.92	0.42	0.61	1.00	1.00	1.00
16	0.94	0.59	0.67	1.00	1.00	1.00
17	0.93	0.30	0.67	1.00	1.00	1.04
18	0.92	0.34	0.60	1.00	1.00	1.00
19	0.93	0.44	0.54	1.00	1.00	1.00
20	0.93	0.46	0.62	1.00	1.00	1.00
21	0.94	0.40	0.69	1.00	1.00	1.00
22	0.89	0.33	0.48	1.00	1.00	1.00
23	0.93	0.34	0.58	1.00	1.00	1.00
24	0.89	0.39	0.59	1.00	1.00	1.00
Total	**0.94**	**0.30**	**0.66**	**1.00**	**1.00**	**1.06**

Table 3.12: Ratio of the spaces required by VE-D-SR and VED.

Nodes	**Mean**	**Min**	**5th perc.**	**Median**	**95th perc.**	**Max**
5	1.11	0.50	0.68	0.95	1.15	18.16
6	0.98	0.60	0.76	0.98	1.22	1.85
7	0.98	0.34	0.74	0.98	1.24	1.99
8	0.99	0.59	0.75	0.98	1.23	1.93
9	0.93	0.57	0.67	0.97	1.08	1.29
10	0.97	0.52	0.72	0.97	1.15	2.53
11	0.96	0.49	0.73	0.97	1.18	1.72
12	0.96	0.40	0.74	0.98	1.13	1.47
13	0.97	0.50	0.74	0.99	1.09	1.93
14	0.95	0.44	0.67	0.99	1.15	1.24
15	0.96	0.52	0.73	0.98	1.12	1.28
16	0.97	0.49	0.70	1.00	1.15	1.76
17	0.97	0.48	0.72	1.00	1.13	1.26
18	0.94	0.50	0.65	0.98	1.13	1.29
19	0.95	0.48	0.60	0.99	1.13	1.26
20	0.97	0.55	0.67	1.00	1.15	1.20
21	0.96	0.44	0.74	1.00	1.14	1.26
22	0.93	0.46	0.65	0.97	1.09	1.26
23	0.96	0.43	0.68	1.00	1.13	1.22
24	0.93	0.52	0.60	0.95	1.09	1.16
Total	**0.97**	**0.34**	**0.70**	**0.99**	**1.14**	**18.16**

Table 3.13: Ratio of the times required by VE-D-SR and VED.

alternative to the arc reversal algorithm by Tatman and Shachter (1990), the only one that could evaluate IDs super value nodes.

A problem of all these algorithms, including ours, is that they occasionally introduce redundant variables—see Section 3.1.3. Several algorithms have been proposed in the literature for detecting structurally redundant variables by analyzing the graph (Faguiouli and Zaffalon, 1998; Nielsen and Jensen, 1999; Nilsson and Lauritzen, 2000; Shachter, 1998; Vomlelova and Jensen, 2004), but none of them can analyze IDs with super value nodes. One of the advantages of the algorithm that we have proposed is that it rarely introduces redundant variables (see the experimental results in Sec 3.5.2). However, in some real-world applications it might be desirable to ensure that the decision-support system does not include any redundant variables at all, and for this reason in a future work we will propose an algorithm for eliminating them in the case of IDs with super value nodes. This algorithm must take into account the distinction between *structurally redundant* and *quasi-structurally redundant* variables (cf. Sec. 3.1.3), which is one of the contributions of this paper.

Another element of crucial importance for the efficiency of algorithms for IDs (as in the case of Bayesian networks) is finding an efficient elimination order. In the first experiments that we carried out, our algorithm randomly selected the elimination order inside each $\mathbf{C}_i$ (we should recall that $\mathbf{C}_i$ is the set of variables unknown for decision D_i and known for D_{i+1}). In these experiments our algorithm was faster than AR in general, but it usually required more memory. Thus we realized that one of the advantages of AR is that it automatically detects sink nodes, i.e., nodes having no outgoing arcs towards chance or decision nodes (Olmsted, 1983; Shachter, 1986)[9]. When we forced our VE algorithm to use the same elimination order as AR, VE was able to outperform AR not only in time efficiency, but also by requiring less storage space, as shown in Section 3.5.2.[10]

However, it might be desirable to have a method for finding the optimal elimination order for VE, which is not necessarily optimal for AR. However, given that finding the optimal elimination ordering for a Bayesian network is NP-complete, we conjecture that finding the optimal elimination order for VE is also NP-complete. For this reason, we should concentrate our efforts on developing heuristics that return near-optimal orderings.

[9]AR takes profit of sink nodes by eliminating them without performing any numerical computation. In turn, VE can take profit of them because they lead to unity potentials—see Sec. 3.4.3.

[10]Fortunately, it is possible to feed our algorithm with the elimination order used by AR without the necessity of completely executing the algorithm by Tatman and Shachter: it suffices to reverse arcs and delete nodes from the graph without performing any numerical computation. Given that these operations only focus on the neighbors of each node, the cost of obtaining the elimination order is absolutely negligible compared to the cost of operating with potentials.

This is a very difficult issue even when the ID has only one utility node (Gómez and Bielza, 2004), and becomes much more complex when the utility function is given by an ADG of potentials, since the basic operations of our algorithm (distribution and variable elimination) treat sum nodes very differently to product nodes, and chance variables differently to decision variables. A solution to this problem might be to examine different orderings, as in (Gómez and Bielza, 2004): we can make a qualitative evaluation of the ID by operating on the ADGoP, without doing any numerical computation, in order to know the sizes of the potentials involved and to estimate the time and memory space required by a candidate elimination order.

There is another line of improvement for our algorithm. The reason why VE and VE-D do not always perform better than AR is that, even though they try to preserve the separability of the utility function when eliminating a variable, sometimes the subsequent elimination of other variables merges the potentials that we wanted to keep separate. In this way some distributions become counterproductive, first because they increase the storage space and second because they prevent the algorithm from detecting common factors, as shown in Section 3.2.2. Therefore, the methods *unfork* and *distribute*, which in the current version of the algorithm only focus on the variable to be eliminated, should be refined in order to take into account the effect of the next eliminations.

We might try to solve both problems at the same time: we might assess the cost of different elimination orderings and different distribution strategies—also analyzing the number of redundant variables introduced by each one of them—and then perform the numerical computations for the optimal combination. However, it would be necessary to prove empirically that the time spent on the qualitative evaluation of several possibilities is compensated by finding a more efficient path for the evaluation of the ID.

Finally, it would be interesting to research how the ideas of lazy evaluation for IDs without super value nodes introduced by Madsen and Jensen (1999) and extended in (Vomlelova and Jensen, 2004) could be integrated with our algorithm and applied to more general IDs. This might avoid unnecessary multiplications and subsequent divisions, and also avoid redundant variables in the policies.

As conclusion, as we said at the beginning, we wanted to develop a variable-elimination (VE) algorithm for IDs with super value nodes having five advantages over Tatman and Shachter's (Tatman and Shachter, 1990) arc reversal (AR) algorithm: it were faster, required less memory, avoided redundant variables, simplified sensitivity analysis, and could save time and memory space for IDs containing canonical models.

We conducted some experiments to discover whether we achieved the first three ob-

jectives. The analysis of 2,000 IDs randomly generated shows, in the first place, that on average VE is around 10 times faster than AR, especially for large IDs and in 5% of cases it is at least 30 times faster; for some IDs, it was between 100 and 340 times faster. In contrast, the cases in which AR is faster than VE are infrequent and the differences are much smaller: for IDs having more than 6 nodes, AR could never be twice as fast as VE.

In the second place, AR requires on average 3 or 4 times more memory space than VE, with a median ratio of about 2. For 5% of the IDs, AR needs at least 10 times more space. In several cases, it needed between 20 and 60 times more space. On the contrary, the cases in which VE required more memory are infrequent and the differences are much smaller.

Third, our experiments showed that for each case in which AR introduces fewer redundant variables than VE, there are over 50 in which VE gives better results. A version of VE with division of probability potentials (VE-D) performs even better: for each case in which AR introduces fewer redundant variables, there were almost 100 in which VE-D outperformed AR. Given that the time and space efficiency of VE-D is very similar to that of VE, we recommend VE-D as the default algorithm for evaluating IDs with super-value nodes.[11] One weakness of VE is that it does not compute the expected utility of each decision option, while AR and VE-D do. Given that in general, human decision makers are interested in knowing such expected utilities, VE is only useful for autonomous decision systems, not for decision-support systems.

Forth, our algorithm can simplify sensitivity analysis by keeping track of which potentials have been involved in the computation of the (new) potentials on which maximizations are performed —see Section 3.2.2.

With respect to the fifth objective, we have not tested empirically the time and space savings that our variable elimination algorithm can provide for IDs containing canonical models, such as the noisy OR/MAX (Díez and Druzdzel, 2006). However, the important savings obtained by the integration of variable elimination and canonical models in the case of Bayesian networks (see Section 3.1.1 and (Díez and Galán, 2003)) indicate that similar savings might be obtained for IDs.

In summary, we conclude that we have attained, to different degrees, the five objectives set at the beginning of our research.

A minor contribution of this chapter is the empirical evaluation of the *subset rule* (Tatman and Shachter, 1990), which—as far as we know—had never been tested before. Our experiments have shown that for most IDs the impact of this rule is null or almost

[11] This was an unexpected conclusion because, as we stated in the introduction, the motivation for developing a variable elimination algorithm was to avoid the division of potentials.

null and, contrary to our expectations, it can either increase or decrease the time and space spent by the algorithms (see Section 3.5.2).

Chapter 4

Explanation of the reasoning in influence diagrams

Bayesian networks and influence diagrams are probabilistic graphical models widely used for building diagnosis- and decision-support expert systems. Explanation of both the model and the reasoning is important for debugging these models, for alleviating users' reluctance to accept their advice, and for using them as tutoring systems. Influence diagrams have demonstrated to be effective tools for building decision support systems, but the explanation in that formalism is hard because inference in probabilistic graphical models have no relation with human reasoning. We are going to describe some explanation options for influence diagrams that we have implemented. Some of them have been programmed in Elvira and are available through its GUI.

4.1 Introduction

Influence diagrams (IDs) are a probabilistic graphical model for representing and solving Bayesian decision problems (see Section 2.3 for a description about IDs).

In the context of expert systems, either probabilistic or heuristic, the development of explanation facilities is important for three main reasons (Lacave and Díez, 2002; Lacave, 2003). First, because the construction of those systems with the help of human experts is a difficult and time-consuming task, prone to errors and omissions. An explanation tool can help the experts and the knowledge engineers taking part in the project to *debug* the system when it does not yield the expected results and even before a malfunction occurs. Second, because human beings are reluctant to *accept* the advice offered by a machine if they are not able to understand how the system arrived at those recommendations; this reluctance is especially clear in medicine (Wallis and Shortliffe, 1984). And third, because an expert system used as an intelligent *tutor* must be able to communicate the apprentice

the knowledge it contains, the way in which the knowledge has been applied for arriving at a conclusion, and what would have happened if the user had introduced different pieces of evidence (what-if reasoning).

These reasons are especially relevant in the case of probabilistic expert systems, because the elicitation of probabilities is more difficult than the assessment of uncertainty in heuristic expert systems and because, even though probabilistic reasoning is just a formalization of (a part of) common-sense reasoning, the algorithms for the computation of probabilities and utilities are very different from the way a human being would draw conclusions from a probabilistic model.

Unfortunately, the explanation methods proposed so far are still unsatisfactory, as shown by the fact that most expert systems and commercial tools available today, either heuristic or probabilistic, have virtually no explanation capability (Lacave and Díez, 2002, 2004). Despite the practical interest of this issue, very little research is currently carried out about explanation in probabilistic graphical models. As an attempt to palliate this shortcoming, in this paper we describe some methods for explaining both the model and the reasoning of probabilistic expert systems as influence diagrams, which have been implemented in Elvira, a public software tool developed as a joint project of several Spanish universities. We also discuss how such methods respond to the needs that we have detected when building and debugging medical expert systems (Díez et al., 1997; Lacave and Díez, 2003; Luque et al., 2005) and when teaching probabilistic graphical models to pre- and postgraduate students of computer science and medicine Díez (2004). Section 4.3 describes the explanation methods for IDs in Elvira, and its content is based on (Luque and Díez, 2006) and (Lacave et al., 2007).

The main output of an evaluation algorithm for IDs is the optimal strategy. Software tools like Elvira usually present each optimal policy in the form a policy table, which contains a column for each configuration of the informational predecessors of the decision. In some cases the policy tables can be huge. For example, the policy table for the ID of the mediastinal staging of non-small cell lung cancer has 15552 columns. We have felt the need of having a alternative representation to policy tables that could summarize the optimal policy. In Section 4.4 we describe the representation that we have proposed for that purpose.

Influence diagrams can contain numerical parameters that are not known with certainty. After evaluating an influence diagram, the decision analyst investigates whether these results obtained (the optimal strategy and the maximum expected utility) depend on (are sensitive) to the uncertainty about the parameters of the model. The investiga-

tion of the variations of parameters in PGMs is performed by using sensitivity analysis techniques (see Section 2.7). In the context of IDs with SV nodes we needed to have implemented algorithms of sensitivity analysis for using it with an influence diagram of mediastinal staging of non-small cell lung cancer. Section 4.5 presents a description of the algorithms implemented.

4.2 Preliminaries

We explain here the conversion of an influence diagram (ID) into a Bayesian network (BN), which constitutes the basis of explanation capabilities of influence influence diagrams presented in this chapter. A description of Bayesian networks and influence diagrams were exposed in Sections 2.2 and 2.3.

If we have a strategy $\Delta = \{P_D | D \in \mathbf{V}_D\}$ for an ID, it can be used to convert the ID into a BN, called *Cooper policy network* (CPN), as follows: each decision D is replaced by a chance node with probability potential P_D and parents $IPred(D)$, and each utility node U is converted into a chance node whose parents are its functional predecessors, $FPred(U)$—see Fig. 4.2. The values of each new chance variable U are $\{+u, \neg u\}$ and its probability is $P_{CPN}(+u|fPred(U)) = norm_U(U(fPred(U)))$, where $norm_U$ is a linear transformation that maps the utilities $U(fPred(U))$ from the interval $[\alpha_U, \beta_U]$ onto the interval $[0, 1]$ (Cooper, 1988), where α_U and β_U are defined as:

$$\alpha_U = \min_{fPred(U)} \psi_U(fPred(U)) \tag{4.1}$$

$$\beta_U = \max_{fPred(U)} \psi_U(fPred(U)) \ . \tag{4.2}$$

The joint distribution of the CPN is:

$$\begin{aligned} &P_{CPN}(\mathbf{v}_C, \mathbf{v}_D, \mathbf{v}_U) \\ &\quad = P_\Delta(\mathbf{v}_C, \mathbf{v}_D) \prod_{U \in \mathbf{V}_U} P_U(u|pa(U)) \end{aligned} \tag{4.3}$$

Given two configurations $\mathbf{r}$ and $\mathbf{r}'$defined over two set of variables, $\mathbf{R} \subseteq \mathbf{V}_C \cup \mathbf{V}_D$ and $\mathbf{R}' \subseteq (\mathbf{V}_C \cup \mathbf{V}_D)$, such that $\mathbf{R} \cap \mathbf{R}' = \varnothing$ and $P(\mathbf{r}) \neq 0$, and U a utility node, it is easy to prove that

$$P_\Delta(\mathbf{r}') = P_{CPN}(\mathbf{r}') \tag{4.4}$$

$$P_\Delta(\mathbf{r}'|\mathbf{r}) = P_{CPN}(\mathbf{r}'|\mathbf{r}) \tag{4.5}$$

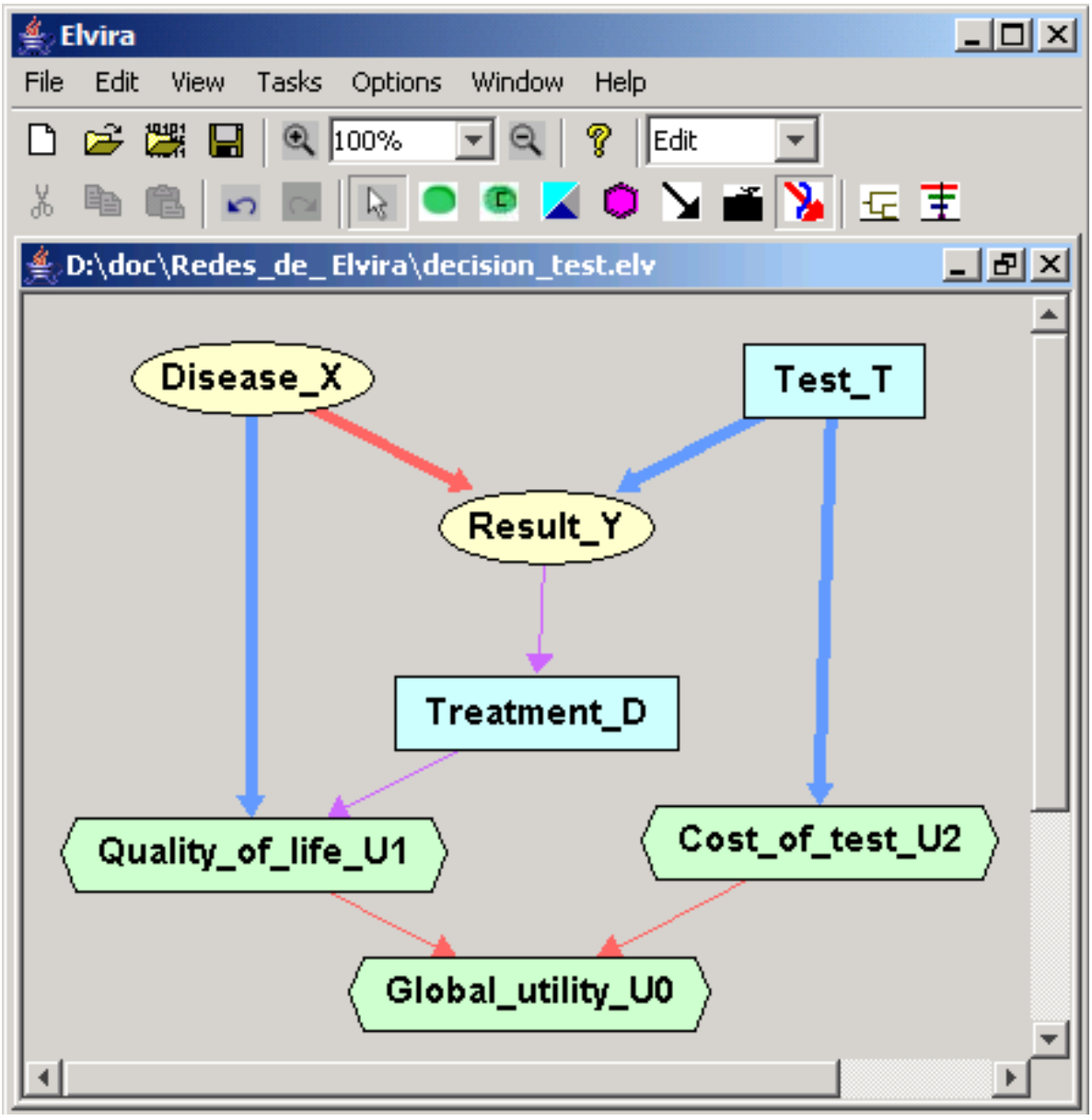

Figure 4.1: ID with two decisions (rectangles), two chance nodes (ovals) and three utility nodes (hexagons). Please note that there is a directed path T–Y–D–U_1–U_0 including all the decisions and the global utility node U_0.

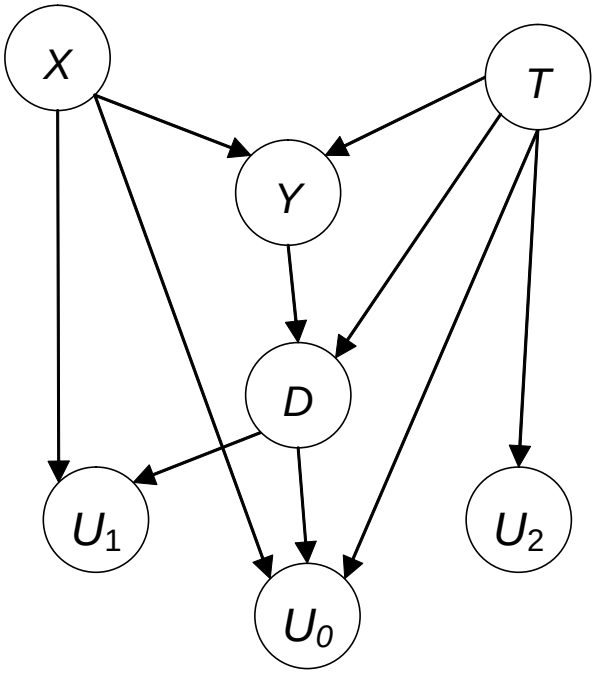

Figure 4.2: Cooper policy network (CPN) for the ID in Figure 4.1. Please note the addition of the non-forgetting link $T \rightarrow D$ and that the parents of node U_0 are no longer U_1 and U_2 but $FPred(U_0) = \{X, D, T\}$, which were chance or decision nodes in the ID.

$$EU_U(\Delta) = norm_U^{-1}(P_{CPN}(+u)) \tag{4.6}$$

$$EU_U(\Delta, \mathbf{r}) = norm_U^{-1}(P_{CPN}(+u|\mathbf{r})) \tag{4.7}$$

In Section **??** we will use these equations to compute on a CPN the probabilities and expected utilities to be displayed in the GUI.

4.3 Explanation of influence diagrams in Elvira

Elvira[1] is a tool for building and evaluating probabilistic graphical models (Elvira Consortium, 2002). A brief description of Elvira has been exposed in Section 2.8. We describe in this section the explanation capabilities for influence diagrams that have been implemented in Elvira.

[1] At `http://www.ia.uned.es/~elvira` it is possible to obtain the source code and several technical documents about Elvira.

4.3.1 Explanation of the model

The explanation of IDs in Elvira is based, to a great extent, on the methods developed for explanation of BNs. One of the methods that have proven to be more useful is the automatic colorings of links. The definitions in (Lacave et al., 2006) for the sign of influence and magnitude of influence, inspired on (Wellman, 1990), have been adapted to ordinary utility nodes as follows:

Definition 4.3.1 *Let U be an ordinary utility node having $\alpha_U \neq \beta_U$ (see Equations 4.1 and 4.2) and $Pa(U) = \{A\} \cup \mathbf{B}$. The* magnitude of the influence *(MI) for the link $A \rightarrow U$ is*

$$MI(A,U) = norm_U(\max_{a,\mathbf{b}} |\psi_U(a,\mathbf{b}) - \psi_U(a_0,\mathbf{b})|) \tag{4.8}$$

We say that A positively influences *variable U iff $MI(A,U) \neq 0$ and*

$$\forall a, \forall a', \forall \mathbf{b}, a > a' \Longrightarrow \psi_U(a,\mathbf{b}) \geq \psi_U(a',\mathbf{b}) \tag{4.9}$$

We also say that the link is positive.

The definitions of *negative influence* and *negative link* are analogous. When $MI(A,U) = 0$ the influence of link $A \rightarrow U$ is said to be *null*; in that case, link $A \rightarrow U$ should be removed. When the influence is neither positive nor negative nor null, then it is said to be *undefined*.

The *magnitude of the influence* for a link $A \rightarrow U$ gives a relative measure of how much variable A influences a utility node U. Then, the influence of a link $A \rightarrow U$ is positive when the higher the value of A the higher the utility of U. Cooper's transformation $norm_U$ guarantees that the magnitude of the influence is normalized.

For instance, in Fig. 4.1 the link $X \rightarrow Y$ is colored in red because it represents a positive influence: the presence of the disease increases the probability of a positive result of the test. The link $X \rightarrow U_1$ is colored in blue because it represents a negative influence: the disease decreases the expected quality of life. The link $D \rightarrow U_1$ is colored in purple because its influence is undefined: the treatment is beneficial for patients suffering from X but detrimental for healthy patients.

Additionally, when a policy has been assigned to a decision node, either by calculating its optimal policy or by imposing a policy (see Sec. 4.3.4), the corresponding probability distribution P_D can be used for coloring the incoming links of that node, as shown in Figure 4.1.

The coloring of links in Elvira has been very useful for debugging IDs, by detecting probability and utility tables whose numerical values do not agree with the qualitative influences assessed by the expert.

4.3.2 Displaying the results of inference

In Section 4.2 we have seen that, given a strategy Δ, an ID can be converted into a Cooper policy network (CPN), which is a true Bayesian network. Consequently, all the explanation capabilities for BNs are also available for IDs by exploiting such transformation.

The information displayed for nodes depends on the kind of node—see Fig. 4.3. Chance and decision nodes display bars and numbers corresponding to the probabilities of their states, $P_\Delta(v)$, a marginal probability of $P_\Delta(\mathbf{v}_C, \mathbf{v}_D)$, defined by Equation 4.4. $P_\Delta(v)$ is the probability that a chance variable V takes a certain value v, or the probability that the decision maker chooses option v for decision V (Nilsson and Jensen, 1998). $P_\Delta(v)$ can be computed on the Cooper policy network (CPN) by means of Equation 4.4. Each utility node U displays the expected utility $EU_U(\Delta)$, defined by Equation 2.9, which is computed by propagating on the CPN and transforming back with the use of Equation 4.6. The guide bar (black line) indicates the range of the utilities.

Links pointing into a decision node D are drawn with the color and thickness indicated in Section 4.3.1, by examining the policy P_D (returned by the evaluation of the ID) as if it were the conditional probability table of a chance node. Non-forgetting links added during the evaluation of the diagram (Olmsted, 1983; Shachter, 1986), such as link $T \rightarrow D$ in Fig. 4.3, are drawn as a discontinuous arrows.

Elvira, as most software tools for IDs, can show the utility table associated to each decision. For instance, in Table 4.1 each column corresponds to a configuration (t, y) of the informational predecessors of decision D and each cell contains the expected utility of option d given t and y provided that every future decision will be made optimally $EU(d|iPred(d)) = EU(d|t, y)$. The order of the variables in *IPred(D)* in that table is chosen to make it compatible with the partial order induced by the ID, i.e., the order in which the observations and decisions are known by the decision maker during the decision process.

This table is used by the evaluation algorithm to compute the optimal policy; in this example, $d_{opt} = \arg\max_d EU(d|t, y)$, as shown in Table 4.2. A toggle allows the user to view either the expected utilities for a decision (Table 4.1) or the optimal policy (Table 4.2).

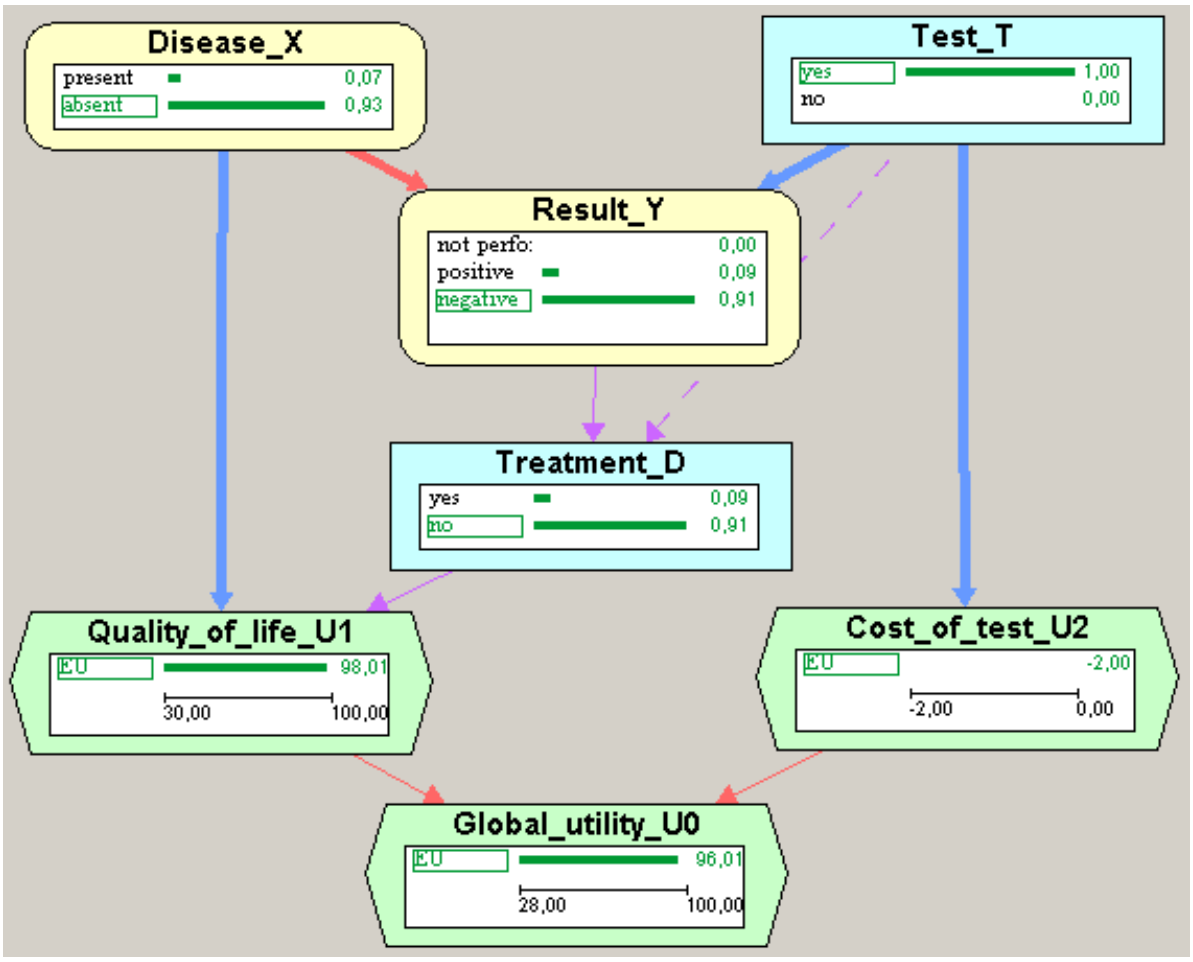

Figure 4.3: ID resulting from the evaluation of the ID in Figure 4.1. It shows the probability $P_\Delta(v)$ of each chance and decision node and the expected utilities.

Table 4.1: Expected utilities for decision D in the ID in Figure 4.1.

Test_T	...	yes	yes	no	no	no
Result_Y	...	positive	negative	not performed	...	...
yes	...	81.05	87.93	89.3	...	...
no	...	49.32	97.51	95.1	...	...

The highest utility in each column is highlighted in red. We have contracted the columns that represent impossible scenarios, i.e., configurations such that $P(iPred(D))$=0.

Table 4.2: Optimal policy for decision D in the ID in Figure 4.1.

Test_T	...	yes	yes	no	no	no
Result_Y	...	positive	negative	not performed	...	...
Treatment_D	...	yes	no	no	...	...

4.3.3 Introduction of evidence

Elvira's ability to manage several evidence cases simultaneously in BNs is also available for IDs, as shown in Fig. 4.4. The evidence is introduced in the ID by using its corresponding Cooper policy network. Given evidence $\mathbf{e}$, Elvira displays for each chance and decision node V the probability $P_\Delta(v|\mathbf{e})$ (cf. Eqs. 2.5 and 4.5), and for each utility node U the expected utility $EU_U(\Delta, \mathbf{e})$ (cf. Eqs. 2.7 and 4.7).

Clarifying the concept of evidence in influence diagrams

In order to avoid confusions, we must mention that the meaning of evidence in Elvira is very different from its meaning in some methods oriented to the computation of the value of information in IDs, such as (Shachter and Peot, 1992; Dittmer and Jensen, 1997; Ezawa, 1998). For those methods, the introduction of evidence $\mathbf{e}$ leads to a different decision problem in which the values of the variables in $\mathbf{E}$ would be known with certainty before making any decision. For instance, introducing evidence $\{+x\}$ in the ID in Fig. 4.1 would imply that X would be known when making decisions T and D. Therefore, the expected utility of the new decision problem, which can be called "Ezawa's scenario" (Ezawa, 1998), would be

$$\max_t \sum_y \max_d P(y|{+x}:t,d) \cdot \underbrace{(U_1(+x,d)+U_2(t))}_{U_0(+x,d,t)}$$

where $P(y|{+x}:t,d) = P(+x,y:t,d)/P(+x:t,d) = P(+x,y:t)/P(+x) = P(y|\ +x:t)$. In spite of the apparent similarity of this expression with Equation 2.14, the optimal strategy changes significantly from "test, and treat only if the result is positive" to "always

treat, without testing", because if we knew with certainty that the disease X is present the result of the test would be irrelevant. The MEU for the new decision problem would be $U_0(+x, +d, \neg t) = U_1(+x, +d)$. In contrast, the introduction of evidence in Elvira does not lead to a new decision scenario nor to a different strategy, since the strategy is determined *before* introducing the "evidence". Put another way, when introducing evidence in Elvira we adopt the point of view of an external observer of a system that includes the decision maker as one of its components. The probabilities and expected utilities given by Equations 2.6 and 2.7 are those corresponding to the subpopulation indicated by $\mathbf{e}$ when the decision maker applies strategy Δ. For instance, given the evidence $\{+x\}$, the probability $P_\Delta(+t|+x)$ shown by Elvira is the probability that a patient suffering from X receives the test, which is 100% (it was 0% in Ezawa's scenario), and $P_\Delta(+d|+x)$ is the probability that he receives the treatment; contrary to Ezawa's scenario, this probability may differ from 100% because of false negatives. The expected utility for a patient suffering from X isA second difference is that Elvira admits the possibility of analyzing non-optimal strategies, as we will see below. And the third significant difference is that the evidence introduced in Elvira may include "findings" for decision variables. For instance, $\mathbf{e} = \{+d\}$ would represent the subpopulation of patients who have received therapy, and $P_\Delta(+x|+d)$ is the probability that a patient receiving therapy has disease X. We must stress that the two approaches are not rivals. They correspond to different points of view when considering evidence in IDs and can complement each other in order to perform a better decision analysis and to explain the reasoning. We have implemented first the options that, in our opinion, can be more useful, but in the future we will implement as well Ezawa's method and the possibility of computing the expected value of perfect information (EVPI).

Example

Fig. 4.4 shows two evidence cases. In this example, Δ is the optimal strategy obtained when evaluating the ID, because no policy was imposed by the user. The first evidence case in Fig. 4.4 is the *prior case*, which was also displayed in Fig. 4.3. Its probabilities and expected utilities are those of the general population. The second evidence case is given by $\mathbf{e} = \{+y\}$; i.e., it displays the probabilities and utilities of the subpopulation of patients in which the test has given a positive result. Node Y is colored in gray to highlight the fact that there is evidence about it. The probability $P_\Delta(+x|+y)$, represented by a red bar, is 0.70; the green bar close to it represents the probability of $+x$ for the prior case, i.e., $P_\Delta(+x)$, which equals $P(+x)$ because the decision maker's actions do

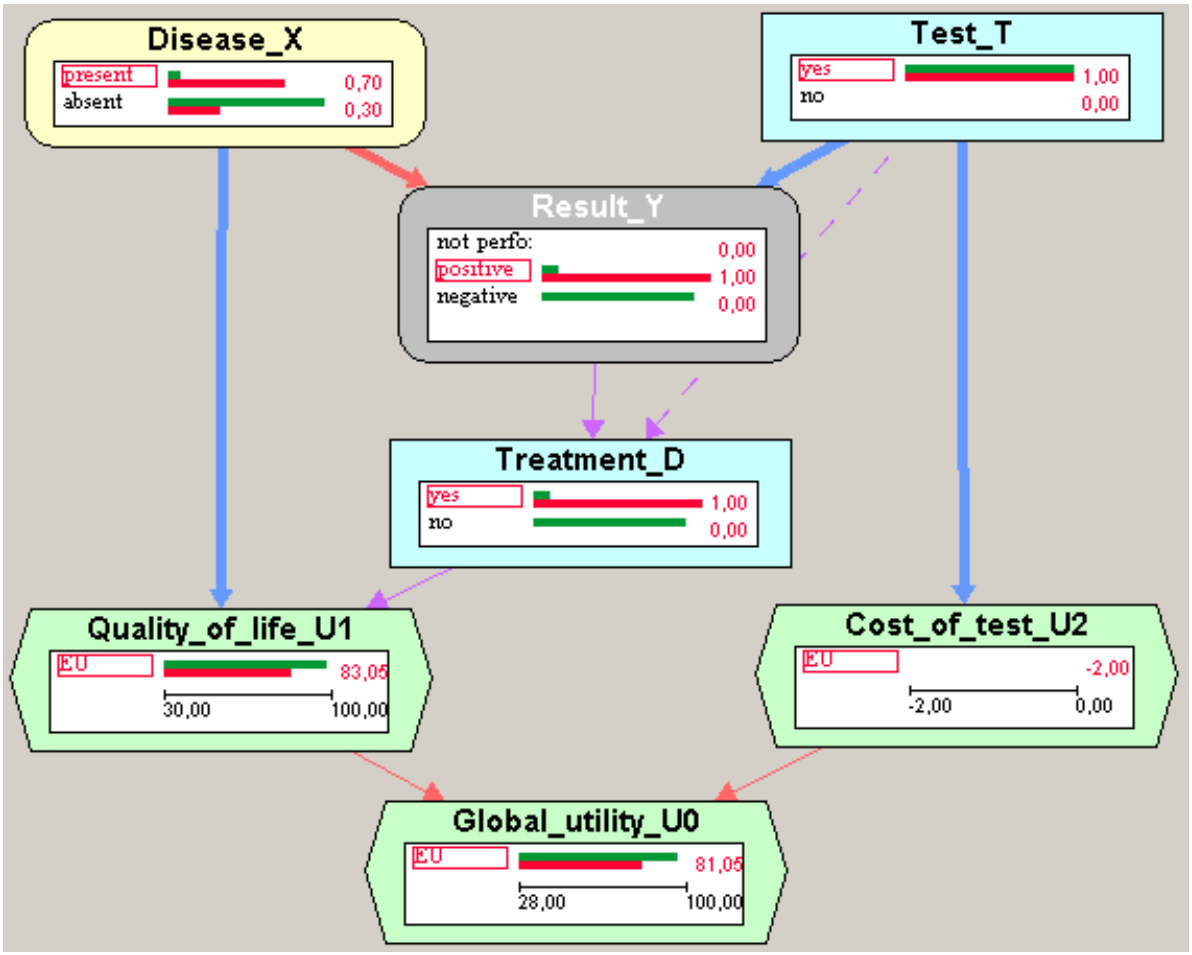

Figure 4.4: ID resulting from the evaluation of the ID in Figure 4.1. It shows two evidence cases: the prior case (no evidence) and the case in which $\mathbf{e} = \{+y\}$.

not affect X. The red bar is longer than the green one because $P_\Delta(+x|+y) > P_\Delta(+x)$, as it was expected from the fact that link $X \rightarrow Y$ is positive. The global utility for the second evidence case, $EU(\Delta, \{+y\})$, represented by a red bar in node U_0, is smaller than $EU(\Delta, \varnothing)$, the expected utility for the general population, represented by a green bar, because the presence of the symptom worsens the prognosis. The red bar for Treatment=yes, which represents $P_\Delta(+d|+y)$, is 1.00 because the optimal strategy determines that all symptomatic patients must be treated. Similarly, $P_\Delta(+t|+y) = 1.00$ because a positive result of the test implies that the test has been done.

Debugging influence diagrams by introducing evidence

The possibility of introducing evidence in Elvira has been useful for building IDs in medicine (Luque et al., 2005): before having this explanation facility, when we were interested in computing the posterior probability of a certain diagnosis given a set of

findings, we needed to manually convert the ID into a BN by removing decision and utility nodes. Each time the ID was modified, even slightly, we had to repeat this conversion, which was tedious and time consuming. (When building medical expert systems, the time of interaction with the experts is a precious resource that must not be waisted.) This was the reason for implementing a facility that allowed us to compute the probabilities directly on the ID, which is much more convenient.

4.3.4 What-if reasoning: analysis of non-optimal strategies

In Elvira it is possible to have a strategy in which some of the policies are imposed by the user and the others are computed by maximization. The way of imposing a policy consists in setting a probability distribution P_D for the corresponding decision D by means of Elvira's GUI; the process is identical to editing the conditional probability table of a chance node. In fact, such a decision will be treated by the inference algorithms as if it were a chance node, and the maximization will be performed only on the rest of the decision nodes.

This way, in addition to computing the optimal strategy (when the user has imposed no policy), as any other software tool for IDs, Elvira also permits to analyze how the expected utilities and the rest of the policies would vary if the decision maker chose a non-optimal policy for some of the decisions (what-if reasoning).

The reason for implementing this explanation facility is that when we were building a certain medical influence diagram (Luque et al., 2005) our expert wondered why the model recommended not to perform a certain test. We wished to compute the a posteriori probability of the disease given a positive result in the test, $P_\Delta(+x|+y)$, but we could not introduce this "evidence", because it was incompatible with the optimal policy (not to test): $P_\Delta(+y) = 0$. After we implemented the possibility of imposing non-optimal policies (in this case, performing the test) we could see that the posterior probability of the disease remained below the treatment threshold even after a positive result in the test, and given that the result of the test would be irrelevant, it was not worthy to do it.

4.3.5 Utility plots as a way of explanation

By introducing evidence about Y in the ID of Figure 4.3 we can see that $P(+x|+y) = 0.83$; this means that the prevalence of X in the subpopulation $\{+y\}$ is 0.83, which is above the 0.17 threshold that can be seen in the utility plot of Figure 2.18. In contrast, $P(+x|\neg y) = 0.015 < 0.17$. This explains why the optimal policy for D is to treat only

TBNA	positive	positive	positive	positive	negative	negative	negative	negative	no_result	no_result	no_result	no_result
Decision_TBNA	yes	yes	no	no	yes	yes	no	no	yes	yes	no	no
CT_scan	positive	negative	positive	negative	positive	negative	positive	negative	positive	negative	positive	negative
Decision_PET	no	no	yes	yes	no	no	yes	yes	yes	yes	no	no

Figure 4.5: Policy table for *Decision_PET* in Elvira.

after a positive result of the test. In the construction of more complex IDs this kind of analysis done with the help of utility plots has been useful for understanding why some tests are necessary or not, and why sometimes the result of a test is irrelevant, as discussed in the previous section. Thus, utility plots, which were described in Section 2.7, have been an important explanation tool.

4.4 Policy tables and policy trees

The optimal strategy of an influence diagram, consisting of an optimal policy for each decision, can be obtained by evaluating it with Elvira. For example, Figure 4.5 displays optimal policy for the node *Decision_PET* of ID of Figure 6.9, in the form of a *policy table*, which contains a column for each configuration of its informational predecessors (see Section 2.3). The bottom cell in each column displays the optimal decision, i.e. either to do the PET or not.

However, given that the size of the policy tables grow exponentially with the number of informational predecessors, we prefer to present the optimal policy of each decision in the form of a *policy tree* (see Figures 6.10 to 6.11)[2] , which is much more understandable. A policy tree (PT) is similar to decision trees (Raiffa and Schlaifer, 1961): it consists of chance and decision nodes, and arcs labeled with the states of the nodes (see an example of PT for *Decision_PET* in Figure 6.10). As in decision trees, the ancestors of a decision node in the PT are informational predecessors in the ID. The leaves of the PT indicate the optimal decision of the corresponding scenario. In contrast with decision trees, a PT only represent scenarios that are possible by following the optimal strategy. This reduces enormously the size of the representation and makes it more understandable for the human expert. For example, the policy table for decision *Treatment* of MEDIASTINET has 15552 columns, which is the size of the state space of the informational predecessors of *Treatment*. In contrast, the PT of *Treatment* in MEDIASTINET disregarding costs has 9 leaves (see Figure 6.10), and the PT when considering costs has only 5 leaves (see Figure 6.11).

[2]The names of some variables of MEDIASTINET have been abbreviated in the figures to have a more compact representation of the trees.

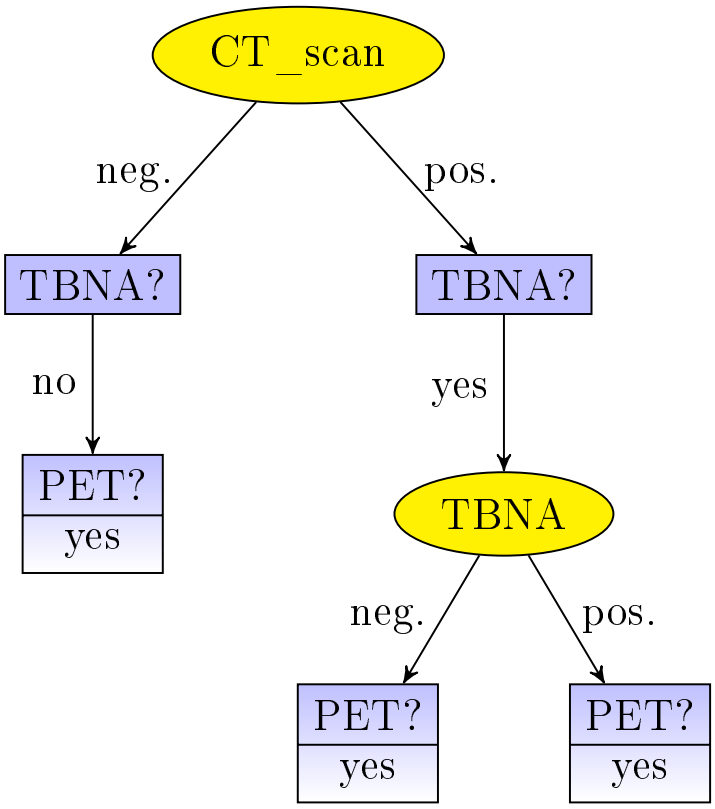

Figure 4.6: A policy tree example.

4.5 Sensitivity analysis

Section 2.7 describes the basic concepts of SA in probabilistic graphical models. We describe in this section some algorithms for performing SA in influence diagrams with super-value nodes. In particular, we first define a new measure of sensitivity analysis for influence diagrams (Section 4.5.1). We also present how to built an auxiliary influence diagram (Section 4.5.2) that will be used for computing three measures of sensitivity analysis:

- the policy change thresholds (see Section 2.7.1),
- the expected value of perfect information (see Section 2.7.2),
- and the sensitivity of each decision to each parameter (see Section 4.5.1).

4.5.1 Sensitivity of a decision to a parameter

We want to analyze the changes in the optimal policies when varying a parameter. We have felt the need of defining the next measure for this purpose.

The *sensitivity of a decision D_i to a parameter* Θ, which will be denoted as $sens(D_i, \Theta)$, can be calculated by defining a Boolean variable, C_i, which indicates if the uncertainty associated to a parameter can change the optimal policy of D_i. Thus, this metrics of sensitivity is defined as:

$$\begin{aligned} \text{sens}(D_i, \Theta) &= P(+c_i) && (4.10)\\ &= \int_{\Theta} P(+c_i|\theta)\ P(\theta)\ d\theta && (4.11)\\ &= \int_{\Theta} \sum_{\mathbf{x}} P(+c_i|\mathbf{x}, \theta)\ P(\mathbf{x}|\theta)\ P(\theta)\ d\theta && (4.12)\\ &= \int_{\Theta} \sum_{\mathbf{x}} \sum_{d_i} P(+c_i|d_i, \mathbf{x}, \theta_j)\ P(+d_i|\mathbf{x}, \theta)\ P(\mathbf{x}|\theta)\ P(\theta)\ d\theta && (4.13) \end{aligned}$$

where $\mathbf{X} = iPred(D_i)$ is the set of informational predecessors of decision D_i in the original influence diagram and $\mathbf{x}$ represents each configuration of $iPred(D_i)$.

We know that $P(d_i|\mathbf{x}, \theta_j)$ is equal to 1 when d_i is the optimal decision option for the configuration $\mathbf{x}$ in the diagram given for θ, and 0 otherwise. Thus, in the summation over

d_i it is enough to consider the optimal policy for each configuration:

$$sens(D_i, \Theta) = \int_{\Theta} \sum_{\mathbf{x}} P(+c_i|\delta^{\Theta}_{D_i}(\mathbf{x}, \theta_j), \mathbf{x}, \theta_j)\ P(\mathbf{x}|\theta)\ P(\theta)\ d\theta \tag{4.14}$$

where$\delta^{\Theta}_{D_i}$ is the optimal policy for D_i in the influence diagram of parameter Θ.

On the other hand, $P(+c_i|\delta^{\Theta}_{D_i}(\mathbf{x}, \theta_j), \mathbf{x}, \theta_j)$ is equal to 1 when there is no change in the policy when compared to the reference ID; otherwise, its value is 0. Thus, if $\mathbf{X}'_{\theta}$ is the subset of configurations of $\mathbf{X}$ in which the optimal decision option given by $\delta^{\Theta}_{D_i}$ is different from the reference ID:

$$sens(D_i, \Theta) = \int_{\Theta} \sum_{\mathbf{x} \in \mathbf{X}'_{\theta}} P(\mathbf{x}|\theta)\ P(\theta)\ d\theta \tag{4.15}$$

$$= \int_{\Theta} \sum_{\mathbf{x} \in \mathbf{X}'_{\theta}} P(\mathbf{x}, \theta)\ d\theta \tag{4.16}$$

4.5.2 Computation of sensitivity analysis measures

We are going to compute three measures of sensitivity analysis:

- the policy change thresholds (see Section 2.7.1),
- the expected value of perfect information (see Section 2.7.2),
- and the sensitivity of each decision to each parameter (see Section 4.5.1).

We are going to describe how to built an ID, named *augmented ID*, for calculating these measures. After that, we expose the algorithms for the computations.

Augmented ID

Let us assume that we have a parameter Θ in an influence diagram I. Let us denote by Y one the nodes of I whose probability or utility potential ϕ_Y contains Θ. Let $\mathbf{X}$ be the parents of Y in the reference diagram. An *augmented ID* for Θ is derived from I by making the following changes:

- Adding a discrete node,Θ.
- Adding an arc from Θ to each of the nodes Y, and another from Θ to the first decision of I .

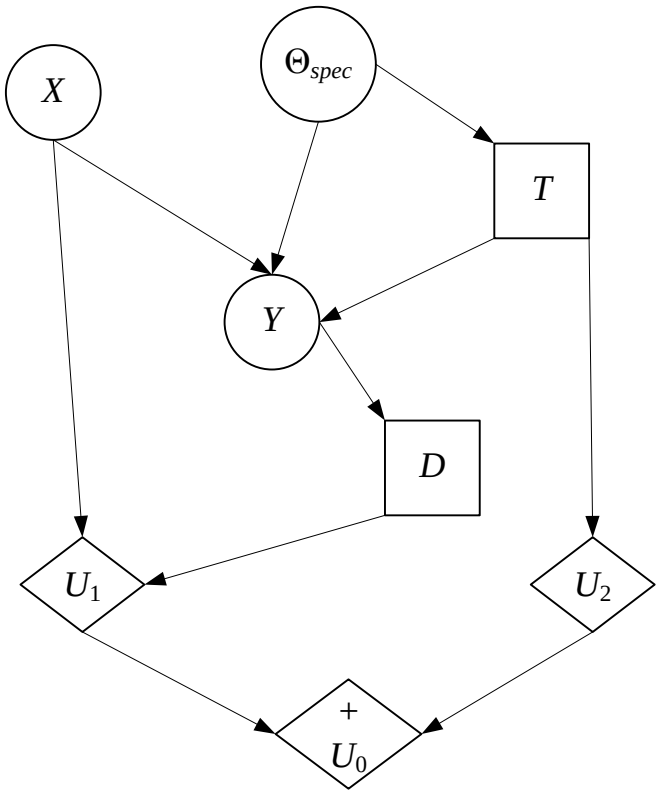

Figure 4.7: Augmented ID for parameter Θ_{sens} of test problem (Figure 2.1).

For example, let us consider the ID for the *test problem*, shown in Figure 2.1. The augmented ID for the parameter Θ_{sens}, which represents the specificity of test Y, spec $= P(+y|+t,+x)$, is shown in Figure 4.7.

The augmented ID has the same set of probability and utility potentials as I , except for node Y, which is modified to incorporate the new parent, Θ.

If Y is a chance node, its potential in I is $P(y|\mathbf{x})$. Let y^{Θ} the value of Y and $\mathbf{x}^{\Theta}$ the configuration of $\mathbf{X}$ such that Θ represents the value of $P(y^{\Theta}|\mathbf{x}^{\Theta})$. The potential of Y in the augmented ID is:

$$P'(y|\mathbf{x},\theta_i) = \begin{cases} \theta_i & \text{if } y = y^{\Theta} \wedge \mathbf{x} = \mathbf{x}^{\Theta} \\ sem(y,\theta_i,P(y|\mathbf{x})) & \text{if } y \neq y^{\Theta} \wedge \mathbf{x} = \mathbf{x}^{\Theta} \\ P(y|\mathbf{x}) & \text{otherwise} \end{cases} \tag{4.17}$$

where $\text{sem}(y,\theta_i,P(y|\mathbf{x}))$ is a function that assigns the residual mass probability $1-\theta_i$ into the states of Y different from y^{Θ}.

A possibility for $\text{sem}(y,\theta_i,P(y|\mathbf{x}))$ is to divide $1-\theta_i$ proportionally to the probabilities in I. However, there are some cases in which we know that $1-\theta_i$ has to be redistributed differently. For example, in Figure 4.7, we would always assign $1-\theta_{spec}$ to the state $\neg y$

and 0 to the state *no-result*.

If Y is a utility node in I, the utility of Y in the augmented ID is:

$$U(\mathbf{x}, \theta_i) = \begin{cases} \phi_Y(\mathbf{x}) & \text{if } \mathbf{x} \neq \Pi_\Theta \\ \theta_i & \text{otherwise} \end{cases} \tag{4.18}$$

Parameter Θ is assumed to be characterized by a continuous probability distribution. The augmented ID will be used in Section 4.5.2 to compute some measures of sensitivity analysis, which requires to evaluate it. We could model Θ as a continuous variable in the ID, but evaluating it with exact methods would be impossible. One possibility for performing an approximate evaluation would be to use Monte Carlo methods for evaluating it. However, we have instead preferred to discretize Θ as follows:

- For distributions with bounded domain, we partition the entire interval. For example, in beta distributions the discretization partitions the interval $[0, 1]$.
- For a distribution whose domain is unbounded, such as the normal or log-normal, we select an interval $[a, b]$ that ensures that the interval selected accumulates a high percentage of the mass probability of the continuous distribution. For example by taking $[\mu - k \cdot \sigma, \mu + k \cdot \sigma]$ and using $k = 3.5$ for a normal distribution, this interval accumulates 99.953 % of the probability mass.

If $[a, b]$ is the interval selected for the discretization, then we partition it uniformly, by taking the medium point of each subinterval, i.e., we select N points $\{\theta_0, \theta_1, ..., \theta_{N-1}\}$ such that the distance between contiguous points is $h = (b - a)/N$, the first point is $\theta_0 = a + h/2$, and $\theta_i = \theta_{i-1} + h$ for $i \geq 0$.

The points of the discretization are assigned as states of node Θ.

The probability distribution of Θ is given by:

$$P(\theta_i) = \alpha \cdot f_\Theta(\theta_i)$$

where f_Θ is the density function of distribution of Θ, and α is a normalization constant to ensure $\sum_i P(\theta_i) = 1$.

Computation on the augmented ID

In this section we suppose that we have a numerical parameter Θ to investigate in an ID I, and we have built the augmented ID I' for Θ. The optimal policies of a decision D_i in I and I' are denoted by δ_{D_i} and $\delta_{D_i}^\Theta$ respectively.

We describe next how to compute the measures of sensitivity analysis by using I and I'. We have sometimes omitted some parameters when writing invokations to the algorithms if their values were clear from the context.

Policy change thresholds We first define the Boolean function $change(D_i, \theta_j)$, which is equal to *true* if there exists a configuration $\mathbf{x}$ of $iPred(D_i)$ such that $\delta_{D_i}(\mathbf{x}) \neq \delta^{\Theta}_{D_i}(\mathbf{x}, \theta_j)$, and it is *false* otherwise. This function is calculated by Algorithm 4.1, which only analyzes the policy changes in the scenarios that are possible in I and I' according to their respective optimal strategies.

Algorithm 4.1 Calculus of the change of the policy of a decision to a each value of a parameter

Input: I : an influence diagram
D_i: a decision of I;
θ_j: a value of a parameter of I;
I': augmented ID for Θ;
$\delta_{D_i}(\mathbf{x})$: optimal policy of D in I;
$\delta^{\Theta}_{D_i}(\mathbf{x})$: optimal policy of D in I';
$P(\mathbf{x})$: probability of $iPred(D_i)$ in the CPN of I;
Output: $change(D_i, \theta_j)$

1. $\mathbf{X} := iPred(D_i)$ in I;
2. calculate $P'(\mathbf{x}|\theta_j)$ in the CPN of I';
3. *change* := *false*;
4. $\mathbf{x}$:= first configuration of $\mathbf{X}$;
5. **while** $\neg$*change* **do**
6. *change*:=$(P(\mathbf{x}) > 0) \wedge (P'(\mathbf{x}|\Theta) > 0) \wedge (\delta_{D_i}(\mathbf{x}) \neq \delta^{\Theta}_{D_i}(\mathbf{x}, \theta_j))$;
7. $\mathbf{x}$:= next configuration of $\mathbf{x}$;
8. **end while**
9. **return** *change*;

When function $change(D_i, \theta_j)$, has been calculated for every value θ_j of a parameter Θ it is very easy to approximate the policy change thresholds. We assume that for a different value from the discretization points we have $change(D_i, \theta) = change(D_i, \theta_j)$ if $\theta \in [\theta_j - h/2, \theta_j + h/2)$. We find an approximate set of intervals where the policy does not change by using Algorithm 4.2, which calculates the thresholds by using function *change*, computed by Algorithm 4.1. The output of Algorithm **??** is a list of intervals $thresholds(D_i, \Theta)$ where the policy is the same as in the reference case. The algorithm uses Boolean variables *doesCurrentChange* and *doesNextChange* to detect when there are adjacent points where the policy changes in one point and does not change in the other one; indexes l and u are used to delimit the interval where the policy does not change and thus add it to the list $thresholds(D_i, \Theta)$.

Algorithm 4.2 Calculus of the policy change thresholds for a parameter

Input: I : an influence diagram
D_i: a decision of I;
Θ: parameter of I;
I': augmented ID for Θ.
Output: $thresholds(D_i, \Theta)$.

1. $doesCurrentChange := change(D_i, \theta_0)$;
2. **if** doesCurrentChange=false **then**
3. $l := 0$;
4. **end if**
5. $thresholds := \emptyset$;
6. **for** $j := 1$ **to** $N - 1$ **do**
7. $doesCurrentChange := change(D_i, \theta_j)$;
8. **if** $doesCurrentChange \neq doesNextChange$; **then**
9. **if** $doesNextChange$ **then**
10. $u := j$;
11. $thresholds := thresholds \cup \{[\theta_l - h/2, \theta_u - h/2)\}$;
12. **else**
13. $l := j$;
14. **end if**
15. $doesCurrentChange := doesNextChange$;
16. **end if**
17. **end for**
18. **return** $thresholds$;

The intersection of the policy change thresholds for Θ for every decision D_i in the influence diagram is the interval where the strategy does not change.

Expected value of perfect information A classic metric used in SA (Felli and Hazen, 1998) is the expected value of perfect information (EVPI), defined in Section 2.7. Let us have the notation:

- EU_V^{Θ}: maximum expected utility (MEU) of I' ;
- EU_F^{Θ}: expected utility (EU) of using the reference strategy in I'.

The EVPI of Θ can be calculated as:

$$EVPI(\Theta) = EU_V^{\Theta} - EU_F^{\Theta}$$

Algorithm 4.3 calculates last equation.

Algorithm 4.3 Calculus of the expected value of perfect information for a parameter

Input: I : an influence diagram
D_i: a decision of I;
Θ: parameter of I;
I': augmented ID for Θ.
Output: $EVPI$: the EVPI of Θ.
1. EU_V^{Θ} := MEU of I';
2. I'' := construct an ID from I' by replacing the decisions by chance nodes by using the optimal policies of I;
3. EU_F^{Θ} := MEU of I'';
4. $EVPI(\Theta) := EU_V^{\Theta} - EU_F^{\Theta}$;

Sensitivity of a decision to a parameter Calculating Equation 4.15 by exact methods can be very complicated, even impossible. We instead propose to calculate it by using the discretization explained above.

Let $\{\theta_j\}_{0 \leq j \leq N-1}$ be the resultant domain of discretizing Θ. The value of sensitivity $sens(D_i, \Theta)$ is approximated as:

$$sens(D_i, \Theta) = \sum_j \sum_{\mathbf{x} \in \mathbf{X}'_\theta} P(\mathbf{x}, \theta) \tag{4.19}$$

and is calculated by Algorithm 4.4.

Algorithm 4.4 Calculus of the sensitivity of a decision to a parameter

Input: I : an influence diagram
D_i: a decision of I;
Θ: parameter of I;
I': augmented ID for Θ;
$\delta_{D_i}(\mathbf{x})$: optimal policy of D in I;
$\delta^{\Theta}_{D_i}(\mathbf{x})$: optimal policy of D in I';
Output: $sensitivity(D_i, \Theta)$.

1. calculate $P(\mathbf{x}, \theta)$ in the CPN of the augmented ID;
2. $sens(D_i, \Theta) := 0$;
3. **for all** configuration $(\mathbf{x}, \vartheta)$ **do**
4. **if** $\delta_{D_i}(\mathbf{x}) \neq \delta^{\Theta}_{D_i}(\mathbf{x}, \theta)$ **then**
5. $sens(D_i, \Theta) := sens(D_i, \Theta) + P(\mathbf{x}, \theta)$;
6. **end if**
7. **end for**

4.6 Discussion

We have described new explanation methods that we have developed for both the model (the knowledge encoded in the influence diagram) and the reasoning (the strategies returned by it), which proved to be very useful in the construction and debugging of MEDIASTINET, and helped to convince the expert that the recommendations given by our model are reasonable.

The coloring of links in Elvira has been very useful for debugging IDs, by detecting probability and utility tables whose numerical values do not agree with the qualitative influences assessed by the expert.

When an influence diagram has been evaluated with Elvira, it displays graphically the probabilities of chance and decision variables and the expected utilities.

The possibility of introducing evidence in Elvira has been useful for building IDs in medicine (Luque et al., 2005): before having this explanation facility, when we were interested in computing the posterior probability of a certain diagnosis given a set of findings, we needed to manually convert the ID into a BN by removing decision and utility nodes. The concept of evidence in Elvira differs from the proposal by Ezawa (1998).

We have included in Elvira the possibility of having a strategy in which some of the policies are imposed by the user and the others are computed by maximization. This has allowed to analyze non-optimal strategies with the medical expert, which were very useful for debugging the model.

We have seen how utility plots in Elvira has been useful for understanding why some tests are necessary or not, and why sometimes the result of a test is irrelevant,

Policy trees, which have been proposed as a compact way of representing the optimal policies returned by the ID, have been very useful for a model such as MEDIASTINET, in which the biggest policy table contains 15552, while the corresponding policy tree contains only 5 or 9 leaves, depending on the evaluation criterion. In the future these explanation facilities could be used to convert influence diagrams into tutoring systems.

We have implemented some sensitivity analysis algorithms which, given the uncertainty in the parameters, allow us to compute:

- the probability of a change in the strategy;
- the intervals of the parameters where the optimal policies do not change; and
- the expected value of perfect information for each parameter.

These algorithms presented allows us to perform probabilistic SA to an ID with SV nodes, when the uncertainty of every parameter is given by a continuous distribution, which can be unbounded (for example the beta distribution) or unbounded (for example the normal). It performs n-way independent analysis.

As future work, we could investigate some kind of explanation to tell to the expert which variables are having more influence in the strategy and in the numerical value of the maximum expected utility. We could try to find some quantitative measure for this purpose. The work initiated in this direction by Elizalde et al. (2008) for Markov decision processes would be a possible start point.

The possibility of introducing the evidence in the way proposed by Ezawa (1998) in the GUI of Elvira would be very interesting and would complement the concept of evidence that is now implemented.

We could also investigate how to simplify policy trees, which in some cases can still be big for a human expert.

Our current implementation of policy trees assumes that the decision maker acts optimally in every decision. However, policy tables present the optimal option for every scenario of the problem. We could try to find some kind of the tree representation of all the scenarios of the problem, but without having to present the entire decision tree.

With regard to sensitivity analysis, we could try to do research on SA for IDs with SV nodes. In particular, there are basically two issues to be investigated: how to perform n-way joint analysis; and how to calculate with exact methods the policy change thresholds.

Finally, building the utility plots with exact methods would also be very useful for debugging the ID with the expert.

Chapter 5

An anytime algorithm for evaluating unconstrained influence diagrams

Unconstrained influence diagrams (UIDs) extend the language of influence diagrams to cope with decision problems in which the order of the decisions is unspecified. Thus, when solving a UID we not only look for an optimal policy for each decision, but also for a so-called step-policy specifying the next decision given the observations made so far. However, due to the complexity of the problem temporal constraints can force the DM to act before the solution algorithm has finished, and, in particular, before an optimal policy for the first decision has been computed. This chapter addresses this problem by proposing an anytime algorithm that computes an optimal strategy and at any time provides a qualified recommendation for the first decision of the problem. The algorithm performs a heuristic-based search in a decision tree representation of the problem. Experiments indicate that the proposed algorithm performs significantly better under time constraints than dynamic programming.

The content of this chapter is based on (Luque et al., 2008).

5.1 Introduction

An influence diagram (ID) is a framework for representing and solving Bayesian decision problems with a linear temporal ordering of decisions (Howard and Matheson, 1984). However, in many domains finding an ordering of the decisions is an integral part of the decision problem, and in these situations the use of IDs would require all decision orderings to be explicitly specified in the model, possibly using artificial nodes and states. Examples of such decision problems include troubleshooting and medical diagnosis.

Unconstrained influence diagrams (UIDs) were introduced to represent and solve decision problems of this type (Jensen and Vomlelova, 2002); as a special case this also

includes decision problems with a linear temporal ordering of the decisions. An optimal strategy in this framework consists not only of an optimal policy for each decision, but also of a step-strategy that prescribes the next decision to consider given the observations and decisions made so far. Such strategies are computable using dynamic programming in a way similar to that for traditional IDs (Shachter, 1986; Shenoy, 1992; Jensen et al., 1994; Madsen and Jensen, 1999).

Unfortunately, many real world problems have an inherent complexity that makes evaluation through exact methods intractable when time is scarce (as an example, Fig. 5.1 shows a UID model for jaundice management in infants (Bielza et al., 1999)). Moreover, even if you had the time for solving the problem, storing the solution as a simple lookup table may be a problem: the number of possible past scenarios to consider in a policy may be intractably large.

Recently, a new approach for solving UIDs (Ahlmann-Ohlsen et al., 2009) has been proposed to reduce the memory space used during the evaluation. However, by considering the evaluation time, the method by Ahlmann-Ohlsen et al. (2009) is significantly slower than dynamic programming-based algorithm for UIDs (Jensen and Vomlelova, 2002).

In this paper we present an anytime algorithm for solving UIDs. The algorithm provides a solution whenever it is stopped, and given sufficient time it will eventually provide a correct solution.

In comparison, the standard evaluation algorithm for UIDs (Jensen and Vomlelova, 2002) is a backward algorithm employing dynamic programming like most algorithms for IDs. It starts computing an optimal policy for the last decision and moves backwards in time until it reaches the first decision. If the process is stopped prematurely, the algorithm may provide a policy, however, the prescription for the first decision is completely uninformed. Furthermore, as described above, all effort so far may be spent on calculating a policy for a distant decision with an enormous space for the past; a task which will decrease considerably in size when you actually approach the point of the decision. If you consider a situation with a decision maker impatiently awaiting advice on what to do, he most probably wants to get an informed advice on the first decision rather than receiving detailed prescriptions for the last decisions.

To address this problem the proposed anytime algorithm starts with the first decision and works its way forward in time. Due to the nature of the problem, we cannot be sure of the policy for the first decision before the entire problem has been solved. However, the algorithm will over time gradually improve the probability of choosing the best decision.

5.2 Unconstrained influence diagrams

UIDs were proposed in (Jensen and Vomlelova, 2002) to represent decision problems in which the order of decisions is not linear, and for which the decision maker is interested in the best ordering as well as an optimal choice for each decision. A detailed exposition of the representation language of UIDs and UID solution can be found in Section 2.4.

To illustrate the representational power of UIDs, consider the diagram in Fig. 5.1(a). In the Ictneo system there is a sequence of treatment stages. At each stage there is a non-ordered set of treatments to decide depending on the previous treatments and their result. To represent this situation with 36 different orderings in an ID we are forced to introduce a large amount of artificial variables each with a large amount of artificial states. The unspecified order of the treatments is easily represented in a UID. A part of it is shown in Fig. 5.1(b).

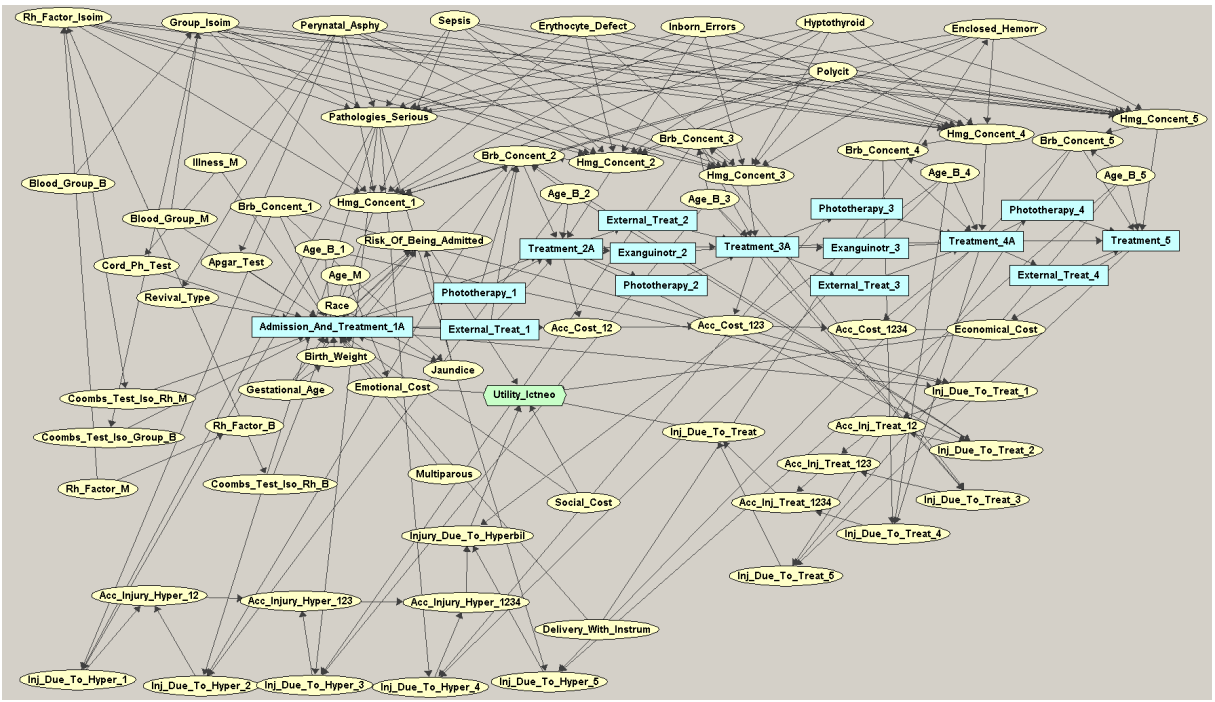

Figure 5.1: (a) A UID model for Ictneo.

5.3 An anytime algorithm for solving UIDs

In general, the basic idea with an anytime algorithm is that time constraints may cause the user to be unable to wait for the standard solution algorithm to finish. Thus, it should be possible to stop the algorithm at any time, and the algorithm should then provide an

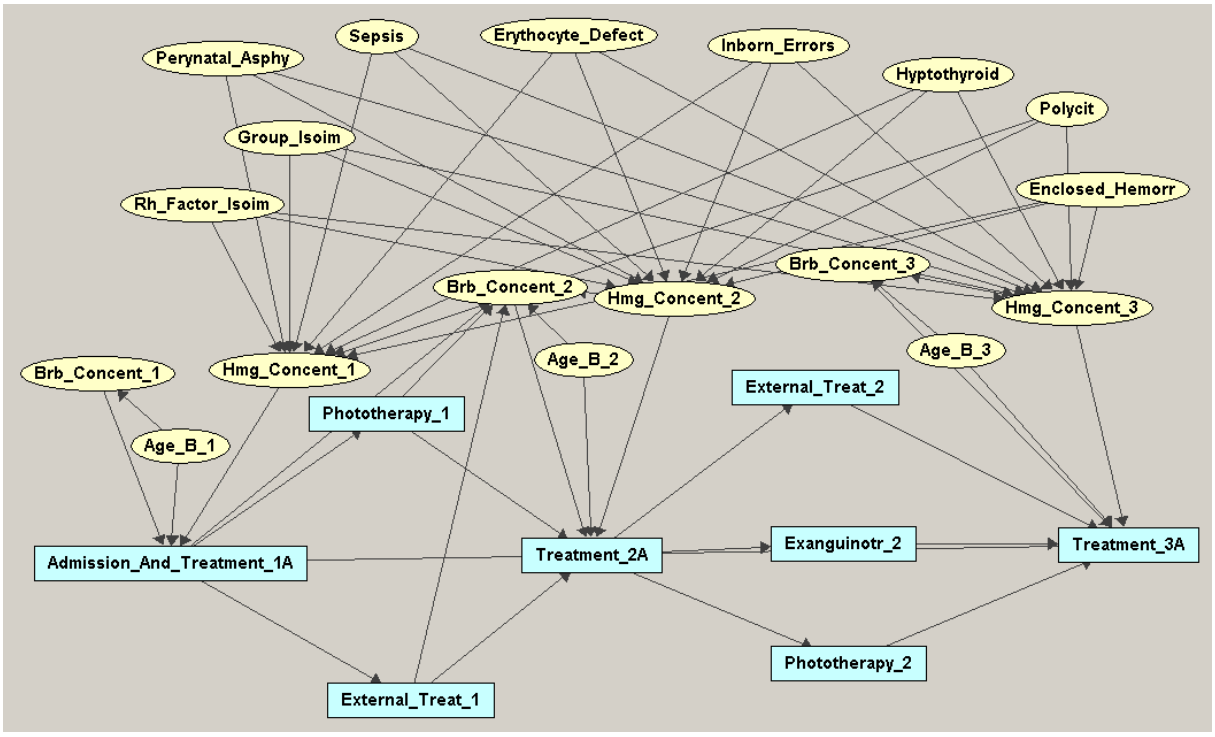

Figure 5.2: (b) A partial UID model of Ictneo

approximate solution. With this requirement we may settle for an algorithm that may take longer than the standard algorithm, but which in the mean time can provide a better approximate solution than the standard algorithm.

With respect to UIDs, the standard algorithm provides a strategy by solving the problem in reverse temporal order. If the algorithm is stopped prematurely, it can provide a strategy, which consists of choosing completely randomly for the decisions which have not yet been dealt with, and to follow the calculated optimal policies for the last decisions. In this way, it can be said that you have an anytime algorithm; it provides a strategy whenever it is stopped, the expected utility of the strategy never decreases over time, and eventually, the algorithm provides an optimal strategy.

However, this is not satisfactory. If the user stops the algorithm prematurely, it is because she needs to take the first decision, but the algorithm does not give her any clue on what to do first. Therefore, the aim of an anytime algorithm for solving UIDs (or decision graphs in general) is to provide more and more informed advice on what to do first.

We propose a forward search performed in a decision tree (Raiffa and Schlaifer, 1961). The tree is built from the root toward the leaves, and it keeps a list of triggered nodes (the

current leaves in the tree constructed so far) as candidates for expansion.[1] A triggered node X is *expanded* by adding its children to the tree and calculating the expected utility of the path from the root to X using a *heuristic function* to estimate the maximum expected utility obtainable at the children of X. We give the details of the algorithm in Section 5.3.1, and our experiments are described in Section 5.5. The main results of our experiments can be summarized as follows. For very small UIDs dynamic programming is faster than tree search; for small UIDs, our version of tree search provides a very reliable proposal of the first decision faster than dynamic programming; for large UIDs our tree search methods provides a reliable first decision much faster than dynamic programming.

5.3.1 A search based solution Algorithm

A UID can be converted into a decision tree (possibly using a dummy source node), which in turn can be used as a computational structure for solving the corresponding decision problem (disregarding complexity issues). A decision tree (Raiffa and Schlaifer, 1961) is a rooted tree in which the leaves are utility nodes and the non leaf nodes are either decision nodes (square shaped) or chance nodes (circular shaped). The decisions on the possible orderings are made explicit in the model by partitioning the decision nodes into either ordinary decisions or branching point decisions.

The past of a node X (denoted by past(X)) is the configuration specified by the labels associated with the arcs on the path from the root to X; if X is a value node then past(X) is called a *scenario*.

The quantitative part of the decision tree consists of probabilities and utilities. Each arc from a chance node A is associated with a probability $P(A = a \mid \text{past}(A))$, where $A = a$ is the label of the arc. These probabilities can be found by converting the UID into a Bayesian network: value nodes are removed, and decision nodes are replaced by chance nodes having no parents and with an arbitrary probability distribution. Finally, with each value node V in the decision tree, we associate the utility $\psi(\text{past}(V))$ of the scenario past(V). These utilities can be read directly from the UID model.

The decision tree represents each scenario in the decision problem explicitly; hence the size of the tree can grow exponentially in the number of variables. The size can, however, be reduced by collapsing identical subtrees, a procedure also know as *coalescence* (Olmsted, 1983). The opportunities for exploiting coalescence can be automatically detected in the GS-DAG of the UID.

[1] The terminology is borrowed from AO* search algorithms (Nilsson, 1980), from which the proposed algorithm has been inspired.

Instead of building the decision tree in full and solving it using the "average-out and fold-back" algorithm (Raiffa and Schlaifer, 1961), we propose to build the tree from the root toward the leaves. A heuristic function h should provide an estimate of the maximum expected utility obtainable at every node in the decision tree. Thus, at any point in time we have a *partial decision tree* in which the heuristic can be used to estimate the maximum expected utility at the leaf nodes. These estimates can in turn be propagated upward in the tree, which gives an estimate of the maximum expected utility of the nodes in the explored part of the tree, and, in particular, an estimate of the optimal policy for the decision nodes in this part.

A collection of optimal policies for a subset of the decision nodes is called a *partial strategy* Δ', and the partial strategy based on the heuristic function is called a *partial heuristic strategy* $\hat{\Delta}'$. Clearly, the closer the heuristic function is at estimating the maximum expected utility of the triggered nodes in the partial decision tree, the closer $\hat{\Delta}'$ will be at Δ'.

A partial strategy can always be extended to a full (not necessarily optimal) strategy by assigning random policies to the decision nodes in the unexplored part of the tree. When we have a set of policies S, we define the *uniform extension* of S as a strategy Δ such that every policy in S is in Δ and the rest of the policies in Δ are random with a uniform distribution.

Performing the search

The search/construction of the coalesced decision tree starts with the tree consisting of a single root node together with its children (such a tree stump is always uniquely identifiable). From this the method iteratively expands a node consistent with the UID specification.

When a node is expanded, its outgoing links are added to the decision tree as well as any successor node not already in the tree; the node to be expanded is always selected among the triggered nodes/leaves. When a node is added to the decision tree, a heuristic estimate of the maximum expected utility for that node is calculated. The values are propagated upwards, thereby possibly updating the current partial heuristic strategy.[2]

The choice of which node to expand is non-deterministic. We have experimented with three selection schemes: (*i*) expand the node X with highest probability $P(\text{past}(X))$ of occurring (decision nodes are given an even probability distribution), (*ii*) expand the

[2]Note that the search/exploration process is closely related to heuristic search algorithms in AND/OR graphs (Nilsson, 1980), where chance nodes are AND nodes and decision nodes and branching points are OR nodes.

node X with highest weight $w(X) = P(\text{past}(X)) \cdot h(X)$, where h is the heuristic function estimating the expected utility of node X, and (*iii*) expand the node of lowest depth, i.e., perform a breadth first search. Preliminary experiments suggest that the latter provides the best results, and this is therefore the selection scheme used in the tests documented in Section 5.5.

In summary, a triggered node X is selected for expansion based on its probability $P(\text{past}(X))$ of occurring, and a heuristic function is used to estimate the maximum expected utility of the trigger nodes, i.e., the leaves in the partial decision tree. Thus, at any time during the search we have a partial decision tree for which a heuristic based strategy can be computed.

5.3.2 Selecting a Heuristic Function

The choice of heuristic function not only determines the policies being computed, but it may in fact also be used to prune irrelevant parts of the tree thereby reducing complexity. A special class of heuristic functions are the so-called admissible heuristic functions:

Definition 5.3.1 *A heuristic function h is said to be* admissible *if $h(N) \geq \text{MEU}(N)$ for any node N in the decision tree.*

An admissible heuristic can be exploited during the search: Consider a decision node whose children X and Y are the roots in two subtrees. If the subtree defined by Y have been explored and $h(X) \leq \text{MEU}(Y)$, then we need not explore the subtree rooted at X.

Obviously, we would like the heuristic function h to define a tight upper bound on the expected utility, and relative to the computational complexity of solving the decision tree we would also like for h to be easy to compute.

An Admissible Heuristic

A possible heuristic function could be

$$h_U(X) = \max_{l \in \mathcal{L}} \psi(\text{path}(X, l)), \tag{5.1}$$

where $\mathcal{L}$ is the set of leaf nodes in the subtree rooted at X and $\psi(\text{path}(X, l))$ is the sum of the utilities associated with l and the path from X to l.

It is trivial to see that h_U is admissible. Moreover, h_U has the advantage of being computationally efficient, since it can be evaluated by max-marginalizing out the variables appearing in the domains of the utility potentials. The number of required

max-marginalizations is at most $|\mathbf{V}_C \cup \mathbf{V}_D|$. In contrast to the dynamic programming approach (Section 2.4.3), the complexity of computing this heuristic does not depend on the number of possible paths in the GS-DAG as max-operations commute.

Unfortunately, preliminary experiments have shown that h_U yields a very loose bound on the expected utility. For certain UIDs the estimated optimal policy for the first decision failed to stabilize over time, and in fact a random policy would on average have provided a similar solution in terms of expected utility. Since we have not been able to define an alternative computationally efficient admissible heuristic, we have instead been looking for a nonadmissible heuristic.

A Nonadmissible Heuristic

The estimation given by the admissible heuristic h_U can be extremely far from the maximum expected utility. However, since it provides an upper bound on the expected utility, we can use it in combination with a lower bound to derive a good approximation to the expected utility.

As a lower bound, we use the expected utility of the uniform extension of the current partial heuristic strategy; decision nodes in the unexplored part of the decision tree are treated as chance nodes with a uniform distribution. Relative to the computational complexity of solving the UID, this heuristic can be calculated efficiently by sum-marginalizing out the variables in the utility and probability potentials. The number of required marginalizations is at most $|\mathbf{V}_C \cup \mathbf{V}_D|$ and does not depend on the number of paths in the GS-DAG as sum-marginalizations commute (this also means that we are not required to follow an admissible elimination order consistent with the UID).

It is easy to see that this heuristic provides a lower bound on the MEU. If we denote the lower bound heuristic by h_L, then we have the following inequalities for any node in the decision tree:

$$h_L(X) \leq \mathrm{MEU}(X) \leq h_U(X).$$

If all the variables in the future of node X are chance variables, i.e., if $\mathrm{future}(X) \subseteq \mathbf{V}_C$, then $h_L(X) = \mathrm{MEU}(X)$. Furthermore, as the number of decision nodes in $\mathrm{future}(X)$ increases the larger the difference $\mathrm{MEU}(X) - h_L(X)$ will be. The opposite holds for the heuristic $h_U(X)$.

In order to derive a heuristic closer to the actual expected utility, we define the nonadmissible heuristic h as a weighted linear combination of h_L and h_U:

$$h(X) = w_L(X)h_L(X) + w_U(X)h_U(X),$$

where

$$w_L(X) = \alpha(X) \cdot k_X \cdot c(X); \quad w_U(X) = \alpha(X) \cdot d(X).$$

Here $c(X)$ and $d(X)$ are the number of chance and decision nodes in future(X), respectively, and $\alpha(X)$ is a normalizing factor ensuring $w_L(X) + w_U(X) = 1$. By varying the parameter k_X between 0 and $+\infty$, one can achieve any desired mixture of conservatism and optimism as defined by the two heuristics; note that k_X may be the same for all nodes.

One potential difficulty with this heuristic is how to choose a good value for k_X. To alleviate this problem, we propose to update k_X automatically as the tree is expanded. The intuition underlying the updating method is that we would in general expect the heuristic to be more precise the closer we get to the leaves: After a node X has been expanded we first estimate the expected utility of its children (using h and the current value for k_X). These estimates are then propagated upward in the tree: If X is a chance node, then the value propagated to X is $\widehat{\mathrm{EU}}(X) = \sum_{Y \in \mathrm{ch}(X)} h(Y)$ and if X is a decision node then the value is $\widehat{\mathrm{EU}}(X) = \max_{Y \in \mathrm{ch}(X)} h(Y)$. By treating $\widehat{\mathrm{EU}}(X)$ as an accurate estimate of the expected utility for X we calculate a new value for k_X by setting $\widehat{\mathrm{EU}}(X) = h(X)$:

$$k_X := \frac{(\widehat{\mathrm{EU}}(X) - h_U(X))d(X)}{(h_L(X) - \widehat{\mathrm{EU}}(X))c(X)}.$$

Note that k_X will always be non-negative, and that the update is not guaranteed to get us closer to the true expected utility, since we might have started off with the correct value for k_X.

To shed some light on the properties of the updating procedure we can consider the difference between $\widehat{\mathrm{EU}}(X)$ and $h(X)$. Specifically, assume that X is a chance variable with children ch(C). Then $h(X) - \widehat{\mathrm{EU}}(X)$ can be written as:

$$h(X) - \widehat{\mathrm{EU}}(X) = \sum_{Y \in \mathrm{ch}(X)} P(X \to Y)([m_U(X)(h_U(Y) - h_L(Y))] + [w_U(X)(\max_{Z \in \mathrm{ch}(X)} h_U(Z) - h_U(Y))]),$$

where $m_U(X) = w_U(X) - w_U(Y) \leq 0$ and Y is any child of X. Thus, the updated value for k_X can be derived from:

$$\sum_{Y \in \mathrm{ch}(X)} P(X \to Y)([m_U(X)(h_U(Y) - h_L(Y))] + [w_U(X)(\max_{Z \in \mathrm{ch}(X)} h_U(Z) - h_U(Y))]).$$

The right term in the inner sum is always non-negative, thereby decreasing $\widehat{\mathrm{EU}}(X)$ relative to $h(X)$. This decrease is maximized if only a single subtree contains the utility value $h_U(X)$. By decreasing $\widehat{\mathrm{EU}}(X)$ with a value proportional to $\max_{Z \in \mathrm{ch}(X)} h_U(Z) - h_U(Y)$ therefore compensates for $h(X)$ being calculated without considering how the maximum values are distributed in the subtrees rooted at the children of X; we can consider the calculation of $h_U(X)$ as assuming that the highest utility value appears in all subtrees rooted at the children of X.

The left term is non-positive, since $m_U(X)$ is non-positive, hence it correspond to a relative increase in the estimated expected utility. The value $m_U(X)$ corresponds to the weight that is shifted from the lower bound to the upper bound when X is expanded (recall that the weights are determined by the number of succeeding chance and decision variables). Thus, in accordance with the definition of the weight, the influence from the lower bounds is decreased whereas the influence from the upper bounds is increased.

If X is a decision variable, then the updating procedure is easier justified: Before X is expanded the subtrees rooted at the children of X jointly defines the heuristic value for X. However, as X is a decision variable only a single subtree defines the expected utility, and this is also reflected in the updating of $\widehat{\mathrm{EU}}(X)$.

5.4 Evaluation metrics

We are going to define some metrics and plots that can help us to know how good is an anytime algorithm when evaluating an unconstrained influence diagram and that will be necessary when presenting the experimental results. Some of the proposals in this section can also be applied to influence diagrams, while others are specific for UIDs.

The metrics are oriented to measure the quality of the strategy given by an anytime algorithm when it has to compute the policies for the first decisions, and, after the anytime algorithm stops, the decision maker will have enough time to compute the optimal policies for the rest of decisions.

First, we will discuss first the concept of anytime strategy. After that we will present the evaluation metrics used in the experiments.

5.4.1 Anytime strategies

In Section 5.3.1 we saw that a partial strategy can always be extended to a full strategy by assigning random policies to the decision nodes in the unexplored part of the tree. That anytime strategy Δ is consequently a function of the time t, so it can be denoted

by $\widehat{\Delta}(t)$. It provides the decision maker with a sequence of prescriptions, each one for a decision of the problem.

However, the necessity of having a full strategy covering all the decisions can be relaxed in some situations. The decision maker can need the help of an anytime algorithm to obtain a prescription for the first n decisions in the decision tree[3], but, after that, it will have time to continue the evaluation in the rest of decisions. The anytime strategy over the time, when only requiring prescription for the first n decisions of the decision tree, and the rest of the policies can be computed exactly by the dynamic programming algorithm, is denoted by $\widehat{\Delta}^n(t)$.

For example, if $n = 0$, then we have the situation in which the entire problem is evaluated by dynamic programming (DP). If $n = 1$, then we need the anytime algorithm to calculate the policy for the first decision, and the rest of the problem is evaluated by DP, and so on.

We have defined $\widehat{\Delta}^n(t)$ to monitor the performance of the algorithms and to calculate some evaluation metrics, which are presented below.

In the rest of the section we will denote by I a UID, and by $depth(I)$ the maximum number of decisions in a path from the root to a leaf in the decision tree of I.

5.4.2 Definitions of evaluation metrics

The evaluation metrics are divided in two sets depending on the value they basically represent: (a) the expected utility of a strategy, and (b) the probability of being right in the decisions made. After that, we will describe some metrics to summarize the results.

Metrics of the expected utility of the anytime strategy

The first plot that we can think about when measuring the performance of anytime algorithm is the plot of the expected utility (EU) of the anytime strategy $\widehat{\Delta}^i(t)$. Let us denote $EU(\widehat{\Delta}^i(t))$ by $EU^i(t)$. A very interesting property that relates them for every instant time t:

$$EU(\widehat{\Delta}^i(t)) \geq EU(\widehat{\Delta}^{i+1}(t)),$$

where i is an integer such that $0 \leq i \leq depth(I)$. It is trivial to see that $MEU(I) = EU^0(t)$.

[3]We talk about number of decisions in the decision tree instead of referring to the UID. This is because in the decision tree the decisions not only correspond to decision nodes in the UID but also to branch points that model the possible orderings.

Metrics of the frequency of selection of the right decision

We can measure the goodness of the decisions made in any level of the decision tree in which we have to make a decision. Then, let i be an integer number such that $0 \leq i \leq depth(I)$, then the *frequency of optimal selection in the i-th level of decisions in the tree*, denoted by $FreqDec^i(t)$, is defined as:

$$FreqDec^i(t) = \frac{\sum_{D \in decisions(i)} FreqDec(D,t)}{|decisions(i)|} \tag{5.2}$$

where $decisions(i)$ is a set that contains every decision D whose path from the root of the tree to D has exactly i decisions; and $FreqDec(D,t)$ takes the value 1 if the decision in node D in instant time t is the optimal one and 0 otherwise. Equation 5.2 calculates the mean of $FreqDec(D,t)$ over all the scenarios defined by $decisions(i)$. However, a more sophisticated equation could calculate a weighted mean by multiplying $FreqDec(X,t)$ by the probability of the scenario defined by X.

The global goodness of the anytime algorithm when making the first n decisions can be measured by accumulating the values $FreqDec^i(t)$, having $1 \leq i \leq n$. Then, the *accumulative frequency of optimal selection in the n-th level of decisions*, denoted by $AccFreqDec^n(t)$, is defined as:

$$AccFreqDec^n(t) = \sum_{i=1}^{n} FreqDec^i(t)$$

$AccFreqDec^n(t)$ gives a measure of the number of decisions being optimal in the first n levels of decisions in the decision tree.

5.4.3 Normalization of the metrics

In Section 5.5.2 we will present a summary of the experimental results. That summary required the metrics of all the UIDs evaluated to be combined into an individual metrics. For example, if we are interested in summarizing the results for $FreqDec^1(t)$, then we have to find a way to combine the values of $FreqDec^1(t)$ for all the UIDs evaluated. However, given that the graphs and the numbers in different UIDs can be completely different, the summary for $FreqDec^1(t)$ does not simply consist in calculating the mean of the values of the individual UIDs. Instead of that, the method used here is based on a normalization of the evaluation of every UID, and considering the dynamic programming evaluation as reference for the time axis.

Let us assume that the evaluation of a UID with an algorithm γ begins in the instant time 0 and is stopped in the instant time τ_γ. We also assume that dynamic programming (DP) algorithm is always stopped when it finishes its evaluation, and the instant time of stopping is denoted by τ_{DP}.

Let $f_\gamma(t)$ be a function over the time, measuring some aspect of the evaluation of a UID with an algorithm γ. The function $f_\gamma(t)$ has the domain $[0, \tau_\gamma]$, which is the interval time while the algorithm γ has been running. Then, the normalization of $f_\gamma(t)$, denoted by $f'_\gamma(t)$, is a function whose domain is $[0, \tau_\gamma/\tau_{DP}]$ and is defined by:

$$f'_\gamma(t) := \frac{f_\gamma(t \cdot \tau_{DP}) - f_{DP}(0)}{f_{DP}(\tau_{DP}) - f_{DP}(0)}$$

It is easy to see that f'_{DP} is a function whose range and domain is $[0, 1]$, and that verifies $f'_{DP}(0) = 0$ and $f'_{DP}(1) = 1$. Moreover, for any algorithm γ we have $f'_\gamma(t) \leq 1$.

Having the normalization $f'_{\gamma,r}(t)$ of every function $f_{\gamma,r}(t)$ for every algorithm γ and every UID r, we construct a *summary function* $f''_\gamma(t)$ *for each algorithm* γ, taking the mean over the different templates, the different graphs of each template and the different UIDs of each graph. We also obtain that the summary function $f''_{DP}(t)$ satisfies the same properties that the individual functions f'_{DP}: its range and domain is $[0, 1]$, and verifies $f''_{DP}(0) = 0$ and $f''_{DP}(1) = 1$. When talking about plot in this section, we will refer to the corresponding summary function.

Justification and interpretation of the normalized metrics

The normalization of the metrics described above uses DP evaluation as reference for normalizing the metrics because the objective of our anytime algorithm is to provide a policy for the first decision(s) better than the proposed by DP algorithm. We therefore need to see the quality of the answer of the anytime algorithm over the time, comparing it with the obtained by DP in the same instant time. The decision maker can be impatiently waiting advice on what to do, but he wants an informed advice on the first decision before of the end of the evaluation of DP.

The interpretation of the normalization of the metrics needs also another explanation. Every normalized metric will be presented in a two-axis graph. The X-axis represents the time of the evaluation, and can be seen as the fraction of the time spent by DP algorithm. The Y-axis represents how better is the anytime strategy than having random policies for the decision nodes in question. Random policies are attached with the value 0 in the

Y-axis. Moreover, it is clear that the anytime strategy would be at most as good as an optimal strategy, which is therefore represented in the Y-axis with value 1.

For example, let us consider the normalized metric for $AccFreqDec^3(t)$ in Figure B.8. This metrics accumulates the number of decisions right in the first 3 levels with decisions in the decision tree. Then, the value 0 for the normalized metric of $AccFreqDec^3(t)$ would correspond to the number of decisions right when we assign random policies in the decisions of the first 3 levels of the tree. Thus, if all the decisions in the first 3 levels are binary, random policies in them would give us a value for $AccFreqDec^3(t)$ of 3/2 (see Equation 5.4.3). And with optimal policies the value 1 in the Y-axis would be correspond to the value of 3 for $AccFreqDec^3(t)$.

5.5 Experiments

We have performed a series of experiments for assessing the efficiency of testing three algorithms: the dynamic programming-based (DP), breadth first search with admissible heuristic (BF-A) and with non-admissible heuristic (BF-N). The first problem we faced is that we only have a few real-world examples of UIDs, and these are not complex enough for comparing the algorithms among themselves. The repositories of graphical probabilistic models available on Internet do not contain UIDs. For this reason, we have created on randomly generated UIDs, but imposing some structures that can appear in some real domains. Moreover, the UIDs generated had been solved with exact algorithms so that we could obtain the exact solution and compare with our proposals.

5.5.1 Generation of UIDs

Vomlelova (2003) proposed an algorithm for generating randomly UIDs. However, we have not used it because of two reasons. First, that algorithm generated UIDs having a structure that does not match with the UIDs that can appear in real domains. And second, the UIDs generated rarely present a high number of possible paths in the GS-DAG, so the evaluation are so fast that do not require the use of an anytime algorithm.

Alternatively, we have used a different approach for generating the UIDs. We have divided the generation of a UID in two phases: (a) obtain the structure of the UID (nodes, their kinds and the arcs); and (b) generate the numbers of the probability and utility potentials.

We have used 4 methods for generating the structure of UID. Each method corresponds to a template having some parameters. Templates try to vary the structure of the UID

by representing some patterns that can appear in a real problem with partial order of decisions. For example, let us consider the UID of Figure 5.5. It can be seen like the representation of a medical decision problem where the unobservable variable H represents a disease, O_1 and O_2 indicates two vaccinations for H, and O_3 and O_4 represents medical tests. The doctor first has to decide which vaccinations are necessary, and in which order, and, after that, decides on which tests (and in which order) are performed to the patient. Finally, decision D_0 represents the decision about the treatment.

We describe the parameters for each method and their corresponding pseudocode that generates the structure of the UID.

Algorithm 5.1 is called *Template 1*, and its parameter is *nDec*, the number of decision nodes. Figure **??** presents an example of UID generated according to *Template 1*, when $nDec = 3$.[4]

Algorithm 5.1 Template 1

Input: *nDec* is a parameter described in Section 5.5.1.
Output: the graph of a new UID.

1. create a utility node U;
2. **for** $i = 1$ to *nDec* **do**
3. create a new decision node D_i and an observable node node O_i;
4. add the arcs (D_i, O_i), (D_i, U) and (O_i, U);
5. **end for**

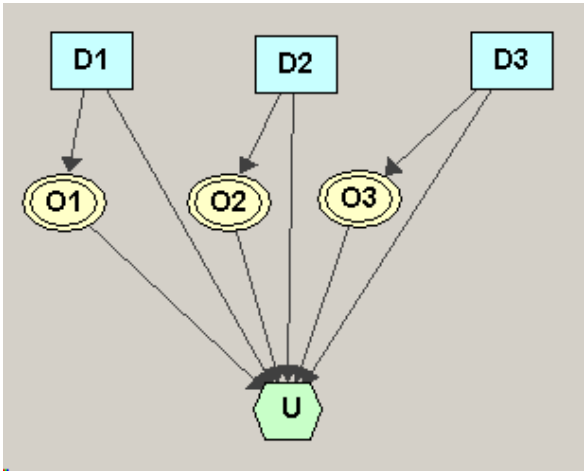

Figure 5.3: Example of *Template 1*

Algorithm 5.2 is called *Template 2*, and its parameters are *nDec1* and *nDec2*, that correspond to the numbers of decision nodes in the first and second subparts of the graph

[4]The UIDs generated have been represented in Elvira system under the assumption exposed by Jensen and Vomlelova (2002) that states that each decision node D has a cost, which does not have be represented graphically when only depends on D.

respectively. Figure 5.4 presents an example of UID generated according to *Template 2*, when $nDec1 = 2$ and $nDec2 = 2$.

Algorithm 5.2 Template 2

Input: $nDec1$ and $nDec2$ are parameters described in Section 5.5.1.
Output: the graph of a new UID.

1. create an observable node O_0;
2. **for** $i = 1$ to $nDec1$ **do**
3. create a new decision node D_i,a new observable node O_i and a new utility node U_i;
4. add the arcs (D_i, O_i), (O_i, O_0) and (O_i, U_i);
5. **end for**
6. create a utility node U_0;
7. **for** $i = nDec1+1$ to $nDec1+nDec2$ **do**
8. create a new decision node D_i and a new observable node O_i;
9. add the arcs (D_i, O_i), (O_i, U_0) and (O_0, O_i);
10. **end for**

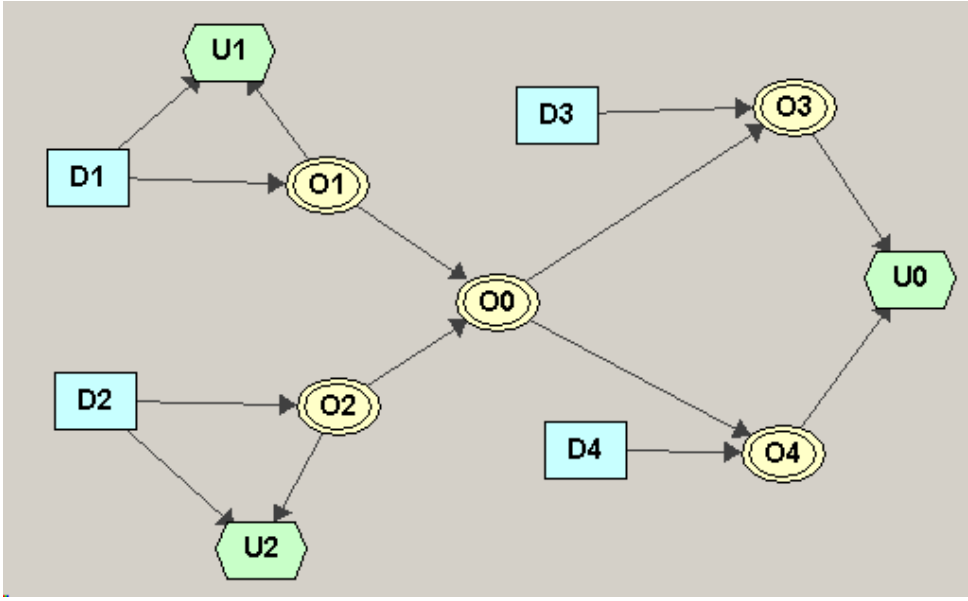

Figure 5.4: Example of *Template 2*

Algorithm 5.3 is called *Template 3*, and its parameters are $nDec1$ and $nDec2$, that correspond to the numbers of decision nodes in the first and second subparts of the graph respectively. Figure 5.5 presents an example of UID generated according to *Template 3*, when $nDec1 = 2$ and $nDec2 = 2$.

Algorithm 5.4 is called *Template 4*, and its parameter is $nDec$, the number of decision nodes. Figure 5.6 presents an example of UID generated according to *Template 4*, when $nDec = 3$.

Finally, the complete method for generating a UID is presented in Algorithm 5.5.

Algorithm 5.3 Template 3

Input: *nDec1* and *nDec2* are parameters described in Section 5.5.1.
Output: the graph of a new UID.

1. create a non-observable node H;
2. **for** $i = 1$ to *nDec1* **do**
3. create a new decision node D_i,a new observable node O_i and a new utility node U_i;
4. add the arcs (D_i, O_i), (D_i, U_i), (O_i, H) and (O_i, U_i);
5. **end for**
6. create a new decision node D_0 and a new utility node U_0;
7. **for** $i =$*nDec1*$+1$ to *nDec1*$+$*nDec2* **do**
8. create a new decision node D_i, a new observable node O_i;
9. add the arcs (D_i, O_i), (O_i, U_0), (O_i, D_0), (H, O_i) and (D_i, U_0);
10. **end for**
11. create a new utility node U and add the arcs (O_0, U) and (D_0, U);

Algorithm 5.4 Template 4

Input: *nDec* is a parameter described in Section 5.5.1.
Output: the graph of a new UID.

1. create a non-observable node H and a utility node U;
2. add the arc (H, U);
3. **for** $i = 1$ to *nDec* **do**
4. create a new decision node D_i and a new observable node O_i;
5. add the arcs (D_i, O_i) and (O_i, H);
6. **end for**

5.5.2 Experimental results

Using the procedures for generating a UID from the templates (see Algorithms 5.1 to 5.4), we have created four different graphs of UID according to Template 1, and three graphs for each of the other three templates (numbered 2, 3 and 4). We have obtained 50 realizations for each graph by generating randomly the numbers of the probability and utility potentials. That amounts a total of $(4 + 3 \times 3) \times 50 = 650$ UIDs.

Each UID was evaluated with three algorithms: the dynamic programming-based (DP), breadth first search with admissible heuristic (BF-A) and with non-admissible heuristic (BF-N).

The algorithms were implemented in Java 6.0 with the Elvira software package.[5] The tests were run on an Intel Core 2 computer (2.4 GHz) with 2 GB of memory under Windows XP.

[5]The Elvira program was developed as a collaborative project of several Spanish universities (Elvira Consortium, 2002). The source code, a user manual, and other documents can be downloaded from `www.ia.uned.es/~elvira`.

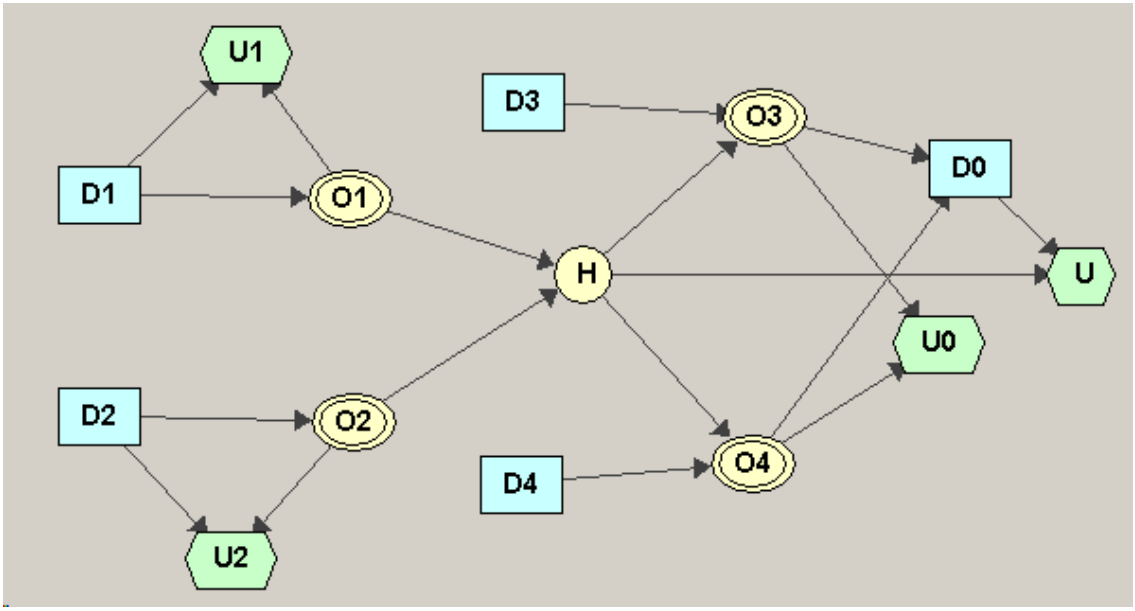

Figure 5.5: Example of *Template 3*

Algorithm 5.5 Generate a UID

Input: the template to use to create the graph of the UID and its corresponding parameters;

Output: a generated UID;

1. create the graph of the UID by using a template (see Algorithms 5.1, 5.2, 5.3 and 5.4);
2. randomly generate a probability table for each chance node (observable or non-observable);
3. randomly generate a utility table for each utility node;

Experimental results

It is important to emphasize that all results are normalized as we have described in Section 5.4.3.

Comparison of DP and BF-A Appendix B contains all the figures and table of this comparison. Figures B.3 to B.5 shows the comparison of $EU^n(t)$, $n = 1, 2, 3$, between DP and BF-A. Moreover, Figures B.6 to B.8 shows the value of $AccFreqDec^n(t)$, $n = 1, 2, 3$.

The results obtained by letting the anytime algorithm run for e.g. 25% of the time required by dynamic programming are listed in the second column in Table B.1; $EU^i(t)$ and $AccFreqDec^i(t)$ correspond to the two measures described in Section 5.4.2. In particular, $AccFreqDec^1(t)$ denotes the frequency of selecting the best initial decision (i.e., a branching point decision).

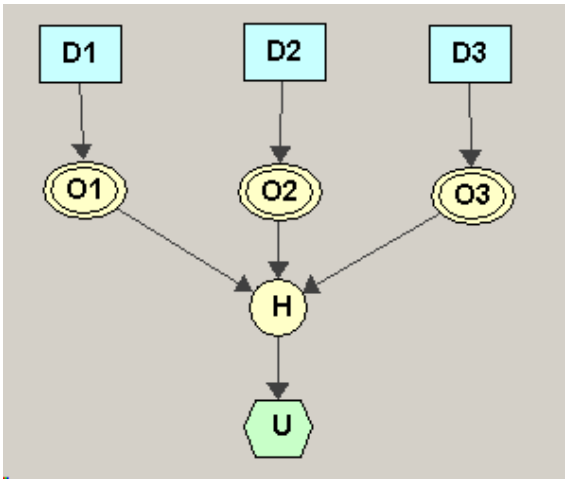

Figure 5.6: Example of *Template 4*

	0.0 %	**25.0 %**	**50.0 %**	**75.0 %**	**100.0 %**
$EU^1(t)$	0	0,538	0,613	0,56	0,586
$EU^2(t)$	0	0,64	0,754	0,891	0,94
$EU^3(t)$	0	0,609	0,724	0,83	0,872
$AccFreqDec^1(t)$	0	0,475	0,522	0,483	0,531
$AccFreqDec^2(t)$	0	0,464	0,536	0,558	0,607
$AccFreqDec^3(t)$	0	0,34	0,413	0,439	0,484

Table 5.1: Table for BF-N

Comparison of DP and BF-N Figures 5.17 to 5.19 shows the comparison of $EU^n(t)$, $n = 1, 2, 3$, between DP and BF-N. Moreover, Figures 5.20 to 5.22 shows the value of $AccFreqDec^n(t)$, $n = 1, 2, 3$.

The results obtained by letting the anytime algorithm run for e.g. 25% of the time required by dynamic programming are listed in the second column in Table 5.1; $EU^i(t)$ and $AccFreqDec^i(t)$ correspond to the two measures described in Section 5.4.2. In particular, $AccFreqDec^1(t)$ denotes the frequency of selecting the best initial decision (i.e., a branching point decision).

From the results we clearly see that the algorithm improves over time with respect to all the recorded characteristics. Moreover, if we perform a careful analysis of Table 5.1 we can obtain additional conclusions. Then, let us clarify first the table and, after that, we will analyze it.

Let us focus on $AccFreqDec^3(t)$. This metrics accumulates the number of decisions right in the first 3 levels with decision in the decision tree. Then, the value 0 in the table for $AccFreqDec^3(t)$ would correspond to the number of decisions right when we assign

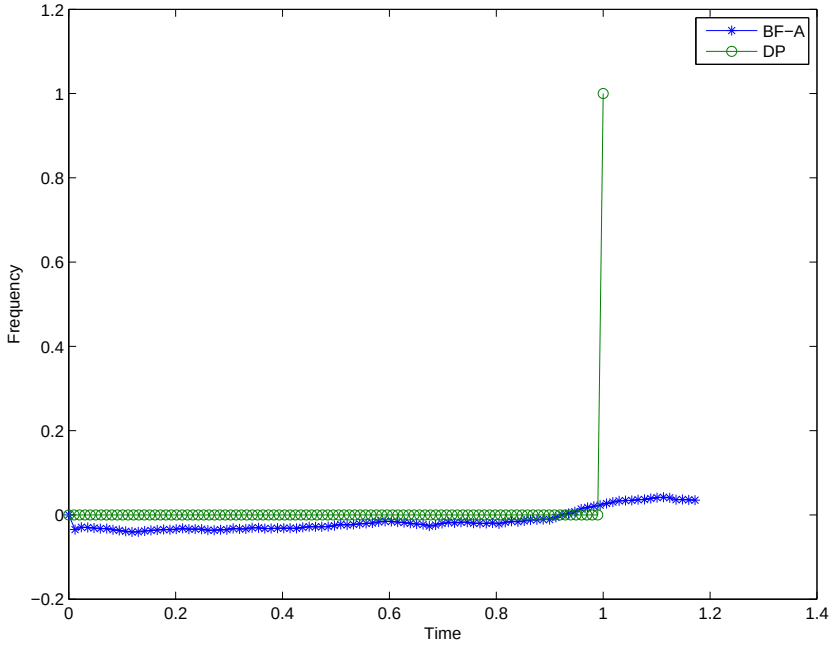

Figure 5.7: Comparison of $FreqDec(t)$ between DP and BF-A

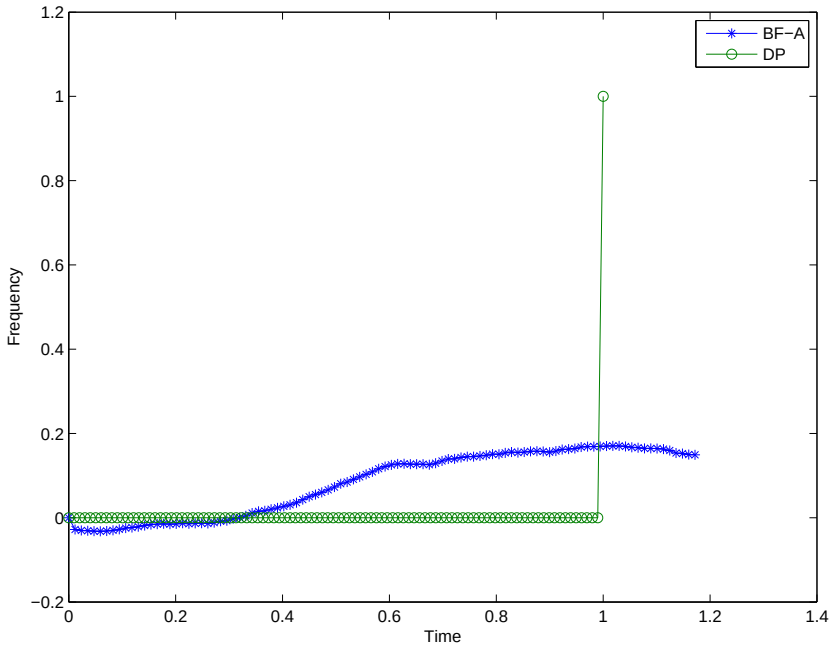

Figure 5.8: Comparison of $FreqOpt(t)$ between DP and BF-A

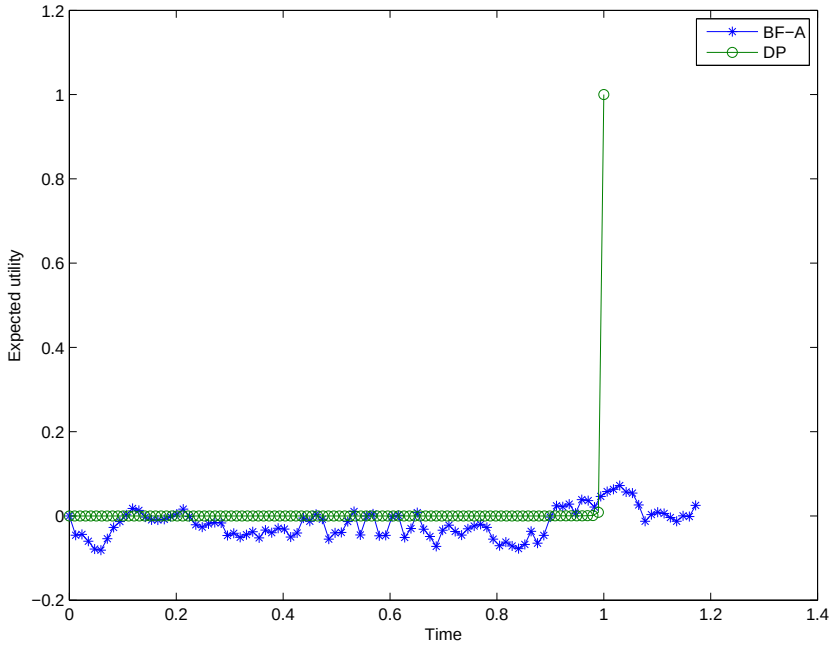

Figure 5.9: Comparison of $EU^1(t)$ between DP and BF-A

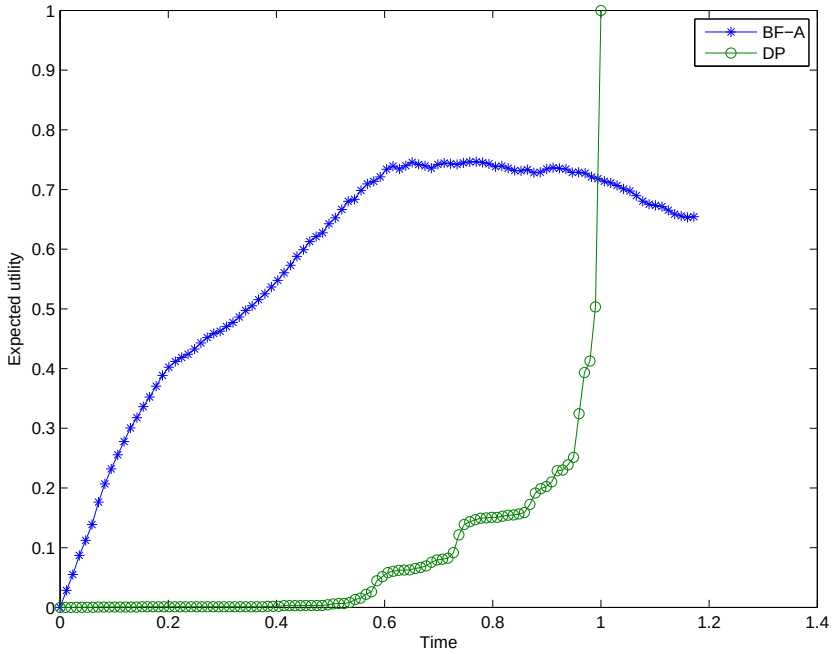

Figure 5.10: Comparison of $EU^2(t)$ between DP and BF-A

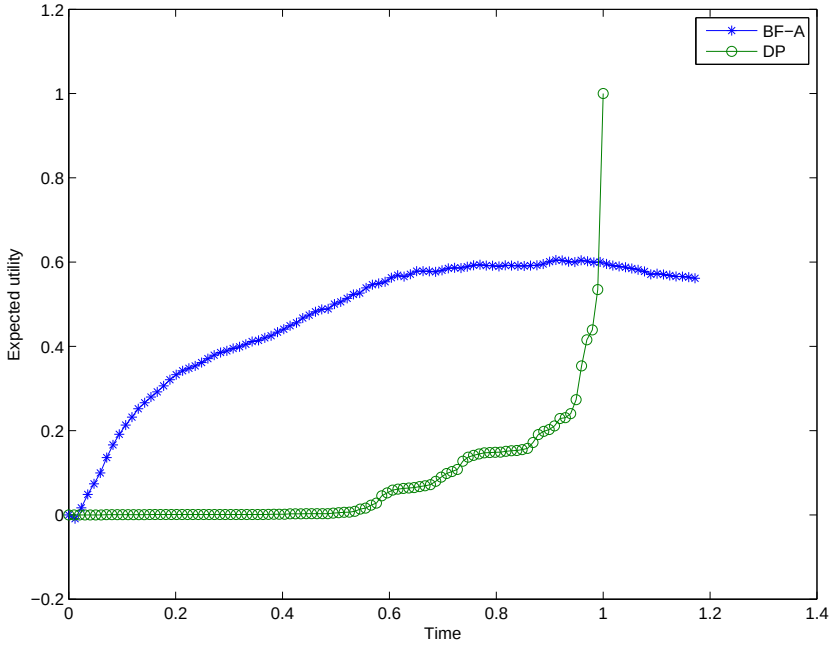

Figure 5.11: Comparison of $EU^3(t)$ between DP and BF-A

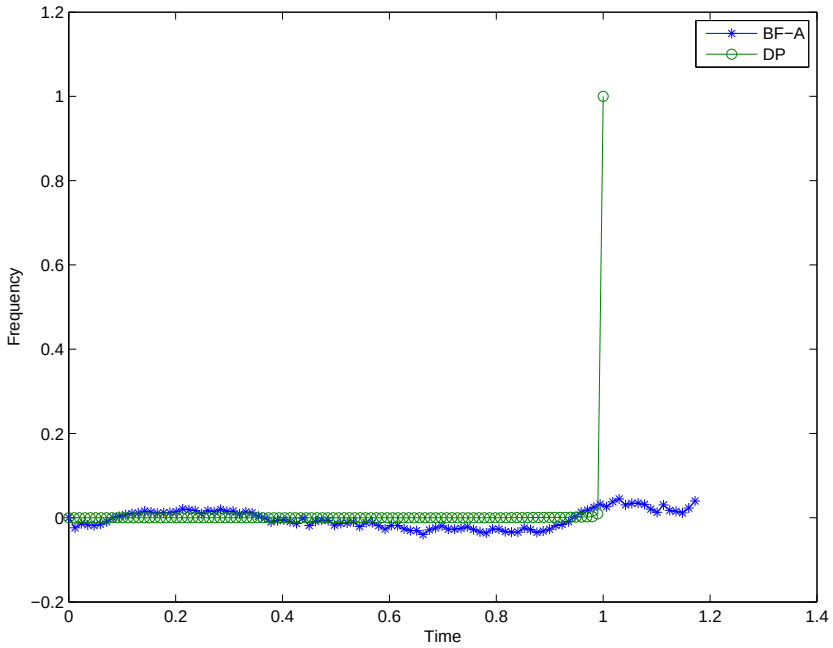

Figure 5.12: Comparison of $AccFreqDec^1(t)$ between DP and BF-A

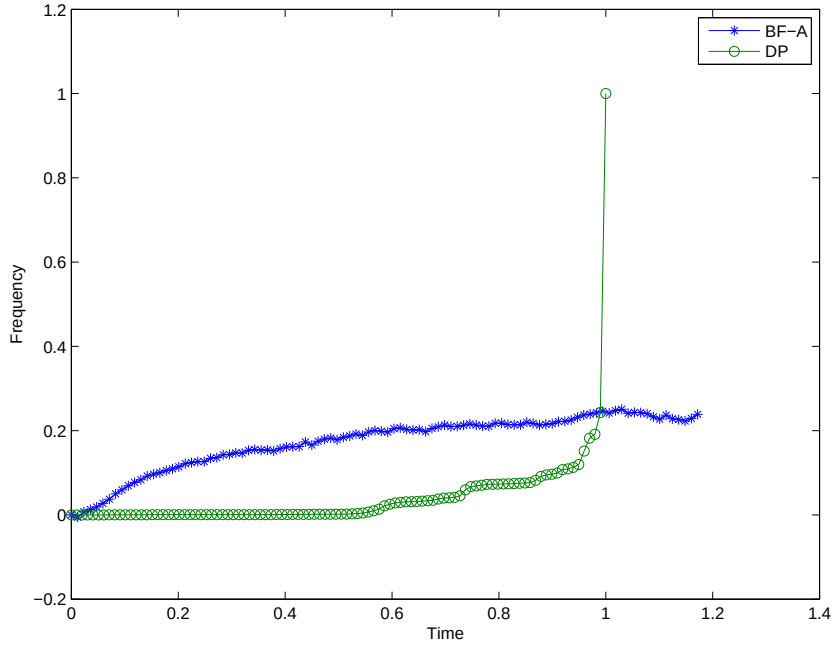

Figure 5.13: Comparison of $AccFreqDec^2(t)$ between DP and BF-A

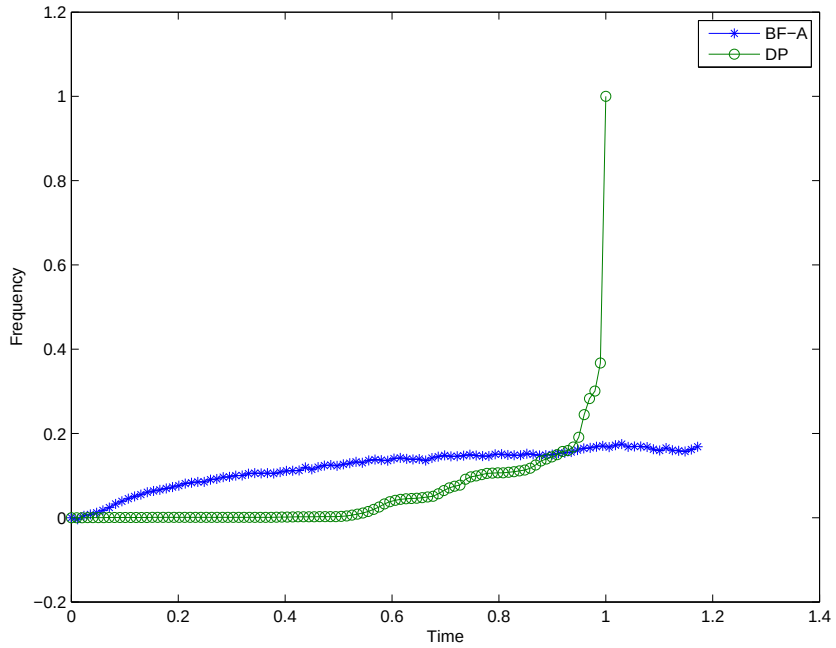

Figure 5.14: Comparison of $AccFreqDec^3(t)$ between DP and BF-A

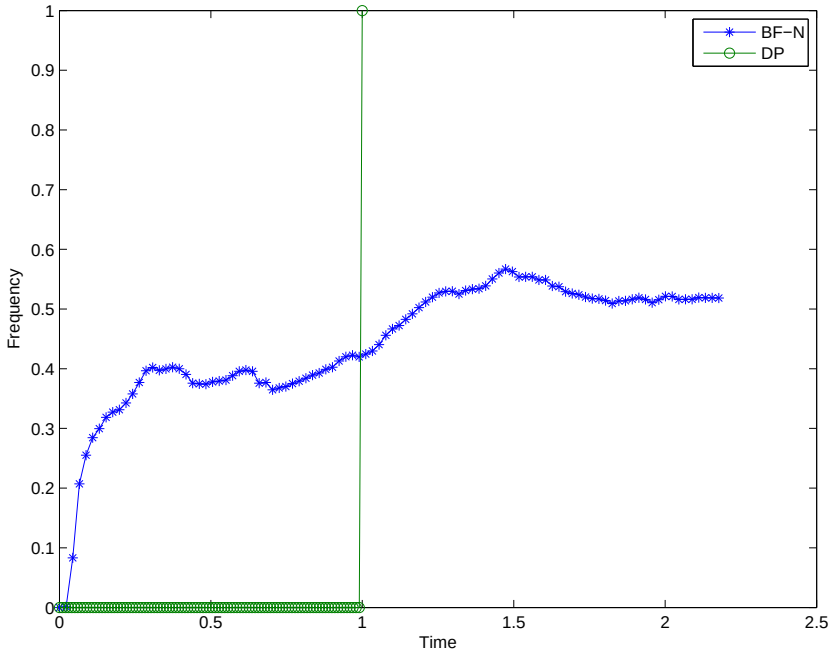

Figure 5.15: Comparison of $FreqDec(t)$ between DP and BF-N

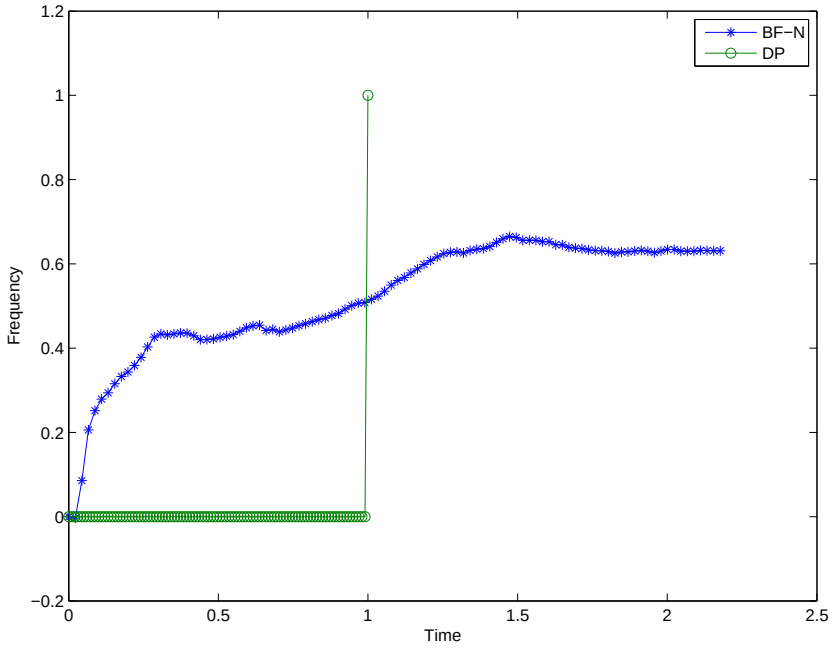

Figure 5.16: Comparison of $FreqOpt(t)$ between DP and BF-N

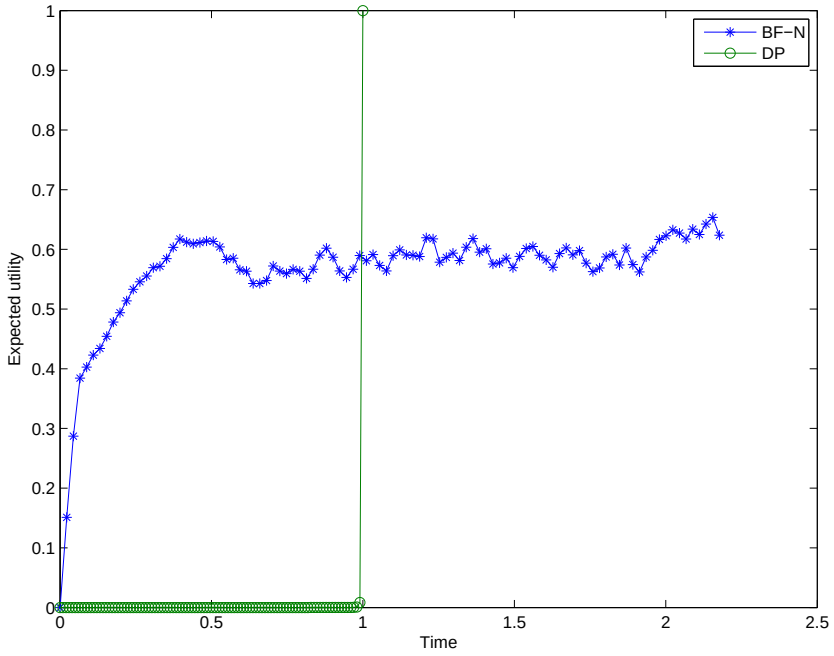

Figure 5.17: Comparison of $EU^1(t)$ between DP and BF-N

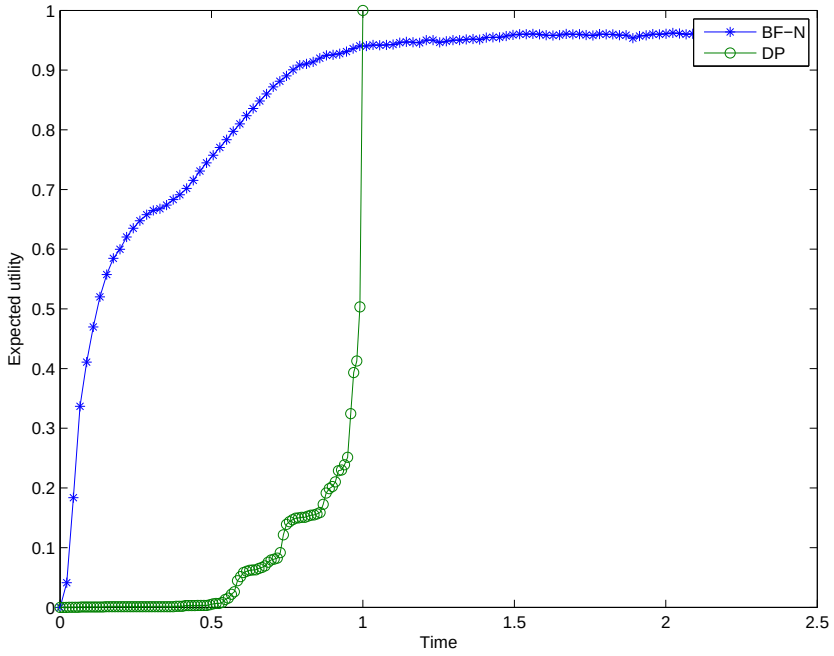

Figure 5.18: Comparison of $EU^2(t)$ between DP and BF-N

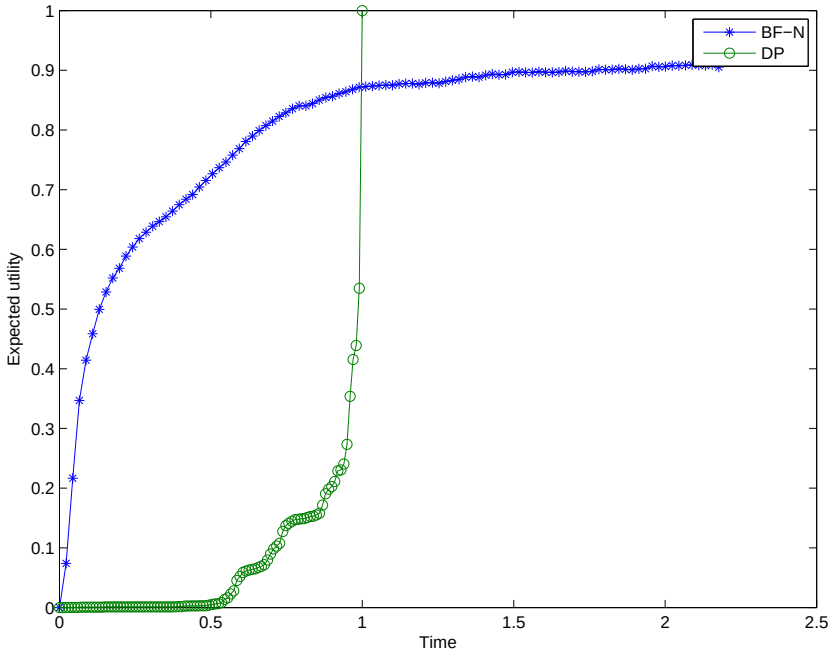

Figure 5.19: Comparison of $EU^3(t)$ between DP and BF-N

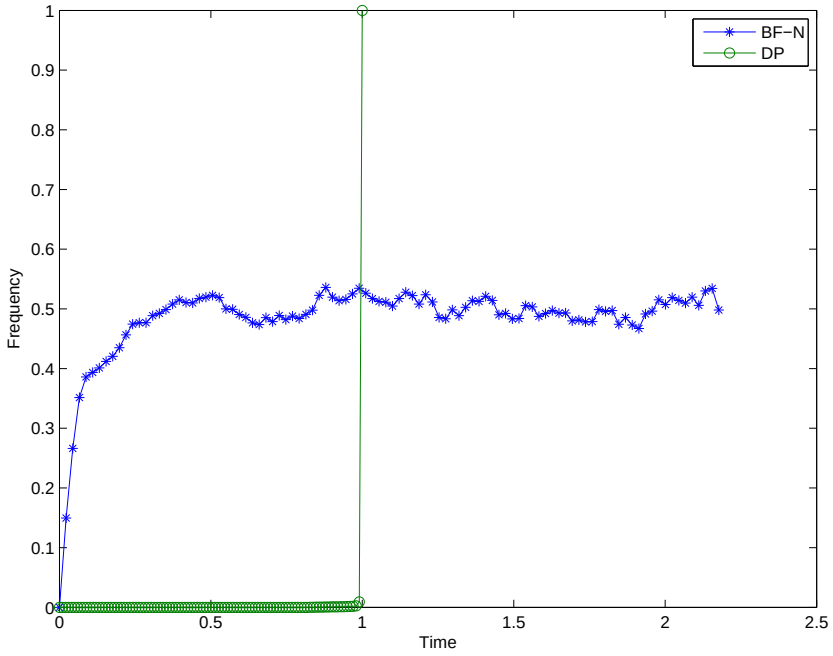

Figure 5.20: Comparison of $AccFreqDec^1(t)$ between DP and BF-N

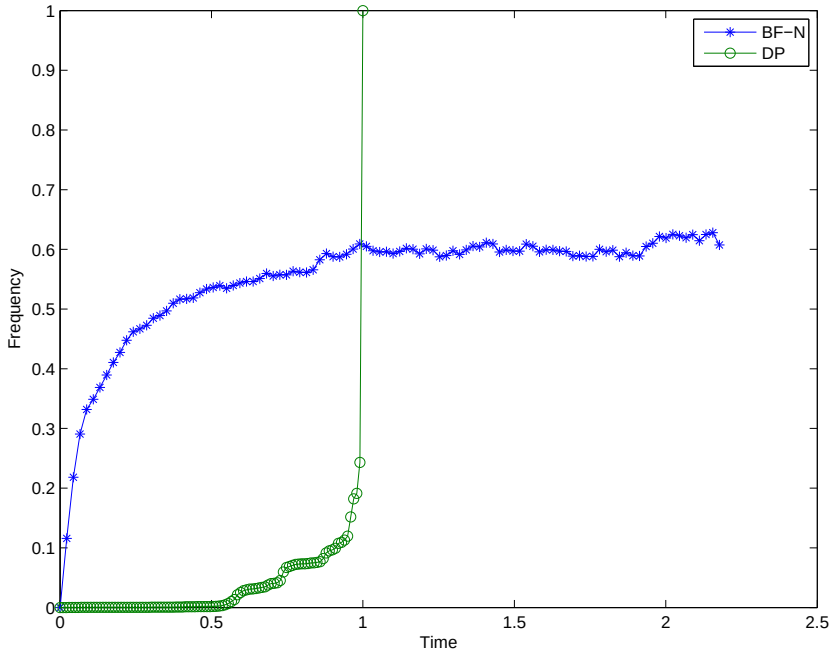

Figure 5.21: Comparison of $AccFreqDec^2(t)$ between DP and BF-N

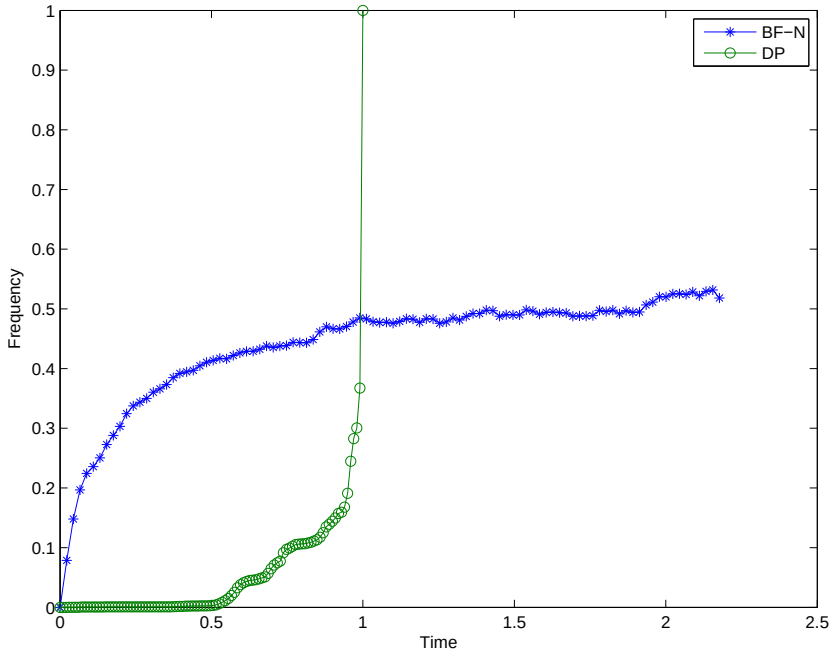

Figure 5.22: Comparison of $AccFreqDec^3(t)$ between DP and BF-N

random policies in the decisions of the first 3 levels of the tree. For example, if all those decisions in the first 3 levels are binary, random policies in those decisions would give us a value for $AccFreqDec^3(t)$ of 3/2, which is the expected number of decisions right in the first 3 levels of decisions having uniform strategy. And with optimal strategy the value 1 in the table would be assigned to the value of 3 for $AccFreqDec^3(t)$. For example, for the cell of 25% in $AccFreqDec^3(t)$ in the table, the value 0.291 comes from normalizing $3/2 + (3 - 3/2) \times 0.291 = 1.9365$ (see Equation 5.4.3), which would be the number of decisions right in the first 3 levels of the tree.

Showing an example for $EU^i(t)$ would be similar to the example of $AccFreqDec^3(t)$. For example, the value 0 in the table for $EU^i(t)$ would correspond to the expected utility of a strategy where we have random policies for the first i levels of decisions. That value would obviously depend on the particular UID realization, i.e. its probability and utility potentials. The value 1 would correspond to the expected utility having optimal strategy, i.e. the maximum expected utility.

The interpretation of the table is direct from the explanation of these examples. It shows how much better is the following strategy provided by the anytime algorithm, compared in a scale delimited by a real interval specified by two extremes: (i) the lower extreme is what we obtain having random policies in the decision nodes in question, and (ii) the upper extreme is what we obtain having the optimal policies. A positive value in the table would therefore means that the policies computed are better than just using random policies.

If we analyze the table carefully, we can appreciate the tendency of $AccFreqDec^i(t)$ or $EU^i(t)$ for a specific instant of time when i increases. Given that the algorithm performs a breadth first search, we should expect to have that the lower i the better the results values in the table. The anytime algorithm was designed to have an answer to the first decisions of the tree, so we would expect not to have so promising results for the last decisions because we assume that we would then have the possibility of calculating its policies with enough time. That tendency can be observed by checking $AccFreqDec^1(t)$ and $AccFreqDec^3(t)$ in the table.

However, we have an increase of value from $AccFreqDec^1(t)$ to $AccFreqDec^2(t)$ that contradicts that general rule. The reason can be found by thinking in the topology of the GS-DAGs of the UIDs generated. The UIDs present several initial decisions whose order is unspecified. That means that the first level of decisions correspond to a branch point for deciding the first decision, the second level corresponds to choosing a decision option, the third corresponds again to a branch point, and so on. So, the first and the third

levels correspond to branch points, while the second one corresponds to a decision node choosing its decision option. Moreover, the minimum number of initial decisions in the UIDs generated is 2, and even for some UIDs we have more initial decisions. However, all the decision variables are binary. Consequently, choosing the best decision to be made is more difficult than choosing the best decision option, as the number of possible decisions is greater than the number of possible options. That makes the values for $AccFreqDec^2(t)$ in the table be better than for $AccFreqDec^1(t)$ and $AccFreqDec^3(t)$.

The behavior for $EU^i(t)$ for a specific instant of time when i increases can be justified with the same reasoning as for $AccFreqDec^i(t)$.

We can extract other conclusion from the table. The results for $AccFreqDec^i(t)$ are better than for $EU^i(t)$, by fixing a specific level i and a instant time t. When computing $EU^i(t)$, it is positive that the policies of the algorithm for the decisions are good in terms of expected utility, in spite of not being optimal. However, when computing $AccFreqDec^i(t)$, choosing a good decision is not enough as we require it to choose the correct decision; and discern which decision is the best one can be difficult when the differences in expected utility between the different decisions are small.

Finally, we must summarize the two main important conclusions from the table: the algorithm improves over time with respect to all the recorded characteristics, and always gives better results than using a uniform strategy.

5.6 Applications of the proposed method

At the beginning of the chapter we described the problem that we wanted to address. The origin was that many real world problems have an inherent complexity that makes evaluation through exact methods intractable when time is scarce. Moreover, even if you had the time for solving the problem, storing the solution as a simple lookup table may be a problem: the number of possible past scenarios to consider in a policy may be intractably large. However, the user needs to take the first decision, so it needs the algorithm to give her any clue on what to do first. That was the objective of our proposed algorithm. Here we are going to describe some real applications of our proposed method.

The first scenario of application was briefly mentioned previously. It corresponds to a **medical decision problem**. Bielza et al. (1999) proposed a system for the jaundice management in infants. The system, named Ictneo, includes a large number of uncertain factors and decisions, to better define when treatment is required and/or should be changed, to decrease the costs and risks, and to take into account the preferences of par-

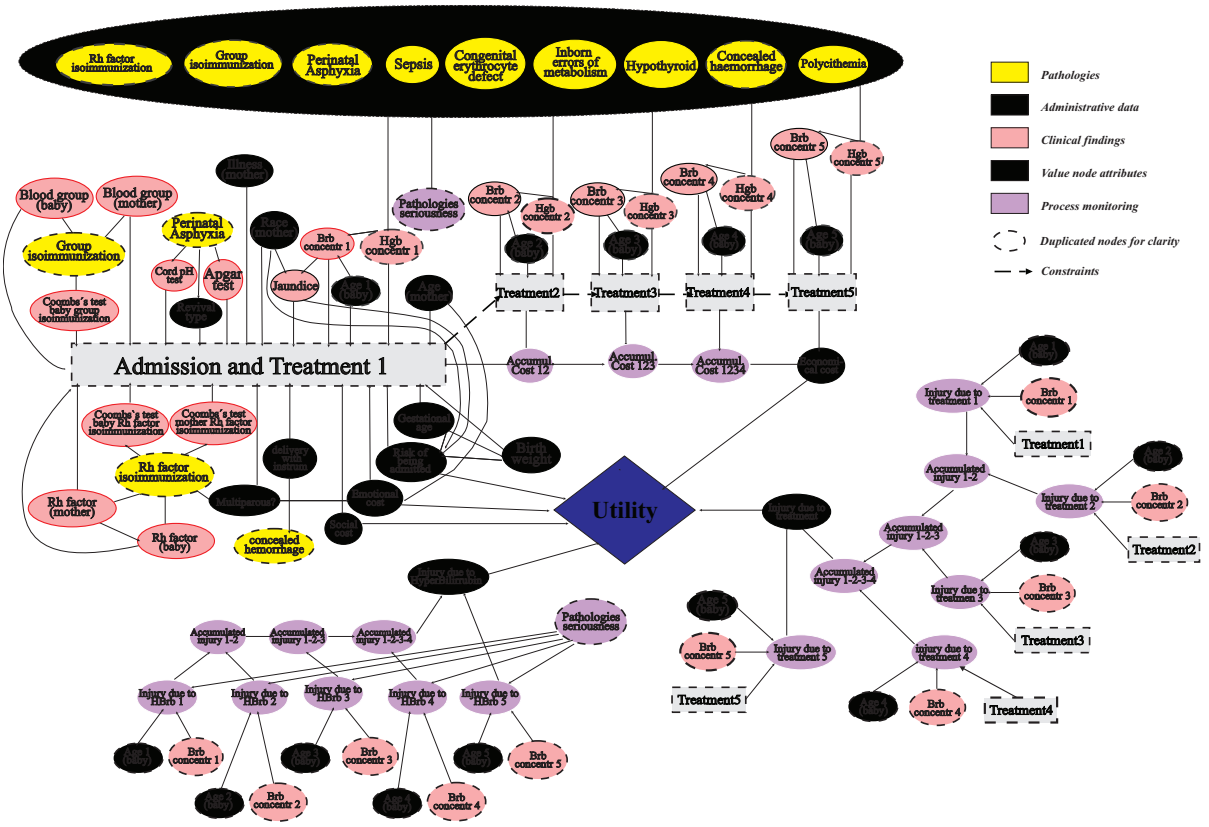

Figure 5.23: Ictneo

ents and doctors. An influence diagram representation of Ictneo is shown in Figure 5.23.

In the jaundice management in infants, the doctor first decides whether or not to admit the baby to hospital and, possibly, confines it to the Intensive Care Unit. If the baby is admitted, it is necessary to control the bilirubin levels, carrying out different tests and giving the patient some of the prescribed treatments. After each treatment stage, the effects on the baby are observed repeating the process as many times as necessary until the problem is over, i.e., the infant is discharged or she receives a treatment that falls outside of the scope of the problem.

The maximum storage space requirements during the problem-solving process were 1.66×10^{14} positions. The authors had to reduce the enormous complexity through:

- evidence propagation operations;
- incorporating asymmetries;
- simplifying the sequence of treatments: to have a sequence of 12 decision nodes for at most 72 nodes. Thus, the time between consecutive treatments was established to 6 hours.

These modifications make the influence diagram could be evaluated. However, light modifications to the requirements of the model can easily make it intractable. For example, we could be interested in refining the sequence of treatments to consider not only therapies separated by 6 hours, but by less hours. Furthermore, we could want to introduce additional medical tests in each slice of time of the decision process, and to look for what is the best ordering of them. These are situations that can appear in a real medical decision problem and would easily make the problem intractable because of the exponential growth of the space and time complexity. Moreover, even if the influence diagram can be evaluated, the doctor, who is interested in taking the first decision, could not wait until the end of the evaluation by the dynamic programming algorithm.

The situation could be more complicated if we consider for example the **emergency room of a hospital**. Doctors and nurses have to make decisions very quickly when a patient arrives to the emergency room. As in Ictneo, they can have a diagnosis problem with several available tests which could be performed in different orders. In the emergency room, making the right decisions in a short time could be decisive for saving the life of the patient.

Other situation can appear in the field of economy. In the **stock market**, every weekday, investors have to make decisions about how to invest their money. They typically have screens to receive information about the prices in the market, information about political, economic and social events that could influence the prices, or even the evolution of the prices in other stock markets. Thus, they have to make quick decisions to adapt their portfolio to the changing situation.

A similar situation appears when a person is **betting in real-time on the Internet** while watching a sport event, which usually usually last less than two hours. Significant actions during the match that could influence the result can happen in few minutes. The simultaneous availability of information to the audience through television, radio or Internet is an additional difficulty for people willing to win money in bets. They have to be extremely intelligent and make the right decisions about their bets very quickly if they want to win money.

Other scenarios appear in **natural disasters**. Forest fires, earthquakes and tsunamis are some examples of natural disasters that human beings have suffered along history. Nowadays it is still quite difficult to predict when and where can happen the next natural disaster. The decisions made by authorities of countries or regions during the first hours of the disaster influence highly the death toll as a consequence of the disaster. Thus, reducing the time of response of the authorities and making the right decisions for minimizing the

effects could save many lives.

We can also find **industrial** applications. Failures in the mains electricity or in nuclear plants can require responses with different time constraints. However, they have in common that the faster the response the smaller the effects caused by the failure. Other applications include automatic cars or autonomous robots, which have to make quick decisions when they are navigating through changing environments.

Recommender systems on the Internet are other example of application of our method. These systems use information filtering techniques to attempt to present information items (movies, music, books, web pages, etc.) that are likely of interest to the user. The recommendations are chosen automatically by the system typically in less than a second. Thus, the decisions about what to recommend must be made within severe temporal constraints.

We could also find many other fields of application. We would just to think in real situations when we have to make decisions and the temporal constraints impose us to make decisions as fast as possible.

Finally, it is important to notice that some of the situations presented here, or other that can be imagined by the reader, could present decision making where all the decisions are totally ordered. However, our method can still be applied to these situations because an influence diagram with a total order of decisions is a particular case of unconstrained influence diagram.

5.7 Discussion

We have presented an anytime algorithm for evaluating unconstrained influence diagrams. Our method performs a breadth first search in the decision tree, and, by using a non-admissible heuristic provides a qualified recommendation for the first decision. The use of an admissible heuristic was disregarded. We also presented an outline of some possible applications of the method.

We have performed some experiments to assess the performance of the algorithms. The quality of the recommendations were measured by considering:

- the frequency with which the anytime algorithm returns the correct decision options (relative to the optimal strategy) for all decisions down to the ith level in the decision tree;

- the expected utility of following the strategy prescribed by the anytime algorithm

or dynamic programming for the first i levels of decisions, followed by the optimal strategy for the remaining decisions.

From the results we could see that the anytime algorithm improves over time with respect to all the recorded characteristics. We have thus reached our initial objective.

As a future work, after performing the experiments, we can clearly see that if the purpose is to compute the entire optimal strategy then the backward evaluation performed by dynamic programming approach is more efficient. However, our method has demonstrated that forward search can give qualified recommendations for the first decisions without any information about the policies for the last ones.

Thus, it appears as a possible future work to find a forward-backward method that could combine:

- the efficiency of the backward propagation given by dynamic programming; and
- the possibility of having qualified recommendations for the first decisions when using our forward search-based approach.

We could also do research about the equation of our non-admissible heuristic. We are using a linear combination of the lower and upper bounds, but we could look for more sophisticated combinations.

In this way, the form of our heuristic only depends on the ratio of chance and decision variables in the UID, while there can also be other parameters that would make the heuristic to be more precise. For example, it is reasonable to think that the higher the number of possible paths in the GS-DAG the higher the maximum expected utility because we would have more maximizations in the evaluation.

Studying techniques for exploiting the coalescence in the decision tree could also accelerate the convergence of our algorithm and would improve the recommendations for the first decisions. This would require a careful analysis of the UID, much more than what we are doing now in the correct implementation of the algorithm.

Part IV

APPLICATION

Chapter 6

Application: Mediastinal staging of non-small cell lung cancer

Lung cancer is a frequent and devastating disease with a complex evaluation for deciding the best treatment for patients. In this chapter we describe the difficult assessment of operability of a variant of non-small cell lung cancer and present a decision support system built for finding the optimal strategies for this problem, more specifically, to find the most efficient selection of tests and therapy for each patient.

6.1 Introduction

Lung cancer is a very frequent tumor in the developed world and the leading cause of cancer death. Smoking is the main risk factor of lung cancer, with more than 90 per cent of lung cancers thought to be a result of smoking (Alberg and Samet, 2003). Broadly lung cancer can be classified into two major types: small-cell lung cancer (SCLC) and non-small cell lung cancer (NSCLC). The first one appears in 20% of cases and is usually inoperable and only treatable with chemotherapy or chemo-radiotherapy. In contrast, surgery resection remains the optimal treatment for NSCLC when it is limited to the lung, certain adjacent structures, and lymph nodes proximal to the lung (N1-hilar-lymph nodes). However, more than 80% of NSCLC patients can not be treated with surgery because the disease is out of control due to an advanced local extension of the tumor or spreading to other parts of the body (metastasis). A disappointing fact is that a high percentage of surgical patients die of lung cancer during their lifetime. A correct assessment in an early stage of the disease and an accurate selection of patients for surgery (staging phase) is very important to avoid dangerous, painful, and unnecessary surgery in bad prognosis patients.

When there are no distant metastases (i.e. spread to brain, kidney, bones, etc.) medi-

astinal staging, i.e., determining whether malignant mediastinal lymph nodes are present or absent, positive or negative N2-N3 status) is the most important prognostic factor in patients with NSCLC and, consequently, determine the therapeutic strategy. Different techniques are available to study the mediastinum and potentially malignant lymph nodes. There exist non-invasive imaging techniques, such as CT-scan and PET, with high sensitivity but low specificity; there are also minimally invasive endoscopic techniques (TBNA, EBUS, EUS)[1], with low risk, high specificity and varying degrees of sensitivities, as well as more invasive surgical techniques, such as mediastinoscopy, which are considered as the gold standard.

Because of this variety of available tests, each one having pros and cons, there is a vivid debate among specialists about which technologies should be used and in shat order. For this reason, we have developed a decision support-system for the mediastinal staging of non-small cell lung cancer. The system basically consists of an influence diagram built. We have relied on the expert advice of Dr. Carlos Disdier, pneumologist at the Hospital San Pedro de Alcántara, in Cáceres (Spain).

6.2 Medical problem: mediastinal evaluation (staging) of lung cancer

The lungs are a pair of cone-shaped organs made up of spongy, pinkish-gray tissue. They occupy most of the space in the thorax (the part of the body between the base of the neck and diaphragm).

The *mediastinum* is the space between the lungs, in the middle portion of the upper chest, and contains the heart and its large vessels, trachea, esophagus, thymus, and *lymph nodes*. The latter are located within and around the lungs and mediastinum (see Figure 6.1). Other anatomical considerations are out of the scope of this thesis. For a more detailed description of the respiratory system, see, for instance, the web site `http://healthcare.utah.edu/healthinfo/adult/respiratory/sitemap.htm`.

Lung cancer is one of the most frequent pathologies of the respiratory system. *Cancer* is a term used for naming a family of diseases in which abnormal cells divide without control. Cancer cells can invade nearby tissues and can spread to other parts of the body through the blood and lymph system. *Metastasis* is the spreading of cancer to other parts of the body.

[1] CT-scan stands for computer tomography, PET for position emission tomography, TBNA for transbronchial needle aspiration, EBUS for endobronchial ultrasound, and EUS for endoscopic ultrasound.

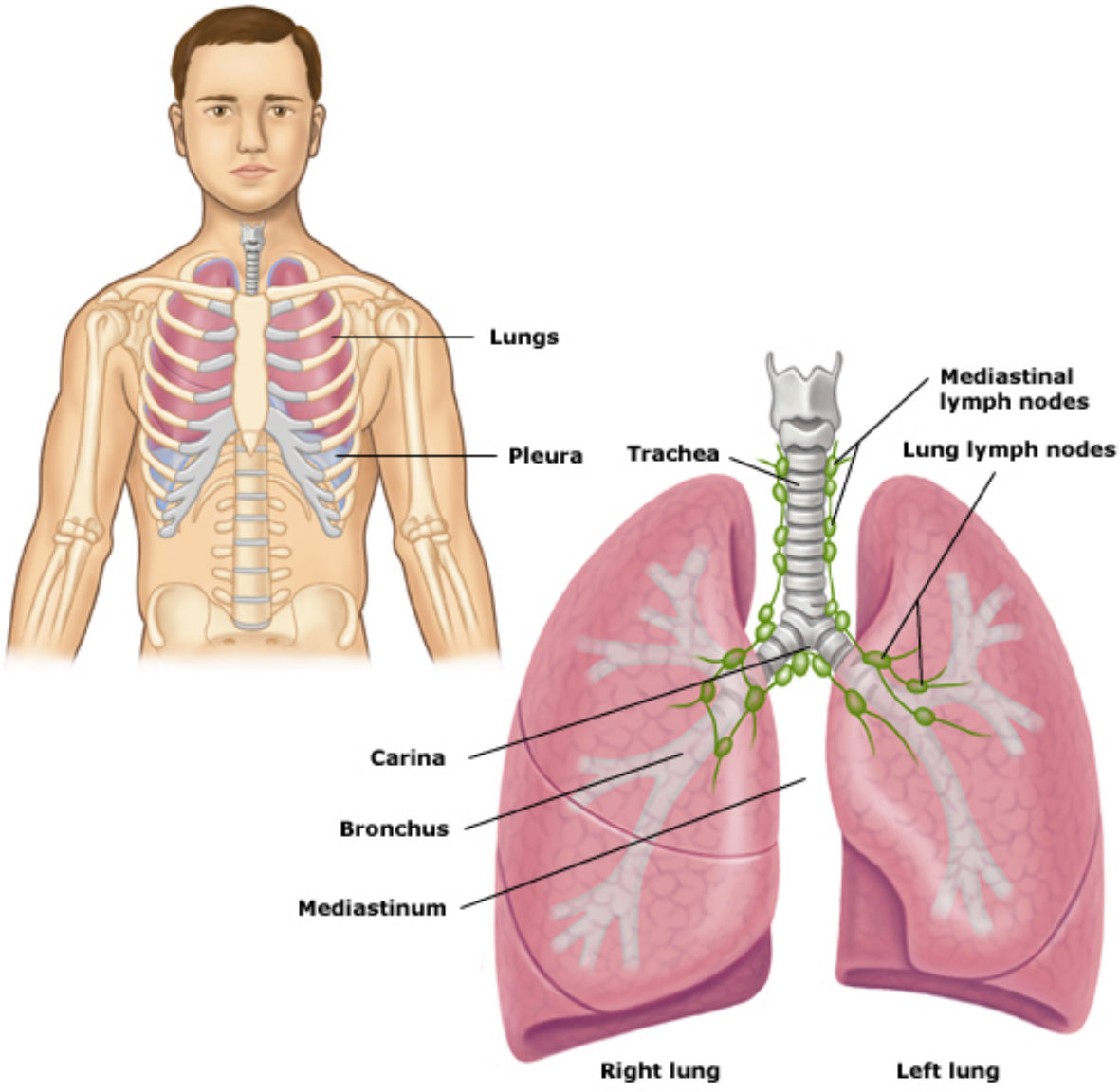

Figure 6.1: Anatomy of the lungs.

Tumor is an abnormal mass of tissue that results when cells divide more frequently than normal cells or do not die when they should. Tumors may be *benign* (not cancerous) or *malignant* (cancerous).

Lung cancer usually initiates in the lining of the bronchi, but can also start in other areas of the respiratory system, including the trachea, bronchioles, or alveoli. Lung cancer usually does not cause *symptoms* when it first develops, but they often become present after the tumor begins growing and develop over a period of many years. A cough is one the most common symptoms of lung cancer.

When a patient is suspected of having lung cancer, in the light of the symptoms and an abnormal chest radiograph, the first step is to confirm with biopsies the nature of the tumor lesions. Then, the most frequently technique used for diagnosis is bronchoscopy, which allows the doctor to view the inside of the airways and obtain samples.

After the diagnosis phase, it is necessary to assess the overall state of health to understand the capacity to resist the aggressive treatments for cancer. At the same time, we need to obtain an accurate knowledge about the degree of extension of the tumor and the absence or presence of spread of distant metastasis (staging phase). That knowledge will allow us to select the best treatment for the patient.

6.3 Mediastinal staging of non-small cell lung cancer

6.3.1 Grading and staging of cancer (in general)

The *grading* of cancer consists of classifying tumor cells in terms of how abnormal they appear when examined under a microscope. The objective of a grading system is to provide information about the probable growth rate of the tumor and its tendency to spread. Grading plays an important role when selecting the best treatment for the patient. The systems used to grade tumors vary with each type of cancer.

Staging is the process of determining a descriptor of how much the cancer has spread. The common elements considered in most staging systems are:

- Location of the primary tumor.
- Tumor size and number of tumors.
- Lymph node involvement (spread of cancer into lymph nodes).
- Cell type and tumor grade (how closely the cancer cells resemble normal tissue).

- Presence or absence of metastasis.

Staging is important because:

- it helps the doctor plan a person's treatment;
- it can be used to establish the person's prognosis (likely outcome or course of the disease).
- it contributes to identify the most suitable trials (research studies) that may be suitable for a particular patient.

Staging helps researchers and health care providers exchange information about patients. It also gives them a common language for evaluating the results of clinical trials and comparing the results of different trials.

6.3.2 Types (grading) and staging of lung cancer

Types of lung cancer

Nearly all lung cancers are carcinomas, a type of cancer that begins in the lining or covering tissues of an organ. The tumor cells of each type of lung cancer grow and spread differently, and each type requires different treatment. More than 95 percent of lung cancers belong to the group called *bronchogenic carcinoma*.

Lung cancers are generally divided into two types:

- **Small cell lung cancer (SCLC)**. Sometimes called *oat cell cancer* because the cancer cells may look like oats when viewed under a microscope. It grows rapidly and quickly spreads to other organs. SCLC, which amounts the 20% of patients with lung cancer, is usually non-operable and must be treated with chemotherapy or chemo-radiotherapy.
- **Non small cell lung cancer (NSCLC)**. It is more common than SCLC (80% of patients with lung cancer). The three main kinds of NSCLC are named for the type of cells in the tumor: *squamous cell carcinoma*, *adenocarcinoma*, and *large cell carcinoma*. If this cancer is diagnosed early, patients can be cured by surgery.

TNM staging

The TNM staging system for lung cancer (Lababede et al., 1999), developed by the American Joint Committee on Cancer, remains an important guide to the treatment and

Factor	Meaning
T	The characteristics of primary tumor: size, endobronchial location, local invasion, and other characteristics
N	Regional lymph node involvement
M	Presence of metastasis

Table 6.1: Meaning of T, N and M factors.

prognosis of lung cancer. It defines three factors for achieving a consistent reproducible description of the extent of anatomic involvement:

The **T factor** indicates the characteristics of the primary tumor in size (the smaller, the better), location within the bronchi (better when it is more peripheral), local invasion (better if it is only located in the lung and does not invade nearby structures such as pleura, ribs, and central cardiac, vascular and digestive structures, preventing the removal of the tumor). The classification of the T factor from T1 to T4 is in ascending order of severity.

The **N factor** describes whether or not the cancer has reached lymph nodes. N0 means that there is no cancer in any lymph nodes. N1 refers that there is cancer in the lymph nodes nearest the affected lung. N2 indicates that there is cancer in lymph nodes in the mediastinum but on the same side as the affected lung or there is cancer in lymph nodes just under where the trachea branches off to each lung. N3 implies that there is cancer in lymph nodes on the opposite side of the chest from the affected lung or in the lymph nodes above either collar bone or in the lymph nodes at the top of the lung.

The **M factor** refers to distant metastases. M0 indicates that and involves the inoperability of the patient. They are classified as M0 and M1.

The meaning of each factor is summarized in Table 6.1. The different categories T, N and M are presented in Table 6.2(Lababede et al., 1999).

6.3.3 Preoperative lymph node staging for non-small cell lung cancer

We present in this Section a brief outline of the different diagnostic techniques.

Imaging techniques

Computer tomography It is also called CT (computed tomography) or CAT (computerized axial tomography) scan. It is a diagnostic imaging procedure that uses a combination of X-rays and computer technology to produce cross-sectional images (often called "slices") of the body. A CT scan shows detailed images of structures inside of the body,

T (Primary Tumor)	
Tx	No primary tumor
T0	Tumor which cannot be assessed or is not apparent radiologically or bronchoscopically (malignant cells in bronchopulmonary secretions)
Tis	Carcinoma in situ
T1	Tumor with the following characteristics: 1. size: $\leq$ 3 cm 2. airway location: in lobar (relative to lobe) bronchus or distal (anatomically located far) airways 3. local invasion: none, surrounded by lung or visceral pleura (membrane around the lungs)
T2	Tumor with the following characteristics: 1. size: $>$ 3 cm 2. airway location: involvement of the main bronchus, whose distance to the carina (where the trachea divides into the left and right bronchus) is 2 cm or more, or presence of atelectasis (total or partial collapse of the lung) or obstructive pneumonitis (inflammation of lung tissue) that extends to hilar region but doesn't involve the entire lung 3. local invasion: involvement of visceral pleura
T3	Tumor with the following location or invasion: 1. size: any 2. airway location: tumor in the main bronchus (within 2 cm to the carina), or tumor with atelectasis (lungs are not fully inflated) or obstructive pneumonitis of the entire lung 3. local invasion: invasion of mediastinum, heart, great vessels, trachea, esophagus, vertebral body, or carina; or presence of malignant pleural/pericardial effusion (bloody fluid into the area located around the heart or the pericardium)
T4	Tumor with the following location or invasion: 1. size: any 2. airway location: satellite tumor nodule(s) within the ipsilateral (on the same side) primary-tumor lobe of the lung 3. local invasion: invasion of mediastinum, heart, great vessels, trachea, esophagus, vertebral body, or carina; or presence of malignant pleural/pericardial effusion
N (Lymph Nodes)	
Nx	Regional lymph nodes cannot be assessed
N0	Absence of regional lymph nodes involvement
N1	Presence of metastasis to lymph nodes nearest the affected lung: ipsilateral peribronchial (surrounding a bronchus or the bronchi) and/or ipsilateral hilar lymph nodes (the lymph nodes in the hilum or the triangular depression or indented region at the junction of each lung and its bronchi)
N2	Presence of metastasis to ipsilateral mediastinal lymph nodes (in the mediastinum but on the same side as the affected lung) and/or subcarinal lymph nodes (just under the carina, where the trachea branches off to each lung)
N3	Presence of metastasis to any of the following lymph node groups: contralateral (opposite side) mediastinal, contralateral hilar, ipsilateral or contralateral scalene (relative to one of the scalenous muscles), or supraclavicular (above collar bone)
M (Distant metastasis)	
Mx	Metastasis cannot be assessed
M0	Absence of distant metastasis
M1	Presence of distant metastasis (separate metastatic tumor nodule(s) in the ipsilateral nonprimary-tumor lobe(s) of the lung also are grouped as M1)

Table 6.2: Categories T, N, and M. See also Figures 6.1 and 6.2 for a better understanding of the anatomy of the lungs and the regional lymph nodes.

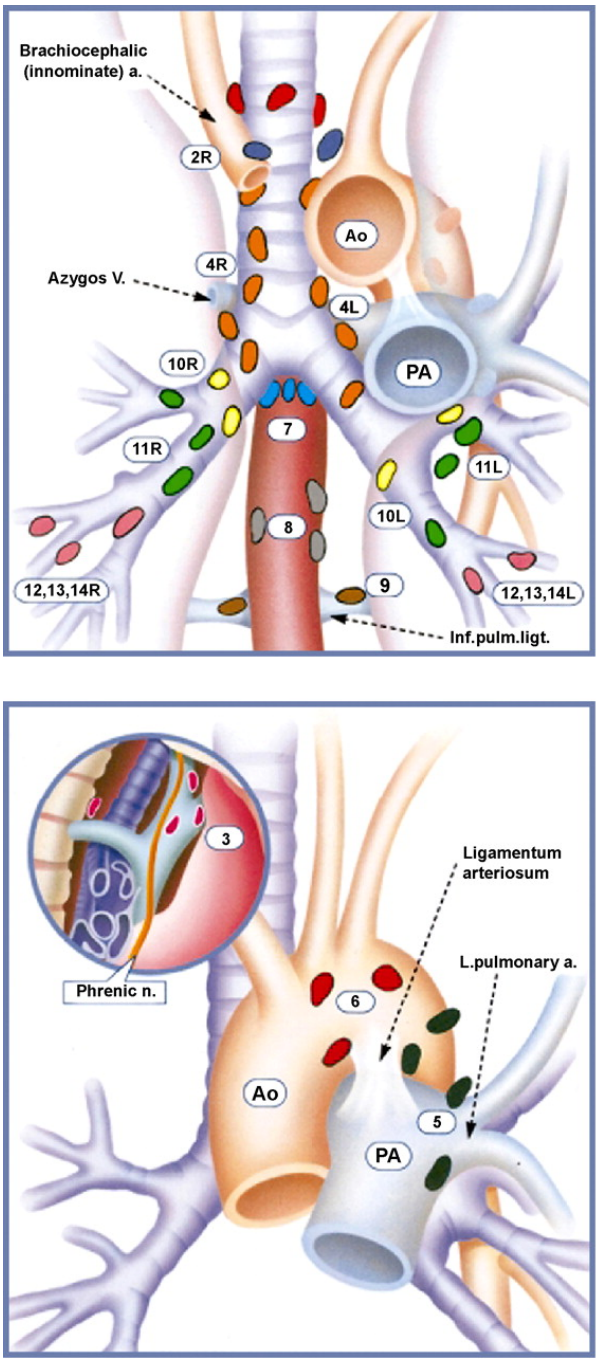

Figure 6.2: Regional lymph node stations for lung cancer staging. [Taken from `http://ejcts.ctsnetjournals.org`].

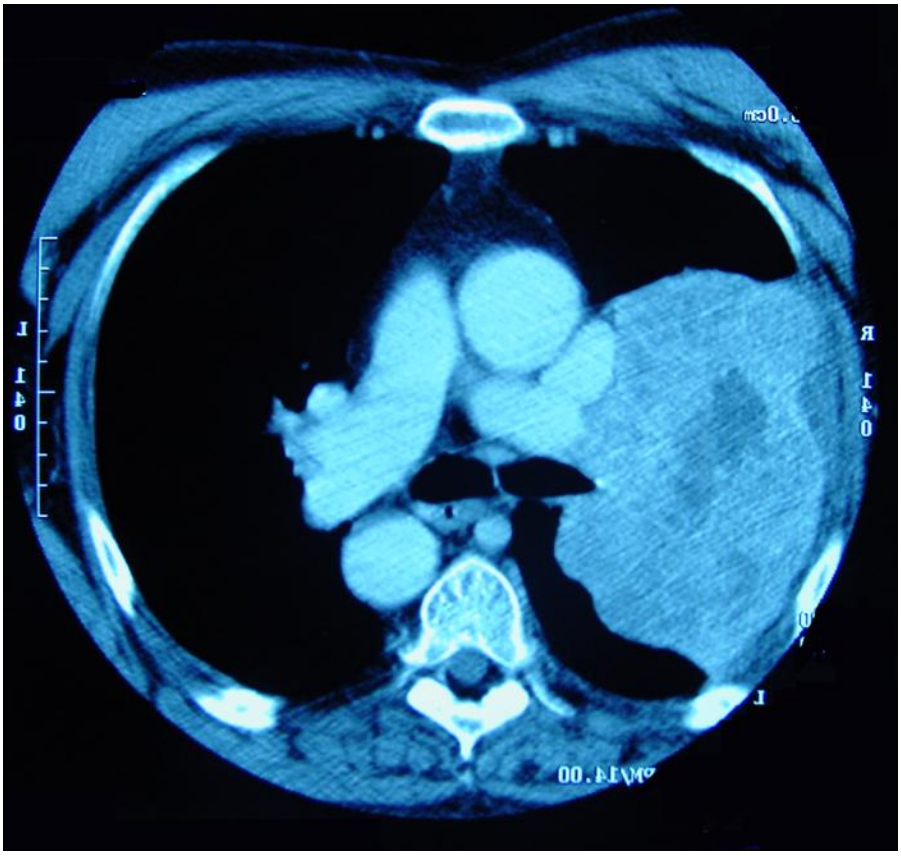

Figure 6.3: Image of a CT scan. [Taken from `http://www.tobacco-facts.info/images_html/lung_cancer_ct-scan-1.htm`].

including the bones, muscles, fat, and organs. CT scans are more detailed than general X-rays.

Computed tomography is currently used to radiographically stage the mediastinum in patients with primary bronchogenic cancer. The accuracy of CT in staging the mediastinum has been controversial over the last years, largely due to variability of patient selection and study design (Lloyd and Silvestri, 2001). Mediastinal nodes larger than 1 cm on the short axis are defined as pathologic.

Several factors affect the accuracy of CT for staging bronchogenic carcinoma. Accuracy is lessened by central tumors, obstructive pneumonia, and prior granulomatous disease.

Figure 6.3 presents the image of a CT scan.

The sensitivity and specificity of CT for mediastinal staging is approximately only 50-70% and 60-80%, respectively (data taken from `http://www.oncoline.nl`).

Position Emission Tomography Positron emission tomography (PET) (see Figure 6.4) was introduced as an alternative or complementary technique for mediastinal staging. Patients are injected with fluorine-18 fluorodeoxyglucose (FDG) and imaged using PET. To

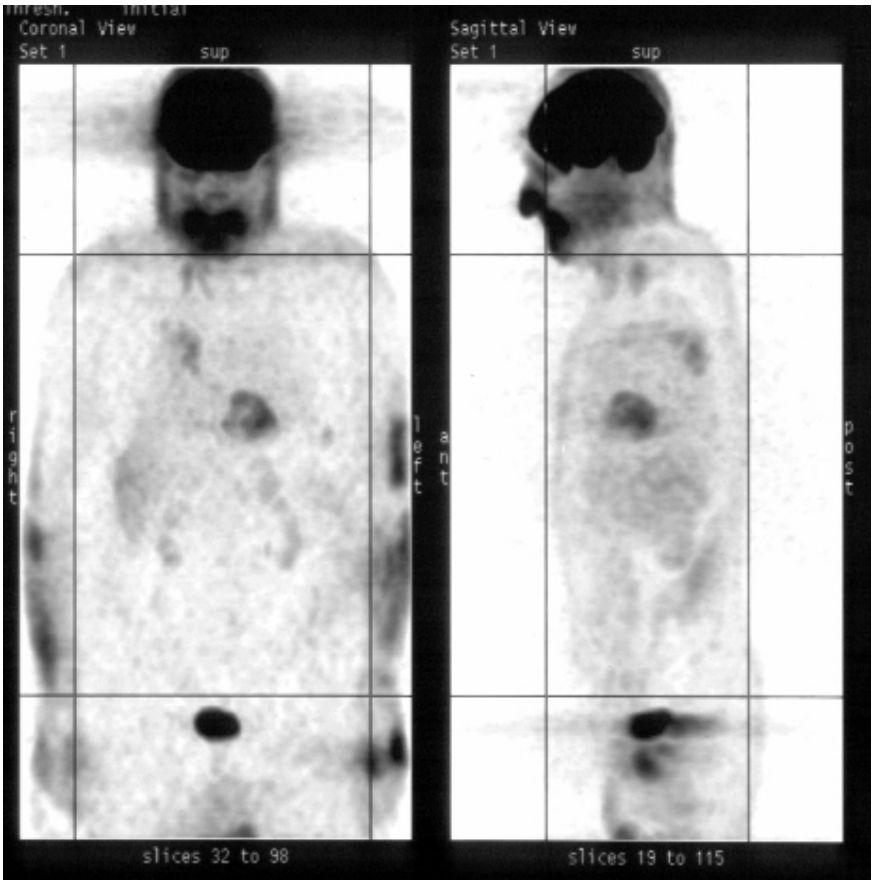

Figure 6.4: A PET image.

identify malignancy, PET depends on demonstrating metabolic differences between tumor and normal tissue. Potential advantages of PET imaging include the identification of tumor foci in normal-sized lymph nodes.

PET appears to be more accurate than conventional CT scanning for staging the mediastinum. Unlike CT, PET relies on increased metabolic activity rather than solely on the size of the lymph nodes. It can identify not only disease in nodes smaller than 1 cm, but also unsuspected extrathoracic metastases. Disadvantages include difficulty with accurate anatomic placement of lesions as well as with extent of local tumor involvement, both necessary for staging.

Figure 6.4 presents the image of a PET.

The sensitivity and specificity of PET for mediastinal staging is approximately 83% and 64% respectively (data taken from `http://www.ncpic.info`).

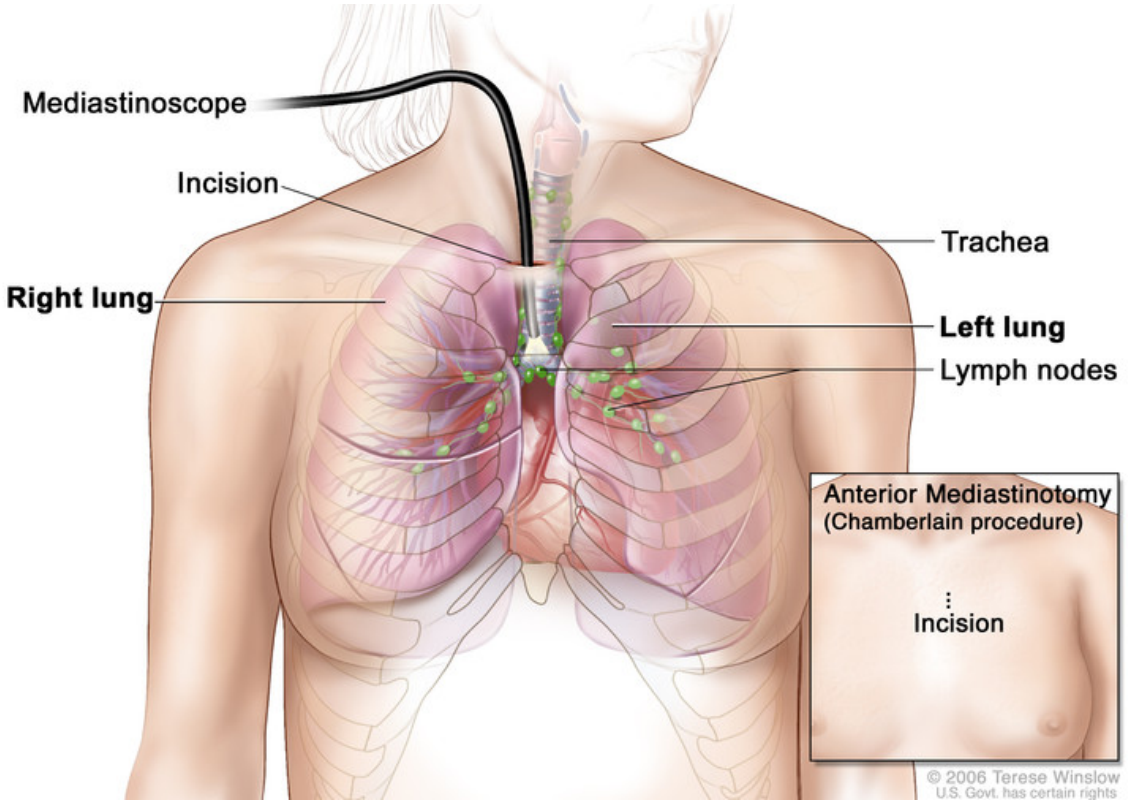

Figure 6.5: A mediastinoscopy image.[Taken from `http://visuals.nci.nih.gov/preview.cfm?imageid=7242&fileformat=jpg`].

Invasive techniques

Mediastinoscopy It is a process in which a small incision is made in the neck (see Figure 6.5). Then a thin scope (mediastinoscope) is inserted through the opening. A tissue sample can be collected through the mediastinoscope and then examined under a microscope. Mediastinoscopy is the historic gold standard for mediastinal staging.

Mediastinoscopy provides access to mediastinal lymph nodes of stations 2, 4, and 7, according to the regional division of lymph nodes presented in Figure 6.2. Lymph nodes of stations 5 and 6 are accessible by *cervical extended mediastinoscopy* and *anterior mediastinoscopy* (see Figure 6.5).

Overall, mediastinoscopy has a reported sensitivity of 87% and specificity of 100% (data taken from (Luke et al., 1986)).

Transbronchial needle aspiration Transbronchial needle aspiration (TBNA) was developed in the 1980s as a method of sampling extrabronchial lesions through a flexible

tube, called bronchoscope. TBNA requires the same preparation as bronchoscopy[2], i.e, conscious sedation, and has a low risk of morbidity and mortality. TBNA can be performed at the same time as diagnostic bronchoscopy, thus avoiding a separate staging intervention. During the TBNA, a needle is passed through the working channel of the bronchoscope and through the wall of the trachea or bronchus into the underlying lymph node. Needles are currently available for both cytology and histology.

A factor limiting the widespread use of TBNA is training. TBNA is the most operator-dependent of the bronchoscopic procedures.

Overall, the sensitivity of TBNA is between 40% and 80% and its specificity is approximately 99% (Lloyd and Silvestri, 2001).

Endobronchial ultrasound (EBUS) The EBUS technology combines ultrasound guidance and a bronchoscopy, which permits to perform real-time transbronchial needle aspiration. Biopsies using EBUS are performed through the trachea using ultrasound, which is less invasive than surgical incisions. EBUS allows the physician to look in areas of the chest where it is traditionally difficult to biopsy. It offers a more precise way of assessing a patient's lymph nodes and determining if lung cancer has spread to other parts of the body. EBUS allows for complete staging of the mediastinum through a less invasive approach.

In an EBUS, the patient is placed under general anesthesia and a small bronchoscope with a special ultrasound at its tip is passed through the patient's mouth down into the trachea (see Figure 6.6). The scope has a small instrument at its tip called a transducer, which can be pointed in different directions to produce images of lymph nodes and other structures in the mediastinum.

In many cases, EBUS offers better sensitivity and specificity without the need to make an incision.

The sensitivity of EBUS is between 90 and 94% and its specificity is 100% (Yasufuku et al., 2006).

Endoscopic ultrasound (EUS) It is an endoscopic technique that allows a miniaturized ultrasound probe to be driven though the mouth into the upper gastrointestinal tract to investigate organs and structures close to the esophagus, stomach, or duodenum, such as the lung (see Figure 6.7). When staging lung cancer, it is done by advancing a fine

[2]A bronchoscopy is the examination of the bronchi using a bronchoscope. Bronchoscopy helps to evaluate and diagnose lung problems, assess blockages, obtain samples of tissue and/or fluid, and/or to help remove a foreign body.

Figure 6.6: EBUS image.[Taken from `http://www.ctsnet.org/graphics/experts/Thoracic/d_rice_endobronc_ultrasnd/Figure-3.jpg`].

needle through the esophagus into adjacent lymph nodes. EUS provides accurate images of the entire posterior mediastinum, including lymph nodes in the subcarinal, aortopulmonary, and paraesophageal regions. This has led to studies of its diagnostic accuracy for mediastinal malignancy.

Some advantages of EUS are its low cost and morbidity. It can be used to biopsy lymph nodes levels that are inaccessible or difficult to access by mediastinoscopy, including the para-aortic and paraesophageal regions. Its advantages over TBNA include the ability to visualize and biopsy nodes smaller than 1 cm with good accuracy. However, EUS cannot biopsy right paratracheal (adjacent to the trachea) and pretracheal nodes due to the air-filled trachea, which blocks the ultrasound signal.

The sensitivity of EUS is 89% and its sensitivity is 100% (Silvestri et al., 1996).

6.3.4 Treatment of lung cancer

We describe in this section the main treatment options for lung cancer, namely surgery, chemotherapy, radiation therapy, combination therapy, and palliative and supportive care. The applicability of each treatment depends on the stage of the tumor.

Surgery

Depending on the type and stage of the cancer, surgery may be used to remove the tumor and some of the lung tissue around it. If a lobe (section) of the lung is removed, the surgery is called a lobectomy. If the entire lung is removed, the surgery is called a pneumonectomy. These operations are done with the patient asleep under general anesthesia. A hospital stay of about 1 week is usually needed. The patient will have some pain after the surgery because the surgeon has to cut through the ribs to get to the lungs.

People who do not have any other lung problems (other than cancer) can often return to their normal activities after a lobe or even an entire lung is removed. However, if they also have diseases such as emphysema or chronic bronchitis (common among heavy smokers), then they may find that their shortness of breath gets worse. For people who cannot have the usual surgery because they have lung disease or other medical problems or because the cancer is widespread, other types of surgery (for example, laser surgery) can be done to relieve symptoms.

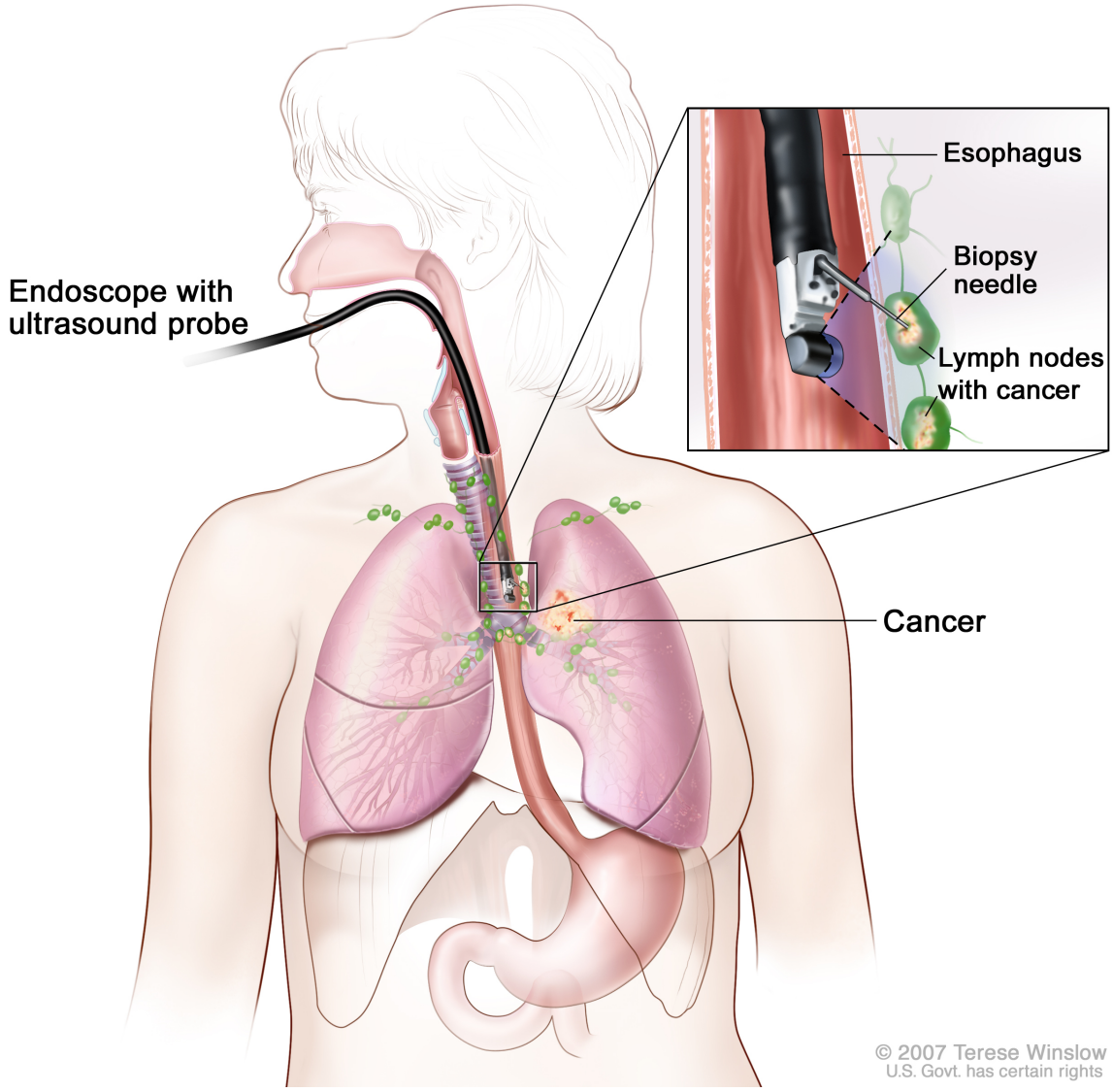

Figure 6.7: EUS image.[Taken from `http://www.meb.uni-bonn.de/cancer.gov/Media/CDR0000466552.jpg`].

Chemotherapy

Chemotherapy refers to the use of drugs to kill cancer cells. The drugs are usually given into a vein or by mouth. Once the drugs enter the bloodstream, they reach all parts of the body. Often several drugs are given at the same time. Depending on the type and stage of lung cancer, chemotherapy may be given as the main treatment or in addition to surgery and/or radiation therapy. Chemotherapy is referred to an adjuvant therapy (that enhances the effectiveness of a medical treatment) when it is used along with surgery or radiation therapy. It is used in this way to reduce the risk that the cancer will recur or spread outside the lung. Doctors who prescribe these drugs generally use a combination of medicines that have proven to be more effective than a single drug. Doctors give chemotherapy in cycles, with each period of treatment followed by a recovery period.

Chemotherapy has side effects. Temporary side effects might include loss of appetite, nausea and vomiting, mouth sores, and hair loss. Because chemotherapy can damage the blood-producing cells of the bone marrow, a drop in white blood cells increases the risk of infection; a shortage of blood platelets can cause bleeding or bruising after minor cuts or injuries; and a decrease in red blood cells (low blood hemoglobin levels) can lead to fatigue. Other effects from anticancer drugs are premature menopause, infertility, or heart or lung damage.

Radiation Therapy

Radiation therapy uses high-energy rays (such as X-rays) to kill or shrink cancer cells. The radiation may come from outside the body (external radiation) or from radioactive materials placed directly in the tumor (internal or implant radiation, also called *brachytherapy*). External radiation is the type most often used to treat lung cancer.

External beam radiation is sometimes used as the main treatment of lung cancer, for example, for those who may not be healthy enough to have surgery or whose cancer has spread too far to be removed by surgery. For other patients, radiation might be used after surgery to kill small areas of cancer that can't be seen and removed during surgery. Radiation can also be used to relieve symptoms such as pain, bleeding, or blockage of air passages by the cancer; or as a treatment of lung cancer that has spread to other organs, such as the bone or brain.

Side effects of external radiation therapy to the chest area may include mild skin reactions, nausea, tiredness, pain on swallowing, and a cough. Radiation to the chest may also cause lung damage and difficulty breathing. Side effects of radiation therapy to the brain usually become most serious 1 or 2 years after treatment and include headaches

and trouble thinking.

Brachytherapy, also known as internal radiation therapy, is rarely used as the initial treatment for a lung cancer, but is sometimes recommended if cancer has returned and is blocking one of the airways.

Combined Therapy

Lung cancer is often initially treated with combination therapy; that is, the combination of surgery with chemotherapy, or radiation therapy, or both, either before or after the surgery. Chemotherapy and radiation therapy may be used before surgery to shrink the tumor so that it can be removed surgically. Chemotherapy and radiation therapy may be given after surgery if there is a chance that the entire tumor was not removed at surgery.

Palliative and Supportive Care

Sometimes patients receive cancer treatment intended to reduce or prevent symptoms, even though is not expected to cure the cancer. Palliative care may include radiation or chemotherapy treatments that relieve symptoms by shrinking the tumor. Some other palliative treatments for lung cancer include laser surgery and photodynamic therapy.

Cancer treatments identify ways to cure some people with lung cancer and to help others live longer by removing or destroying lung cancer cells. Nevertheless, another important goal is controlling symptoms and helping a patient continue to do the things important to her. There are effective and safe ways to treat pain, most other symptoms of lung cancer, and most of the side effects caused by lung cancer treatments. Care to help relieve symptoms is sometimes called palliative care, or supportive care.

Pain is a significant concern for patients with lung cancer. Growth of the cancer around certain nerves may cause severe pain. For most patients, treatment with morphine or other opioids will reduce the pain considerably.

In addition to the supportive care measures for people with advanced cancer, a patient may also benefit from specific measures that relieve some symptoms of lung cancer that are relatively rare with other cancers.

6.4 Construction of MEDIASTINET

In this section, we describe the construction of MEDIASTINET, a decision support-system (DSS) for the mediastinal staging of non-small cell lung cancer (NSCLC). The system

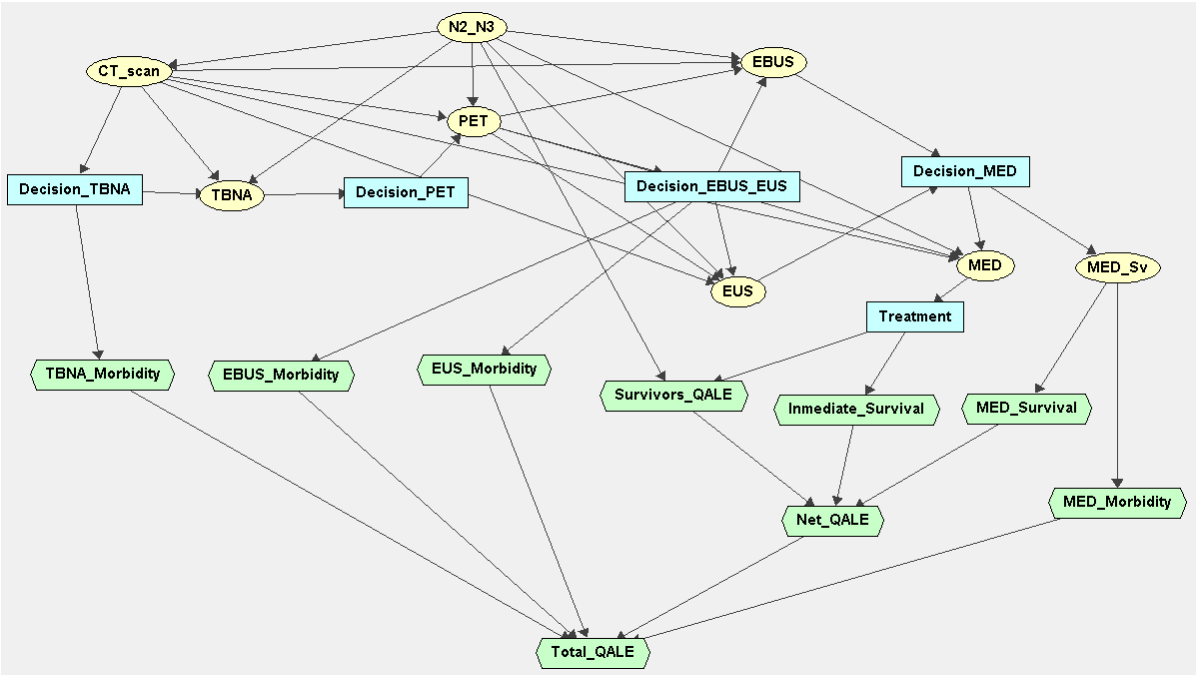

Figure 6.8: Influence diagram of MEDIASTINET.

basically consists of an influence diagram built using Elvira, a free-software package developed as a joint project of several Spanish universities (Elvira Consortium, 2002).

6.4.1 Construction of the structure of the graph

A graph is basically a set of nodes (or variables) and set of arcs relating them (see Section 2.1). We describe in this section how we have built the graph of the model (see Figure 6.8). The process of identifying the variables of the problem, their domains and their relations in MEDIASTINET has been performed with the expert's help. We describe now that work.

Identification of variables

We have identified three types of variables, which correspond to the three types of nodes that can appear in an influence diagram: chance, decision and utility (see Section 2.3).

Chance variables In medical diagnosis, chance variables usually correspond to possible *causes* and *risk factors* of a *disease*, as well as the *symptoms*, *signs* and *laboratory tests* that allow us to confirm or discard the presence of the disease.

Given that our objective is the mediastinal staging of NSCLC, we have included a variable representing the value of N in the TNM classification (see Section 6.3.2). Even

though the N factor takes on four possible values, from N0 to N3, we have modeled it as a binary variable: we have grouped N0 and N1 into a state, which means that the cancer is operable, and N2 and N3 into another state, which means that it is inoperable. This variable has been named *N2N3*, and its domain is *present / absent.*

The laboratory tests that can be performed are represented by the following variables:

- *CT_scan*: the result of computer tomography;
- *TBNA*: the result of TBNA;
- *PET*: the result of PET;
- *EBUS*: the result of EBUS;
- *EUS*: the result of EUS;
- *MED*: the result of mediastinoscopy.

Each of these variables has two states: *positive* and *negative.*

When the mediastinoscopy is performed, there is a small probability that the patient does not survive. We have therefore created the variable *MED_Sv*, whose states are *yes* and *no*, representing whether the patient has survived mediastinoscopy.

Decision variables The decisions are whether to perform a medical test and what possible treatments to apply to a patient.

The set of possible treatments are represented by the variable *Treatment.* Its states are *thoracotomy*, *chemotherapy*, and *palliative.*

The decisions about whether to perform the different laboratory tests have been represented by the following variables:

- *Decision_TBNA*, *Decision_PET*, and *Decision_MED*, whose states are *yes* and *no.*
- *Decision_EBUS_EUS*: The decision about whether to perform EBUS, EUS, both of them, or none. It has four states: *ebus*, *eus*, *ebus+eus* and *none.*

These decisions force us to introduce a new state to some of the chance variables to reflect that when we do not perform a medical test its result is not available. Thus, the state *no_result* is added to the variables *TBNA*, *PET*, *EBUS*, *EUS*, and *MED.*

CT scan is always performed to a patient because it gives very useful information about the thorax and can inform about patients which can be discarded initially from being treated with surgery.

Ordinary utility nodes The decision maker's preferences have been represented by a set of utility nodes.

The QALE (quality-adjusted life expectancy) of the survivors to the medical tests (except the mediastinoscopy) and the treatment is represented by the node *Survivors_ QALE*.

The morbidities due to TBNA, EBUS, EUS, and mediastinoscopy, are depicted by *TBNA_ Morbidity*, *EBUS_ Morbidity*, *EUS_ Morbidity*, and *Med_ Morbidity* respectively, and measured in QALYs.

Med_ Survival indicates if the patient has survived to the mediastinoscopy.

The probability of survival to the treatment is represented by *Immediate_ Survival*.

Super value nodes The ordinary utility nodes presented above have been combined by using super-value nodes, as proposed by Tatman and Shachter (1990). Nodes *Survivors_ QALE*, *Med_ Survival* and *Immediate_ Survival* have been combined into the product node *Net_ QALE*. We have used a product because *Survivors_ QALE* indicate the QALE of the survivors to the medical tests (except to the mediastinoscopy) and the treatments, and we need to multiply it by the probability of survival to the mediastinoscopy (*MED_ Survival*) and to the treatment (*Immediate_ Survival*) to obtain a net qale (*Net_ QALE*).

The nodes *Net_ QALE, TBNA_ Morbidity*, *EBUS_ Morbidity*, *EUS_ Morbidity*, and *Med_ Morbidity* have been added into the node *Total_ QALE*. We have used a sum node because morbidities decreases of the medical tests decrease the QALE of patients (*Net_ QALE*), what have to been reflected in the diagram by a sum, considering that the utilities of nodes representing morbidities will take non-positive values.

Arcs of the graph

Influence diagram of Figure 6.8 contains four kinds of arcs (see Section 2.3):

1. **Arcs into chance nodes.** They represent probabilistic dependencies. In our diagram, an arc from a node representing the decision of a test , such as the arc $Decision\_TBNA \rightarrow TBNA$, indicates that the result (in this case *TBNA*) is only available whether we perform the test ($Decision\_TBNA = yes$).

2. **Arcs into decision nodes.** They imply informational precedence. Based on the "no-forgetting" assumption we have specified in our diagram the minimum set of informational arcs (Nielsen and Jensen, 1999). For example, the arc $CT\_scan \rightarrow Decision\_TBNA$ has been included in MEDIASTINET (see Figure 6.8) to indicate that *CT_ scan* is known in *Decision_ TBNA*. However, the arc *CT_ scan* $\rightarrow$

Decision_PET has not been specified because we are using the no-forgetting assumption.

3. **Arcs into ordinary utility nodes.** They represent functional dependencies. The parents of a utility node indicate the domain of the associated utility function. For example, the arcs into the node *Immediate_Survival* means that the domain of its utility function consists of nodes *N2N3* and *Treatment*.

4. **Arcs into super value nodes.** They indicate the set of utility nodes that are combined into the super value node. For instance, arcs into super value node *Net_QALE* indicate that is the combination of *Survivors_QALE*, *MED_Survival* and *Immediate_Survival*.

6.4.2 Numerical values

When the graph of the ID has been constructed, it is necessary to complete the quantitative part of the ID (see Section 2.3), which consists of a set of probability and utility potentials. For example, for each chance node C we must give a conditional probability potential $p(C|pa(C))$ for each configuration of its parents, $pa(C)$. Then, the table for $p(C|pa(C))$ requires $|dom(C)| \cdot \prod_{X \in pa(C)} |dom(X)|$ numbers, but given the restriction that $\sum_c P(c|pa(C)) = 1$, only some of them are independent.

When trying to elicit the parameters of the model, we asked the expert to estimate only a certain number of independent parameters. In our study we have chosen a set of 61 independent parameters (see Table 6.3). One of them is the prior probability of metastasis (*N2N3*); 46 parameters are conditioned probabilities of the tests; one is the probability of surviving *MED*; 10 are utilities measured in QALYs, and 3 parameters are also utilities that represent the probability of surviving the treatment.

Table 6.3: Independent parameters of MEDIASTINET.

Name of the parameter	Value assigned
prevalence_N2_N3	0.281
sens_CT_scan	0.510
spec_CT_scan	0.857
sens_TBNA_CT_scan_pos	0.460
spec_TBNA_CT_scan_pos	0.904
Continued on next page	

Table 6.3 – continued from previous page

Name of the parameter	Value assigned
sens_TBNA_CT_scan_neg	0.020
spec_TBNA_CT_scan_neg	0.921
sens_PET_CT_scan_pos	0.905
spec_PET_CT_scan_pos	0.775
sens_PET_CT_scan_neg	0.740
spec_PET_CT_scan_neg	0.925
sens_EBUS_CT_scan_pos_PET_pos	0.879
sens_EBUS_CT_scan_pos_PET_neg	0.889
sens_EBUS_CT_scan_neg_PET_pos	0.892
sens_EBUS_CT_scan_neg_PET_neg	0.881
spec_EBUS_CT_scan_pos_PET_pos	0.966
spec_EBUS_CT_scan_pos_PET_neg	0.976
spec_EBUS_CT_scan_neg_PET_pos	0.967
spec_EBUS_CT_scan_neg_PET_neg	0.975
sens_EBUS_CT_scan_pos	0.919
spec_EBUS_CT_scan_pos	0.974
sens_EBUS_CT_scan_neg	0.892
spec_EBUS_CT_scan_neg	0.978
sens_EUS_CT_scan_pos_PET_pos	0.861
sens_EUS_CT_scan_pos_PET_neg	0.868
sens_EUS_CT_scan_neg_PET_pos	0.581
sens_EUS_CT_scan_neg_PET_neg	0.567
spec_EUS_CT_scan_pos_PET_pos	0.935
spec_EUS_CT_scan_pos_PET_neg	0.933
spec_EUS_CT_scan_neg_PET_pos	0.926
spec_EUS_CT_scan_neg_PET_neg	0.938
sens_EUS_CT_scan_pos	0.857
spec_EUS_CT_scan_pos	0.929
sens_EUS_CT_scan_neg	0.762
spec_EUS_CT_scan_neg	0.923
sens_MED_CT_scan_pos_PET_pos	0.800
spec_MED_CT_scan_pos_PET_pos	0.950

Continued on next page

Table 6.3 – continued from previous page

Name of the parameter	Value assigned
sens_MED_CT_scan_pos_PET_neg	0.813
spec_MED_CT_scan_pos_PET_neg	0.941
sens_MED_CT_scan_neg_PET_pos	0.786
spec_MED_CT_scan_neg_PET_pos	0.947
sens_MED_CT_scan_neg_PET_neg	0.800
spec_MED_CT_scan_neg_PET_neg	0.938
sens_MED_CT_scan_pos	0.813
spec_MED_CT_scan_pos	0.929
sens_MED_CT_scan_neg	0.727
spec_MED_CT_scan_neg	0.944
MED_sv_Decision_MED_yes	0.963
morbidity_MED	−0.050
morbidity_EBUS	−0.030
morbidity_EUS	−0.030
morbidity_TBNA	0.000
thor_inmediate_survival	0.909
chem_inmediate_survival	0.980
pall_inmediate_survival	0.981
survivors_qale_N2_N3_pos_thor	0.660
survivors_qale_N2_N3_pos_chem	0.830
survivors_qale_N2_N3_pos_pall	0.500
survivors_qale_N2_N3_neg_thor	3.000
survivors_qale_N2_N3_neg_chem	2.000
survivors_qale_N2_N3_neg_pall	1.250

Data in Table 6.3 were assessed by the expert based on the literature.

We have used a convention for naming the parameters of MEDIASTINET. For example, the parameter *sens_EBUS_CT_scan_po_PET_po* refers to the sensitivity of EBUS when the CT scan is positive positive and the PET is also positive. In contrast, the parameter *sens_EBUS_CT_scan_po* refers to the sensitivity of EBUS when the CT scan is positive and the PET has not been performed.

6.4.3 Cost-effectiveness with MEDIASTINET

The version of MEDIASTINET presented above does not include the economic costs of the diagnostic tests and the treatments. However, in medical decision making costs cannot be ignored. Including the economic cost turns our medical problem into a multiobjective optimization problem (Steuer, 1986). In Section 2.6.1 presented the basic techniques of cost-effectiveness analysis (CEA).

Nevertheless, instead of basing our CEA on the incremental cost-effectiveness ratios (ICERs), which is the standard method, we will apply a different perspective: the maximization of the net benefit, defined in Equation 2.20, because it is easier to integrate with IDs. As we explained in that section, both approaches are equivalent, .i.e, they return the same selection of interventions.

Actually, instead of applying Equation 2.20, we have built our ID by using an equivalent equation obtained by dividing it by λ:

$$NE = E - \lambda^* C, \tag{6.1}$$

where $\lambda^* = 1/\lambda$, and NE denotes the *net effectiveness*, which can be interpreted as the medical benefit minus the economic costs (converted into medical units). The reason for using this equation instead of Equation 2.20 is explained below.

The integration of Equation 6.1 in MEDIASTINET is as follows (see Figure 6.9):

- The cost, C, is represented by a super value node, *Total_Economic_Cost*, whose parents are the utility nodes *Economic_Cost_CT_scan*, *Economic_Cost_PET*, *Economic_Cost_TBNA*, *Economic_Cost_EBUS*, *economic_Cost_EUS*, and *economic_Cost_MED.*

- The effectiveness, E, is represented by another super value node, *Total_QALE*, whose structure of utility nodes was explained in Section 6.4.1.

- The parameter λ^*, the inverse of λ, is represented by an ordinary utility node, *C2E* (cost to effectiveness), without parents.

- The node *Weighted_Economic_Cost* is a super value node of type product, which represents $\lambda^* C$.

- *Net_Effectiveness* is a super value node of type sum, which represents the net effectiveness (Equation 6.1).

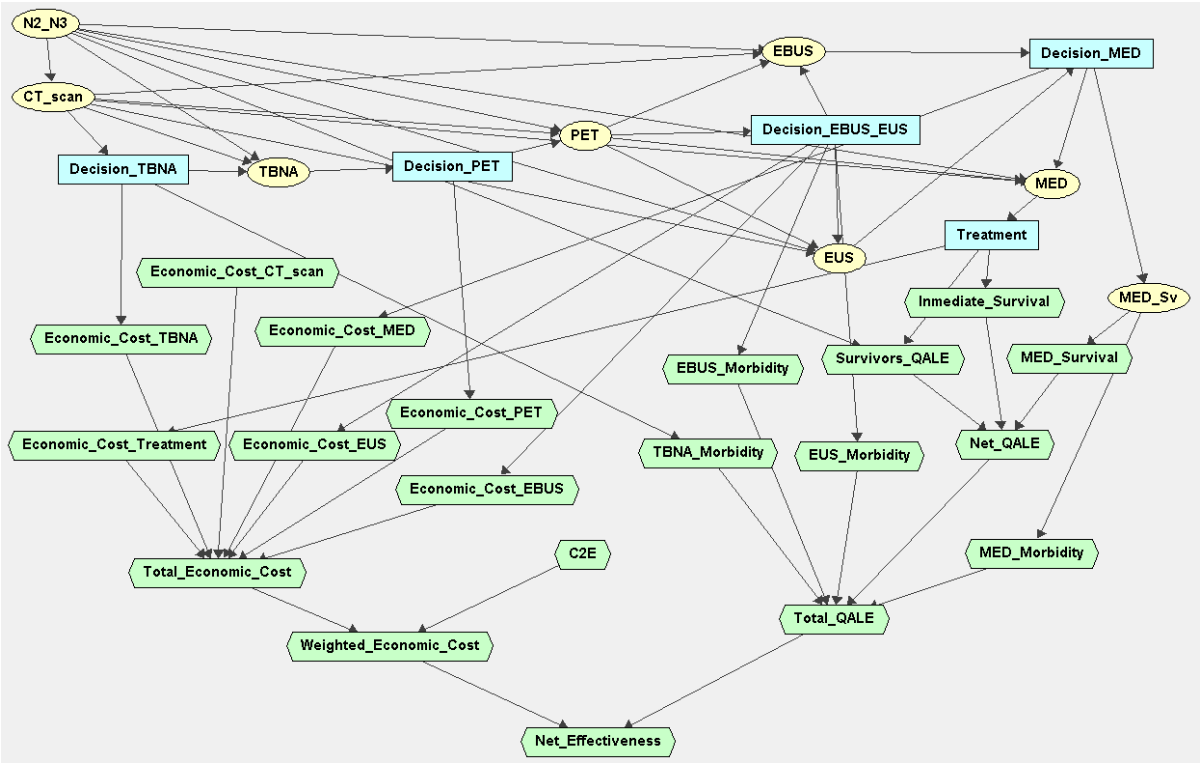

Figure 6.9: A new version of MEDIASTINET, including economic costs.

If we make $\lambda^* = 0$ the evaluation of the ID returns the strategy that maximizes the effectiveness, without taking into account the economic costs. The human expert collaborating with us was very interested in knowing this strategy, which turns out to be different from the one obtained with the value of $\lambda = 30,000$ €/QALY, used as a reference point for the Spanish public health system (Sacristán et al., 2002) (see Section 6.5). In fact, the strategy that maximizes the effectiveness can be seen as a component of the explanation of reasoning.

This justifies why in our ID we have used Equation 6.1 instead of 2.23: because when looking for the maximum-effectiveness strategy (without caring about the costs), it suffices to make $\lambda^* = 0$. In contrast, making $\lambda = +\infty$ in Equation 2.20 would lead to an infinite net benefit (NB $= +\infty$) for any strategy having positive effectiveness, thus turning the maximization into an intractable problem.

Numerical values

Analogously to the the case of Section 6.4.2, we have defined a set of minimal parameters for the new nodes added in the new version of MEDIASTINET (Figure 6.9). The new parameters are shown in Table 6.4. The names of the parameters use a similar convention to those in Table 6.3. For example, *cost_pall* denotes the economic cost of palliative

Name of the parameter	Value assigned
cost_CT	670 €
cost_TBNA	80 €
cost_MED	1620 €
cost_PET	2250 €
cost_EBUS	620 €
cost_EUS	620 €
cost_thor	19646 €
cost_chem	11242 €
cost_pall	3000 €
lambda	30000 €/QALY

Table 6.4: Parameters of economic costs of MEDIASTINET.

treatment. The last parameter in table, *lambda*, denotes the term λ in Equation 2.23, which is the inverse of λ^*.

.

Data in Table 6.4 were assessed by the expert based on the literature.

6.5 Optimal strategies

In this section we show two strategies returned by MEDIASTINET with two different criteria: the maximization of the effectiveness (disregarding costs) and the net benefit.

6.5.1 Maximum-effectiveness strategy

The strategy that maximizes the effectiveness can be obtained from the version of MEDIASTINET that does not include economic costs (see Figure 6.8). However, instead of maintaining two versions of the ID, with and without costs, we have only the one with costs (see Fig. 6.9). As explained above, the maximum-effectiveness strategy can be obtained from this ID by making $\lambda^* = 0$ (see Eq. 6.1), i.e., by setting the utility node *C2E* to 0.

The policy tree of Figure 6.10, which corresponds to the decision *Treatment*, depicts the whole optimal strategy obtained by evaluating MEDIASTINET disregarding costs. We examine each policy:

- **Decision_TBNA**: Perform the TBNA only when CT scan is positive.
- **Decision_PET**: Perform the PET.

- **Decision_EBUS_EUS**: Perform the EBUS if we have negative CT scan and positive PET, or if we have positive CT scan, and the PET and the TBNA have contrary result, i.e., positive PET and negative TBNA, or viceversa. It recommends never to perform the EUS.
- **Decision_MED**: Never perform the mediastinoscopy.
- **Treatment**: Apply chemotherapy if the last test performed is positive; otherwise, it is better to apply thoracotomy.

Our expert's judgment

After obtaining the optimal strategy we have presented it to the expert to know his opinion about the policies obtained. We have examined the scenarios presented in the policy tree to find out the coincidences between what the expert would do and the recommendations given by the policies. We told the expert to imagine a hypothetical situation in which economic costs are disregarded.

Let us focus on Figure 6.10, which contains the entire strategy. The expert agreed completely with the policies for decisions *Decision_TBNA* and *Treatment.* Additionally, he agreed in never performing the EUS (which is decided in *Decision_EBUS_EUS*).

Then, discussion with the expert focused on seeing which of the tests *PET*, *EBUS*, and *MED*, should be performed. He agreed in the tests performed for the case of CT scan being positive. However, when CT scan is negative, he would slightly vary the decisions made after performing the TBNA:

- If the result is positive, then he would not perform any additional tests.
- Otherwise, he would perform the EBUS instead of the PET, and would also perform the mediastinoscopy if the EBUS is negative.

With respect to the main discrepancy between the expert and our strategy, which is the decision of performing the PET, he told us that the system looks intelligent by deciding to perform it, because according to the information encoded it does not take into account that performing the *PET* causes a delay and forces the patient to go again to the hospital a different day.

6.5.2 Maximum-benefit strategy

Analogously we have evaluated MEDIASTINET with economic costs. In the reference case, we have set the parameter *lambda* to a value of $\lambda = 30,000$ €/QALY, used as a reference

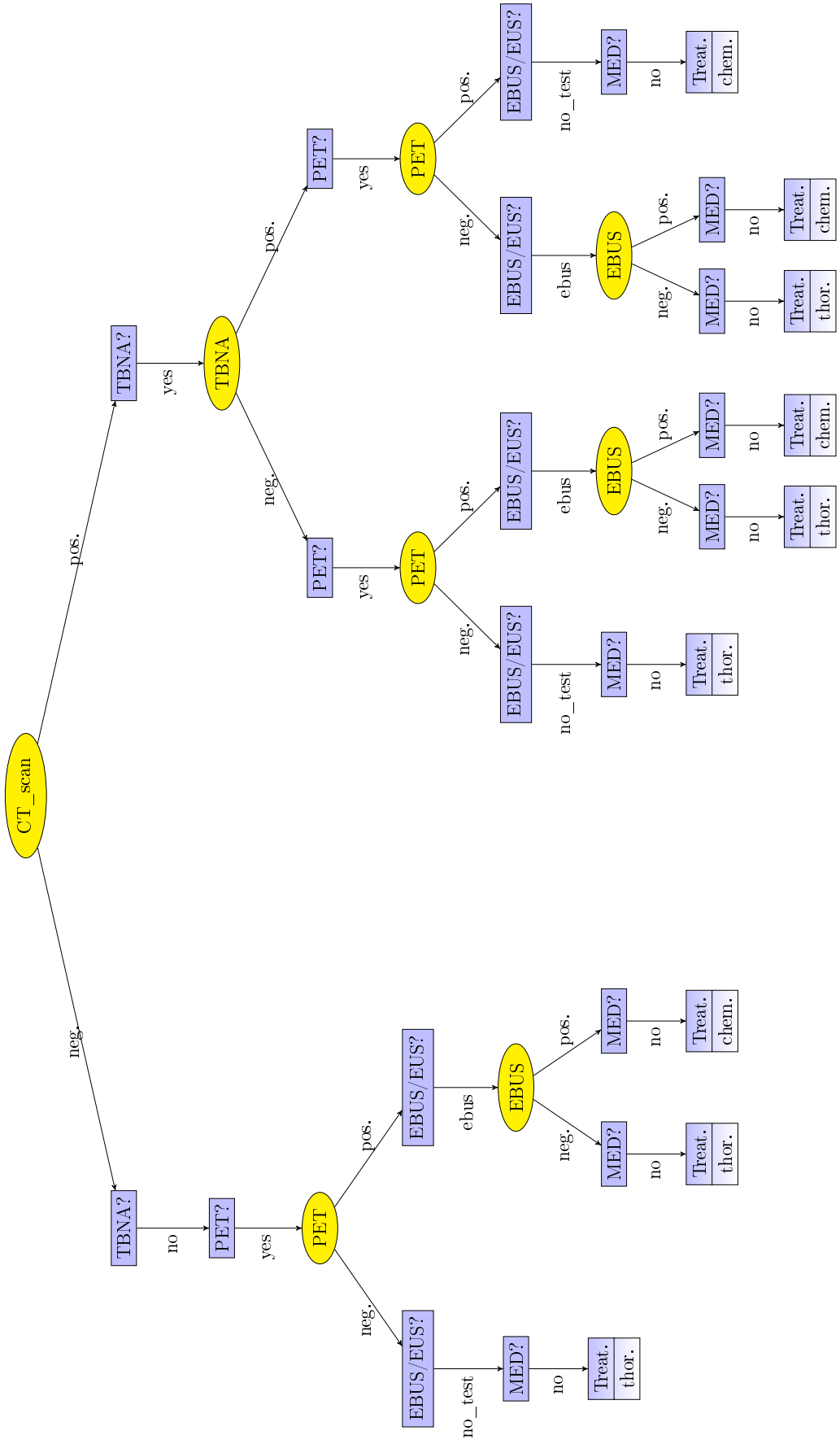

Figure 6.10: Optimal strategy for MEDIASTINET (disregarding costs).

point for the Spanish public health system (Sacristán et al., 2002). A different value of *lambda* could lead to a different strategy, as we would see in Section 6.6.

The policy tree of Figure 6.11, which corresponds to the decision *Treatment*, depicts the optimal strategy. We can examine in Figure 6.11 the optimal policy for the 5 decisions in MEDIASTINET.

We describe the 5 policies:

- **Decision_TBNA**: Perform the TBNA if CT scan is positive.
- **Decision_PET**: Never perform the PET.
- **Decision_EBUS_EUS**: Perform the EBUS if CT scan or the TBNA are negative. It also prescribes never to perform the EUS.
- **Decision_MED**: Never perform the mediastinoscopy.
- **Treatment**: Apply chemotherapy if the last test performed is positive; otherwise, it is better to apply thoracotomy.

Our expert's judgment In this case, which is more realistic (in practice costs cannot be ignored) the expert agreed completely with the strategy given by MEDIASTINET.

However, he would vary some policies in some situations:

- If the hospital does not have the possibility of performing EBUS then it would replace it by the PET.
- In the scenario of positive CT scan and negative results in the TBNA and the EBUS, he would decide initially not to perform the mediastinoscopy, would perform the PET and then would perform the mediastinoscopy if the PET is positive.

The first situation could be analyzed in the GUI of Elvira by forcing the policy of the EBUS to never to perform it. Analyzing the second situation would require to modify the structure of MEDIASTINET by changing the sequence of decision nodes and test results.

6.5.3 Comparison of both strategies

If we read carefully the two optimal strategies returned by MEDIASTINET with the criteria of the maximization of the effectiveness (disregarding costs) and the net benefit, presented above, we find some discrepancies in the recommendations. We review the optimal policy given for each decision to see the differences.

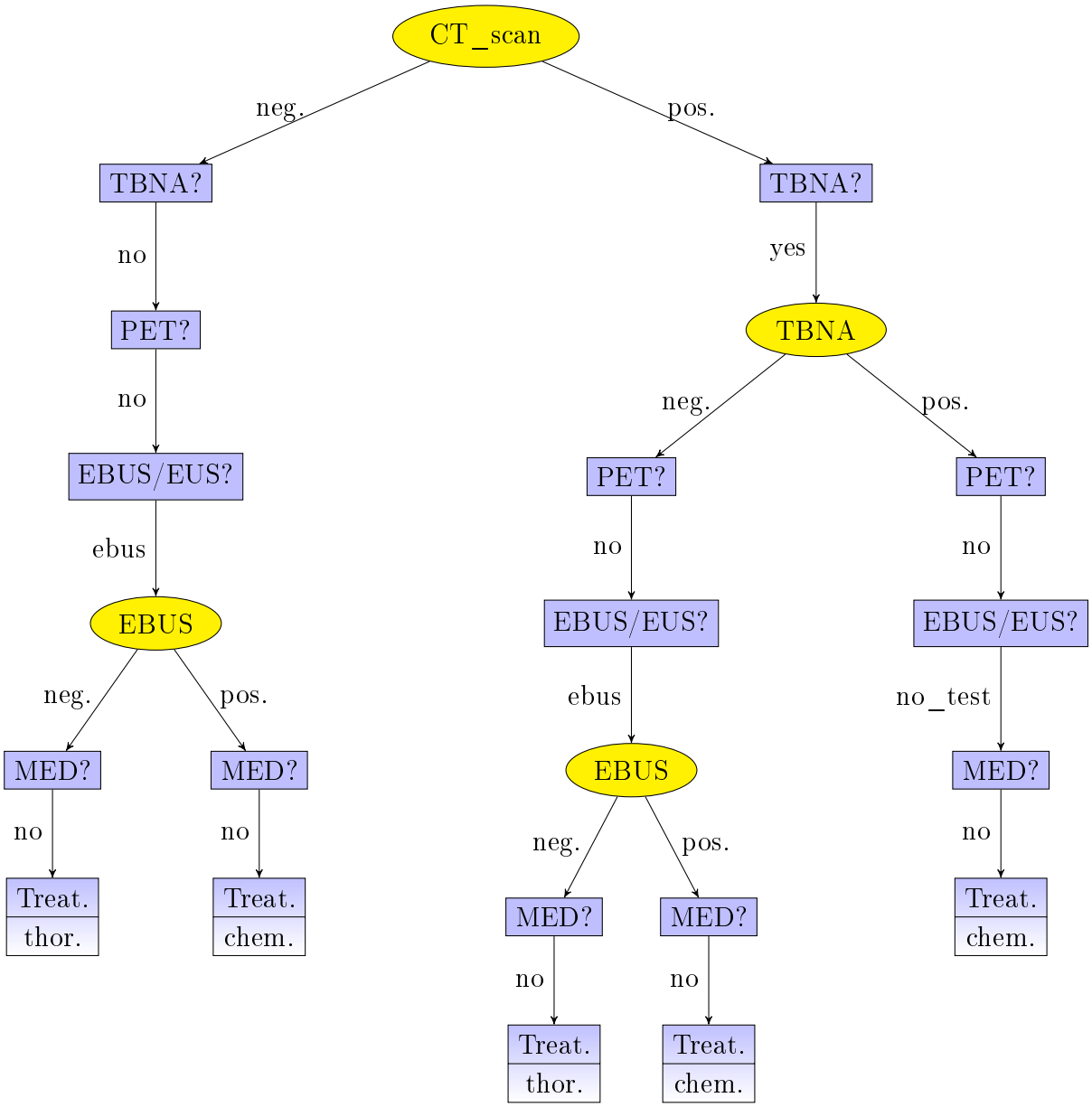

Figure 6.11: Optimal strategy for MEDIASTINET with economic costs.

The policies for *Decision_ TBNA*, *Decision_ MED* and *Treatment* are identical, and the *EUS* is never performed.

With respect to the discrepancies, when we disregard costs the PET is always performed, but never performed if we consider them. When CT scan is negative, the EBUS is not performed if the PET is negative when disregarding costs, but it is performed if we considering them.

6.6 Sensitivity analysis in MEDIASTINET

We are interested in determining whether the conclusions obtained (optimal strategy and MEU) for MEDIASTINET hold in spite of the uncertainty relative to the construction of the model. We therefore needs to perform SA in the ID of MEDIASTINET.

6.6.1 Uncertainty on the numerical parameters of MEDIASTINET

Section 6.4.2 presents the numerical parameters of MEDIASTINET. We have performed SA over each of these parameters by following the methodology described in Section 4.5. Thus, we have built an augmented ID for each parameter. For example, Figure 6.12 shows the graph of the augmented ID of MEDIASTINET for performing SA on the prevalence of the *N2N3*. Node named *Iteration*, corresponds to the node that was called Θ in the formal exposition of Section 4.5.

Each uncertain parameter of MEDIASTINET has been characterized with a probability distribution. To indicate that a parameter is modeled with a normal distribution with mean μ and variance σ^2, we write $\mathcal{N}(\mu, \sigma^2)$. To present that a parameter is attached to a beta distribution with parameters α and β we use the notation $\mathcal{B}(\alpha, \beta)$. Finally, if it is modeled with a log-normal distribution we write Log-$\mathcal{N}(\mu, \sigma^2)$.

Table 6.5 displays the distribution used for each parameter. The second columns refers to the Parameter Number (PN), which is a number used to identify each parameter in following tables and figures. Using the PN instead of the parameter name will allows us to save space specially in tables.

The parameters of distributions have been assessed by the expert based on medical literature. He usually found in the literature that there had values for the mean and confidence interval for each parameter. These values were used to determine the necessary parameters for each distribution. For each parameter we have the mean as reference value.

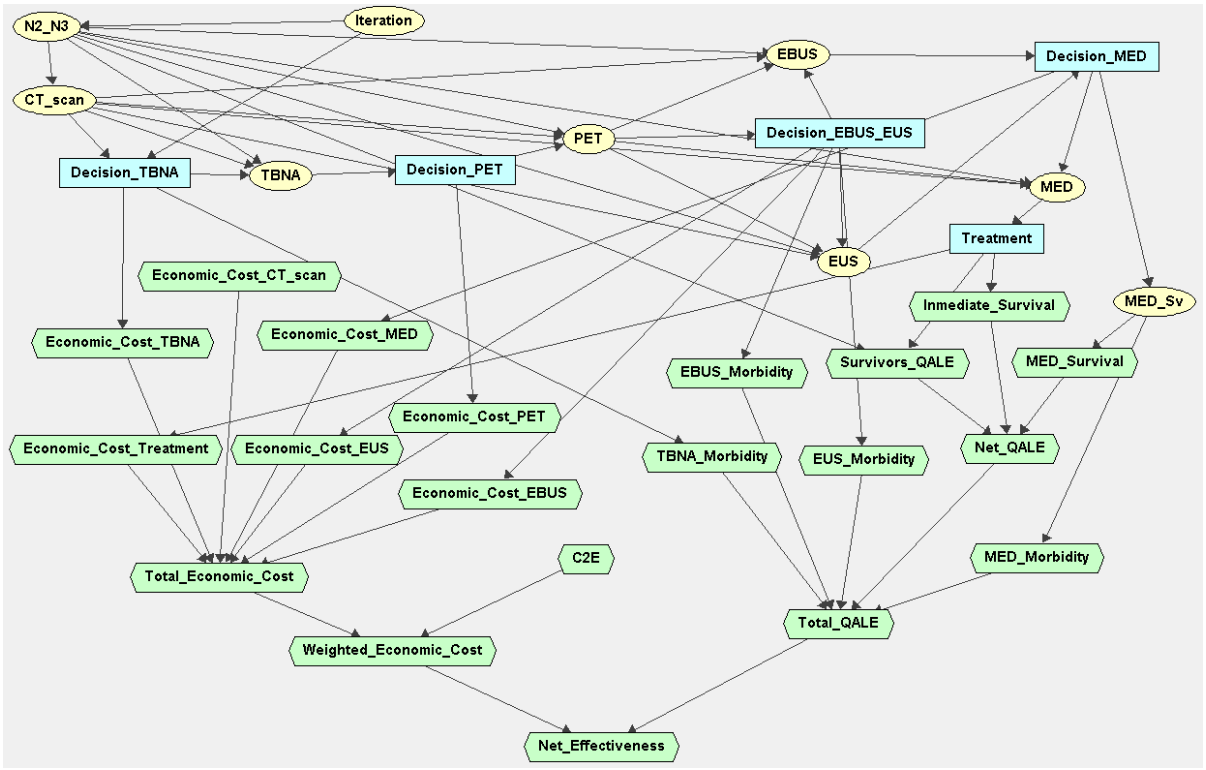

Figure 6.12: Augmented influence diagram of MEDIASTINET for performing SA over the prevalence of node *N2N3*.

Table 6.5: Probability distribution assigned to each parameter of MEDIASTINET. The second columns refers to the Parameter Number (PN), which is a number used to indentify each parameter in following tables and figures.

Name of the parameter	PN	Probability distribution
prevalence_N2N3	1	$\mathcal{B}(144.0, 369.0)$
sens_CT	2	$\mathcal{B}(74.0, 71.0)$
spec_CT	3	$\mathcal{B}(317.0, 53.0)$
sens_TBNA_CT_po	4	$\mathcal{B}(57.0, 67.0)$
spec_TBNA_CT_po	5	$\mathcal{B}(104.0, 11.0)$
sens_TBNA_CT_ne	6	$\mathcal{B}(2.0, 98.0)$
spec_TBNA_CT_ne	7	$\mathcal{B}(129.0, 11.0)$
sens_PET_CT_po	8	$\mathcal{B}(76.0, 8.0)$
spec_PET_CT_po	9	$\mathcal{B}(62.0, 18.0)$
sens_PET_CT_ne	10	$\mathcal{B}(57.0, 20.0)$
spec_PET_CT_ne	11	$\mathcal{B}(86.0, 7.0)$
sens_EBUS_CT_po_PET_po	12	$\mathcal{B}(29.0, 4.0)$

Continued on next page

Table 6.5 – continued from previous page

Name of the parameter	PN	Probability distribution
sens_EBUS_CT_po_PET_ne	13	$\mathcal{B}(32.0, 4.0)$
sens_EBUS_CT_ne_PET_po	14	$\mathcal{B}(33.0, 4.0)$
sens_EBUS_CT_ne_PET_ne	15	$\mathcal{B}(37.0, 5.0)$
spec_EBUS_CT_po_PET_po	16	$\mathcal{B}(28.0, 1.0)$
spec_EBUS_CT_po_PET_ne	17	$\mathcal{B}(40.0, 1.0)$
spec_EBUS_CT_ne_PET_po	18	$\mathcal{B}(29.0, 1.0)$
spec_EBUS_CT_ne_PET_ne	19	$\mathcal{B}(39.0, 1.0)$
sens_EBUS_CT_po	20	$\mathcal{B}(34.0, 3.0)$
spec_EBUS_CT_po	21	$\mathcal{B}(37.0, 1.0)$
sens_EBUS_CT_ne	22	$\mathcal{B}(33.0, 4.0)$
spec_EBUS_CT_ne	23	$\mathcal{B}(44.0, 1.0)$
sens_EUS_CT_po_PET_po	24	$\mathcal{B}(31.0, 5.0)$
sens_EUS_CT_po_PET_ne	25	$\mathcal{B}(33.0, 5.0)$
sens_EUS_CT_ne_PET_po	26	$\mathcal{B}(18.0, 13.0)$
sens_EUS_CT_ne_PET_ne	27	$\mathcal{B}(17.0, 13.0)$
spec_EUS_CT_po_PET_po	28	$\mathcal{B}(29.0, 2.0)$
spec_EUS_CT_po_PET_ne	29	$\mathcal{B}(28.0, 2.0)$
spec_EUS_CT_ne_PET_po	30	$\mathcal{B}(25.0, 2.0)$
spec_EUS_CT_ne_PET_ne	31	$\mathcal{B}(30.0, 2.0)$
sens_EUS_CT_po	32	$\mathcal{B}(24.0, 4.0)$
spec_EUS_CT_po	33	$\mathcal{B}(26.0, 2.0)$
sens_EUS_CT_ne	34	$\mathcal{B}(16.0, 5.0)$
spec_EUS_CT_ne	35	$\mathcal{B}(24.0, 2.0)$
sens_MED_CT_po_PET_po	36	$\mathcal{B}(12.0, 3.0)$
spec_MED_CT_po_PET_po	37	$\mathcal{B}(19.0, 1.0)$
sens_MED_CT_po_PET_ne	38	$\mathcal{B}(13.0, 3.0)$
spec_MED_CT_po_PET_ne	39	$\mathcal{B}(16.0, 1.0)$
sens_MED_CT_ne_PET_po	40	$\mathcal{B}(11.0, 3.0)$
spec_MED_CT_ne_PET_po	41	$\mathcal{B}(18.0, 1.0)$
sens_MED_CT_ne_PET_ne	42	$\mathcal{B}(12.0, 3.0)$
spec_MED_CT_ne_PET_ne	43	$\mathcal{B}(15.0, 1.0)$
sens_MED_CT_po	44	$\mathcal{B}(13.0, 3.0)$

Continued on next page

Table 6.5 – continued from previous page

Name of the parameter	PN	Probability distribution
spec_MED_CT_po	45	$\mathcal{B}(13.0, 1.0)$
sens_MED_CT_ne	46	$\mathcal{B}(8.0, 3.0)$
spec_MED_CT_ne	47	$\mathcal{B}(17.0, 1.0)$
MED_sv_Decision_MED_yes	48	$\mathcal{B}(52.0, 2.0)$
morbidity_MED	49	$\mathcal{N}(-0.05, 1.0E-4)$
morbidity_EBUS	50	$\mathcal{N}(-0.03, 3.6E-5)$
morbidity_EUS	51	$\mathcal{N}(-0.03, 3.6E-5)$
morbidity_TBNA	52	$\mathcal{N}(-1.0E-4, 4.0E-10)$
thor_inmediate_survival	53	$\mathcal{B}(20.0, 2.0)$
chem_inmediate_survival	54	$\mathcal{B}(100.0, 2.0)$
pall_inmediate_survival	55	$\mathcal{B}(52.0, 1.0)$
surv_qale_N2_N3_po_thor	56	$\mathcal{N}(0.66, 0.017424)$
surv_qale_N2_N3_po_chem	57	$\mathcal{N}(0.83, 0.027556)$
surv_qale_N2_N3_po_pall	58	$\mathcal{N}(0.5, 0.01)$
surv_qale_N2_N3_ne_thor	59	$\mathcal{N}(3.0, 0.36)$
surv_qale_N2_N3_ne_chem	60	$\mathcal{N}(2.0, 0.16)$
surv_qale_N2_N3_ne_pall	61	$\mathcal{N}(1.25, 0.0625)$
cost_CT	62	$\mathcal{N}(670.0, 17956.0)$
cost_TBNA	63	$\mathcal{N}(80.0, 256.0)$
cost_MED	64	$\mathcal{N}(1620.0, 104976.0)$
cost_PET	65	$\mathcal{N}(2250.0, 202500.0)$
cost_EBUS	66	$\mathcal{N}(620.0, 15376.0)$
cost_EUS	67	$\mathcal{N}(620.0, 15376.0)$
cost_thor	68	$\mathcal{N}(19646.0, 9223372.036854776)$
cost_chem	69	$\mathcal{N}(11242.0, 5055302.560000001)$
cost_pall	70	$\mathcal{N}(3000.0, 360000.0)$
lambda	71	Log-$\mathcal{N}(30000.0, 9223372.036854776)$

We have used discrete variables in the ID. Thus, each continuous distribution has been discretized by taking 100 points of an interval of the domain of the parameter as explained in Section 4.5. The intervals partitioned for normal and log-normal distributions of parameters μ and σ^2 have been $[\mu - k \cdot \sigma, \mu + k \cdot \sigma]$ and $[e^{\mu - k \cdot \sigma}, e^{\mu + k \cdot \sigma}]$ respectively, by

using $k = 3.5$, which accumulates 99.953 % of the probability mass.

6.6.2 Results of the SA

We have performed a SA of MEDIASTINET by recording the metrics defined in Section 4.5. We present here the results.

Policy change thresholds

Table 6.6 presents the policy change thresholds (PCTs) for every parameter of the influence diagram. For each parameter, its PCT in the table is the intersection of the PCTs obtained for the different decisions of MEDIASTINET. The second column indicates the reference value for the parameter.

Table 6.6: Intervals where the optimal strategy changes. The correspondence of the first column (PN) is given by Table 6.5. Each of the rest columns indicates the interval where the optimal policy of the corresponding decision changes. Each row shows the results for each parameter of MEDIASTINET.

PN	Ref. value	Intervals
prevalence_N2N3	0.281	**[0.259,0.284], [0.864,1.0]**
sens_CT	0.510	**[0.407,0.531]**
spec_CT	0.857	**[0.0,0.099], [0.815,0.864]**
sens_TBNA_CT_po	0.460	**[0.37,0.815]**
spec_TBNA_CT_po	0.904	**[0.877,0.914], [0.975,1.0]**
sens_TBNA_CT_ne	0.020	**[0.012,0.383]**
spec_TBNA_CT_ne	0.921	**[0.914,1.0]**
sens_PET_CT_po	0.905	**[0.0,1.0]**
spec_PET_CT_po	0.775	**[0.0,1.0]**
sens_PET_CT_ne	0.740	**[0.0,1.0]**
spec_PET_CT_ne	0.925	**[0.0,1.0]**
sens_EBUS_CT_po_PET_po	0.879	**[0.0,1.0]**
sens_EBUS_CT_po_PET_ne	0.889	**[0.0,1.0]**
sens_EBUS_CT_ne_PET_po	0.892	**[0.0,1.0]**
sens_EBUS_CT_ne_PET_ne	0.881	**[0.0,1.0]**
spec_EBUS_CT_po_PET_po	0.966	**[0.0,1.0]**
spec_EBUS_CT_po_PET_ne	0.976	**[0.0,1.0]**

Continued on next page

Table 6.6 – continued from previous page

PN	Ref. value	Intervals
spec_EBUS_CT_ne_PET_po	0.967	**[0.0,1.0]**
spec_EBUS_CT_ne_PET_ne	0.975	**[0.0,1.0]**
sens_EBUS_CT_po	0.919	**[0.914,0.951]**
spec_EBUS_CT_po	0.974	**[0.938,1.0]**
sens_EBUS_CT_ne	0.892	**[0.765,1.0]**
spec_EBUS_CT_ne	0.978	**[0.975,1.0]**
sens_EUS_CT_po_PET_po	0.861	**[0.0,1.0]**
sens_EUS_CT_po_PET_ne	0.868	**[0.0,1.0]**
sens_EUS_CT_ne_PET_po	0.581	**[0.0,1.0]**
sens_EUS_CT_ne_PET_ne	0.567	**[0.0,1.0]**
spec_EUS_CT_po_PET_po	0.935	**[0.0,1.0]**
spec_EUS_CT_po_PET_ne	0.933	**[0.0,1.0]**
spec_EUS_CT_ne_PET_po	0.926	**[0.0,1.0]**
spec_EUS_CT_ne_PET_ne	0.938	**[0.0,1.0]**
sens_EUS_CT_po	0.857	**[0.0,0.963]**
spec_EUS_CT_po	0.929	**[0.0,1.0]**
sens_EUS_CT_ne	0.762	**[0.0,1.0]**
spec_EUS_CT_ne	0.923	**[0.0,1.0]**
sens_MED_CT_po_PET_po	0.800	**[0.0,1.0]**
spec_MED_CT_po_PET_po	0.950	**[0.0,1.0]**
sens_MED_CT_po_PET_ne	0.813	**[0.0,1.0]**
spec_MED_CT_po_PET_ne	0.941	**[0.0,1.0]**
sens_MED_CT_ne_PET_po	0.786	**[0.0,1.0]**
spec_MED_CT_ne_PET_po	0.947	**[0.0,1.0]**
sens_MED_CT_ne_PET_ne	0.800	**[0.0,1.0]**
spec_MED_CT_ne_PET_ne	0.938	**[0.0,1.0]**
sens_MED_CT_po	0.813	**[0.0,1.0]**
spec_MED_CT_po	0.929	**[0.0,0.951]**
sens_MED_CT_ne	0.727	**[0.0,1.0]**
spec_MED_CT_ne	0.944	**[0.0,1.0]**
MED_sv_Decision_MED_yes	0.963	**[0.0,0.963]**
morbidity_MED	−0.050	**[-0.085,-0.015]**

Continued on next page

Table 6.6 – continued from previous page

PN	Ref. value	Intervals
morbidity_EBUS	−0.030	**[-0.033,-0.015]**
morbidity_EUS	−0.030	**[-0.051,-0.0090]**
morbidity_TBNA	0.000	**[0.0,0.0]**
thor_inmediate_survival	0.909	**[0.79,0.84], [0.901,0.938]**
chem_inmediate_survival	0.980	**[0.938,1.0]**
pall_inmediate_survival	0.981	**[0.0,1.0]**
surv_qale_N2_N3_po_thor	0.660	**[0.643,0.734]**
surv_qale_N2_N3_po_chem	0.830	**[0.78,0.852]**
surv_qale_N2_N3_po_pall	0.500	**[0.15,0.548]**
surv_qale_N2_N3_ne_thor	3.000	**[2.611,2.767], [2.922,3.13]**
surv_qale_N2_N3_ne_chem	2.000	**[1.879,2.363]**
surv_qale_N2_N3_ne_pall	1.250	**[0.375,2.103]**
cost_CT	670.000	**[201.0,1139.0]**
cost_TBNA	80.000	**[73.78,136.0]**
cost_MED	1620.000	**[1466.0,2754.0]**
cost_PET	2250.000	**[1025.0,3825.0]**
cost_EBUS	620.000	**[186.0,721.8]**
cost_EUS	620.000	**[186.0,1054.0]**
cost_thor	19646.000	**[18118.0,20155.0], [25588.0,29323.0]**
cost_chem	11242.000	**[10756.0,12699.0]**
cost_pall	3000.000	**[1159.0,5100.0]**
lambda	30000.000	**[26217.0,32591.0]**

We can see that there are many parameters whose PCT is the entire interval of its domain. For example, all the parameters of PET has the interval $[0, 1]$ has PCT.

In contrast, there are some parameters whose PCT is more restrictive. For example, the reference value for the sensitivity of CT scan is 0.51. The PCT given by the table is $[0.41, 0.574]$. It means the reference value of sensitivity of CT scan is not very far from the thresholds, and some policy could change if the value of sensitivity varies.

EVPI

Table 6.7 shows the EVPI of each parameter of MEDIASTINET. Most of the parameters presents a value of 0.0 when presenting their EVPI rounded to 3 decimals. The parameter with the highest EVPI is *lambda*, whose value 0.503 is much higher than the second parameter with highest EVPI (*surv_ qale_ N2_ N3_ ne_ thor* with a value of 0.052).

Other important observation is obtained from columns of EU_F and EU_V (see Section 4.5.2). Most of the cells in these two columns have a value of 1.471 or similar values, except the two cells for *lambda*. Its value of EU_F is 0.471, very far from the obtained for the rest of parameters.

Table 6.7: EVPI in MEDIASTINET.

Name of the parameter	EU_F	EU_V	EVPI ($EU_V - EU_F$)
prevalence_N2N3	1.471	1.471	0.000
sens_CT	1.471	1.471	0.000
spec_CT	1.471	1.471	0.000
sens_TBNA_CT_po	1.471	1.471	0.000
spec_TBNA_CT_po	1.471	1.471	0.000
sens_TBNA_CT_ne	1.471	1.471	0.000
spec_TBNA_CT_ne	1.471	1.471	0.000
sens_PET_CT_po	1.471	1.471	0.000
spec_PET_CT_po	1.471	1.471	0.000
sens_PET_CT_ne	1.471	1.471	0.000
spec_PET_CT_ne	1.471	1.471	0.000
sens_EBUS_CT_po_PET_po	1.471	1.471	0.000
sens_EBUS_CT_po_PET_ne	1.471	1.471	0.000
sens_EBUS_CT_ne_PET_po	1.471	1.471	0.000
sens_EBUS_CT_ne_PET_ne	1.471	1.471	0.000
spec_EBUS_CT_po_PET_po	1.471	1.471	0.000
spec_EBUS_CT_po_PET_ne	1.471	1.471	0.000
spec_EBUS_CT_ne_PET_po	1.471	1.471	0.000
spec_EBUS_CT_ne_PET_ne	1.471	1.471	0.000
sens_EBUS_CT_po	1.471	1.471	0.000
spec_EBUS_CT_po	1.471	1.471	0.000
sens_EBUS_CT_ne	1.471	1.471	0.000

Continued on next page

Table 6.7 – continued from previous page

Name of the parameter	EU_F	EU_V	EVPI ($EU_V - EU_F$)
spec_EBUS_CT_ne	1.471	1.471	0.000
sens_EUS_CT_po_PET_po	1.471	1.471	0.000
sens_EUS_CT_po_PET_ne	1.471	1.471	0.000
sens_EUS_CT_ne_PET_po	1.471	1.471	0.000
sens_EUS_CT_ne_PET_ne	1.471	1.471	0.000
spec_EUS_CT_po_PET_po	1.471	1.471	0.000
spec_EUS_CT_po_PET_ne	1.471	1.471	0.000
spec_EUS_CT_ne_PET_po	1.471	1.471	0.000
spec_EUS_CT_ne_PET_ne	1.471	1.471	0.000
sens_EUS_CT_po	1.471	1.471	0.000
spec_EUS_CT_po	1.471	1.471	0.000
sens_EUS_CT_ne	1.471	1.471	0.000
spec_EUS_CT_ne	1.471	1.471	0.000
sens_MED_CT_po_PET_po	1.471	1.471	0.000
spec_MED_CT_po_PET_po	1.471	1.471	0.000
sens_MED_CT_po_PET_ne	1.471	1.471	0.000
spec_MED_CT_po_PET_ne	1.471	1.471	0.000
sens_MED_CT_ne_PET_po	1.471	1.471	0.000
spec_MED_CT_ne_PET_po	1.471	1.471	0.000
sens_MED_CT_ne_PET_ne	1.471	1.471	0.000
spec_MED_CT_ne_PET_ne	1.471	1.471	0.000
sens_MED_CT_po	1.471	1.471	0.000
spec_MED_CT_po	1.471	1.471	0.000
sens_MED_CT_ne	1.471	1.471	0.000
spec_MED_CT_ne	1.471	1.471	0.000
MED_sv_Decision_MED_yes	1.471	1.471	0.000
morbidity_MED	1.471	1.471	0.000
morbidity_EBUS	1.471	1.471	0.000
morbidity_EUS	1.471	1.471	0.000
morbidity_TBNA	1.471	1.471	0.000
thor_inmediate_survival	1.471	1.475	0.003
chem_inmediate_survival	1.471	1.471	0.000

Continued on next page

Table 6.7 – continued from previous page

Name of the parameter	EU_F	EU_V	EVPI ($EU_V - EU_F$)
pall_inmediate_survival	1.471	1.471	0.000
surv_qale_N2_N3_po_thor	1.471	1.472	0.001
surv_qale_N2_N3_po_chem	1.471	1.480	0.009
surv_qale_N2_N3_po_pall	1.471	1.474	0.003
surv_qale_N2_N3_ne_thor	1.471	1.523	0.052
surv_qale_N2_N3_ne_chem	1.471	1.494	0.023
surv_qale_N2_N3_ne_pall	1.471	1.471	0.000
cost_CT	1.471	1.471	0.000
cost_TBNA	1.471	1.471	0.000
cost_MED	1.471	1.471	0.000
cost_PET	1.471	1.471	0.000
cost_EBUS	1.471	1.471	0.000
cost_EUS	1.471	1.471	0.000
cost_thor	1.471	1.473	0.002
cost_chem	1.471	1.473	0.002
cost_pall	1.471	1.471	0.000
lambda	0.544	1.047	0.503

Sensitivities

Table 6.8 presents the values of sensitivities for each decision and each parameter of MEDIASTINET. Last column indicates the maximum of the sensitivities of each parameter, while last row indicates the maximum of the sensitivities of each decision. Cell in the last row and last column indicates the maximum of the table.

Table 6.8: Sensitivities in MEDIASTINET. The correspondence of the first column (PN) is given by Table 6.5.

Parameter	TBNA	PET	EB/EU	MED	Treat.	Max.
prevalence_N2N3	0.105	0.000	0.000	0.033	0.000	**0.105**
sens_CT	0.061	0.000	0.000	0.025	0.000	**0.061**
spec_CT	0.004	0.000	0.000	0.026	0.000	**0.026**

Continued on next page

Table 6.8 – continued from previous page

Parameter	TBNA	PET	EB/EU	MED	Treat.	Max.
sens_TBNA_CT_po	0.005	0.000	0.000	0.000	0.000	**0.005**
spec_TBNA_CT_po	0.037	0.000	0.000	0.029	0.000	**0.037**
sens_TBNA_CT_ne	0.290	0.000	0.000	0.000	0.052	**0.290**
spec_TBNA_CT_ne	0.252	0.000	0.000	0.000	0.000	**0.252**
sens_PET_CT_po	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_PET_CT_po	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_PET_CT_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_PET_CT_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EBUS_CT_po_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EBUS_CT_po_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EBUS_CT_ne_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EBUS_CT_ne_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EBUS_CT_po_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EBUS_CT_po_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EBUS_CT_ne_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EBUS_CT_ne_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EBUS_CT_po	0.064	0.000	0.003	0.026	0.000	**0.064**
spec_EBUS_CT_po	0.000	0.000	0.001	0.007	0.000	**0.007**
sens_EBUS_CT_ne	0.014	0.000	0.000	0.000	0.000	**0.014**
spec_EBUS_CT_ne	0.225	0.000	0.025	0.000	0.011	**0.225**
sens_EUS_CT_po_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EUS_CT_po_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EUS_CT_ne_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EUS_CT_ne_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EUS_CT_po_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EUS_CT_po_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EUS_CT_ne_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EUS_CT_ne_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EUS_CT_po	0.004	0.000	0.004	0.000	0.000	**0.004**
spec_EUS_CT_po	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_EUS_CT_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_EUS_CT_ne	0.000	0.000	0.000	0.000	0.000	**0.000**

Continued on next page

Table 6.8 – continued from previous page

Parameter	TBNA	PET	EB/EU	MED	Treat.	Max.
sens_MED_CT_po_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_MED_CT_po_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_MED_CT_po_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_MED_CT_po_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_MED_CT_ne_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_MED_CT_ne_PET_po	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_MED_CT_ne_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_MED_CT_ne_PET_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
sens_MED_CT_po	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_MED_CT_po	0.000	0.000	0.000	0.035	0.000	**0.035**
sens_MED_CT_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
spec_MED_CT_ne	0.000	0.000	0.000	0.000	0.000	**0.000**
MED_sv_Decision_MED_yes	0.000	0.000	0.000	0.044	0.000	**0.044**
morbidity_MED	0.000	0.000	0.000	0.000	0.000	**0.000**
morbidity_EBUS	0.218	0.000	0.000	0.000	0.000	**0.218**
morbidity_EUS	0.000	0.000	0.000	0.000	0.000	**0.000**
morbidity_TBNA	0.000	0.000	0.000	0.000	0.000	**0.000**
thor_inmediate_survival	0.315	0.000	0.038	0.029	0.027	**0.315**
chem_inmediate_survival	0.011	0.000	0.000	0.000	0.000	**0.011**
pall_inmediate_survival	0.000	0.000	0.000	0.000	0.000	**0.000**
surv_qale_N2_N3_po_thor	0.120	0.000	0.106	0.033	0.000	**0.120**
surv_qale_N2_N3_po_chem	0.325	0.000	0.000	0.033	0.040	**0.325**
surv_qale_N2_N3_po_pall	0.154	0.000	0.000	0.000	0.042	**0.154**
surv_qale_N2_N3_ne_thor	0.445	0.000	0.246	0.071	0.237	**0.445**
surv_qale_N2_N3_ne_chem	0.401	0.000	0.150	0.045	0.149	**0.401**
surv_qale_N2_N3_ne_pall	0.000	0.000	0.000	0.000	0.000	**0.000**
cost_CT	0.000	0.000	0.000	0.000	0.000	**0.000**
cost_TBNA	0.263	0.000	0.000	0.000	0.000	**0.263**
cost_MED	0.000	0.000	0.000	0.023	0.000	**0.023**
cost_PET	0.000	0.001	0.000	0.000	0.000	**0.001**
cost_EBUS	0.155	0.000	0.000	0.000	0.000	**0.155**
cost_EUS	0.000	0.000	0.000	0.000	0.000	**0.000**

Continued on next page

Table 6.8 – continued from previous page

Parameter	TBNA	PET	EB/EU	MED	Treat.	Max.
cost_thor	0.197	0.000	0.140	0.030	0.001	**0.197**
cost_chem	0.245	0.000	0.000	0.030	0.003	**0.245**
cost_pall	0.000	0.000	0.000	0.000	0.000	**0.000**
lambda	0.337	0.047	0.366	0.099	0.423	**0.423**
Maximum	**0.445**	**0.047**	**0.366**	**0.099**	**0.423**	**0.445**

We can observe in Table 6.8 how the sensitivities are 0.0 in most of cases, which indicates that the optimal strategy would not change when varying these parameters. The highest value of sensitivity corresponds to *surv_gale_N2_N3_ne_thor*, which presents a sensitivity of 0.445 when analyzing the policy changes of *Decision_TBNA*. Other parameters with high sensitivity are *lambda* (0.382) when analyzing *Treatment*, and *sens_TBNA_CT_ne* (0.29) when considering the changes in *Decision_TBNA*. We can observe that the only parameter that affects the policies of all the decisions is *lambda*.

With regards to the decisions, the policy that can be more affected by changes in the parameters is *Decision_TBNA* (0.445), *Treatment* (0.382) and *Decision_EBUS_EUS* (0.366). Sensitivities of *Decision_PET* and *Decision_MED* are very small.

6.7 Discussion

We have built an influence diagram, MEDIASTINET for the mediastinal staging of non-small cell lung cancer. The parameter λ, which in cost-effectiveness analyses represents the amount of money that the decision maker is willing to pay to obtain a unit of effectiveness, has been included in the ID by introducing a utility node that represents $1/\lambda$.

First, we have evaluated the ID with $\lambda = +\infty$, i.e., $(1/\lambda) = 0$, to obtain the strategy that maximizes the effectiveness, disregarding economic costs. Then, we have evaluated it again with $\lambda = 30,000$ €/QALY, which is accepted as the shadow cost-effectiveness equivalence in Spain.[3]

The expert said that the optimal strategies yielded by MEDIASTINET were very reasonable and "logic", and that the system was "quite intelligent."

[3]In this context, *shadow* means that this value has not been explicitly stated by the health authorities; but, it has been estimated by some researchers (Sacristán et al., 2002) by analyzing which interventions are included in the Spanish public health system and which have been excluded.

Given that we do not know with precision the values of the parameters, we have represented this uncertainty by assigning, with the expert's help, a probability distribution to each independent parameter of the model (see Section 6.6.1). We were glad to realize, after performing the sensitivity analysis, that the optimal policies resulting from MEDIASTINET were very robust to the variations in the parameters (see Section 6.6.2) because they:

- present policy change tresholds with enough range of variation in most of the cases;
- generally have a value of 0.0 for the EVPI;
- the sensivities are 0.0 or a very small value in most of the cases.

There are are only two parameters that can have significant on the strategy: *surv_ qale_ N2_ N3_ ne_ thor* and *lambda*. In spite of the fact that *surv_ qale_ N2_ N3_ ne_ thor* is the parameter with the highest sensitivity in a decision, the parameter that reflects to have more impact is *lambda*, as is shown by the value of sensitivity for three decisions, and its high value of EVPI. This indicates us that knowing with more certainty the value of λ has very important benefits for the strategy because it increases its expected value.

As future work, the expert suggested us two possible research lines regarding to the application. First, he is very interested in considering the possibility of having partial order among the decisions of the influence diagrams. Thus, a representation like unconstrained influence diagrams would be one possibility for that purpose.

Second, he would like to include in the model the possibility of repeat some decisions, which is named by the expert as *restaging*.

We want to emphasize that all the influence diagrams of MEDIASTINET and the augmented IDs for the SA, the tables and figures of the strategies of this chapter have been generated automatically by a Java program whose entries are the numerical values of probability distributions of parameters and their type (normal, log-normal, beta). These values are contained in a file, which can easily modified to obtain in few minutes all the influence diagrams, their optimal strategies and the SA tables and figures. Parameters were provided by the expert, which made a reasonable search in the literature. Other experts could disagree in a numerical parameter of distributions or even in its type. However, we have tried to implement the decision-support system to have the possibility of modifying it easily.

Part V

CONCLUSION

Chapter 7

Conclusions

We end this dissertation by presenting the main contributions (Section 7.1) and the lines open for future research (Section 7.2).

7.1 Main contributions

In this dissertation we have considered the representation, solution, and analysis of medical decision problems with probabilistic graphical models.

We have proposed a new algorithm for evaluating influence diagrams (IDs) with super-value (SV) nodes, which has five advantages over the arc reversal (AR) algorithm of Tatman and Shachter (1990): it is faster in general, requires less memory in most of the cases, introduces fewer redundant variables, simplifies sensitivity analysis, and can achieve further savings of time and memory space for IDs containing canonical models, such as the noisy OR or the noisy MAX (see Chapter 3).

We have developed new explanation methods—see Chapter 4—for both the model (the knowledge encoded in the influence diagram) and the reasoning (the strategies returned by the influence diagram), which proved to be very useful in the construction and debugging of MEDIASTINET, and helped to convince the expert that the recommendations given by our model are reasonable. In particular, *policy trees*, which we have been proposed as a compact way of representing the optimal policies returned by the ID, have been very useful for a model such as MEDIASTINET, in which the biggest policy table contains 15552 columns, while the corresponding policy tree contains only 5 or 9 leaves, depending on the evaluation criterion. In the future these explanation facilities could be used to convert influence diagrams into tutoring systems.

We have implemented some sensitivity analysis algorithms (see Section 4.5) which, given the uncertainty in the parameters, allow us to compute:

- the probability of a change in the strategy;

- the intervals of the parameters where the optimal policies do not change; and
- the expected value of perfect information for each parameter.

We have developed an anytime algorithm for evaluating unconstrained influence diagrams, a type of probabilistic graphical model that admits a partial ordering of the decisions. The purpose of the method was to provide a qualified recommendation for the first decisions when the decision maker has no time to wait until the standard algorithm had computed the optimal strategy (see Chapter 5). The quality of the recommendations were measured by considering:

- the frequency with which the anytime algorithm returns the correct decision options (relative to the optimal strategy) for all decisions down to the ith level in the decision tree;
- the expected utility of following the strategy prescribed by the anytime algorithm or dynamic programming for the first i levels of decisions, followed by the optimal strategy for the remaining decisions.

From the results we could see that the algorithm improves over time with respect to all the recorded characteristics.

As an application of probabilistic graphical models, we have built a decision-support system for the mediastinal staging of non-small cell lung cancer (Chapter 6). The parameter λ, which in cost-effectiveness analyses represents the amount of money that the decision maker is willing to pay to obtain a unit of effectiveness, has been included in the ID by introducing a utility node that represents $1/\lambda$.

First, we have evaluated the ID with $\lambda = +\infty$, i.e., $(1/\lambda) = 0$, to obtain the strategy that maximizes the effectiveness, disregarding economic costs. Then, we have evaluated it again with $\lambda = 30,000$ €/QALY, which is accepted as the shadow cost-effectiveness equivalence in Spain.[1]

The expert said that the optimal strategies yielded by MEDIASTINET were very reasonable and "logic", and that the system was "quite intelligent."

Given that we do not know with precision the values of the parameters, we have represented this uncertainty by assigning, with the expert's help, a probability distribution to each independent parameter of the model (see Section 6.6.1). We were glad to realize,

[1] In this context, *shadow* means that this value has not been explicitly stated by the health authorities; but, it has been estimated by some researchers (Sacristán et al., 2002) by analyzing which interventions are included in the Spanish public health system and which have been excluded.

after performing the sensitivity analysis, that the optimal policies resulting from MEDIASTINET were very robust to the variations in the parameters (see Section 6.6.1): the parameter with the highest impact on the strategy is λ, as expected.

7.2 Future work

There are some open lines for future research.

With respect to the variable-elimination algorithm for IDs with SV nodes, presented in Chapter 3, the three main issues that should be investigated are the following:

1. to avoid introducing all redundant variables, if possible;

2. to develop heuristic rules for this algorithm, mainly to find near-optimal elimination orderings, but also to decide how to distribute potentials in sum-product combinations; and

3. how to combine our algorithm with the proposals for lazy evaluation of traditional IDs, in order to reduce the computational cost.

With respect to the explanation of the reasoning in IDs (Chapter 4) there are also three main issues to be investigated:

1. to find some method that could help to explain to the expert which variables are having more influence in the optimal strategy and in the maximum expected utility;

2. to find very simple representations of optimal policies not only by assuming that the decision maker acts optimally in every decision, but also considering the non-optimal scenarios of the problem.

3. to solve two questions related with SA in IDs with SV nodes: how to perform n-way joint analysis; and how to calculate with exact methods the policy change thresholds.

The anytime algorithm proposed, presented in Chapter 5, could be improved in three ways:

1. to combine our method with some dynamic programming-based approach that could accelerate the convergence towards to the optimal strategy;

2. to find alternative forms for the non-admissible heuristic used in the search; and

3. to exploit the coalescence when building the decision tree.

With respect to the application of MEDIASTINET (Chapter 6) we have two research lines suggested by the expert:

1. to consider the possibility of having partial order among the decisions of the influence diagram;

2. to represent in the model the possibility of repeat some medical tests.

Bibliography

Ahlmann-Ohlsen, K. S., Jensen, F. V., Nielsen, T. D., Pedersen, O., and Vomlelova, M. (2009). A comparison of two approaches for solving unconstrained influence diagrams. *International Journal of Approximate Reasoning*, 50:153 – 173.

Alberg, A. J. and Samet, J. M. (2003). Epidemiology of lung cancer. *Chest*, 123:21–49.

Arias, M. (2009). *Carmen: una herramienta de código abierto para sistemas gráficos probabilistas.* PhD thesis, Universidad Nacional de Educación a Distancia, Madrid.

Arias, M. and Díez, F. J. (2008). Carmen: An open source project for probabilistic graphical models. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM'08)*, pages 25–32, Hirtshals, Denmark.

Bielza, C., Ríos, S., and Gómez, M. (1999). Influence diagrams for neonatal jaundice management. In *AIMDM '99: Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making*, pages 138–142, London, UK. Springer-Verlag.

Bielza, C., Ríos-Insua, D., and Ríos-Insua, S. (1996). Influence diagrams under partial information. In *Bayesian Statistics*, volume 5, pages 491–497. Oxford University Press.

Cheng, J. and Druzdzel, M. J. (2000). AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188.

Clemen, R. T. and Reilly, T. A. (2001). *Making Hard Decisions*. Duxbury, Pacific Grove, CA.

Cooper, G. F. (1988). A method for using belief networks as influence diagrams. In *Proceedings of the 4th Workshop on Uncertainty in AI*, pages 55–63, University of Minnesota, Minneapolis, MN.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

Dechter, R. (1996). Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI'96)*, pages 211–219, Portland, OR. Morgan Kaufmann, San Francisco, CA.

Demirer, R. and Shenoy, P. P. (2001). Sequential valuation asymmetric decision problems. *Lecture Notes in Computer Science*, pages 252–265.

Díez, F. J. (1994). *Sistema Experto Bayesiano para Ecocardiografía*. PhD thesis, Dpto. Informática y Automática, UNED, Madrid. In Spanish.

Díez, F. J. (2004). Teaching probabilistic medical reasoning with the Elvira software. In Haux, R. and Kulikowski, C., editors, *IMIA Yearbook of Medical Informatics*, pages 175–180. Schattauer, Sttutgart, Germany.

Díez, F. J. (2007). Teoría probabilista de la decisión en medicina. Informe Técnico CISIAD-07-01, UNED, Madrid. In Spanish.

Díez, F. J. and Druzdzel, M. J. (2006). Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid, Spain.

Díez, F. J. and Galán, S. F. (2003). Efficient computation for the noisy MAX. *International Journal of Approximate Reasoning*, 18:165–177.

Díez, F. J., Mira, J., Iturralde, E., and Zubillaga, S. (1997). DIAVAL, a Bayesian expert system for echocardiography. *Artificial Intelligence in Medicine*, 10:59–73.

Dittmer, S. L. and Jensen, F. V. (1997). Myopic value of information in influence diagrams. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI'97)*, pages 142–149, Providence, RI. Morgan Kaufmann, San Francisco, CA.

Doubilet, P., Begg, C. B., Weinstein, M. C., Braun, P., and McNeil, B. J. (1985). Probabilistic sensitivity analysis using monte carlo simulation: A practical approach. *Medical Decision Making*, 5(2):155–177.

Drummond, M. F., Sculpher, M. J., Torrance, G. W., O'Brien, B. J., and Stoddart, G. L. (2005). *Methods for the Economic Evaluation of Health Care Programmes*. Oxford University Press, USA, 3 edition.

Elizalde, F., Súcar, L. E., Luque, M., Díez, F. J., and Reyes, A. (2008). Policy explanation in factored markov decision processes. In Jensen, F. V. and Kjærulff, U., editors, *Proceedings of the Forth European Workshop on Probabilistic Graphical Models*, pages 97–104.

Elvira Consortium, T. (2002). Elvira: An environment for creating and using probabilistic graphical models. In *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 1–11, Cuenca, Spain.

Ezawa, K. (1998). Evidence propagation and value of evidence on influence diagrams. *Operations Research*, 46:73–83.

Faguiouli, E. and Zaffalon, M. (1998). A note about redundancy in influence diagrams. *International Journal of Aproximate Reasoning*, 19:231–246.

Felli, J. C. and Hazen, G. B. (1998). Sensitivity analysis and the expected value of perfect information. *Medical Decision Making*, 18(1):95–109.

Gold, M.R., S.-J. R. L. W. M. C. (1996). *Cost-Effectiveness in Health and MedicineCost-Effectiveness in Health and Medicine*. Oxford University Press, New York.

Gómez, M. and Bielza, C. (2004). Node deletion sequences in influence diagrams using genetic algorithms. *Statistics and Computing*, 14:181–198.

Henrion, M. and Druzdzel, M. (1990). Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence (UAI'90)*, pages 17–32, Cambridge, MA.

Howard, R. A. and Matheson, J. E. (1984). Influence diagrams. In Howard, R. A. and Matheson, J. E., editors, *Readings on the Principles and Applications of Decision Analysis*, pages 719–762. Strategic Decisions Group, Menlo Park, CA.

Jensen, F., Jensen, F. V., and Dittmer, S. L. (1994). From influence diagrams to junction trees. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 367–373, San Francisco, CA. Morgan Kaufmann.

Jensen, F. V. and Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, second edition.

Jensen, F. V. and Vomlelova, M. (2002). Unconstrained influence diagrams. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI'02)*, pages 234–241, San Francisco, CA. Morgan Kaufmann.

Lababede, O., Meziane, M. A., and Rice, T. W. (1999). TNM Staging of Lung Cancer: A Quick Reference Chart. *Chest*, 115(1):233–235.

Lacave, C. (2003). *Explanation in Causal Bayesian Networks. Medical Applications*. PhD thesis, Dept. Inteligencia Artificial. UNED, Madrid, Spain. In Spanish.

Lacave, C., Atienza, R., and Díez, F. J. (2000). Graphical explanation in Bayesian networks. In *Proceedings of the International Symposium on Medical Data Analysis (ISMDA-2000)*, pages 122–129, Frankfurt, Germany. Springer-Verlag, Heidelberg.

Lacave, C. and Díez, F. J. (2002). A review of explanation methods for Bayesian networks. *Knowledge Engineering Review*, 17:107–127.

Lacave, C. and Díez, F. J. (2003). Knowledge acquisition in Prostanet, a Bayesian network for diagnosing prostate cancer. In *Proceedings of the Seventh International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'2003)*, volume 2774 of *Lecture Notes in Computer Science*, pages 1345–1350, Oxford, UK. Springer, Berlin, Germany.

Lacave, C. and Díez, F. J. (2004). A review of explanation methods for heuristic expert systems. *Knowledge Engineering Review*, 19:133–146.

Lacave, C., Luque, M., and Díez, F. J. (2007). Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, 37:952–965.

Lacave, C., Oniśko, A., and Díez, F. J. (2006). Use of Elvira's explanation facilities for debugging probabilistic expert systems. *Knowledge-Based Systems*, 19:730–738.

Lloyd, C. and Silvestri, G. A. (2001). Mediastinal Staging of Non-Small-Cell Lung Cancer. *Cancer Control*, 8(4):311–317.

Luke, W. P., Pearson, F. G., Todd, T. R., Patterson, G. A., and Cooper, J. D. (1986). Prospective evaluation of mediastinoscopy for assessment of carcinoma of the lung. *Journal of Thoracic and Cardiovascular Surgery*, 91(1):53–56.

Luque, M. and Díez, F. J. (2006). Decision analysis with influence diagrams using Elvira's explanation capabilities. In Studeny, M. and Vomlel, J., editors, *Proceedings of the Third European Workshop on Probabilistic Graphical Models*, pages 179–186.

Luque, M. and Díez, F. J. (2008). Variable elimination for influence diagrams with super-value nodes. Technical Report DIA-08-01, Dpto. Inteligencia Artificial, UNED, Madrid, Spain.

Luque, M., Díez, F. J., and Disdier, C. (2005). Influence diagrams for medical decision problems: Some limitations and proposed solutions. In Holmes, J. H. and Peek, N., editors, *Proceedings of the Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP'05)*, pages 85–86.

Luque, M., Nielsen, T. D., and Jensen, F. V. (2008). An anytime algorithm for evaluating unconstrained influence diagrams. In Jensen, F. V. and Kjærulff, U., editors, *Proceedings of the Forth European Workshop on Probabilistic Graphical Models*, pages 177–184.

Madsen, A. and Jensen, F. V. (1999). Lazy evaluation of symmetric Bayesian decision problems. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 382–390, San Francisco, CA. Morgan Kaufmann.

Marr, D. (1982). *Vision*. Freeman, New York.

Ndilikilikesha, P. C. (1994). Potential influence diagrams. *International Journal of Approximate Reasoning*, 10(3):251 – 285.

Nielsen, S. H., Nielsen, T. D., and Jensen, F. V. (2007). Multi-currency influence diagrams. In Salmerón, A. and Gámez, J. A., editors, *Advances in Probabilistic Graphical Models*, pages 275–294. Springer, Berlin, Germany.

Nielsen, T. D. and Jensen, F. V. (1999). Welldefined decision scenarios. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 502–511, San Francisco, CA. Morgan Kaufmann.

Nielsen, T. D. and Jensen, F. V. (2003). Sensitivity analysis in influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 33:223–234.

Nilsson, D. and Jensen, F. V. (1998). Probabilities of future decisions. In *Proceedings from the International Conference on Informational Processing and Management of Uncertainty in knowledge-based Systems*, pages 1454–1461.

Nilsson, D. and Lauritzen, S. (2000). Evaluating influence diagrams using LIMIDs. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 436–445, San Francisco, CA. Morgan Kaufmann.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA.

Olmsted, S. M. (1983). *On Representing and Solving Decision Problems*. PhD thesis, Dept. Engineering-Economic Systems, Stanford University, CA.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Pearl, J. (1994). From conditional oughts to qualitative decision theory. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 454–462, Seattle, WA. Morgan Kaufmann, San Mateo, CA.

Pearl, J. (2000). *Causality. Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, UK.

Pradhan, M., Provan, G., Middleton, B., and Henrion, M. (1994). Knowledge engineering for large belief networks. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 484–490, Seattle, WA. Morgan Kaufmann, San Francisco, CA.

Raiffa, H. and Schlaifer, R. (1961). *Applied Statistical Decision Theory*. MIT press, Cambridge.

Sacristán, J. A., Oliva, J., del Llano, J., Prieto, L., and Pinto, J. (2002). ¿Qué es una tecnología sanitaria eficiente en España? *Gaceta Sanitaria*, 16:334–343.

Sember, P. and Zukerman, I. (1989). Strategies for generating micro explanations for Bayesian belief networks. In *Proceedings of the 5th Workshop on Uncertainty in Artificial Intelligence*, pages 295–302, Windsor, Ontario.

Shachter, R. and Peot, M. (1992). Decision making using probabilistic inference methods. In *Proceedings of the 8th Annual Conference on Uncertainty in Artificial Intelligence (UAI-92)*, pages 276–28, San Mateo, CA. Morgan Kaufmann.

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

Shachter, R. D. (1998). Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI'98)*, pages 480–487, San Francisco, CA. Morgan Kaufmann.

Shenoy, P. P. (1992). Valuation based systems for Bayesian decision analysis. *Operations Research*, 40:463–484.

Silvestri, G. A., Hoffman, B. J., Bhutani, M. S., Hawes, R. H., Coppage, L., Sanders-Cliette, A., and Reed, C. E. (1996). Endoscopic ultrasound with fine-needle aspiration in the diagnosis and staging of lung cancer. *The Annals of Thoracic Surgery*, 61(5):1441–1445.

Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation and Application.* John Wiley, New York.

Takikawa, M. and D'Ambrosio, B. (1999). Multiplicative factorization of the noisy-MAX. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 622–630, Stockholm, Sweden. Morgan Kaufmann, San Francisco, CA.

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

van der Gaag, L. C. and Coupe, V. M. H. (2000). Sensitivity analysis for threshold decision making with Bayesian belief networks. *Lecture Notes in Artificial Intelligence*, 1792:37–48.

Vomlelova, M. (2003). Unconstrained influence diagrams - experiments and heuristics. In *Sixth Workshop on Uncertainty Processing (WUPES'2003)*, Hejnice, Czech Republic.

Vomlelova, M. and Jensen, F. V. (2002). An extension of lazy evaluation for influence diagrams avoiding redundant variables in the potentials. In *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 186–193, Cuenca, Spain.

Vomlelova, M. and Jensen, F. V. (2004). An extension of lazy evaluation for influence diagrams avoiding redundant variables in the potentials. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(Supplement-1):1–17.

von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, NJ.

Wallis, J. W. and Shortliffe, E. H. (1984). Customized explanations using causal knowledge. In Buchanan, B. G. and Shortliffe, E. H., editors, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, chapter 20, pages 371–388. Addison-Wesley, Reading, MA.

Wellman, M. P. (1990). Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303.

Yasufuku, K., Nakajima, T., Motoori, K., Sekine, Y., Shibuya, K., Hiroshima, K., and Fujisawa, T. (2006). Comparison of endobronchial ultrasound, positron emission tomography, and ct for lymph node staging of lung cancer. *Chest*, 130(3):710.718.

Appendix A

Appendix of Chapter 3

A.0.1 Proof of Theorem 3.2.1

Before proving the theorem, we introduce a definition and a lemma.

Definition A.0.1 *The number of summands of the expansion of a ToP rooted at node n, denoted by $s(n)$, is defined recursively as follows. If n is a terminal node, then $s(n) = 1$. If n has m children, $n_1, \ldots, n_m$, and n is of type sum, then $s(n) = \sum_{i=1}^{m} s(n_i)$; if n is of type product, $s(n) = \prod_{i=1}^{m} s(n_i)$.*

Lemma A.0.1 *When the method* distribute *(Algorithm 3.1) is applied to a node n having a child of type sum, n_1, then $s(n'_{1,l}) < s(n)$ for each child $n'_{1,l}$ of n_1 in the new ToP (see Figure 3.4).*

Proof. We have that $s(n) = s(n_1) \cdot \ldots \cdot s(n_m)$, which implies that $s(n) \geq s(n_1)$. Given that n_1 has more than one child and $s(n_1) = \sum_{l=1}^{k} s(n_{1,l})$, then $s(n_1) > s(n_{1,l})$ for all l, $s(n_1) > 1$, and $s(n) > s(n_2)$. If $n_{1,l}$ was a terminal node, then $s(n'_{1,l}) = s(n_2)$ and $s(n'_{1,l}) < s(n)$. If $n_{1,l}$ was a non-terminal node then $s(n'_{1,l}) = s(n_{1,l}) \cdot s(n_2) < s(n_1) \cdot s(n_2) \leq s(n_1) \cdot \ldots \cdot s(n_m) = s(n)$, which proves the lemma. ■

Proof. [Proof of Theorem 3.2.1]We prove it by induction on the number of summands of the root of the tree, $s(r)$, taking into account that the number of children of every node is finite. If $s(r) = 1$ then the tree has only one terminal node or one product node having a finite number of leaves, and clearly the algorithm terminates. Let us now assume that the theorem holds for every tree such that $s(r) \leq k$ and let us examine a tree such that $s(r) = k + 1$, where $k \geq 1$. If r is of type sum, then each subtree of r has at most k summands (because r has at least two children), and therefore the *unfork* method terminates for each child of r and for r itself. If r is of type product, then at least one of the children of r, say n_i, must be of type sum (otherwise $s(r)$ would be 1). Therefore, the number

of summands for the other children of r is at most $(k+1)/2$, and $(k+1)/2 \leq k$, which means that *unfork* terminates for those children. Similarly, the number of summands of each child of n_i is at most k, which means that the algorithm terminates for each child of n_i and for n_i itself. When all the children of r have processed the *unfork* message, it may happen that two of them, say n_1 and n_2, depend on A. It is then necessary to distribute one of them, say n_2, wrt the other, as shown in Figure 3.4, and to send again the message *unfork* to n_1. Since the lemma above states that $s(n'_{1,l}) < s(n)$, then $s(n'_{1,l}) \leq k$, and the *unfork* method terminates for the children of n_1 and, consequently, for n_1 itself. If r has still other children that depend on A, they must also be distributed wrt n_1, but the process terminates for each of those other children, and given that the number of children of r is finite, the whole process terminates. ■

A.0.2 Proof of Theorem 3.2.2

Proof. We prove the theorem by induction on the depth of the ToP, d. Clearly, the theorem holds for $d = 1$. Let us assume that the theorem holds for any tree whose depth is not greater than d and that there is a tree t of depth $d+1$, whose root r has m children, $n_1, \ldots, n_m$, such that each node n_i represents a potential ψ_i. If r is a sum node, the potential represented by r is:

$$\psi = \psi_1 + \ldots + \psi_m \ .$$

Therefore,

$$\sum_A \psi = \sum_A \psi_1 + \ldots + \sum_A \psi_m \ ,$$

and, according with the induction hypothesis, each potential $\sum_A \psi_i$ can be obtained by summing out A on the terminal nodes that depend of A. If r is a product node, at most one of its children will depend on A, because the tree is non-forked. If none of them depends on A, then $\sum_A \psi = \psi$ and the theorem holds. If one potential, say ψ_j, depends on A, then

$$\sum_A \psi = \sum_A \prod_{i=1}^{m} \psi_i = \left(\prod_{i \neq j} \psi_i \right) \sum_A \psi_j \ .$$

Since the depth of the tree rooted at n_j is d, the theorem holds because of the induction hypothesis. ■

A.0.3 Correctness of the algorithm VE-D

We prove now the correctness of VE-D (cf. Section 3.4.1), which eliminates the variables by applying Algorithm 3.7 iteratively.

Given an ID, let $\{V_1, \ldots, V_n\}$ be a valid elimination sequence for that ID, i.e., a sequence in which the first variables are those in $\mathbf{C}_n$, then D_n, then those in $\mathbf{C}_{n-1}$, and so on; the last variables are D_1 and those in $\mathbf{C}_0$. Because of Equations 3.1 and 3.2,

$$MEU = \operatorname*{op}_{v_n} \ldots \operatorname*{op}_{v_1} P(\mathbf{v}_C : \mathbf{v}_D) \cdot \psi_{U_0}(fPred(U_0)) \ , \tag{A.1}$$

where op is an operator that depends on the type of variable to be eliminated,

$$\operatorname*{op}_{v_i} = \begin{cases} \sum_{v_i} & \text{if } V_i \in \mathbf{V}_C \\ \max_{v_i} & \text{if } V_i \in \mathbf{V}_D \ , \end{cases} \tag{A.2}$$

and $P(\mathbf{v}_C|\mathbf{v}_D)$ is a family of probability distributions defined as follows:

$$P(\mathbf{v}_C|\mathbf{v}_D) = \prod_{V_i \in \mathbf{V}_C} P(v_i|pa(V_i)) \tag{A.3}$$

i.e., for each configuration $\mathbf{v}_D$ we have a probability distribution defined on $\mathbf{V}_C$.

We define $\mathbf{V}^0$ as the set of all the variables, $\mathbf{V}^0 = \mathbf{V}_C \cup \mathbf{V}_D$, and $\mathbf{V}^i$ as the set of variables remaining after eliminating V_i:

$$\forall i,\ 1 \leq i \leq n, \quad \mathbf{V}^i = \mathbf{V}^{i-1} \backslash \{V_i\} \ . \tag{A.4}$$

Clearly, $\mathbf{V}^n = \varnothing$. Analogously, $\mathbf{V}_C^i$ is the set of chance variables remaining after eliminating V_i, $\mathbf{V}_C^i = \mathbf{V}^i \cap \mathbf{V}_C$, and $\mathbf{V}_D^i = \mathbf{V}^i \cap \mathbf{V}_D$. Therefore, $\mathbf{V}_C^{i-1}$ contains all the variables remaining when V_i is to be eliminated.

Lemma A.0.2 *If V_i is a decision, then $P(\mathbf{v}_C^{i-1}|\mathbf{v}_D)$ does not depend on $\check{\mathbf{V}}_D^i$, where $\check{\mathbf{V}}_D^i = \mathbf{V}_D \backslash \mathbf{V}_D^i$.*

Please note that $\mathbf{V}_C^{i-1}$ is the set of chance variables remaining before eliminating V_i, $\mathbf{V}_D^i$ is the set of decisions remaining after eliminating V_i, and therefore $\check{\mathbf{V}}_D^i$ includes V_i and the decisions eliminated before V_i.

Before proving the lemma, we illustrate it with an example.

Example A.0.1 *Coming back to the ID in Example 3.4.1 on page 77 (Fig. 3.9), Equation A.3 tells us that*

$$P(\mathbf{v}_C|\mathbf{v}_D) = P(a,b,c,e|d_1,d_2) = P(a)\cdot P(b|a)\cdot P(c|a,d_1)\ . \tag{A.5}$$

The only valid elimination sequence is $\{V_1 = A, V_2 = D_2, V_3 = C, V_4 = D_1, V_5 = B\}$. *The first decision eliminated is* V_2 (D_2) *and the second one is* V_4 (D_1). *Let us focus on the former. In this case,* $i = 2$, $\mathbf{V}^{i-1} = \{D_2, C, D_1, B\}$, $\mathbf{V}_C^{i-1} = \{C, B\}$, *and* $P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = P(c,b|d_1,d_2)$. *We also have* $\mathbf{V}_D^i = \{D_1\}$ *and* $\check{\mathbf{V}}_D^i = \{D_2\}$. *The lemma states that* $P(c,b|d_1,d_2)$ *does not depend on* d_2, *which is obvious, because* $P(c,b|d_1,d_2) = \sum_a P(a,b,c|d_1,d_2) = \sum_a P(a)\cdot P(b|a)\cdot P(c|a,d_1)$ *and none of the factors inside the last summatory depends on* d_2. *Let us focus now on the second decision eliminated,* D_1. *Then,* $i = 4$, $\mathbf{V}^{i-1} = \{D_1, B\}$, $\mathbf{V}_C^{i-1} = \{B\}$, *and* $P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = P(b|d_1,d_2)$. *We also have* $\mathbf{V}_D^i = \varnothing$ *and* $\check{\mathbf{V}}_D^i = \{D_1, D_2\}$. *The lemma states that* $P(b|d_1,d_2)$ *does not depend on* d_1 *nor on* d_2. *This result is not obvious, because apparently* $P(b|d_1,d_2)$ *depends on* d_1*:*

$$P(b|d_1,d_2) = \sum_a \sum_c P(a,b,c,e|d_1,d_2) = \sum_a \sum_c P(a)\cdot P(b|a)\cdot P(c|a,d_1)\ . \tag{A.6}$$

Now, we prove the lemma.

Proof. We build a Bayesian network (BN) as follows: we create a chance node for each variable in $\mathbf{V}$. If V_i is a chance variable in the ID, we draw a link from each node that was a parent of V_i in the ID; the conditional probability distribution of V_i in the BN is the same as in the ID. If V_i is a decision in the ID, then V_i has no parents in the BN; we assign it an arbitrary distribution, for instance, a uniform probability. [The BN for the ID in Figure 3.9 is shown in Figure A.1.] If $P_{BN}(\mathbf{v})$ is the join probability of the BN, then it follows from Equation A.3 that

$$P(\mathbf{v}_C|\mathbf{v}_D) = P_{BN}(\mathbf{v}_C|\mathbf{v}_D) \tag{A.7}$$

and, consequently,

$$P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = P_{BN}(\mathbf{v}_C^{i-1}|\mathbf{v}_D)\ . \tag{A.8}$$

Now we focus on the decision V_i. In the BN, $\mathbf{V}_C^{i-1}$ is conditionally independent of $\check{\mathbf{V}}_D^i$ given $\mathbf{V}_D^i$, for the following reason: In the BN the nodes that correspond to decisions in the ID do not have parents. Therefore, any path departing from a decision V_j in $\check{\mathbf{V}}_D^i$ (V_j can be V_i itself) must pass through a child of V_j, that we call X, which was a chance variable in the ID. This node X and its descendants have been eliminated before V_i and

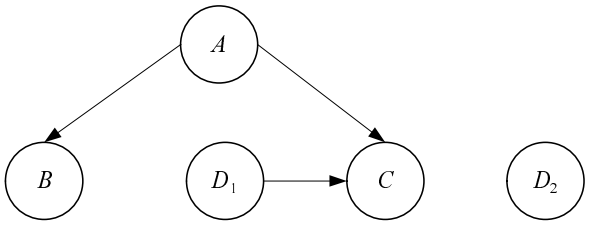

Figure A.1: Bayesian network for the ID in Figure 3.9 (see the proof of Lemma A.0.2).

before V_j, because a descendant of V_j can not be an informational predecessor of any of these decisions. Additionally, no node in $\mathbf{V}_D^i$ is a descendant of V_j. This implies that any path from any node in $\check{\mathbf{V}}_D^i$ to the any node in $\mathbf{V}_C^{i-1}$ is inactive given $\mathbf{V}_D^i$, in the sense of d-separation Pearl (1988), and consequently $\mathbf{V}_C^{i-1}$ is conditionally independent of $\check{\mathbf{V}}_D^i$ given $\mathbf{V}_D^i$.[1] From the fact that $\mathbf{V}_C^{i-1}$ is conditionally independent of $\check{\mathbf{V}}_D^i$ given $\mathbf{V}_D^i$, we conclude thatp

$$P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = P_{BN}(\mathbf{v}_C^{i-1}|\underbrace{\mathbf{v}_D^i, \check{\mathbf{v}}_D^i}_{\mathbf{v}_D}) = P_{BN}(\mathbf{v}_C^{i-1}|\mathbf{v}_D^i) \ , \tag{A.9}$$

which proves that $P(\mathbf{v}_C^{i-1}|\mathbf{v}_D)$ does not depend on $\check{\mathbf{V}}_D^i$. ■

We define ϕ^0 as the set of all the probability potentials, $\phi^0 = \{P(v_i|pa(V_i)) \ |V_i \in \mathbf{V}_C\}$, and ϕ^i as the list of probability potentials (LoPP) handled by VE-D *after* eliminating variable V_i. We denote by $\Pi\phi^i$ the product of all the potentials in ϕ^i.

Proposition A.0.1 *At each step of algorithm VE-D, the list of probability potentials (LoPP) represents the probability of the chance variables that have not been eliminated yet:*

$$\forall i, \ 0 \leq i \leq n, \quad \Pi\phi^i = P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) \ . \tag{A.10}$$

Proof. We prove it by induction on i. When $i = 0$, i.e., before eliminating any variable, the LoPP contains all the conditional probability potentials that define the ID, i.e., ϕ^0; in this case, the proposition holds because of Equation A.3. Let us assume that it holds for $i - 1$:

$$\Pi\phi^{i-1} = P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) \ . \tag{A.11}$$

[1] Coming back to Example A.0.1, when eliminating D_2 ($i = 2$), we have $P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = P(b, c|d_1, d_2) = P_{BN}(b, c|d_1, d_2)$. Because of d-separation (see Figure A.1), $P_{BN}(b, c|d_1, d_2) = P_{BN}(b, c|d_1)$, which explains why $P(b, c|d_1, d_2)$ does not depend on D_2.

When eliminating D_1 ($i = 4$), we have $P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = P(b|d_1, d_2) = P_{BN}(b|d_1, d_2)$. Again, because of d-separation, $P_{BN}(b|d_1, d_2) = P_{BN}(b)$, which explains why $P(b|d_1, d_2)$ does not depend on D_1 nor on D_2.

We divide ϕ^{i-1} in two sets: ϕ_+^{i-1} contains the potentials that depend on V_i and ϕ_-^{i-1} those that do not. If V_i is a chance variable, then $\mathbf{V}_C^i = \mathbf{V}_C^{i-1} \backslash \{V_i\}$ and

$$P(\mathbf{v}_C^i|\mathbf{v}_D) = \sum_{v_i} P(\underbrace{\mathbf{v}_C^i, v_i}_{\mathbf{v}_C^{i-1}}|\mathbf{v}_D) = \sum_{v_i} \Pi\phi^{i-1} = \Pi\phi_-^{i-1} \underbrace{\sum_{v_i} \Pi\phi_+^{i-1}}_{\phi_{V_i}^*} = \Pi\phi^i \ . \tag{A.12}$$

Algorithm 3.7 just implements this equation, because it leaves in the LoPP the potentials that do not depend on V_i, namely ϕ_-^{i-1}, and replaces those that depend on V_i with a new potential $\phi_{V_i}^*$ computed by multiplying all those potentials and summing out V_i. If V_i is a decision, then $P(\mathbf{v}_C^{i-1}|\mathbf{v}_D)$ does not depend on V_i, because of Lemma A.0.2. Given that $P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = \Pi\phi_+^{i-1} \cdot \Pi\phi_-^{i-1}$ and no potential in ϕ_-^{i-1} depends on V_i, then $\Pi\phi_+^{i-1}$ can not depend on V_i. Therefore,

$$P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = \Pi\phi^{i-1} = \Pi\phi_-^{i-1} \cdot \underbrace{\text{project}_{V_i} \Pi\phi_+^{i-1}}_{\phi_{V_i}^*} = \Pi\phi^i \ . \tag{A.13}$$

Given that $\mathbf{V}^i = \mathbf{V}^{i-1} \backslash \{V_i\}$ and V_i is a decision, then $\mathbf{V}_C^i = \mathbf{V}_C^{i-1}$ and

$$P(\mathbf{v}_C^i|\mathbf{v}_D) = P(\mathbf{v}_C^{i-1}|\mathbf{v}_D) = \Pi\phi^i \ . \tag{A.14}$$

■

Theorem A.0.1 *We have*

$$\forall i,\ 0 \leq i \leq n, \quad \text{MEU} = \underset{v_n}{\text{op}} \ldots \underset{v_{i+1}}{\text{op}}\ \Pi\phi^i \cdot \psi^i \ , \tag{A.15}$$

where ψ^i is the potential represented by the ADG of utility potentials (ADGoUP) after algorithm VE-D has eliminated variable V_i.

Proof. We prove it by induction on i. The theorem holds for $i = 0$ because $\Pi\phi^0 = P(\mathbf{v}_C|\mathbf{v}_D)$ and originally the ADGoUP represents the utility of the ID: $\psi^0 = \psi_{U_0}(fPred(U_0))$. Let us assume that it holds for $i-1$:

$$MEU = \underset{v_n}{\text{op}} \ldots \underset{v_i}{\text{op}}\ \Pi\phi^{i-1} \cdot \psi^{i-1} \ . \tag{A.16}$$

If V_i is a chance variable,

$$\operatorname*{op}_{v_i} \Pi\phi^{i-1} \cdot \psi^{i-1} = \sum_{v_i} \Pi\phi^{i-1} \cdot \psi^{i-1} \tag{A.17}$$

$$= \phi_-^{i-1} \sum_{v_i} \Pi\phi_+^{i-1} \cdot \psi^{i-1} \tag{A.18}$$

$$= \underbrace{\phi_-^{i-1} \cdot \phi^*_{V_i}}_{\phi^i} \underbrace{\sum_{v_i} \frac{\phi_{V_i}}{\phi^*_{V_i}} \cdot \psi^{i-1}}_{\psi^i}, \tag{A.19}$$

where ϕ_{V_i} and $\phi^*_{V_i}$ are defined as in Algorithm 3.7: $\phi_{V_i} = \Pi\phi_+^{i-1}$ and $\phi^*_{V_i} = \phi_{V_i}$. When comparing this equation with Algorithm 3.7, is is clear that

$$\operatorname*{op}_{v_i} \Pi\phi^{i-1} \cdot \psi^{i-1} = \phi^i \cdot \psi^i \tag{A.20}$$

If V_i is a decision, then $\Pi\phi^{i-1}$ does not depend on V_i and

$$\operatorname*{op}_{v_i} \Pi\phi^{i-1} \cdot \psi^{i-1} = \max_{v_i} \Pi\phi^{i-1} \cdot \psi^{i-1} \tag{A.21}$$

$$= \Pi\phi^{i-1} \max_{v_i} \psi^{i-1} \tag{A.22}$$

$$= \Pi\phi_-^{i-1} \cdot \underbrace{\Pi\phi_+^{i-1}}_{\phi^*_{V_i}} \cdot \underbrace{\max_{v_i} \psi^{i-1}}_{\psi^i}. \tag{A.23}$$

As $\Pi\phi_+^{i-1}$ does not depend on V_i, $\Pi\phi_+^{i-1} = \text{project}_{V_i}\Pi\phi_+^{i-1} = \phi^*_{V_i}$. On the other hand, $\psi^i = \max_{v_i} \psi^i$, which implies that Equation A.20 also holds when V_i is a decision. This result, together with Equation A.16, proves the theorem. ■

Corollary A.0.1 *For every ID, Algorithm 3.7 returns the MEU and an optimal policy.*

Appendix B

Appendix of Chapter 5

B.1 Figures and table of experimental results for comparing DP and BF-A

	0.0 %	**25.0 %**	**50.0 %**	**75.0 %**	**100.0 %**
$EU^1(t)$	0	-0,026	-0,04	-0,028	0,052
$EU^2(t)$	0	0,434	0,645	0,745	0,716
$EU^3(t)$	0	0,363	0,502	0,59	0,598
$AccFreqDec^1(t)$	0	0,011	-0,017	-0,024	0,029
$AccFreqDec^2(t)$	0	0,127	0,18	0,215	0,244
$AccFreqDec^3(t)$	0	0,086	0,124	0,149	0,169

Table B.1: Table for BF-A

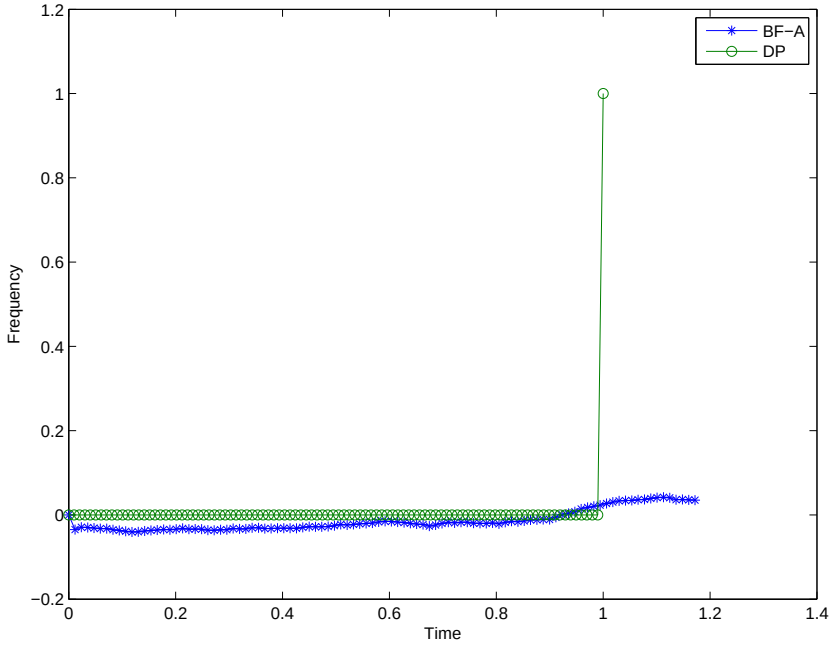

Figure B.1: Comparison of $FreqDec(t)$ between DP and BF-A

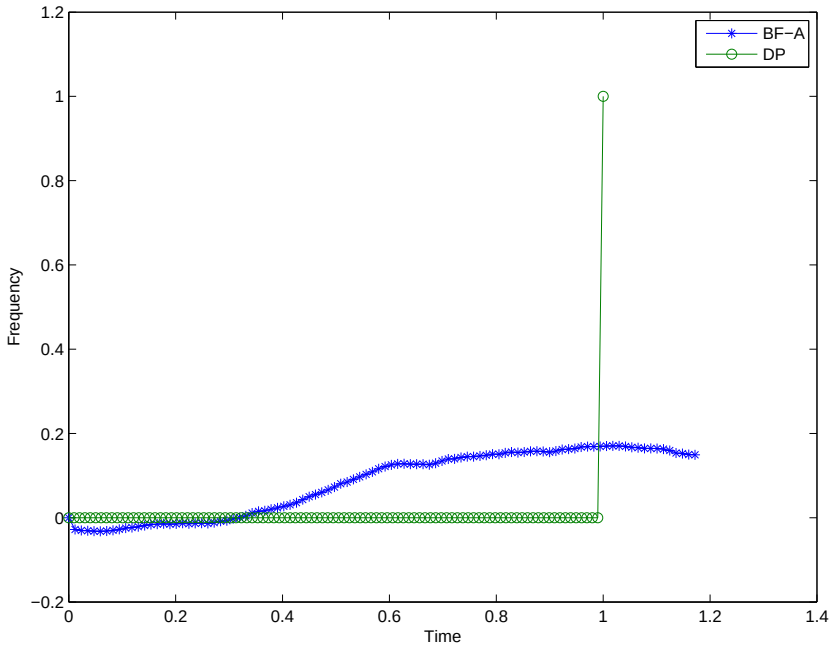

Figure B.2: Comparison of $FreqOpt(t)$ between DP and BF-A

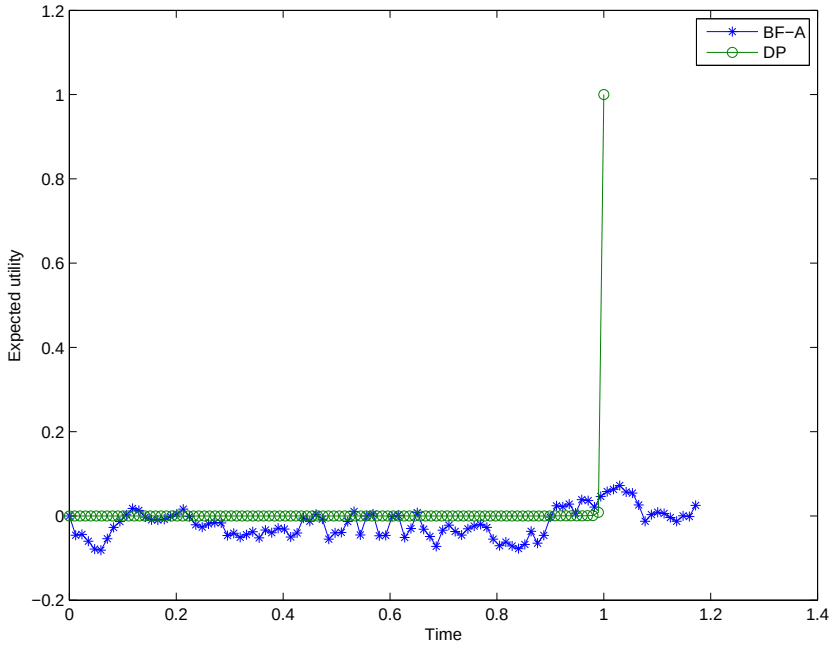

Figure B.3: Comparison of $EU^1(t)$ between DP and BF-A

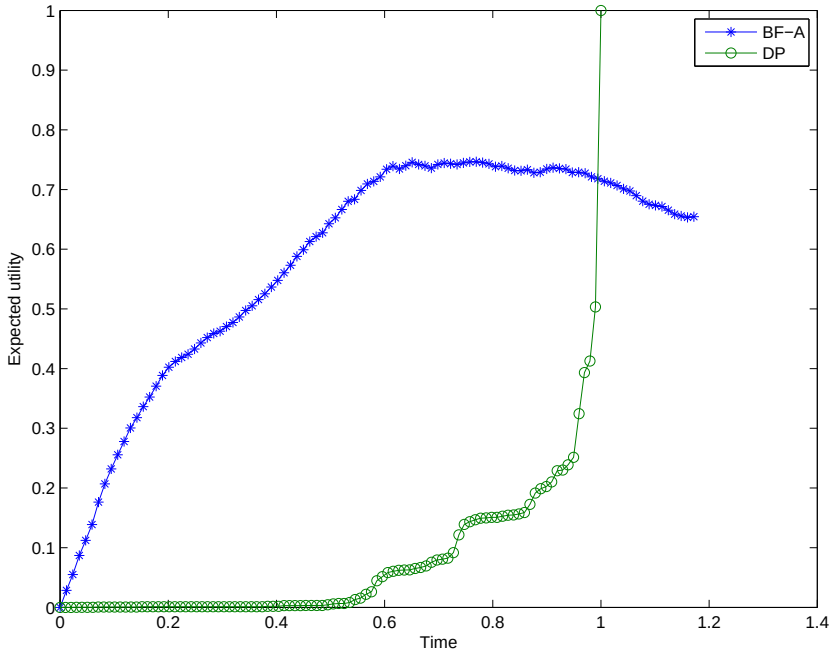

Figure B.4: Comparison of $EU^2(t)$ between DP and BF-A

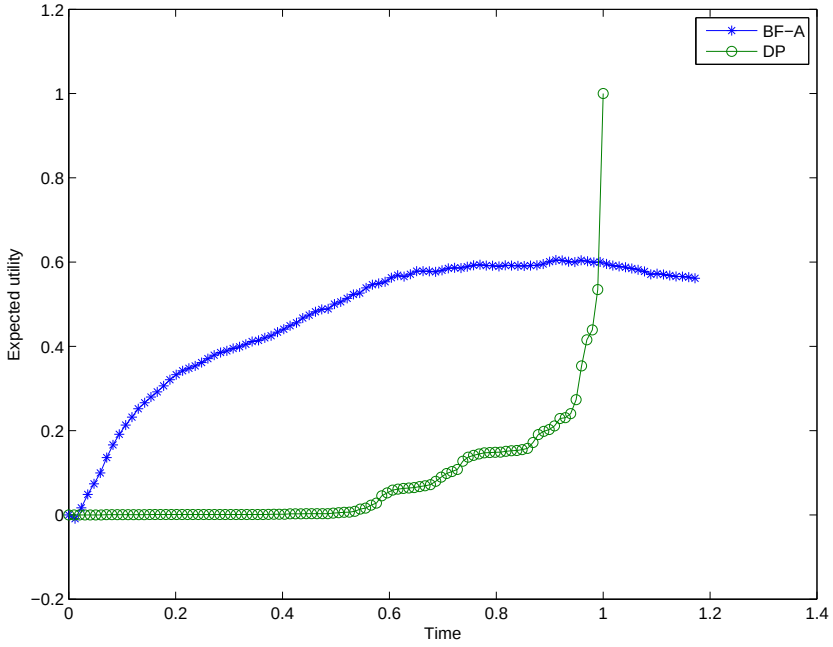

Figure B.5: Comparison of $EU^3(t)$ between DP and BF-A

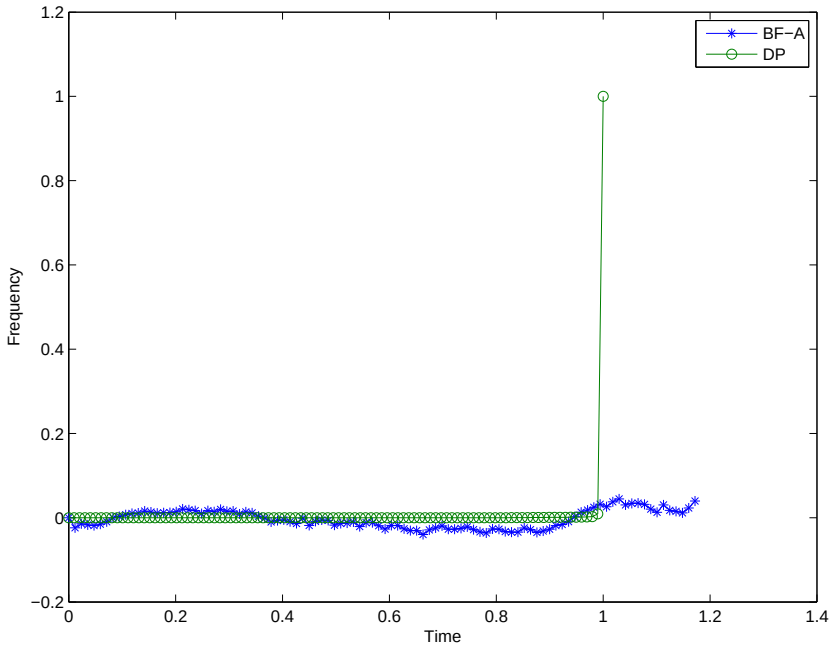

Figure B.6: Comparison of $AccFreqDec^1(t)$ between DP and BF-A

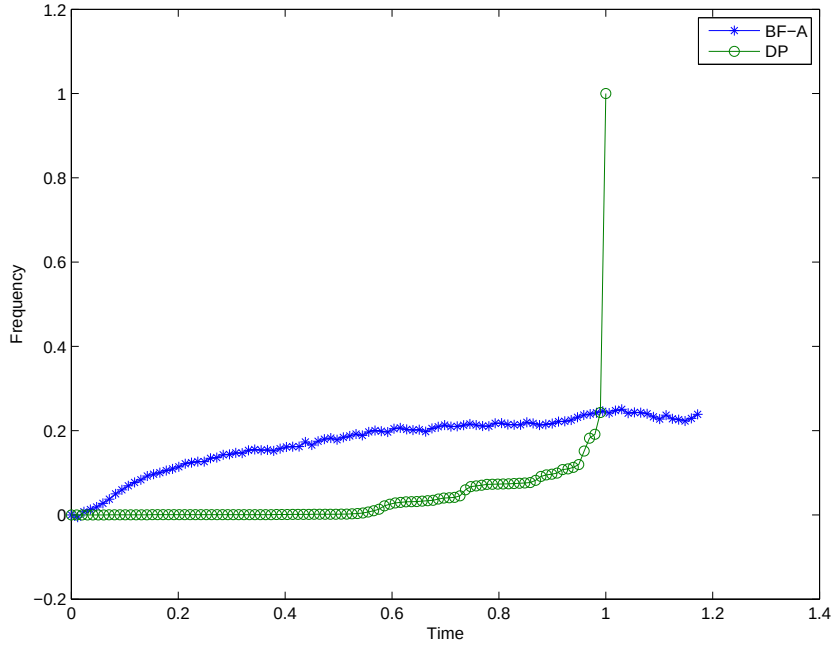

Figure B.7: Comparison of $AccFreqDec^2(t)$ between DP and BF-A

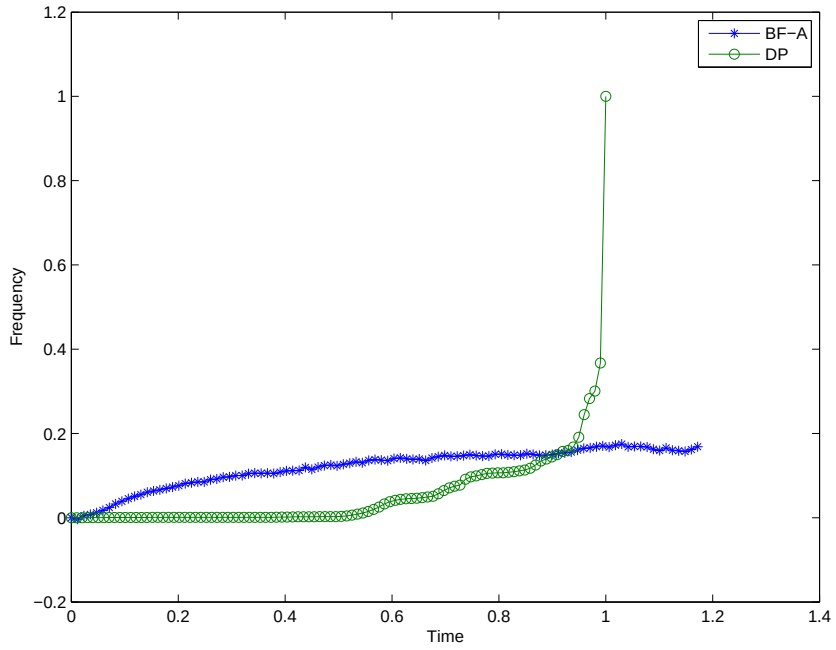

Figure B.8: Comparison of $AccFreqDec^3(t)$ between DP and BF-A

Appendix C

Software developed in Elvira

We have used Elvira for most of the software developed in this thesis. The author of this thesis has been an Elvira developer during the doctorate period[1], having full access to Elvira repository and with permissions for modifying the source code. Most of the algorithms implemented in this thesis have been uploaded to Elvira repository and made publicly available. They are usually accessible through both the GUI and the API. The main contributions of this thesis to Elvira software are the next:

- Functionalities for IDs with SVNs, which can be accessed through Elvira GUI and its API:
  - Edition and evaluation of IDs with SVNs.
  - Two solution algorithms:
    - Tatman and Shachter's algorithm.
    - Variable elimination for IDs with SVNs.
- Explanation capabilities for IDs through Elvira GUI and its API. Since most of the explanations integrated are graphical we recommend the user to access them via Elvira GUI.
- Functionalities of UIDs through the API of Elvira:
  - Edition and evaluation of UIDs.
  - Two evaluation algorithms:
    - Dynamic programming algorithm.
    - Anytime algorithm for solving UIDs.

[1] The author was granted by Elvira project during the first months of the thesis and his work in that period consisted basically in programming in Elvira.

The integration of the functionalities of UIDs in Elvira GUI is in an advanced stage, but it has not still been uploaded to Elvira repository.

Apart from the software developed in Elvira, we have also built an ID for the mediastinal staging of non-small lung cancer. It is stored in a file that can be edited, debugged and evaluated with Elvira.

Apéndice D

Resumen en Español (Spanish Summary)

D.1 Motivación

Los modelos gráficos probabilísticos (MGP), en particular las redes bayesianas y los diagramas de influencia, fueron desarrollados en los años 80 por investigadores del campo de Inteligencia Artificial, Matemáticas y Economía con el propósito de resolver problemas cuya complejidad excede la capacidad de los métodos existentes hasta entonces. Hoy en día los MGP son aplicados a muchas áreas y existe un interés creciente en el campo académico y en el mundo empresarial. Los MGP permiten resolver problemas que no podrían ser abordados con los método probabilísticos tradicionales o con otras técnicas de Inteligencia Artificial.

Varios grupos de investigación españoles interesados en los MGP fueron surgiendo de forma independiente en diferentes universidades. El trabajo en MGP comenzó en la UNED en 1990 con la tesis doctoral de Díez (1994), que consistió en la construcción del sistema experto DIAVAL, una red bayesiana para el diagnóstico de enfermedades cardíacas por ecocardiografía.

Algunos años más tarde, el Dr. Carlos Disdier Vicente, un neumólogo en el Hospital San Pedro de Alcántara, en Cáceres (España), y Javier Díez, en la UNED, comenzaron la construcción de un diagrama de influencia para la estadificación mediastínica del cáncer de pulmón. Cuando el autor de esta tesis entró a formar parte del grupo de investigación, en la UNED, en 2003, se le asignó como tema de investigación completar la construcción del diagrama de influencia, que se encontraba en un estado muy incipiente.

La investigación de este grupo siempre ha estado guiada por problemas médicos concretos: las necesidades surgidas al construirlos ha motivado el desarrollo de nuevos modelos, algoritmos, y herramientas software, que posteriormente han sido aplicados a otros

problemas, no solamente en medicina. Ha sido también el caso de la presente tesis.

En primer lugar, la forma de la la función que combina las utilidades (esto es, las preferencias del decisor, en nuestro caso, la cantidad y calidad de vida de pacientes con cáncer de pulmón) nos llevó a una estructura de nodos de utilidad en nuestro diagrama de influencia. Esto nos llevó a desarrollar un nuevo algoritmo de evaluación que pudiera mejorar el único algoritmo existente hasta ese momento para ese tipo de diagramas de influencia.

En segundo lugar, durante la interacción con el experto vimos la necesidad de contar con capacidades de explicación para diagramas de influencia, que nos ayudarían en la construcción del modelo y en su depuración. Además, serian también útiles al intentar convencer al experto de los resultados. Por esta razón implementamos nuevos métodos de explicación en Elvira.

Tercero, debido a la incertidumbre en los parámetros del diagrama de influencia, asignados por el experto basado en la literatura y en sus propios datos, implementamos algunas técnicas de análisis de sensibilidad.

Y cuarto, debido a la discusión en la literatura médica acerca del orden óptimo en que los tests para la estadificación mediastínica del cáncer de pulmón deberían ser realizados, y dado que los diagramas influencia requieren un orden total, exploramos el uso de los diagramas de influencia no restringidos (Jensen and Vomlelova, 2002), una representación que permite un orden parcial entre las decisiones, y desarrollamos un nuevo algoritmo "anytime" que proporcionara una recomendación para las primeras decisiones cuando encontrar la estrategia óptima requiriese una excesiva cantidad de tiempo.

En esta tesis describimos los algoritmos y métodos, que no son específicos de medicina, y también el sistema de apoyo a la toma de decisiones para la estadificación mediastínica del cáncer de pulmón, llamado MEDIASTINET.

D.2 Objetivos

Debido a las necesidades descritas en la sección previa, los objetivos de esta investigación pueden resumirse como sigue:

- Desarrollar un algoritmo de eliminación de variables para diagramas de influencia (DIs) con nodos super valor (SV), y compararlo con el algoritmo de inversión de arcos de Tatman and Shachter (1990).

- Tener capacidades de explicación y herramientas de análisis de sensibilidad para DI con nodos SV.

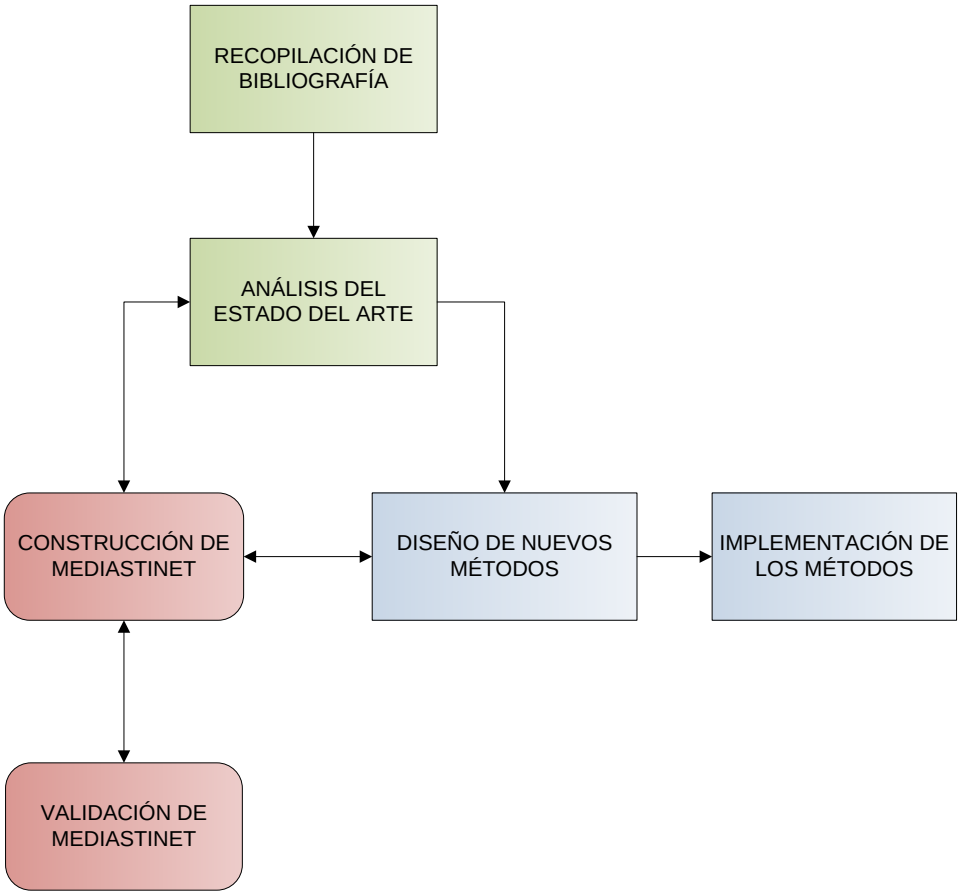

Figura D.1: Fases en el desarrollo de esta tesis doctoral.

- Desarrollar un algoritmo anytime para diagramas de influencia no restringidos.
- Construir y evaluar un sistema para el apoyo a la toma de decisiones para la estadificación mediastínica del cáncer de pulmón, al que hemos denominado MEDISTINET.

D.3 Metodología

La metodología seguida para lograr los objetivos corresponde a varias fases, mostradas en la figura Figure D.1.

La primera fase consistió en la recopilación de la bibliografía sobre modelos gráficos probabilísticos aplicados a la resolución de problemas de decisión. La siguiente fase consistió en el estudio y análisis del estado del arte. A continuación vino la etapa de comenzar la construcción el diagrama de influencia MEDIASTINET con la ayuda del neumólogo, el Dr. Carlos Disdier Vicente. La construcción del diagrama de influencia nos llevó a desarrollar nuevos métodos y a implementarlos como herramientas software. La última fase consistió en la validación del sistema, que llevó a la modificación del modelo con la ayuda

del experto en un proceso iterativo.

Es importante señalar que, después de las fases iniciales (coloreadas en verde en la figura D.1) correspondientes a la recopilación y análisis de la bibliografía, el trabajo estuvo guiado por las necesidades que fueron apareciendo durante la construcción de MEDIASTINET. Por tanto, la realimentación entre las diferentes fases de la investigación fue esencial.

D.4 Organización de la tesis

Hemos estructurado la tesis en cuatro partes.

La parte I explica motivación, objetivos, y metodología de la tesis.

La parte II revisa el estado del arte y presenta los fundamentos matemáticos para el resto de la tesis.

La parte III contiene los avances metodológicos de la tesis. El capítulo 3 describe un algoritmo de eliminación de variables para diagramas de influencia con nodos super-valor. El capítulo 4 presenta las capacidades de explicación y las técnicas de análisis de sensibilidad implementadas para diagramas de influencia con nodos super-valor. El capítulo 4 explica un algoritmo anytime para evaluar diagramas de influencia no restringidos.

La parte IV presenta un sistema de ayuda a la decisión para la estadificación mediastínica del cáncer de pulmón.

Los capítulos de la parte III y IV han sido escritos siguiendo la *jerarquía de niveles,* introducida por Marr (1982). En Ciencias de la Computación, Marr propuso la siguiente jerarquía de niveles para cualquier problema computacional:

- **Teoría** computacional, que indica el objetivo de la computación, su justificación y la teoría que es necesaria para su implementación. En nuestro caso, corresponde básicamente al estudio de los modelos gráficos probabilísticos.

- **Algoritmos**, que consiste en establecer cuáles son las entradas y las salidas, y cuál es el algoritmo que las relaciona.

- **Implementación**, que traduce los algoritmos a un programa que se ejecuta en un ordenador.

Finalmente, la parte **??** presenta las conclusiones y el trabajo futuro.

D.5 Principales contribuciones

En esta tesis hemos considerado la representación, solución de problemas de decisión médicos con modelos gráficos probabilísticos.

Hemos propuesto un nuevo algoritmo para evaluar diagramas de influencia (DI) con nodos super valor (SV), que tiene cinco ventajas sobre el algoritmo de inversión de arcos de Tatman and Shachter (1990): es más rápido en general, requiere menos memoria en la mayoría de los casos, introduce menos variables redundantes, simplifica el análisis de sensibilidad, y puedo conseguir ahorros adicionales de tiempo y espacio de memoria en diagramas de influencia que contengan modelos canónicos, tales como la puerta OR o la puerta MAX con ruido (ver el capítulo 3).

Hemos desarrollado nuevos métodos de explicación—ver el capítulo 4—para el modelo (el conocimiento codificado en el diagrama de influencia) y el razonamiento (las estrategias obtenidas del diagrama de influencia), que han probado ser muy útiles en la construcción y depuración de MEDIASTINET, y ayudaron a convencer al experto de que las recomendaciones dadas por nuestro modelo eran razonables. En particular, los *árboles de políticas*, que han sido propuestos como una forma compacta de representar las políticas óptimas del DI, han sido muy útiles para un modelo como MEDIASTINET, en el que la tabla de la política mayor contiene 15552 columnas, mientras que el correspondiente árbol de política contiene 5 ó 9 hojas, dependiendo del criterio de evaluación. En el futuro estas capacidades de explicación podrían ser usadas para convertir los diagramas de influencia en sistemas de tutorización.

Hemos implementado algunos algoritmos de análisis de sensibilidad (ver la sección 4.5) que, dada la incertidumbre en los parámetros, nos permiten computar:

- la probabilidad de cambio en la estrategia;
- los intervalos de los parámetros donde las políticas no cambian; y
- el valor esperado de la información perfecta para cada parámetro.

Hemos desarrollado un algoritmo anytime para la evaluación de diagramas de influencia no restringidos, y tipo de modelos gráficos probabilísticos que permite una ordenación parcial de las decisiones. El objetivo del método propuesto es proporcionar una recomendación cualificada para las primeras decisiones cuando el decisor no tiene tiempo para esperar hasta que el algoritmo estándar haya computado la estrategia óptima (ver capítulo 5). La calidad de las recomendaciones fue medida considerando:

- la frecuencia con que el algoritmo anytime devuelve la elección correcta (relativa a la estrategia óptima) para todas las decisiones desde la raíz hasta el nivel i en el árbol de decisión;

- la utilidad esperada de seguir la estrategia recomendada por el algoritmo anytime o por programación dinámica para los primeros i niveles de decisiones, seguida por la estrategia óptima para las decisiones restantes.

De los resultados podemos ver que el algoritmo mejora a medida que avanza el tiempo, con respecto a todas las características registradas.

Como aplicación de los modelos gráficos probabilísticos, hemos construido un sistema de ayuda a la decisión para la estadificación mediastínica del cáncer de pulmón (capítulo 6). El parámetro λ, que en análisis coste efectividad representa la cantidad de dinero que el decisor está dispuesto a pagar para obtener una unidad de efectividad, ha sido incluido en el diagrama de influencia introduciendo un nodo de utilidad que representa $1/\lambda$.

Primero, hemos evaluado el diagrama de influencia con $\lambda = +\infty$, es decir, $(1/\lambda) = 0$, para obtener la estrategia que maximiza la efectividad, sin tener en cuenta los costes económicos. Entonces, lo hemos evaluado otra vez con $\lambda = 30,000$ €/AVAC, que es aceptado como la equivalencia coste-efectividad en la "sombra" en España.[1]

El experto dijo que las estrategias óptimas generadas por MEDIASTINET eran muy razonables y "lógicas", y que el sistema era "bastante inteligente".

Dado que no sabemos con precisión los valores de los parámetros, hemos representado esta incertidumbre asignando, con la ayuda del experto, una distribución de probabilidad a cada parámetro independiente del modelo (ver sección 6.6.1). Nos alegramos al darnos cuenta de que, después de realizar el análisis de sensibilidad, las políticas óptimas resultantes de MEDIASTINET eran muy robustas a las variaciones en los parámetros (ver sección 6.6.1): el parámetro con el mayor impacto es λ, como se esperaba.

D.6 Trabajo futuro

Hay algunas líneas abiertas para trabajo futuro.

[1] En este contexto, *sombra* significa que este valor no ha sido explícitamente establecido por las autoridades sanitarias, sino que ha sido estimado por algunos investigadores (Sacristán et al., 2002) tras analizar qué intervenciones son incluidas en el sistema público de saludo de España y cuáles han sido excluidas.

Respecto al algoritmo de eliminación de variables para DIs con nodos SV, presentado en el capítulo 3, las tras principales cuestiones que deberían ser investigadas son las siguientes:

1. evitar introducir todas las variables redundantes, si es posible;
2. desarrollar reglas heurísticas para este algoritmo, principalmente para encontrar ordenaciones de eliminación casi óptimas, y también decidir cómo distribuir los potenciales en combinaciones suma-producto; y
3. cómo combinar nuestro algoritmo con las propuestas para evaluación perezosa de los DIs tradicionales, con el fin de reducir el coste computacional.

Con respecto a la explicación del razonamiento en DIs (capítulo 4) hay tres principales cuestions a investigar:

1. encontrar algún método que pudiera ayudar a explicar al experto qué variables están teniendo más influencia en la estrategia óptima y en la máxima utilidad esperada;
2. encontrar representaciones muy simples de políticas óptimas no sólo asumiendo que el decisor actúa de forma óptima en cada decisión, sino también considerando los escenarios no óptimos.
3. resolver dos cuestiones relacionadas con análisis de sensibilidad en DIs con nodos SV: cómo realizar análisis n-way conjunto; y cómo calcular con métodos exactos los umbrales de cambio de política.

El algoritmo anytime propuesto, presentado en el capítulo 5, podría ser mejorado de varias formas:

1. combinar nuestro método con algún enfoque basado en programación dinámica, que podría acelerar la convergencia hacia la estrategia óptima;
2. encontrar formas alternativas para el heurístico no-admisible usado en la búsqueda; y
3. explotar la coalescencia al construir el árbol de decisión.

Con respecto a la aplicación de MEDIASTINET (capítulo 6) tenemos dos líneas de investigación sugeridas por el experto:

1. considerar la posibilidad de tener un orden parcial entre las decisiones del diagrama de influencia;

2. representar en el modelo explícitamente la posibilidad de repetir algunos tests médicos.

Índice alfabético