Optimization of Policy Trees for Influence Diagrams

Master Thesis

Laura Prada Gracia

September, 2014

Supervisors:

Manuel Luque
Francisco Javier Díez

Universidad Nacional de Educación a Distancia
Máster Universitario en I.A. Avanzada:
Fundamentos, Métodos y Aplicaciones.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This chapter presents the main information related to this project, explaining the motivation of this work, the different stages in the project and the goals set.

1.1 Motivation

Probabilistic graphical models (PGMs) are useful modeling tools for decision making under uncertainty. Uncertainty appears to be an inescapable aspect of most real-world applications. It is a consequence of several factors. We are often uncertain about the true state of the system because our observations about it are partial: only some aspects of the world are observed; for example, the patient's true disease is often not directly observable, and his future prognosis is not yet observed. The true state of the world is rarely determined with certainty by our limited observations, as most relationships are simply not deterministic, at least relative to our ability to model them. Thus, we need to consider different possibilities, and reason not just about what is possible, but also about what is probable.

Most tasks require a person or an automated system to reason: to take the available information and reach conclusions, both about what might be true in the world and about how to act. For example, a doctor needs to take information about a patient — his symptoms, test results, personal characteristics (gender, weight) — and reach conclusions about what diseases he may have and what course of treatment to undertake. PGMs allow us to deal with this big problems that could not be addressed with traditional probabilistic methods. Medicine is the field where more PGMs applications have been built.

Bayesian networks (BNs) and influence diagrams (IDs) are PGMs widely used for building diagnosis- and decision-support expert systems. Explanation of the reasoning is important for users in order to accept them as tutoring systems. Unfortunately, most expert systems have virtually no explanatory capability [21]. This was one of the reasons why OpenMarkov was developed. OpenMarkov is an open-source software tool for editing and evaluating proba-

bilistic graphical models (PGMs) developed by the Research Center for Intelligent Decision-Support Systems of the UNED in Madrid, Spain.

OpenMarkov is able to represent several types of networks, such as Bayesian networks (BNs), Markov networks, or influence diagrams (IDs). After solving an ID, the main output is the optimal strategy. Software tools for PGMs usually present each optimal policy in the form of a policy table, which contains a column for each configuration of the informational predecessors of the decision. One of the main drawbacks is that in some cases these tables can have a large size. For example, in MEDIASTINET, a decision support-system (DSS) for the mediastinal staging of non-small cell lung cancer [26], the biggest policy table has 15,552 columns, most of which correspond to impossible scenarios. For a human expert it is virtually impossible to draw any useful information from this table. For this reason it is necessary to have an alternative representation that summarizes the optimal policy in an understandable way.

1.2 Objectives

In this project we will focus on two decision analysis models: influence diagrams and decision trees. We will use IDs as a model building tool. The purpose of this project is to manipulate the policy tables resulting from the evaluation of ID and return a compact strategy that can be used easily to make a decision.

We analyze different algorithms for creating decision trees, such as Iterative Dichotomiser 3 (ID3) [42]. In this work we propose Policy Trees (PTs) [26] as an alternative representantion of policy tables. PTs have been mathematically formulated by means of the algorithm ID3. During the interaction and implementation of our new algorithm, we felt the need for new methods that would help us to compact the decision trees. Thus as a way of optimization, in this report we propose two methods that reduce the size of the decision trees. Finally, after designing and implementing the new tree algorithm, we evaluate on several influence diagrams.

1.2.1 Report Structure

This report is organized as follows:

Chapter 1: Introduction. The current chapter gives a short introduction to this project, its motivation and its goals.

Chapter 2: State of the art: PGMs and Decision Trees. It describes the main concepts about Probabilistic Graphical Models (PGMs), decision trees, and the software tool OpenMarkov.

Chapter 3: Building and Optimizing Policy Trees. This chapter presents the implementation of a new method of representation called Policy Trees which is based on ID3.

Chapter 4: Evaluation of the new algorithm. In this chapter there are a total of five tryouts in order to show how our algorithm works with different IDs.

Chapter 5: Conclusions and Future Work. It summarizes the work done during this project, exposes the main conclusions and proposes future research lines.

Chapter 2

State of the art: PGMs and Decision Trees

As an introduction, this chapter begins with a section on the importance of information visualization. This chapter provides the basic concepts of PGMs, Decision trees, and also it introduces the reader to the software tool OpenMarkov. We consider the subclass of probabilistic networks known as Bayesian networks (BNs) and influence diagrams (IDs). More specifically, this project focuses on influence diagrams and thus they will be explained in detail. Finally this chapter presents the ID3 algorithm [38, 35] for constructing classification trees.

2.1 Information visualization

Problem solvers in domains like physics and engineering make extensive use of diagrams in problem solving. As Jean-Daniel Fekete and colleagues [11] suggested, the most accepted definition of the Information Visualization comes from Card et al. [6], who describe visualization as "the use of computer-supported, interactive, visual representations of data to amplify cognition". The last three words of their definition communicate the ultimate purpose of visualization, to amplify cognition. Is the amplification of cognition something with a ground truth that is easily and precisely measurable? Clearly it is not. Furthermore, how does one quantify the benefits of an information visualization system? For these and other reasons, information visualization is fundamentally challenging to evaluate [34].

Information visualization augment human memory to provide a larger working set for thinking and analysis and thus become external cognition aids. Card et al. [6] listed a number of key ways in which visuals can amplify cognition:

- Increasing memory.
- Reducing the search for information.

- Enhancing the recognition of patterns.
- Enabling perceptual inference operations.

When two representations are informationally equivalent, their computational efficiency depends on the representation capabilities for recognizing patterns and in the inferences they can carry out directly.

2.2 Basic concepts about PGMs

Bayesian networks and influence diagrams are ideally suited knowledge representations for many situations under uncertainty. These models are often characterized as normative expert systems as they provide model-based domain descriptions, where the model reflects the properties of the problem and probability is used as the calculus for uncertainty. A Bayesian network can be used as the basis for performing inference about the domain. Decision options and utilities associated with these options can be incorporated explicitly into the model, in which case the model becomes an influence diagram, capable of computing expected utilities of all decision options given the information known at the time of decision. Bayesian networks and influence diagrams are applicable to a very large range of domains with uncertainty.

2.2.1 Graphs and probability distributions

The graphical representation of a probabilistic network describes knowledge of a problem domain in a precise manner. The graphical representation is intuitive and easy to comprehend, making it an ideal tool for communication of domain knowledge between experts, users, and systems. Probabilistic graphical models use a graph representation as the basis for encoding a complex distribution over a high-dimensional space.

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 *cycle* is a path in which $X_0 = X_n$, 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 $Pa(X)$. 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)).

We briefly define some necessary terminology for describing trees (see [1] for further details):

1. A *directed (or rooted) tree* is a directed acyclic graph satisfying the following properties:
   (a) There is exactly one node, called the *root*, which no edges enter. The root node contains all the class labels.
   (b) Every node except the root has exactly one entering edge.
   (c) There is a unique path from the root to each node.
2. A node with no proper descendant is called a *leaf (or a terminal)*. All other nodes (except the root) are called *internal nodes*.
3. The *depth of a node* v in a tree is the length of the path from the root to v.

Finally, we introduce some basic definitions about probability theory:

A *random discrete variable* is a variable that is subject to variations due to random chance. When we have a set of variables $\{X_1, ..., X_n\}$, we represent it by $\mathbf{X}$. The set $\mathbf{x} = (x_1, ..., x_n)$ represents the configuration of $\mathbf{X}$ in which each variable X_i takes its corresponding value x_i.

A *discrete probability distribution* is a mapping of all the possible values of a random discrete variable to their corresponding probabilities for a given sample space. It is denoted as:

$$P\left(X = x\right) \tag{2.1}$$

Given a set of discrete variables $\mathbf{X} = \{X_1, ..., X_n\}$, we define the *joint probability* as the probability that all occur simultaneously:

$$\sum_{\mathbf{x}} P\left(\mathbf{x}\right) = \sum_{x_1} ... \sum_{x_n} P\left(x_1, ..., x_n\right) = 1 \tag{2.2}$$

2.2.2 Bayesian networks

A *Bayesian network* (BN) [33] $B = (G, P)$ 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:

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

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

The quantitative information of a BN $B = (G, R)$ 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* $\phi = P(X|Pa(X))$ is $dom(\phi) = \{X\} \cup Pa(X)$.

We assume that each variable X in 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 $dom(X) = (x_1, x_2, ..., x_l)$.

Figure 2.1 shows the graph of a BN. The nodes of the network represent the variables and the arcs of the network represent the properties of (conditional) dependences and independences among the variables as dictated by the distribution.

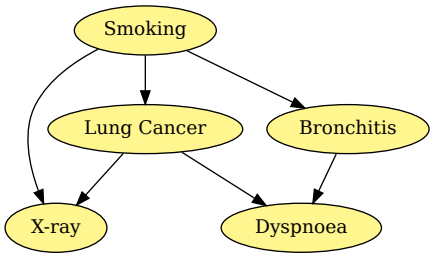

Figure 2.1: The graph of a Bayesian network.

2.2.3 Influence diagrams

An *influence diagram* (ID) is basically a BN augmented with decision nodes and value nodes. Thus, an ID 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* $\mathbf{V}_U$.

Figure 2.2 shows the graph of ID. *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 control of the decision maker. *Utility nodes* (drawn as diamonds) represent the preferences of the decision maker.

Tatman and Schacher (1990) [43] proposed an extended framework of IDs with super value nodes (SVNs). They distinguished two types of utility nodes: *ordinary utility nodes*, whose parents are decision and/or chance nodes, and *super value nodes*, 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.

There are three types of arcs in an ID depending on the type of node they go into: arcs into decision nodes, named *informational arcs*, represent availability of information; arcs into utility nodes represent functional dependency and arcs into super value nodes (whose parents are utility nodes) indicate that

the associated utility function is a combination of the utility functions of the parents.

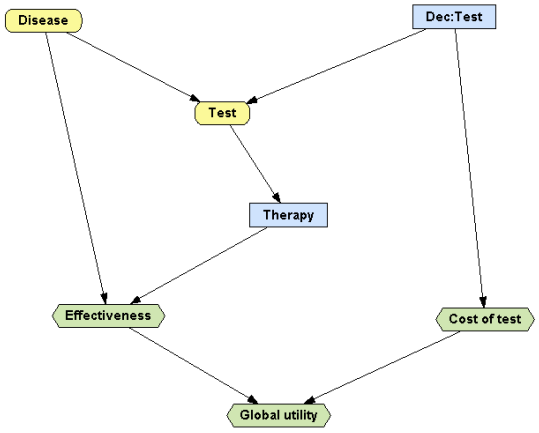

Figure 2.2: An influence diagram involving two decisions, *Dec: Test* and *Therapy*.

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 they 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 $\mathbf{V}_C \cup \mathbf{V}_D$:

$$\mathbf{C}_0 \prec D_0 \prec \mathbf{C}_1 \prec ... \prec D_n \prec \mathbf{C}_n \ . \tag{2.4}$$

An influence diagram is capable of computing the expected utilities of various decision options given the information known at the time of the decision. The values that are known making a decision D_j are known as its *informational predecessors* and is denoted $iPred(D_j)$.

A *policy* for a decision is a family of probability distributions for the options of the decision, such that there is one distribution for each configuration of its informational predecessors, $P_D(d|iPred(D))$. We will focus on policies in which every probability is either 0 or 1, called a *deterministic policy* (see Figure 2.3).

A *strategy* is a set of policies, one for each decision in the influence diagram. Each strategy has an *expected utility*, which depends on the probabilities and utilities that define the influence diagram and on the policies that constitute the strategy. A strategy that maximizes the expected utility is said to be *optimal*. A policy is said to be optimal if it makes part of an optimal strategy.

Finally, the *resolution or evaluation* of an influence diagram consists of finding an optimal strategy and its expected utility, which is the *maximum expected*

utility. Thus, the output of the resolution of an ID are the policy tables for each decision node.

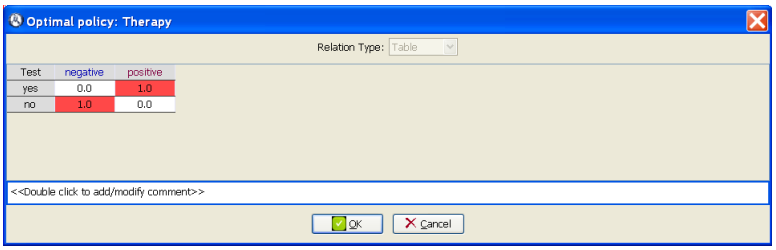

Figure 2.3: Policy table for the decision Therapy.

In real problems, optimal decision tables are very large and virtually impossible to use in practice. In a symmetric decision problem with n variables, where each variable has m possible values, there are m^n scenarios. Given that the sizes of the tables are exponential in the number of informational predecessors, the task of finding explanations is very complex from a computational point of view.

2.2.4 Compact representations of the optimal strategy for an ID

Given that the size of the tables that result from the evaluation of IDs grows exponentially with the number of variables, the search for explanations is not an easy task from a purely computational viewpoint. Several knowledge extraction techniques proposed in the literature might be used for this purpose, such as those that are used to construct tree-based classifiers [10], oblivious read-once decision graphs [19], KBM2L lists [12], and to identify which nodes are relevant for each decision node in an influence diagram [44, 22]. In this project we focus on building decision trees [29] because it is one of the most widely used method for inductive inference. Decision trees have been successfully used in expert systems for capturing knowledge and furthermore they are intuitive and easy to understand for the expert users.

2.3 Classification Trees

2.3.1 Comparison between different types of trees

In order to avoid confusion, we should mention that in this master thesis we refer to three types of trees, which differ in their purpose, their content, and the way in which they are built.

Classification trees: are used to classify instances described by a set of attributes [38, 35]. Each inner node in the tree represent attributes; each branch represents a possible value of the attribute. Leaf nodes represent classes. Classification trees are built from sets of classified instances, usually in the form of

a database, using an automatic learning algorithm, such as ID3 or C4.5. Once they are built, they are used to assign a class to each new instance. Figure 2.4 shows an example of classification tree.

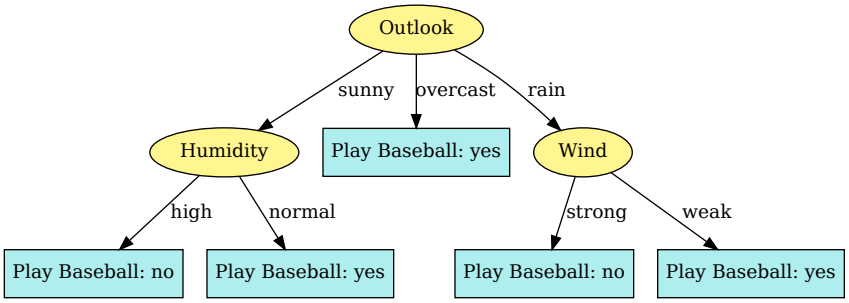

Figure 2.4: Classification tree. The attributes are: Outlook, Humidity, and Wind; and the target attribute is Play Baseball. We want to decide whether the weather is amenable to playing baseball.

Decision trees: are probabilistic models for decision analysis. Inner nodes represent decisions and chance variables. The arcs outgoing from a chance node represent all the possible outcomes of the associated variable, with the corresponding conditional probabilities. The arcs outgoing from a decision node represent the options available for that decision. Utility nodes represent the decision maker's values and preferences. Decision trees can be built manually, i.e., by explicitly encoding its nodes and branches, or generated from a probabilistic graphical model, such as an influence diagram [15]. The evaluation of the decision tree returns the optimal strategy. Figure 2.5 shows an example of decision tree.

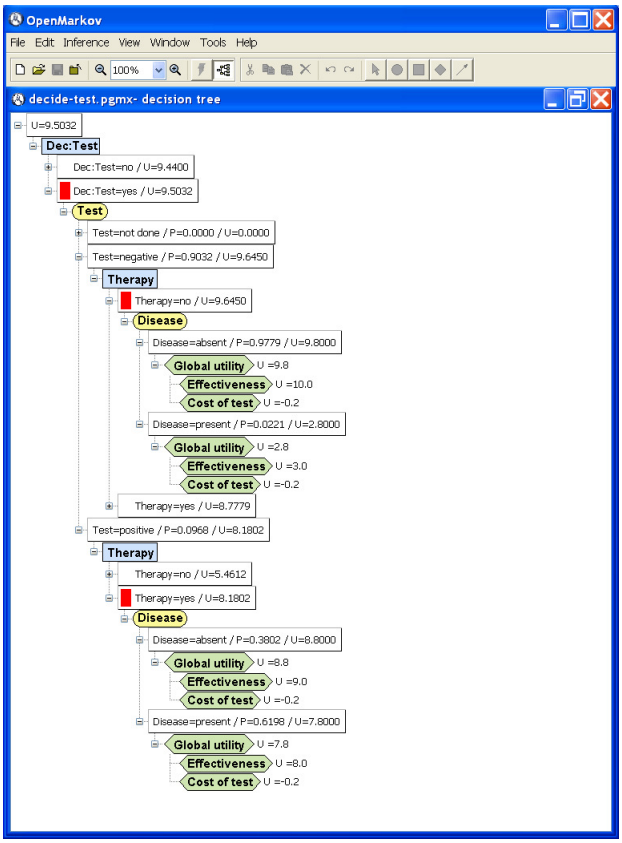

Figure 2.5: Decision tree equivalent to the ID in Figure 2.2. Each path from the root node to a leaf node represents a *scenario*. A red box inside a node denotes the optimal choice in the corresponding scenario. In this figure only the optimal paths are expanded, because we have collapsed the branches that are suboptimal or have null probability.

Policy trees: represent strategies—in general the optimal strategy—for decision models [26]. Their nodes represent decisions and chance variables. Every chance node has several outgoing arcs that represent all the possible outcomes of the associated variable. Every inner decision node has one outgoing arc that represents the optimal option in that scenario; exceptionally, when there is a tie, i.e., when several options have the same expected utility, a decision node may have several outgoing branches. Every leaf node represents the last decision made in a scenario. One way of building a policy tree is to build a decision tree, evaluate it, prune suboptimal branches, and remove the numerical information, namely, the probabilities and utilities. Another way, proposed in [26] and used in this thesis, is to build it from the policy tables resulting from the evaluation of a probabilistic graphical models, such as an influence diagram. Figure 2.6

shows an example of policy tree.

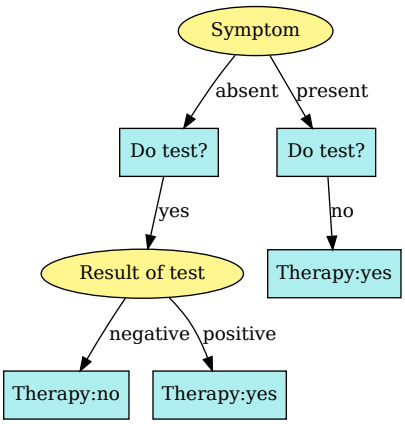

Figure 2.6: Policy Tree. The chance nodes are drawn as yellow circles and decision nodes as blue rectangles.

In the following section we describe in detail classification trees, and the algorithms for building them. In chapter 3 we show how to build and optimize policy trees.

2.3.2 Algorithms for learning Classification Trees

In this section we analyze different classification tree algorithms and show how they differ from each other.

In order to appreciate the complexity of the problem, consider the example by Quinlan [37], in which there is a database with two classes and four discrete attributes, two with two possible values each and two with three. That database supports over 2.2×10^{14} sensible trees. Even for such a simple database an exhaustive examination of all of these trees is impractical. In addition a related problem of trying to find a classification tree with the smallest total external path length consistent with the training database is NP-complete [16]. In light of this evidence there is a strong argument for the application of heuristic algorithms to this problem. Most algorithms for learning classification trees are variations of a core algorithm that employs top-down, greedy search through the space of possible classification trees: the ID3 algorithm.

ID3 (Iterative Dichotomiser 3) was developed in 1986 by Ross Quinlan [35]. It creates a tree, finding for each node (i.e. in a greedy manner) the categorical attribute that yields the largest information gain for categorical targets. It is based on *entropy*. One limitation of ID3 is that it is overly sensitive to attributes with large numbers of values.

C4.5 [36] is an extension of the ID3 algorithm used to overcome its disadvantages; it is able to:

- handle training data with missing values of attributes,
- prune the classification tree after its creation, and
- handle attributes with discrete and continuous values.

C5.0 is Quinlan's latest version of ID3, released under a proprietary license. It uses less memory and builds smaller rule sets than C4.5, while being more accurate.

CART (Classification and Regression Trees) [5] is very similar to C4.5, but supports numerical target variables (*regression trees*) and does not compute rule sets. This software uses a non-parametric decision tree learning technique that produces either classification or regression trees, depending on whether the dependent variable is categorical or numeric, respectively. The CART algorithm looks for the best split by making use of a brute-force (enumerative) procedure. All the possible splits for all the possible variables are generated and evaluated. Such a procedure must be performed everytime a node has to be split and can lead to computational intractability.

SLIQ [27], **FAST** [31] and **GUIDE** [24] also use a greedy, top-down recursive partitioning approach. They primarily differ in the splitting criteria and the ways to avoid overfitting.

Evolutionary algorithms (EAs) are inspired on biological evolution [14, 29]. They use random search to solve optimization problems. They navigate the state-space and find near-optimal solutions. It has been shown that training classifiers with EAs takes longer than with traditional algorithms. However EAs obtain really good accuracy in case of noisy data. In a review, Kokol and colleagues [20], present multiple examples of evolutionary decision-tree applications in the medical domain.

Wei-Yin Loh [25] introduced classification algorithms and compared their capabilities, strengths, and weaknesses.

In the section 2.4 we explain in detail the ID3 algorithm.

2.3.3 Criteria for evaluation Classification Trees

There are three important criteria for evaluating a classification tree: size, accuracy, and understandability.

- *Size*: One should attempt to minimize the size of the induced tree, as measured by the number of nodes or leaves, in order to save memory. If the tree were too big, it would even be impossible to store it on the working memory.
- *Accuracy*: This measure refers to the predictive ability of a tree in terms of classifying an independent set of test data. One can measure this ability in terms of the error rate, i.e., the proportion of incorrect predictions that

a tree makes on the test data. The basic algorithm can produce a very large tree in an effort to correctly classify every instance.

- *Understandability*: Part of the rationale for expert systems is that they should represent knowledge explicitly so that the expert, and to a certain extent the user, can readily understand it. Certainly this is one advantage of trees over other statistical techniques that perform the same function. Clearly deeper trees are more difficult to understand by humans.

2.3.4 Appropriate problems for Classification Trees

One of the reasons because classification tree is attractive is because the information it contains can easily be examined. On the oter hand, one of the weaknesses of the tree representation is its combinatorial explosion when there are many variables.

Although a variety of classification tree learning methods have been developed with different capabilities and requirements, the classification tree learning algorithm is generally suited when:

- Instances are represented as attribute-value pairs.
- The target function has discrete output values.
- The training data contains errors.
- The training data contains missing attribute values.

Many practical problems have been found to fit these characteristics. For example, classification tree learning has been applied to problems such as classifying medical patients by their disease or symptoms. In general, classification trees are commonly used for decision making.

Currently, there are software packages for building and evaluating probabilistic graphical models which provide representations of classification trees, such as deal [4] and Weka [45]. These tools allow the user to build different models or prototypes and select the most suitable one.

2.4 The ID3 algorithm

ID3 begins with the question "which attribute should be tested at the root of the tree?" To answer this question, each instance attribute is evaluated using statistical test to determine how well it alone classifies the training examples. The best attribute is selected and used at the root node of the tree. A descendant of the root node is then created for each possible values of this attribute, and the training examples are sorted to the appropriated descendant node. The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree. This forms a greedy search for an acceptable classification tree, in which the algorithm never backtracks to reconsider earlier choices.

The main ideas behind the ID3 algorithm are:

- Each non-leaf node at the three corresponds to an input attribute, and each arc to a possible value of that attribute. A leaf node corresponds to the expected value of the output attribute when the input attributes are described by the path from the root node to that leaf node.
- *Entropy* is used to determine how informative a particular input attribute is about the output attribute for a subset of the training data. Entropy is a measure of uncertainty in communication systems introduced by Shannon (1948)[41]. It is fundamental in modern information theory.

We summarize the ID3 algorithm (Algorithm 2.1). This algorithm grows the tree top-down until the tree perfectly classifies the training instances or until all attributes have been used.

Algorithm 2.1 ID3 (*Instances, Target_ attribute, Attributes*)

Instances *are the training examples,* ***Target_ attribute*** *is the attribute whose value is to be classified by the tree.* ***Attributes*** *is a list of the input attributes. It returns a classification tree that correctly classifies the given instances.*

1: Create a *Root* node for the tree
2: **if** all *Instances* have the same *value* in *Target_ attribute* **then**
3: **return** the single-node tree *Root*, with label = *value*
4: **end if**
5: **if** *Attributes* is empty **then**
6: **return** the single-node tree *Root*, with label = most common value of *Target_ attribute* in *Instances.*
7: **end if**
8: Otherwise Begin:
9: Choose the attribute A from *Attributes* that best (highest *information gain*) classifies *Instances.*
10: The attribute of *Root* is A.
11: **for** each possible value v_i of A **do**
12: Add a new tree branch below *Root*, corresponding to the test $A = v_i$. Let $Instances_{vi}$ be the subset of *Instances* that have value v_i for A.
13: **if** $Instances_{vi}$ is empty **then**
14: below this new branch add a leaf node with label = most common value of *Target_ attribute* in *Instances.*
15: **else**
16: below this new branch add the subtree: ID3($Instances_{vi}$, *Target_ attribute, Attributes-* $\{A\}$)
17: **end if**
18: **end for**

2.4.1 Which Attribute is the Best Classifier?

We want to select the attribute that is most useful for classifying examples at each node in the tree. For that, ID3 define a statistical property called *information gain*, that measures how well a given attribute separates the training examples according to their target classification.

2.4.1.1 Entropy

In order to define information gain, we begin by defining a measure called *entropy*. In information theory, entropy is a measure of the uncertainty about a source of messages. The more uncertain a receiver is about a source of messages, the more information that receiver will need in order to know what message has been sent.

If the target attribute can take on c different values, the the entropy of a set of instances S relative to this *c-wise* classification is defined as

$$Entropy\,(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{2.5}$$

where p_i is the proportion of S belonging to class i. The logarithm is base 2 because entropy is a measure of the expected encoding length measured in *bits*. Note that the entropy is 0 if all members of S belong to the same class, and the entropy is 1 when the collection contains an equal number of examples of each class. Figure 2.7 shows the form of the entropy function relative to a boolean classification, as p_i varies between 0 and 1.

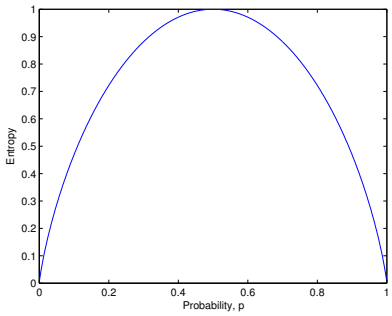

Figure 2.7: The entropy function relative to a boolean classification, where p is the proportion of positive examples in S and varies between 0 and 1.

2.4.1.2 Information Gain

Given entropy as a measure of the impurity in a collection of training examples, the *information gain* is a measure of the effectiveness of an attribute in classifying the training data. It simply measures the expected reduction in entropy caused by partitioning the examples according to this attribute.

The information gain of an attribute A relative to a collection of examples S, is defined as

$$Gain\,(S,A) = Entropy\,(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy\,(S_v) \tag{2.6}$$

where $Values\,(A)$ is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has values v. Note the first term in equation 2.6 is just the entropy of the original collection S, and the second term is the expected value of the entropy after S is partitioned using attribute A. The $Gain\,(S,A)$ is therefore the expected reduction in entropy caused by knowing the value of attribute A.

Information gain is the measure used by ID3 to select the best attribute at each step in growing the tree. ID3 algorithm computes the information gain for each attribute, and selects the one with the highest gain. In case of having attributes with the same information gain, the algorithm selects randomly an attribute.

2.4.1.3 Alternative Measures for Selecting Attributes

There is a natural bias in the information gain measure that favors attributes with many values over those with fewer values. As an extreme example, let us consider that the attribute *Age* takes 15 values, from 15 to 30 years old. If we were to add this attribute to the data in Table 2.8 it would have the highest information gain of any of the attributes because *Age* has so many possible values that is bound to separate the training examples into very small subsets . Because of this, it will have a very high information gain relative to the training examples, despite being a very poor predictor. Thus, it would be selected as the attribute for the root node and lead to a (quite broad) tree of depth.

To avoid this difficulty several alternative measures for selecting attributes have been proposed. One alternative measure that has been used successfully is the gain ratio [35]. This measure penalizes attributes such *Age* by adding a term, called *Split Information*, that is sensitive to how broadly and uniformly the attribute splits the data:

$$SplitInformation\,(S,A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} log_2 \frac{|S_i|}{|S|} \tag{2.7}$$

where S_1 through S_c are the c subsets of examples resulting form partitioning S by the c-valued attribute A. The *Gain Ratio* measure is defined in terms of the earlier *Information Gain* measure, as well as this *Split Information*, as follows

$$GainRatio\,(S,A) = \frac{Gain\,(S,A)}{SplitInformation\,(S,A)} \tag{2.8}$$

Note that the Split Information term discourages the selection of attributes with many uniformly distributed values.

2.4.2 Example Classification Tree

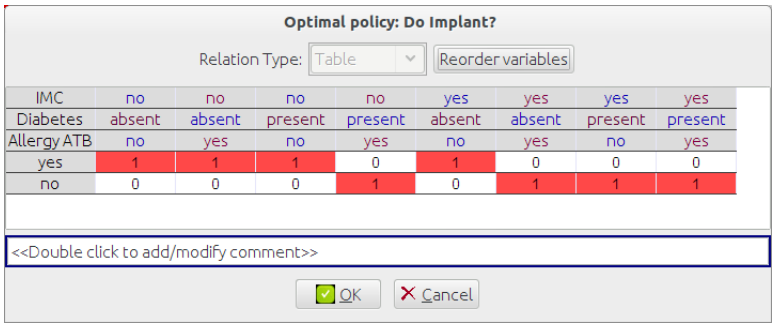

Figure 2.8: Optimal policy table as an example data set.

An example data set will be used to understand how a classification tree is built (Figure 2.8). This policy table comprises two parts: (1) a set of all variable configurations (2) the table content, i.e. the optimal alternative. In this dataset, there are eight training examples and four categorical attributes (the term *attributes* will be used in the following instead of 'variables'): *IMC*, *Diabetes*, *Allergy ATB*, and *Do Implant?*. We are interested in building a system which will enable us to decide whether or not to *Do an Implant* on the basis of the *IMC, Diabetes, and Allergy ATB* factors. From this point on, we will consider the attribute we wish to predict or classify, i.e. *Do Implant?*, as the *output attribute* or *target attribute*, and the other attributes as *input attributes*.

The final classification tree learned by ID3 is shown in Figure 2.9.

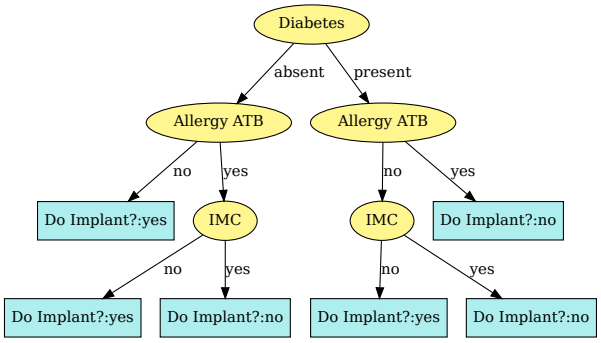

Figure 2.9: A classification tree for the target attribute *Do Implant?*.

According to the *information gain* measure, the *Diabetes* attribute provides the best prediction of the target attribute, *Do Implant?*, over the training examples. Therefore, *Diabetes* is selected as the attribute for the root node, and branches are created below the root for each of its possible values (i.e., absent

and present).

2.4.3 ID3 Capabilities and Limitations

In terms of ID3 search space and search strategy, we can get the next features [29]:

- *ID3 searches complete hypothesis space, relative to the available attributes.* ID3 avoids one of the major risks of methods that search incomplete hypothesis spaces.
- *ID3 maintains only a single current hypothesis as it searches through the space of decision trees.* By determining only a single hypothesis, ID3 loses the capabilities that follow from explicitly representing all consistent hypotheses.
- *ID3 in its pure form performs no backtracking in its search.* Once it selects an attribute to test at a particular level in the tree, it never backtracks to reconsider this choice. Therefore, it is susceptible to the usual risks of hill-climbing search without backtracking: converging to *locally optimal solutions that are not globally optimal.* A possible solution for this problem is post-pruning the tree.
- *ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis.* One advantage of using statistical properties of all the examples (e.g., information gain) is that the resulting search is much less sensitive to errors in individual training examples.
- *ID3 can have difficulties when an input attribute has many possible values*, because the information gain tends to favor attributes which have a large number of values. Quinlan (1986) [35] suggests a solution based on considering the amount of information required to determine the value of an attribute for a set data (see section 2.4.1.3).
- *The computational time required by ID3 grows linearly with the number of training examples and exponentially with the number of attributes.*

2.5 OpenMarkov

OpenMarkov[1] is an open-source software tool for editing and evaluating probabilistic graphical models (PGMs) developed by the Research Centre for Intelligent Decision-Support Systems of the UNED in Madrid, Spain. The development language of OpenMarkov is Java, so that it can run on different platforms.

[1] At `www.openmarkov.org/` it is possible to obtain the source code, executable Java files, technical documents, and additional documentation about OpenMarkov.

OpenMarkov is able to represent several types of networks, such as Bayesian networks (BNs), Markov networks or influence diagrams (IDs). In OpenMarkov a probabilistic network is represented as a generic data structure consisting mainly of a graph, a set of variables, and a set of potentials. Three types of variables can be used: finite-states, numerical, and discretized. There are two types of links: directed and undirected. It also has several types of potentials: uniform, table, delta...

The graphical user interface (GUI) is very similar to those of other software tools for PGMs. It has two main working modes:

- edition - for editing BNs and IDs graphically .
- inference - for finding optimal policies, propagating evidence, and explaining the results.

Chapter 3

Building and Optimizing Policy Trees

The first aim of this chapter is to present the implementation of a new method of representation called Policy Trees (PTs) proposed in [26]. In this work, the PTs have been mathematically formulated by means of the algorithm ID3. Secondly, as a way of optimization we propose two methods to reduce the size of the PTs.

3.1 A new algorithm for building policy trees

A Policy Tree (PT) [26] consists of chance and decision nodes, and arcs labeled with the states of the nodes. The leaves of the PT indicate the optimal decision of the corresponding scenario. PTs only represent scenarios that are possible by following the optimal strategy.

The problem of building a PT given the tables that result from the evaluation of an influence diagram is similar to building a classification tree (CT). Each scenario in the PT, defined by a set of chance and decision variables, is equivalent to an instance defined by a set of attributes in a classification problem. However, there are also important differences.

When viewing the set of scenarios of a decision problem as if they were the instances of a classification problem, we find that they have the following features:

- All the "instances" are correctly classified.
- All the attributes are discrete.
- There are no missing values.
- There is one and only one "instance" for each combination of the values of the "attributes".
- There are as many "classes" as options for the last decision.

- We need a 100% accuracy in the "classification", i.e., we need to build a *perfect tree* [35].

Because of the similarity with CTs, we decided to use the ID3 algorithm to build PTs, but that algorithm had to be adapted to this particular type of "classification". In particular, the requirement 100% accuracy prevents the application of pruning techniques which might lead to suboptimal PTs.

We have used the *Gain Ratio* as the measure to select attributes in order to avoid the natural bias in the *Information Gain*. The result of this adaptation is the Algorithm 3.1. Note the requirement that the decision nodes should appear in the PT in the same order as in the DI.

In the next section we explain how to implement a PT whit an example.

Algorithm 3.1 Policy trees (*ID*)

ID *is the influence diagram. This algorithm returns a Policy Tree (PT).*

1: Identify the order of the decision nodes in the ID.
2: **for** the first decision node **to** the last decision node **do**
3: Read the optimal policy table of the decision node in order to get *Instances*.
4: Remove from *Instances* those cases of suboptimal policies or null probability.
5: **if** it is the first node **then**
6: ID3 (*Instances*, *Target_attribute*, *Attributes*) (algorithm 2.1). It returns the tree structure (PT) corresponding to the decision node.
7: **else** {for the remaining decision nodes}
8: **for** each branch of the PT created in the previous decision node **do**
9: Filter *Instances* in order to get *Filtered Instances**.
10: ID3 (*Filtered Instances*, *Target_attribute*, *Attributes*) (algorithm 2.1). It returns the subtree structure corresponding to the decision node.
11: Add at the end of the corresponding branch of the PT the new subtree.
12: **end for**
13: **end if**
14: **end for**
15: **return** PT

* *Filtered instances* are obtained as a result of filtering the optimal policy table (*Instances*) for those values of the attributes that appeared in the corresponding branch.

3.2 Example Policy Tree

As an example, we explain how to build a PT for the influence diagram shown in Figure 3.1 which has two decisions: *Do test?* and *Therapy*. The decision *Do test?* must be made first because there is a directed path from it to *Therapy*. The informational predecessors of *Do test?* are *Sex* and *Symptom*. Those of *Therapy* are *Sex*, *Symptom*, *Do Test?*, and *Result of Test*.

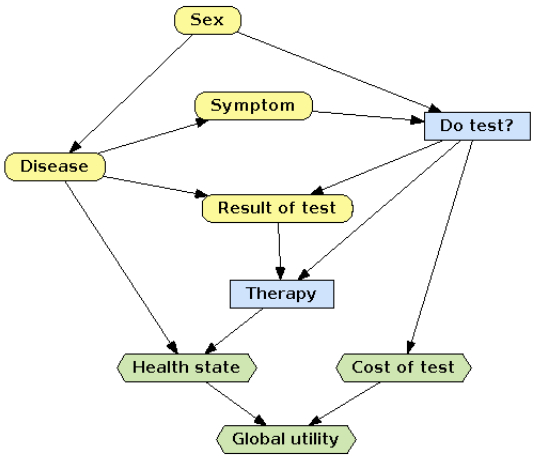

Figure 3.1: Decide-Test influence diagram. Chance nodes: *Sex*, *Symptom*, *Disease* and *Result of test*; Decision nodes: *Do test?* and *Therapy*.

The evaluation of this ID returns two policy tables: Table 3.1 for *Do test?* and Table 3.2 for *Therapy*. These tables contain a row for each configuration of its informational predecessors. The last column in each table displays the optimal decision for each scenario. The policy table for *Therapy* originally contained 24 rows, which is the size of the state space of its informational predecessors, but we have omitted the rows corresponding to suboptimal options or null probability.

Sex	Symptom	Do Test?
Female	Absent	yes
Female	Present	no
Male	Absent	yes
Male	Present	no

Table 3.1: Policy table for *Do Test?*.

Sex	Symptom	Do Test?	Result of Test	Therapy
Female	Absent	no	not-performed	no
Female	Absent	yes	negative	no
Female	Absent	yes	positive	yes
Female	Present	no	not-performed	yes
Female	Present	yes	negative	yes
Female	Present	yes	positive	yes
Male	Absent	no	not-performed	no
Male	Absent	yes	negative	no
Male	Absent	yes	positive	yes
Male	Present	no	not-performed	yes
Male	Present	yes	negative	yes
Male	Present	yes	positive	yes

Table 3.2: Policy table for *Therapy* (short version).

For the construction of the PT (Figure 3.2), first of all our algorithm calculates the subtree for the first decision node *Do Test?* following the ID3 algorithm and considering its predecessors *Sex* and *Symptom* (see Figure 3.2 (A)). Later, for each branch new subtrees for *Therapy* are calculated (see Figure 3.2 (B) and (C)). The subtrees for *Therapy* are calculated following the ID3 algorithm considering its predecessors except of *Symptom* and *Do test?* because these attributes have been already used. The order in which the nodes are traversed from top to down is the chronological order in which decisions are made and/or outcomes of chance events are revealed to the decision-maker. Thus, the decision tree gives a chronological and fully detailed view of the structure of the decision problem. Note that, in this example the attribute *Sex* doesn't have any influence over *Do test?* neither *Therapy,* which means that ID3 will not consider irrelevant variables.

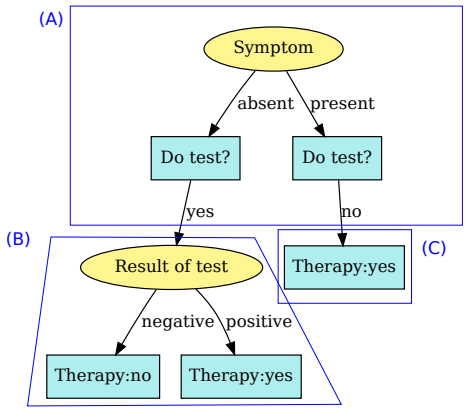

Figure 3.2: Decide-Test policy tree. The blue rectangles indicate subtrees for each decision node, (A) for *Do Test?*, (B) and (C) for *Therapy.*

Furthermore, our algorithm effectively ignores the impossible information states for the decision and chance variables (i.e., *Do Test?* = *yes*, *Result of test* = *not-performed*).

3.3 Policy Tree optimization

In order to reduce the size of the PTs, two methods (Policy Tree restructuring and Policy Tree coalescing) have been implemented in this section. We are assuming here that the following proposed methods are implemented during the representation phase after the evaluation of an influence diagram and given a policy tree.

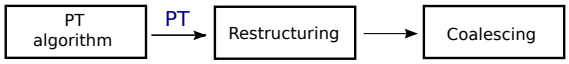

Figure 3.3: Block Diagram.

3.3.1 Policy Tree restructuring

Given a policy tree, we realized that a decision node might have the same value (same optimal policy) in all its appearances in the PT. An example is showed in the section 4.2.4. In this example, the PT recommends to perform the PET and never to perform the MED, in every branch (see Figure 4.11).

We propose a *restructuring method* to simplify these kind of trees, moving those decision nodes to the root of the tree, reducing the number of nodes. In the previous example, the restructuring consists on moving the PET and MED decision nodes to the root of the tree (see Figure 4.12). Note that, in these new PT representation the decision nodes don't appear in the *partial order*.

3.3.2 Policy Tree coalescing

The policy tree solution can be made more efficient by identifying *coalescence* [32]. Coalescence can be a useful mechanism for depicting more compact and convenient representations of decision trees. The use of coalescence has often been ad-hoc for decision trees, and it has been difficult to automate [3]. In Figure 3.4, the decision tree shows two sub-trees that are repeated once (marked in blue lines) as well as two repeated leaves (marked in orange lines). We can exploit this repetition using coalescence.

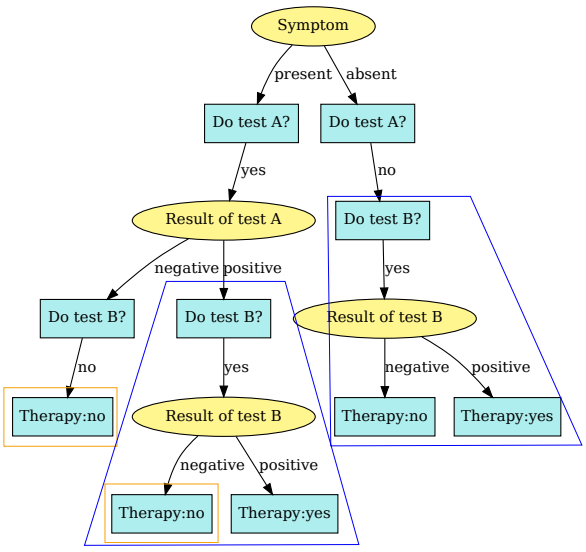

Figure 3.4: Policy tree example. Marked in blue and orange lines those nodes and branches that can be merged by coalescence.

The proposed algorithm for identifying coalescence is described in Algorithm 3.2. The algorithm recursively analyzes branches that are placed "near" in order to find coalescence. Given two close branches, from the leaf nodes and proceeding toward the root node, the algorithm merges only those nodes with the same values and same children. In this project a *coalescence index (ci)* was defined as the maximum distance between two branches that can be merged. For example, if *ci* is equal to *1* (*first order neighbor)*, it means that only adjacent branches can be merged in case of coalescence. In the same way if *ci* is equal to *2* (*second order neighbor*), it means that in addition to those branches merged by *ci*<*2*, if two branches are separated only by another one, they also can be merged and so on. This way with *ci*=*n*, branches which are *m-order neighbors* (being *m*<=*n*) are merged. The maximum value of coalescence index allowed by this algorithm is equal to the half of the number of branches of the Policy Tree.

$$ci = \left\{1, 2, ..., \frac{number\,of\,branches\,of\,PT}{2}\right\} \tag{3.1}$$

Algorithm 3.2 Coalescence (*Policy Tree, ci*)

Coalescence is done during the representation phase given a ***Policy Tree*** *(PT) and* ***ci*** *is the coalescence index.*

1: **for** each *index = {1, 2, 3, ..., ci}* **do**
2: **for** each *branch(j)* **do**
3: Comparison between *branch(j)* and *branch(j+index)*. Identification of repetitive nodes between consecutive nodes in both branches, from the leaf nodes toward the root node.
4: **end for**
5: **end for**
6: **for** each node of the PT **do**
7: **if** the $node_j$ is within a *branch(j)* and has been identified equal to another $node_k$ within a *branch(k)* **then**
8: **if** (children of $node_j$ = children of the $node_k$) **then**
9: Both nodes will merge.
10: **end if**
11: **end if**
12: **end for**

Figure 3.5 shows the simplified decision tree of Figure 3.4 after applying the coalescence method with a coalescence index equal to 2. Note that, coalescing Figure 3.4 to Figure 3.5 yields a saving in number of branches and nodes. The resulted tree is not longer a *rooted tree*, but it is still a *directed acyclic graph*.

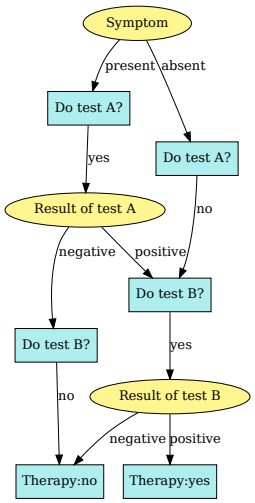

Figure 3.5: Policy tree example after coalescing ($ci = 2$).

Chapter 4

Evaluation of the new algorithm

In this chapter there are a total of five tryouts in order to show how our algorithm works with different IDs.

4.1 Generation of IDs

For testing the proposed algorithm we have created three IDs (Example 1, Example 2, and Example 3) using a method that creates random IDs. Furthermore we have used two more IDs which reflect realistic clinical models:

- Example 4: this example is a decision support-system called MEDIASTINET for the mediastinal staging of non-small cell lung cancer. ML Gallego in his thesis [26] describes how to built this graph, the variables of the problem, their domains, and their relations between them.
- Example 5: this example is a decision support-system called ArthroNET. It is an influence diagram with super-value nodes that represents the knee arthroplasty process and diagnosis of preoperative prosthesis infection. Diego León Guerra in his master thesis [23] describes how to built this graph.

4.2 Experiments performed

The evaluation of every ID yielded a decision table for every decision variable of the ID. Each one contain the optimal decision for each combination of attributes in the tables. For each experiment performed we show the Policy Tree corresponding to the policy table of the last decision node.

Due to the *understandability* of a tree is difficult to quantify or measure, in the present study, the number of nodes and leaves have been selected as

the measure of the tree sizes. For each PT we record the *size* of the tree and *computing time*. The features of the used laptop are: Intel® Core™ i5-3210M CPU (2.60GHz 1600MHz 3MB), Memory 8GB PC3-12800 DDR3 SDRAM 1600 MHz.

Policy Trees will be sketched by Graphviz [13]. Graphviz is open source graph visualization software. In this tree representation, chance nodes will be drawn as yellow circles and decision nodes as blue rectangles.

About the coalescence index, as a rule we choose a value equal to the half of the number of branches of the PT, due to it is the maximum value allowed in the proposed algorithm.

4.2.1 Example 1

The influence diagram in this example is formed by: two chance nodes (*Disease* and *Result of test*), two decision nodes (by order: *Do test?* and *Therapy*) and two Utility nodes (*Health state* and *Cost of test*). The Figure 4.1 shows the influence diagram after computing the posterior probability of each chance and decision node and the expected utility of each utility node. The global maximum expected utility of the influence diagram is 9.43 (value of the *super-value node*).

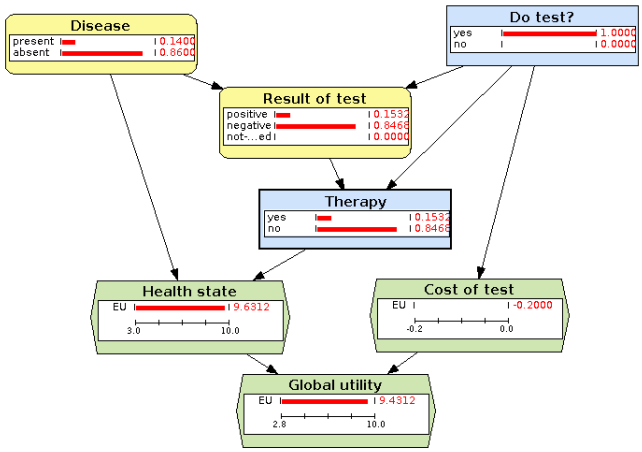

Figure 4.1: Example 1 - Evaluation of the influence diagram.

Observe that on one hand, the optimal policy for the decision node *Do Test?* is to always do the test. On the other hand, the *Therapy* should be applied when the test is done and gives a positive result, and should not be applied when the test is done and gives a negative result (see Figure 4.2).

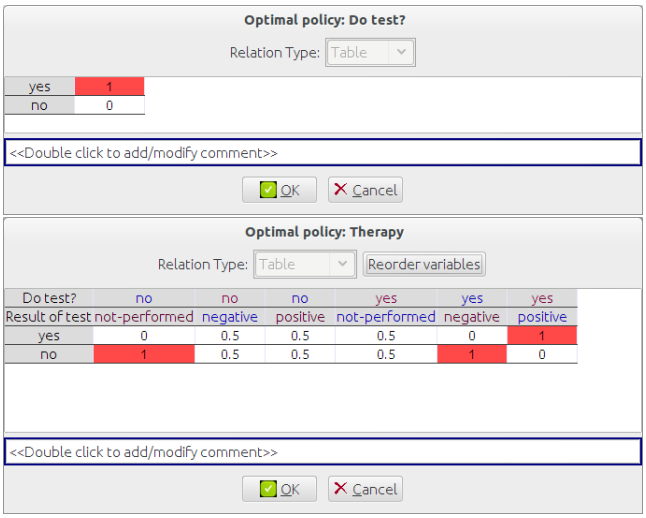

Figure 4.2: Example 1 - Policy Tables for *Do Test?* and *Therapy*.

Finally, after evaluating this ID with our algorithm, we obtain the PT that it is shown in Figure 4.3. Note that, as the optimal policy for the decision node *Do Test?* is to always do the test, the probable scenario (but not optimal): *Do Test?* = *no* and *Therapy* = *no*, is not represented in the PT.

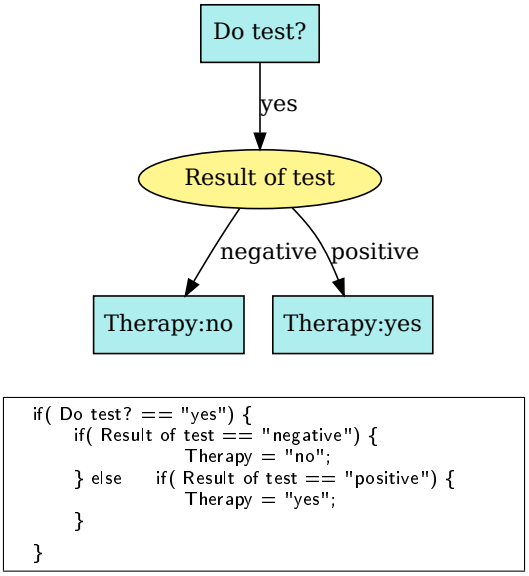

Figure 4.3: Example 1 - Optimal strategy. Policy Tree and inference rules.

4.2.2 Example 2

The influence diagram in this example is formed by: four chance nodes (*GenA, GenB, Enfermedad_Cancer,* and *Resultado_Prueba*), two decision nodes (by order: *Prueba* and *Quimioterapia*) and two Utility nodes (*Vida_en_salud* and *Coste_prueba*). The Figure 4.4 shows the influence diagram after computing the posterior probability of each chance and decision node and the expected utility of each utility node. The policy table for the decision *Quimioterapia* has 24 columns, which is the size of the state space of the *Quimioterapia* informational predecessors (*GenA, GenB, Resultado_Prueba*, and *Prueba*).

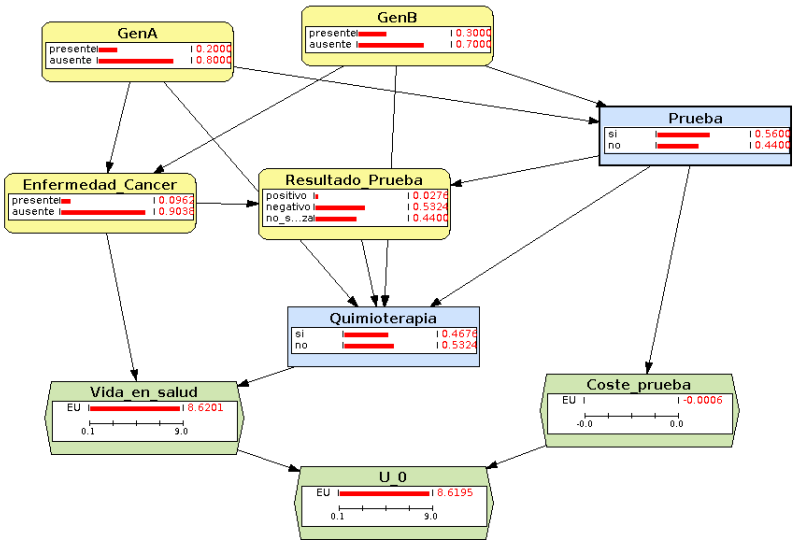

Figure 4.4: Example 2 - Evaluation of the influence diagram.

The policy tree is shown in Figure 4.5. In contrast to the 24 columns of the *Quimioterapia* policy table, this PT has 10 nodes of which 4 are leaves.

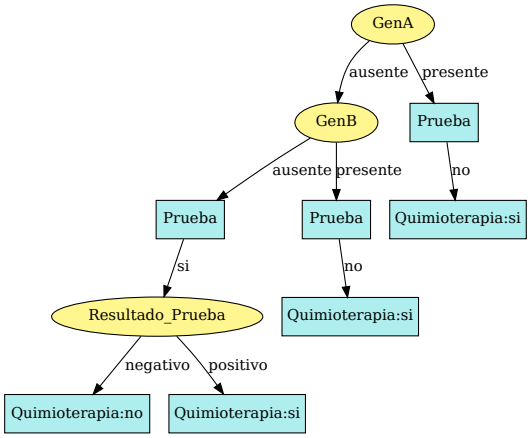

Figure 4.5: Example 2 - Optimal strategy. Policy Tree.

Finally, we have applied the *coalescing method* to the previous PT in order to reduce the size of the tree, Figure 4.6. In this example, using a coalescence index (ci) equal to 2, the tree has reduced its size to 7 nodes of which 2 are leaf nodes. As was mentioned before, the resulting tree is not a *rooted tree* any longer, but it is still a *directed acyclic graph*.

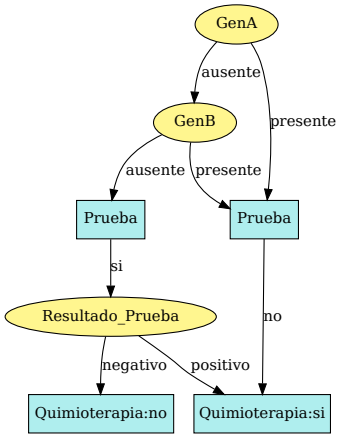

Figure 4.6: Example 2 - Policy Tree after coalescing ($ci = 2$).

4.2.3 Example 3

The influence diagram in this example is formed by: eight chance nodes (*X1, X2, X3, X4, X5, X6, X7,* and *X8*) and three decision nodes (by order: *D1, D2,* and *D3*). The Figure 4.7 shows the influence diagram after computing the posterior probability of each chance and decision node and the expected utility of the utility node *U0*. The policy table for decision *D3* has 1,152 columns, which is the size of the state space of the informational predecessors of *D3* (*X2, X3, X4, X5, X6, X7, X8, D1,* and *D2*).

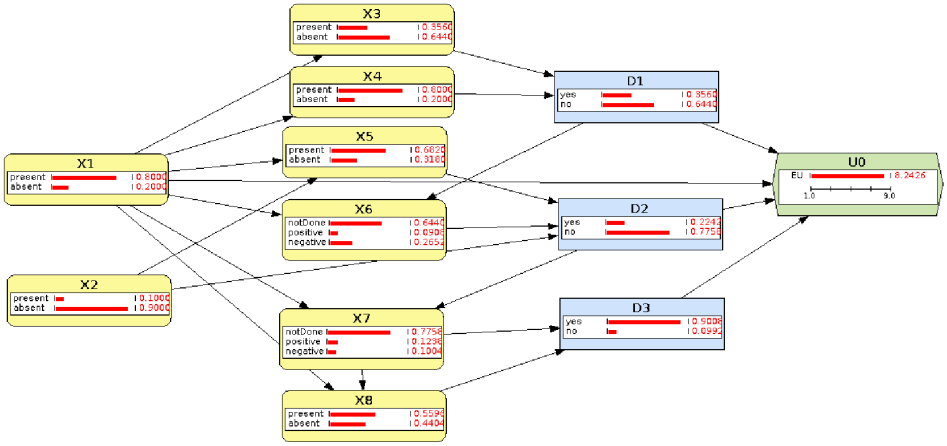

Figure 4.7: Example 3 - Evaluation of the influence diagram.

The policy tree in Figure 4.8 shows the whole optimal strategy. The *X3* variable is located at the root of the PT because this variable classifies better the training data of the decision *D1*. Notice that *X4* is irrelevant for the decision *D1* as well as *X6* is irrelevant for the decision *D2*. This PT has 36 nodes of which 15 are leaves, thus it is easier to interpret in contrast to its policy table.

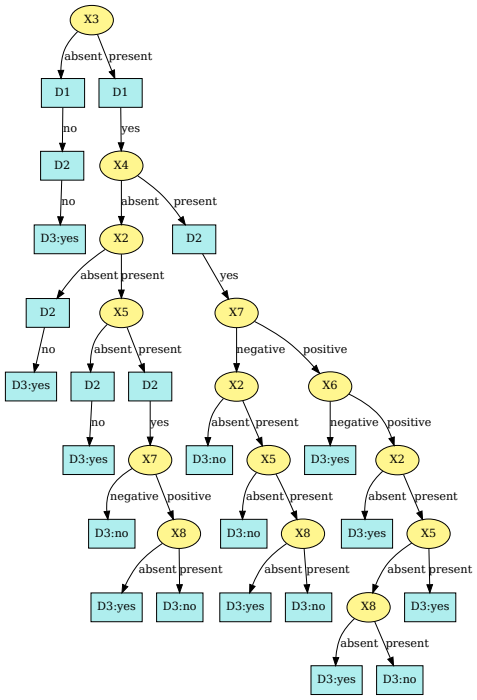

Figure 4.8: Example 3 - Optimal strategy. Policy Tree.

The PT after coalescing using a coalescence index (ci) equal to 7, is depicted on Figure 4.9. This tree has reduced its size to 19 nodes of which 2 are leaf nodes. Thus by coalescing, the PT has reduced its number of nodes to a 52%. This reduction in number of nodes is at expense of having crossing paths, that make harder the tree interpretation. However, this new PT shows clearly the behavior of the *X8* node in the optimal strategy: when the value of *X8* is *absent* *D3* takes the state *yes*, and when the value of *X8* is *present* *D3* takes the state *no*.

Finally, the running time for building the PT after applying the coalescence method was only 3 seconds.

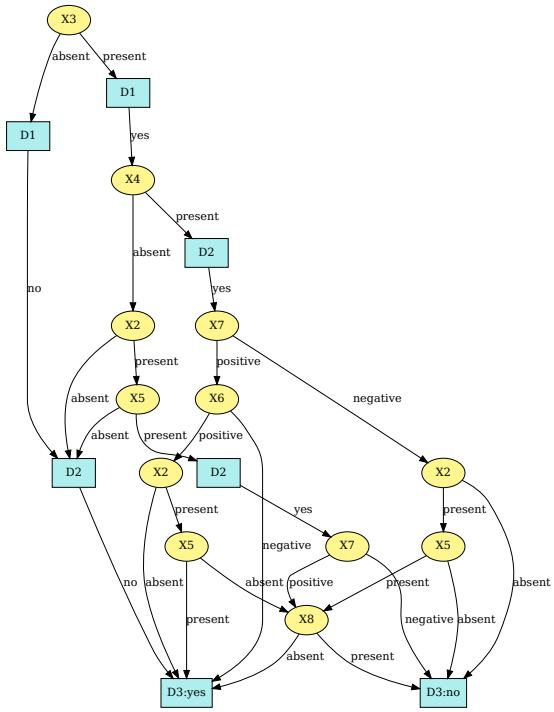

Figure 4.9: Example 3 - Policy Tree after coalescing ($ci = 7$).

4.2.4 Example 4

Figure 4.10 shows a decision support-system for the mediastinal staging of non-small cell lung cancer. The influence diagram in this example is formed by: eight chance nodes (*N2_N3, CT_scan, TBNA, PET, EBUS, EUS, MED,* and *MED_Sv*) and five decision nodes (by order: *Decision_TBNA, Decision PET, Decision_EBUS_EUS, Decision_MED,* and *Treatment*). The policy table for the *Treatment* decision has 15,552 columns, which is the size of the state space of the informational predecessors of *Treatment*.

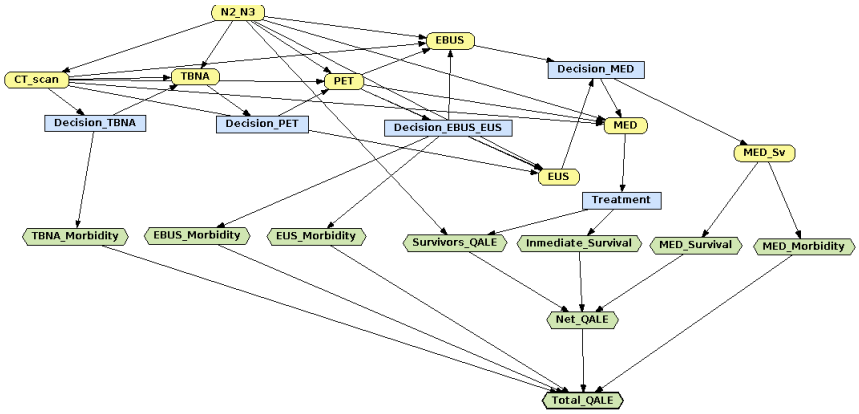

Figure 4.10: Example 4 - Influence Diagram (MEDIASTINET).

The policy tree in Figure 4.11 shows the whole optimal strategy. This PT has 33 nodes of which 9 are leaves and it is easier to interpret in contrast to the 15,552 columns of its policy table. Note that the PT recommends to perform the PET and never to perform the MED, in every branch. Thus, after applying the *restructuring method*, the PT has reduced the number of nodes to 27 (see Figure 4.12). In this new tree, the decision nodes don't appear in the *partial order* although, as we have explained before, the PT has been calculated using this order.

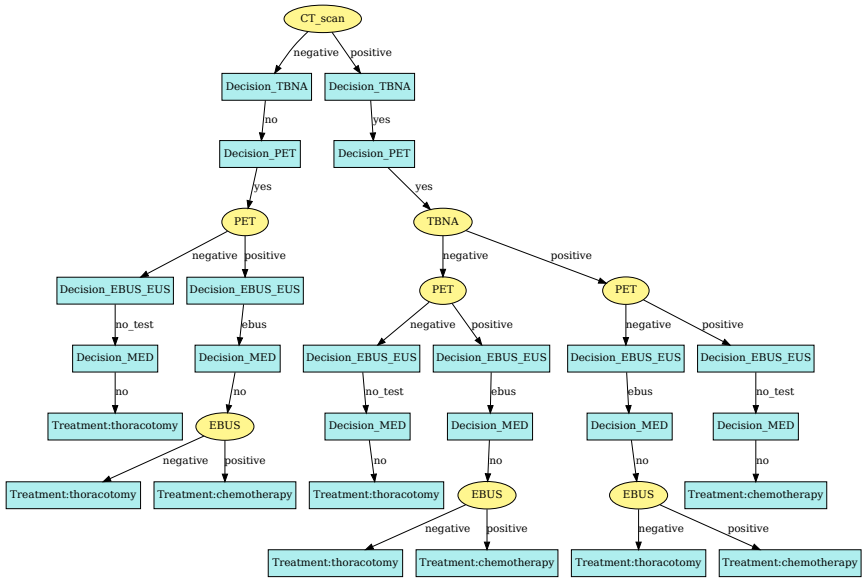

Figure 4.11: Example 4 - Optimal strategy. Policy Tree.

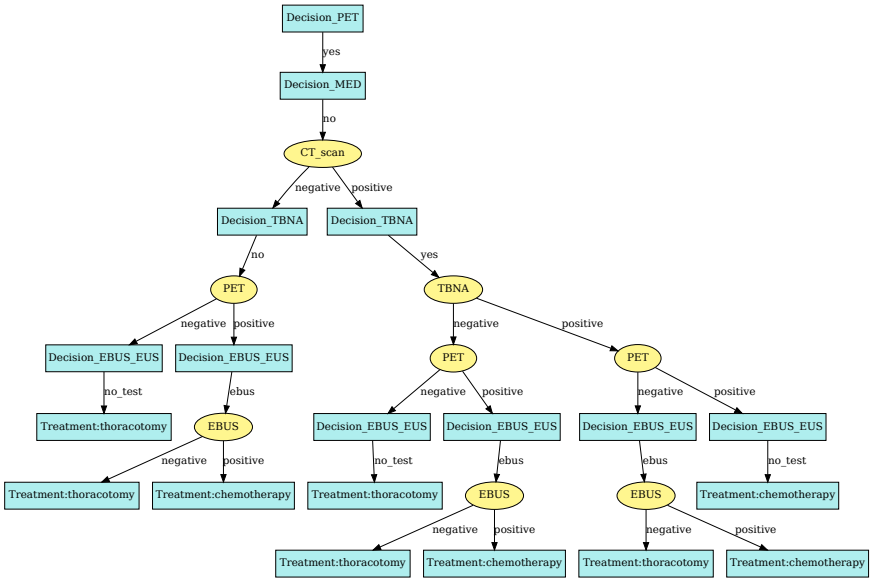

Figure 4.12: Example 4 - Policy Tree after restructuring.

Finally, we have applied the *coalescing method* to the previous PT in order to reduce the size of the tree, Figure 4.13. In this example, using a coalescence

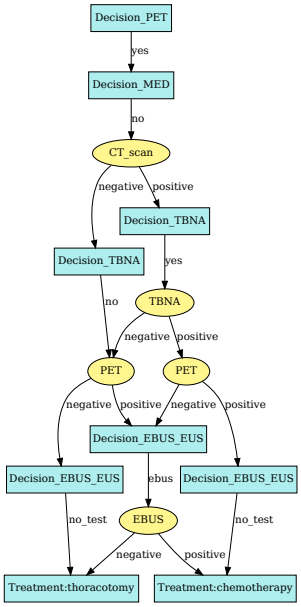

Figure 4.13: Example 4 - Policy Tree after restructuring and coalescing ($ci = 4$).

index (ci) equal to 4, the tree has reduced its size to 14 nodes of which 2 are leaf nodes.

In order to understand the reasons behind the *Treatment*, the PT in Figure 4.13 reveals easily the clinical interpretation. For example, *Scenario 1* (*Decision PET: yes, Decision_MED: no, CT_scan: positive, Decision_TBNA: yes,* ***TBNA: positive***, ***PET: negative***, *Decision_EBUS_EUS: ebus, EBUS: negative,* and *Treatment: thoracotomy*) compared to *Scenario 2* (*Decision PET: yes, Decision_MED: no, CT_scan: positive, Decision_TBNA: yes,* ***TBNA: negative***, ***PET: positive***, *Decision_EBUS_EUS: ebus, EBUS: negative,* and *Treatment: thoracotomy*) reveals that, the suggested optimal strategy is the same either with *TBNA: positive* and *PET: negative,* or *TBNA: negative* and *PET: positive.*

In Table 4.1, we show that after applying the restructuring and coalescing methods, the final PT has reduced its number of nodes to a 42%, resulting a more compact tree. The running time for building the whole PT was only 2 seconds.

	Number of Nodes	Number of leaf Nodes	Time [seconds]
PT	33	9	2
PT restructuring	27	9	2
PT restructuring & coalescing	14	2	2

Table 4.1: Example 4 - Comparison between PT sizes and time computing after applying the restructuring and coalescing methods.

4.2.5 Example 5

The influence diagram ArthroNET (Figure 4.14) is formed by eleven chance nodes (*IMC*, *Diabetes*, *AlergiaATB*, *Isquemia*, *Infección PTR*, *CC_Drenaje*, *PCR*, *VSG*, *Movilidad*, *Ga67 Tc99,* and *Cortes Congelados*) and four decision nodes (by order: *Realizar Implante*, *Realizar Gammagrafías*, *Realizar Biopsia Sinovial,* and *Tratar Infección PTR*).

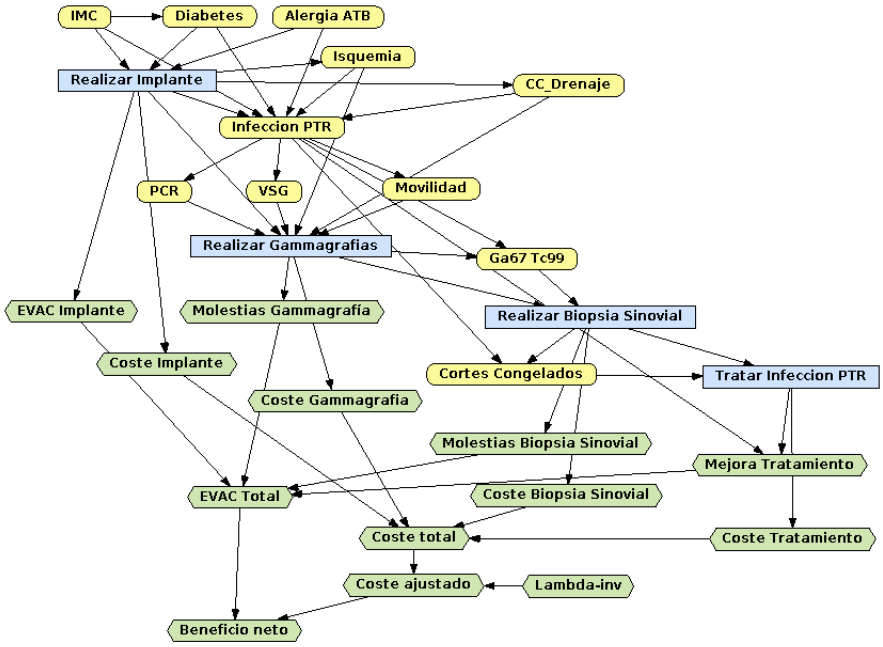

Figure 4.14: Example 5 - Influence Diagram (ArthroNET).

The policy table for decision *Tratar Infección PTR* has 41,472 columns and the final policy tree after evaluating this ID with our algorithm has 673 nodes of which 214 are leaf nodes (the PT figure is not represented due to the huge size of the tree).

In this example first of all we will show the PT after coalescing with a coalescence index equal to 3. After applying the *coalescing method (ci = 3)* the size of the the PT is 264 nodes of which 16 are leaf nodes, reducing the number of nodes to a 41,4%. A section of this final PT is depicted on Figure 4.15. Note that this reduction in number of nodes is at expense of having crossing paths.

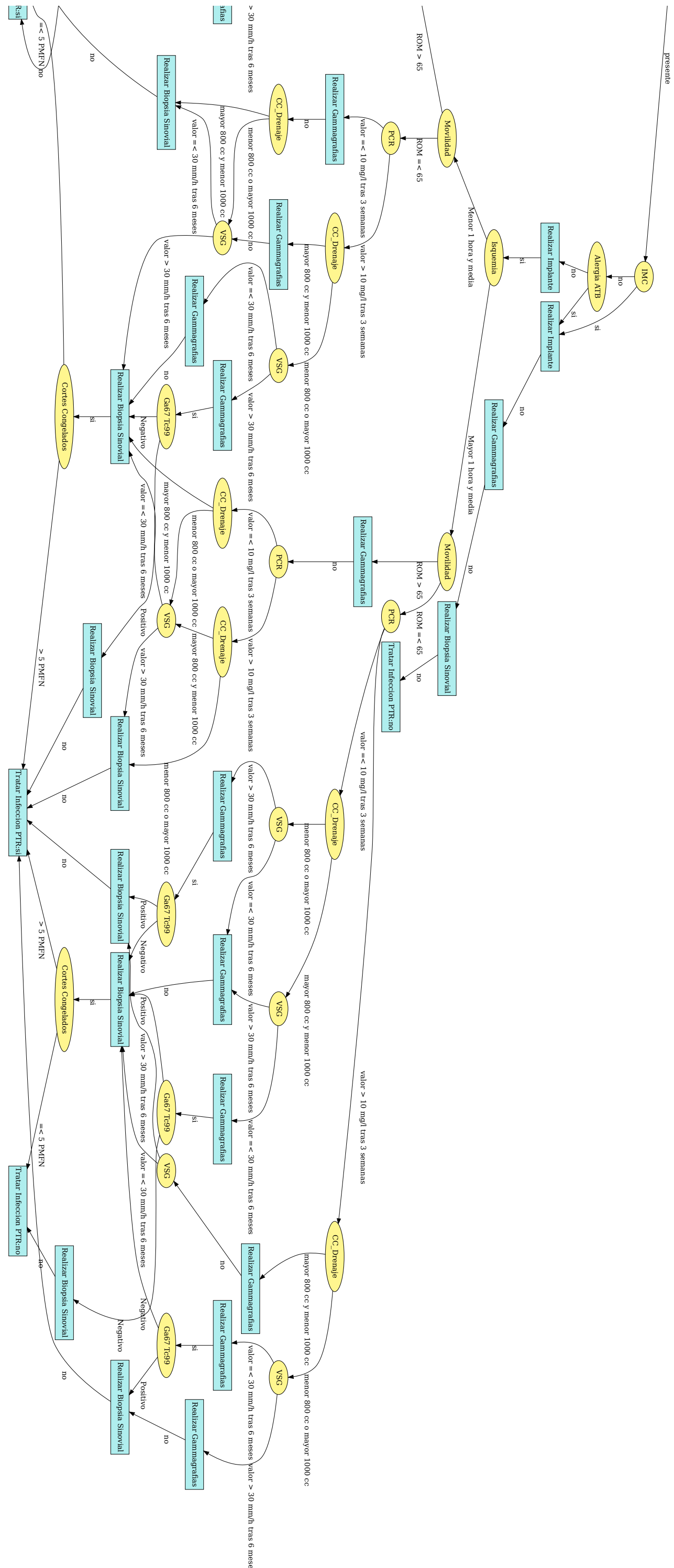

Figure 4.15: Example 5 - A section of the Policy Tree after coalescing ($ci = 3$). The whole tree size is 264 nodes.

Figure 4.16 depicts on the PT tree size (A) and computing time (B), as a result of using different values of coalescence indexes (from 1 to 100). The reduction of nodes in the PT after applying different coalescence indexes is exponential. The minimum size of the coalescing PT is with coalescence index equal to 100 and the computing time in this case is 60 seconds. This PT is depicted on Figure 4.17 (Part 1) and Figure 4.18 (Part 2) and its size is 76 nodes, reducing the number of nodes to a 11.3%. Note that in this case only there are 2 leaf nodes, one for each state of *Tratar Infección PTR*.

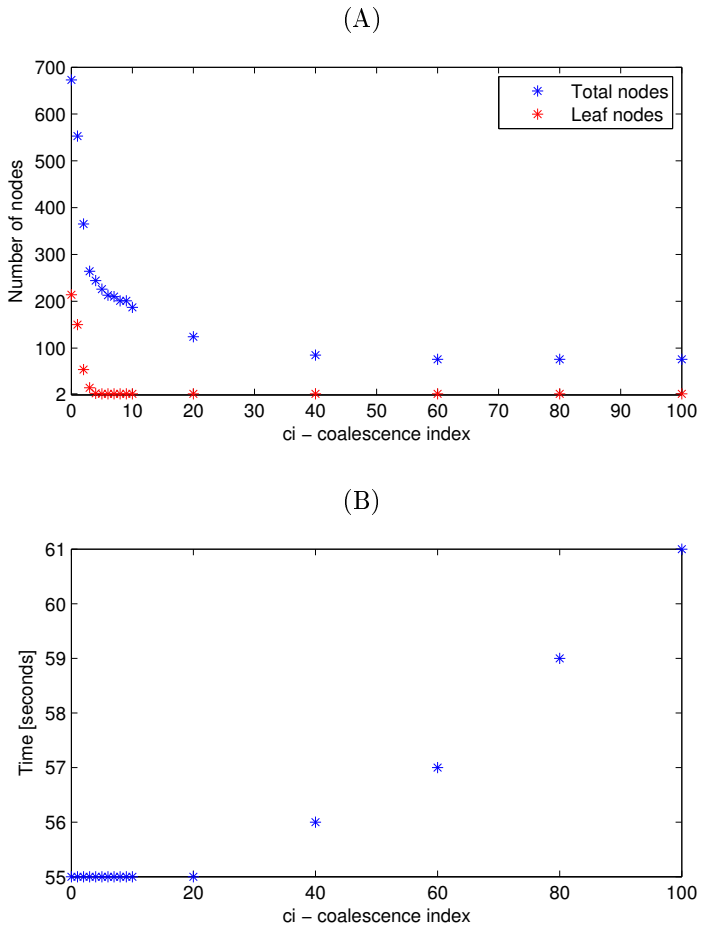

Figure 4.16: (A) Number of nodes and (B) computing time, for different coalescence indexes.

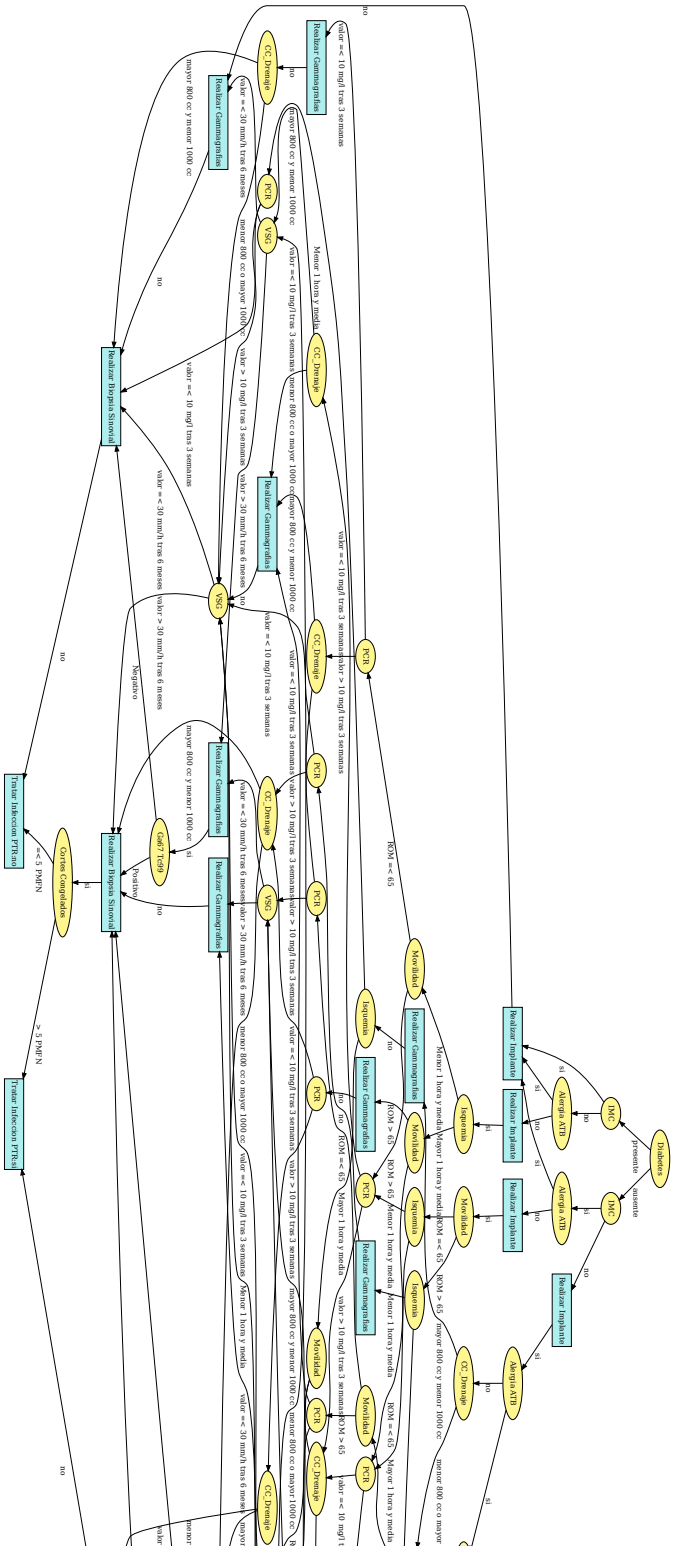

Figure 4.17: Example 5 - Part 1 - PT after coalescing ($ci = 100$). The whole tree size is 76 nodes.

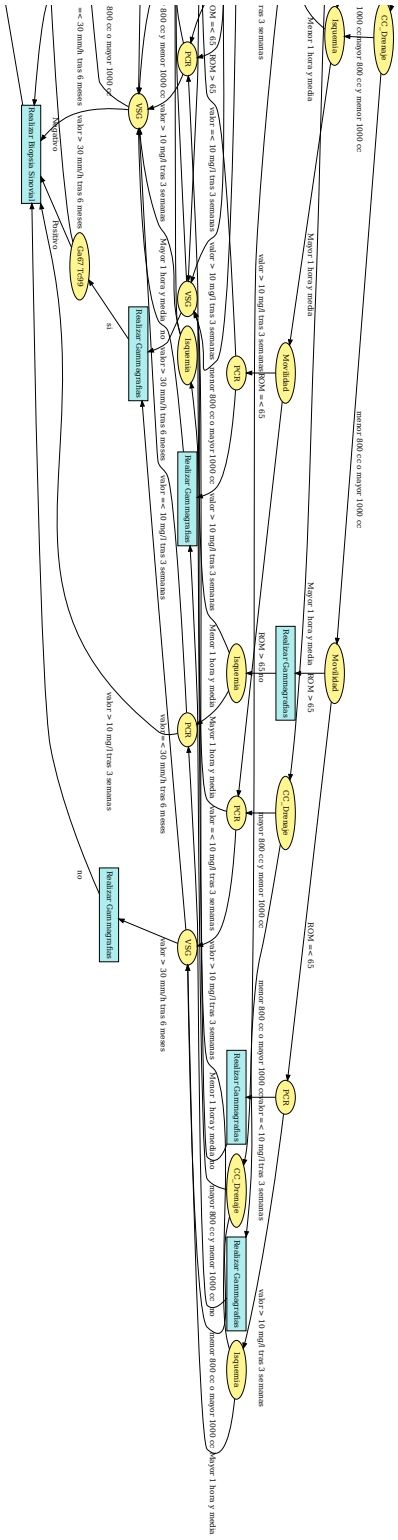

Figure 4.18: Example 5 - Part 2 - PT after coalescing ($ci = 100$). The whole tree size is 76 nodes.

Although the interpretation of the PT in Figure 4.17 and Figure 4.18 is arduous, it enhances the recognition of patterns in the ending nodes of the tree. Clinicians may be interested in examining and comparing scenarios to understand the reasons behind the treatment *Tratar Infección PTR*. For example, Table 4.2 and Table 4.3 present 4 scenarios of optimal strategies that can be extracted easily from the PT. The clinical interpretation of *Scenario 1 (Realizar Implante: si, Realizar Gammagrafías: si,* ***Realizar Biopsia Sinovial: no****, Tratar Infección PTR: no)* compared to *Scenario 2 (Realizar Implante: si, Realizar Gammagrafías: si,* ***Realizar Biopsia Sinovial: si****, Tratar Infección PTR: no)* reveals that, the difference between them can be explained by noting that the *Ga67 Tc99* is different for both scenarios. On the other hand, the clinical interpretation of Scenario 3 (*Realizar Implante: si, Realizar Gammagrafías: si, Realizar Biopsia Sinovial: no, Tratar Infección PTR: no*) compared to Scenario 4 (*Realizar Implante: si, Realizar Gammagrafías: si, Realizar Biopsia Sinovial: no, Tratar Infección PTR: no*) reveals that, despite of the different combination of values of *PCR* and *VSG* the optimal strategy is the same.

# Scenario	Description
Scenario 1	*Diabetes: ausente, IMC: no, Realizar Implante: si, AlergiaATB: si, CC_Drenaje: mayor 800 cc y menor 1000 cc, Isquemia: menor 1 hora y media, Movilidad: ROM < 65, PCR: valor > 10 mg/l tras 3 semanas, VSG: valor =< 30 mm/h tras 6 meses, Realizar Gammagrafías: si,* ***Ga67 Tc99: negativo*** *,* ***Realizar Biopsia Sinovial: no,*** and *Tratar Infección PTR: no.*
Scenario 2	*Diabetes: ausente, IMC: no, Realizar Implante: si, AlergiaATB: si, CC_Drenaje: mayor 800 cc y menor 1000 cc, Isquemia: menor 1 hora y media, Movilidad: ROM < 65, PCR: valor > 10 mg/l tras 3 semanas, VSG: valor =< 30 mm/h tras 6 meses, Realizar Gammagrafías: si,* ***Ga67 Tc99: positivo****,* ***Realizar Biopsia Sinovial: si****,* ***Cortes Congelados: =<5 PMFN,*** *and Tratar Infección PTR: no.*

Table 4.2: Two scenarios of optimal strategies. The decision variables of the ID are shown in blue and the differences between Scenario 1 & Scenario 2 are shown in bold face.

# Scenario	Description
Scenario 3	*Diabetes: ausente, IMC: no, Realizar Implante: si, AlergiaATB: si, CC_Drenaje: mayor 800 cc y menor 1000 cc, Isquemia: menor 1 hora y media, Movilidad: ROM < 65,* ***PCR: valor >10 mg/l tras 3 semanas, VSG: valor =< 30 mm/h tras 6 meses,*** *Realizar Gammagrafías: si, Ga67 Tc99: negativo , Realizar Biopsia Sinovial: no, and Tratar Infección PTR: no.*
Scenario 4	*Diabetes: ausente, IMC: no, Realizar Implante: si, AlergiaATB: si, CC_Drenaje: mayor 800 cc y menor 1000 cc, Isquemia: menor 1 hora y media, Movilidad: ROM < 65,* ***PCR: valor =< 10 mg/l tras 3 semanas, VSG: valor > 30 mm/h tras 6 meses,*** *Realizar Gammagrafías: si, Ga67 Tc99: negativo , Realizar Biopsia Sinovial: no, and Tratar Infección PTR: no.*

Table 4.3: Two scenarios of optimal strategies. The decision variables of the ID are shown in blue and the differences between Scenario 3 & Scenario 4 are shown in bold face.

4.3 Discussion and related work

The main drawback of representing the optimal policy in the form of policy tables is that these tables can have a large size and for a decision maker it can be impossible to draw any useful information from them. In OpenMarkov it is possible to expand this tables into an equivalent decision tree. The strengths of the decision tree representation method are its simplicity and its flexibility. Decision trees are based on the semantics of scenarios and each path in a decision tree from the root to a leaf represents a scenario. These semantics are very intuitive and easy to understand. One of the weaknesses of the decision tree representation method is its combinatorial explosiveness in problems in which there are many variables. It stems from the fact that the number of scenarios is an exponential function of the number of variables in the problem. In a symmetric decision problem with n variables, where each variable has m possible values, there are m^n scenarios. Since a decision tree representation depicts all scenarios explicitly, it is computationally infeasible to represent a decision problem with many variables.

Policy trees have been proposed as an alternative representation to policy tables and decision trees. Whe have demonstrated that PTs provide a powerful formalism for representing comprehensible optimal policies. Furthermore, we have shown that, if a variable of the ID is not relevant in the optimal strategy, the PT simply ignores this variable. As regards time computing, the top down method that is used for constructing PTs is computationally undemanding.

Other authors [8, 12, 2] deal with the problem of the size of policy tables, turning the tables into minimum storage lists (called KBM2L). From the point

of view of the decision maker, these lists are less intuitive than PTs. But these authors found that a good organization reduces the memory requirements to store optimal decision tables and also it extracts qualitative information about variables.

Due to the exponential growth of the tree representation, the PTs are limited to small problems. Coalescence can be a useful mechanism for depicting more compact representations of policy trees. The higher the coalescence index is, the more compact the tree is. But on the other hand, in case of having IDs with many variables, the higher the coalescence index is, the more crossing paths the tree can have. Thus, compacting a tree by coalescence can result in an intricate tree in those cases. Although the coalescence method seems to fail in big problems like in example 5, the human expert can use this method for enhancing the recognition of patterns and comparing scenarios.

In the literature, one of the used methods to improve the induction and the size of classification trees is the use of evolutionary algorithms (EAs) [20]. ID3 is able to generate only a suboptimal tree because anytime a split is chosen a certain subspace of possible trees is not investigated anymore by the algorithm. The power of EAs offers the capability of enhancing the desired characteristics such as accuracy while minimizing the size of the classification tree. On the other hand, it has been shown that training classifiers with EAs takes longer than with traditional algorithms like ID3.

Chapter 5

Conclusions and Future Work

This chapter analyzes the work done during this master thesis, extracting the main technical conclusions. Besides, this chapter pretends to guide readers interested in continuing and extending this work. As in any research project, the work that has been presented in this thesis can be both improved and extended.

5.1 Technical work and conclusions

The main output after solving an ID is the optimal strategy. Software tools such as OpenMarkov usually present each optimal policy in the form of a policy table, which contains a column for each configuration of the informational predecessors of the decision. One of the main drawbacks is that in some cases these tables can have a large size.

In OpenMarkov it is possible to expand this tables into an equivalent decision tree, but as this decision tree depicts all scenarios explicitly, it is computationally infeasible to represent a decision problem with many variables. This evidence, brings to light the need of having an alternative representation to policy tables that could summarize the optimal policy. We have implemented Policy trees (PT) as a compact way of representing the results of the DI on best policies. The visualization of the policy tables as policy trees helps to reduce search for information and enhances the recognition of patterns. Furthermore, if a variable is not relevant in a scenario, the PT simply ignores this variable.

We have shown how one could construct a Policy Tree using the ID3 algorithm. For ID3, the two key concepts are *entropy* (measurement of uncertainty) and *information gain*. Using these parameters, we created a top-down tree that the decision maker can traverse in order to make a decision given a new data set. The recursive partitioning algorithms (like ID3) make use of greedy heuristics to reach a compromise between the tree quality and the computational effort. This kind of greedy approach, that splits the data locally (i.e., in a given node) and only once for each node, allows to grow a tree in a reasonable amount of

time. On the other hand, this rule is able to generate only a suboptimal tree because anytime a split is chosen a certain subspace of possible trees is not investigated anymore by the algorithm. If the optimal tree is included in one of those subspaces there is no chance for the algorithm of finding it.

If a PT has several identical subtrees, then the solution process can be made more efficient by coalescing the subtrees [32]. The notion of coalescence is well known for decision trees, and decision tree figures from the literature often depict coalescence for compactness. It should be noted that automating coalescence in decision trees is not easy because it involves constructing the complete uncoalesced tree and then recognizing repeated subtrees. In this study we have implemented a coalescing method that, on one hand it reduces the number of nodes in the policy tree. However, on the other hand as a drawback, we have shown that in case of having trees of huge dimensions the coalescence method can create a complex tree. In those cases, this coalescing method can be used for recognition of patterns and comparisons between scenarios.

To sum up, the main advantage of the PT is that once the tree has been built, the information it contains (optimal strategy) can easily be examined and it enhances the recognition of patterns. This is an important advantage with respect to black box machine learning techniques, such as neural nets. However, the PTs created through complicated policy tables are often so large that the models they represent are too complex to be understood and the advantage of decision trees is lost.

5.2 Future research lines

- One of the requirements of ID3 is that all attribute values must be discrete values. Thus, **for handling attributes with continuous values, C4.5 [36] might be used** in order to overcome this ID3 disadvantage.

- In this project the *Gain Ratio* measure was used for selecting attributes. However, **a variety of measures for selecting attributes** have been proposed in the literature [28, 18, 9, 7, 17]. Mingers et al. (1989) [28], provides an experimental analysis of the relative significant differences in the sizes of the unpruned trees produced by the different selection measures. He reports significant differences in the sizes of the unpruned trees.

- The ID3 algorithm computes the *Gain Ratio* for each attribute and selects the one with the highest gain. In case of having attributes with the same gain, the algorithm selects randomly an attribute. In order to get the most compact tree, we might need to consider all those equiprobable attributes and by **backtracking** check the best attribute.

- In previous studies [8, 12, 2], the authors propose turning the decision tables into minimum storage lists (called **KBM2L**), which include the same knowledge but are much more compact. These authors found that a good organization reduces the memory requirements to store optimal

decision tables and also it extracts qualitative information about variables. Thus, as there is a growing interest in finding explanations and optimizing the storage space of the decision tables, the future work can involve this research line.

- ID3 is a greedy algorithm that is able to generate only a suboptimal tree. Therefore it seems reasonable to apply alternative methods of optimization to this problem. Taking these considerations into account, we propose **Evolutionary Algorithms** to try to find best exploratory trees [30, 20, 39, 40].

- A policy for a decision is a family of probability distributions for the options of the decision. In this project, we focused on a policy in which every probability is either 0 or 1, called a deterministic policy. However, after evaluating an ID, the policy for a decision node could be an equiprobable probability for each of its values when there is not an unique optimal policy. For those cases, the decision node in the PT should show all its possible values as optimal decisions. In Example 1 if the *Do test?* decision node did not have an unique optimal policy, the PT would be as it is shown in Figure 5.1.

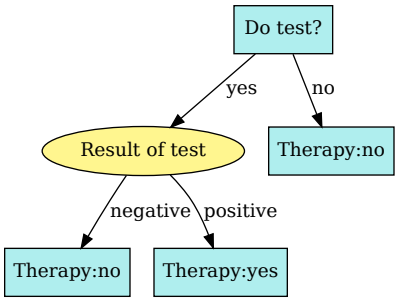

Figure 5.1: Policy Tree proposal for equiprobable strategy of *Do Test?*.

Bibliography

[1] Alfred V Aho and John E Hopcroft. *Design & Analysis of Computer Algorithms*. Pearson Education India, 1974.

[2] Concha Bielza, Juan A Fernández del Pozo, and Peter JF Lucas. Explaining clinical decisions by extracting regularity patterns. *Decision Support Systems*, 44(2):397–408, 2008.

[3] Concha Bielza and Prakash P Shenoy. A comparison of graphical techniques for asymmetric decision problems. *Management Science*, 45(11):1552–1569, 1999.

[4] Susanne G Bøttcher and Claus Dethlefsen. *deal: A package for learning Bayesian networks*. Department of Mathematical Sciences, Aalborg University, 2003.

[5] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[6] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[7] R López De Mántaras. A distance-based attribute selection measure for decision tree induction. *Machine learning*, 6(1):81–92, 1991.

[8] JA Fernández del Pozo, Concha Bielza, and Manuel Gómez. Knowledge organisation in a neonatal jaundice decision support system. In *Medical Data Analysis*, pages 88–94. Springer, 2001.

[9] Tom Dietterich, Michael Kearns, and Yishay Mansour. Applying the weak learning framework to understand and improve c4. 5. In *ICML*, pages 96–104. Citeseer, 1996.

[10] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[11] Jean-Daniel Fekete, Jarke J Van Wijk, John T Stasko, and Chris North. The value of information visualization. In *Information visualization*, pages 1–18. Springer, 2008.

[12] JA Fernández del Pozo, Concha Bielza, and Manuel Gómez. A list-based compact representation for large decision tables management. *European Journal of Operational Research*, 160(3):638–662, 2005.

[13] Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *SOFTWARE - PRACTICE AND EXPERIENCE*, 30(11):1203–1233, 2000.

[14] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[15] J. Matheson. Howard, R. *Influence diagrams. The Principles and Applications of Decision Analysis , Vol. II.* Strategic Decisions Group, Menlo Park, CA, 721-762., 1984.

[16] Laurent Hyafil and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[17] Chen Jin, Luo De-lin, and Mu Fen-xiang. An improved id3 decision tree algorithm. In *Computer Science & Education, 2009. ICCSE'09. 4th International Conference on*, pages 127–130. IEEE, 2009.

[18] Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 459–468. ACM, 1996.

[19] Ron Kohavi. Bottom-up induction of oblivious read-once decision graphs. In *Machine Learning: ECML-94*, pages 154–169. Springer, 1994.

[20] Peter Kokol, Sandi Pohorec, Gregor Štiglic, and Vili Podgorelec. Evolutionary design of decision trees for medical application. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(3):237–254, 2012.

[21] C. Lacave and F. J. Díez. A review of explanation methods for Bayesian networks. *Knowledge Engineering Review*, 17:107–127, 2002.

[22] Steffen L Lauritzen and Dennis Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47(9):1235–1251, 2001.

[23] D. León. A probabilistic graphical model for total knee arthroplasty. Master's thesis, Dept. Artificial Intelligence, UNED, Madrid, Spain, 2011.

[24] Wei-Yin Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12(2):361–386, 2002.

[25] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[26] M. Luque. *Probabilistic Graphical Models for Decision Making in Medicine.* PhD thesis, Universidad Nacional de Educación a Distancia, 2009.

[27] Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. Sliq: A fast scalable classifier for data mining. In *Advances in Database Technology-EDBT'96*, pages 18–32. Springer, 1996.

[28] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine learning*, 3(4):319–342, 1989.

[29] Tom M Mitchell. *Machine learning. 1997*, volume 45. 1997.

[30] Francesco Mola, Raffaele Miele, and Claudio Conversano. Evolutionary algorithms in decision tree induction.

[31] Francesco Mola and Roberta Siciliano. A fast splitting procedure for classification trees. *Statistics and Computing*, 7(3):209–216, 1997.

[32] S.M Olmsted. *On representing and solving decision problems*. PhD thesis, Department of Engineering-Economic Systems, Stanford University, Standford, CA., 1983.

[33] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[34] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116. ACM, 2004.

[35] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[36] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Diego, 1993.

[37] J Ross Quinlan. Decision trees and decision-making. *Systems, Man and Cybernetics, IEEE Transactions on*, 20(2):339–346, 1990.

[38] J Ross Quinlan et al. *Discovering rules by induction from large collections of examples*. Expert systems in the micro electronic age. Edinburgh University Press, 1979.

[39] MD Ryan and VJ Rayward-Smith. The evolution of decision trees. In *Genetic Programming 1998: Proc. 3rd Annual Conf*, pages 350–358, 1998.

[40] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[41] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[42] Mary Slocum. Decision making using id3 algorithm. *InSight: Rivier Academic Journal*, 8(2), 2012.

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

[44] Marta Vomlelová and Finn V Jensen. An extension of lazy evaluation for influence diagrams avoiding redundant variables in the potentials. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(supp01):1–17, 2004.

[45] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, second edition, 2005.