Universidad de Granada

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Máster en Soft Computing y Sistemas Inteligentes

Investigación Tutelada

Approximate inference in Influence Diagrams using binary trees

Julio de 2012

Alumno: Rafael Cabañas de Paz
Director: Dr. D. Andrés Cano Utrera
Director: Dr. D. Manuel Gómez Olmedo

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Decision problems under uncertainty have traditionally been represented and evaluated using *Decision Trees* [17]. However, their main problem is the exponential growth of the representation. An alternative for representing decision problems are *Influence Diagrams* (IDs). They can encode the independence relations between variables, thereby avoiding the exponential growth. Besides, IDs are intuitive enough to communicate with decision makers and experts. At the same time, it is a precise description of information that can be stored and manipulated by a computer.

For complex decision problems, the evaluation of an ID becomes unfeasible due to its computational cost: the set of information states exceeds the storage capacity of PCs or the optimal policy must be obtained in a short period of time. Thus it is necessary to use approximated methods for ID evaluation. For example, *Limited Memory Influence Diagrams* [13], sampling techniques [6, 3]. Some of the deterministic methods use alternative representations for potentials, such as *numerical trees* (NTs) [5]. This representation offers the possibility of taking advantage of *context-specific independences*. NTs can be pruned and converted into smaller trees when potentials are too large, thus leading to approximate algorithms.

Here, we introduce a new kind of tree for potentials representation in which internal nodes always have two children: *binary trees* (BTs). These trees allow the specification of finer grained context-specific independences than those which can be represented with NTs, and should lead to more efficient algorithms. We give detailed descriptions of the algorithms we use to evaluate IDs using BTs. As in the case of NTs, approximate evaluation of IDs can be performed by pruning the trees. All these algorithms were implemented in Java using the Elvira system [1], which is an open-source tool to construct model based decision support systems.

[1]http://leo.ugr.es/elvira/

The present work is organized in the following way: Chapter 2 introduces some concepts about IDs and its evaluation (exact and approximate); Chapter 3 presents key issues about BTs and how they are used during IDs evaluation; Chapter 4 includes the experimental work and results; finally Chapter 5 details my conclusions and lines for future work.

Chapter 2

Preliminaries

2.1 Decision Theory

Research on decision making has mainly focused on several questions about decisions. How should decisions be made? What is the best decision, and how can the decision maker finds, recognizes, and implements it? Questions of this kind are asked about the content of almost any human performance that requires skill or knowledge. What should the person do? How can he or she produce appropriate behaviour? How can the appropriateness of various alternative behaviours be measured and turned into a score?

The goal of a decision is to choose the best option available, and that "best" is a quantitative idea. Consider the following example: someone offers you a bet. You will toss a coin three times. If it comes up heads on all three tosses, you win 1.00 euros. Otherwise, you lose 0.15 euros. In this problem, decision maker has two options: *gambling* or *not gambling*. This is a decision problem under uncertainty, since there are some variables of the problem that the decision maker does not know a priori: each of the coins available, when tossed, are equally likely to land heads or tails and that all tosses are independent of one another.

You should base this trivial decision on a trivial calculation: you will either win 1.00 or lose 0.15 euros. The probability of winning 1 euro is 1/8. The expected value (or utility) of *gambling* can be computed as follows:

$$EU(gambling) = \frac{1}{8} \cdot 1.00 + \frac{7}{8} \cdot (-0.15) = -0.625$$

On the other hand, if the decision is *not gambling*, the expected utility will be:

$$EU(notGambling) = 0$$

Therefore, the best decision for this problem is *not gambling*, since it is the decision that maximizes the expected utility.

In this context of making decision under uncertainty appears the concept of *Decision Support Systems* (DSS), which are computer technology solutions that can be used to support complex decision making and problem solving. The concepts involved in DSS were first articulated in the early 1970s by Gorry and Michael S. Scott Morton; they used the term *management decision systems* (Morton 1971) and defined it "as an interactive computer based system that helps decision makers utilize data and models to solve unstructured problems".

2.2 Influence Diagrams

An influence diagram (ID) [16] is a probabilistic graphical model for reasoning about decision making under uncertainty. It is a graphical representation of a decision problem involving a sequence of decisions and random variables. Like Bayesian networks (BNs), an ID contains *random or chance nodes*, but also two additional types: *decision nodes* (mutually exclusive actions which the decision maker cannot control) and *utility* or *value nodes* (representing decision maker preferences). Utility variables may depend on both: chance variables and decision variables.

An ID consists of a directed acyclic graph over chance nodes, decision nodes, and utility nodes with the following structural properties:

- there is a directed path comprising all decision nodes.
- the utility nodes have no children

For the qualitative specification, it is required that:

- the decision nodes and the chance nodes have a finite set of mutually exclusive states.
- the utility nodes has no states.
- to each chance node there is required a conditional probability function over its parents (*probability potential*).
- to each utility node there is attached a real-valued function over its parents (*utility potential*).

Links represent dependencies: probabilistic for links into chance nodes, informational for links into decision nodes (states for decision parents are known before the decision is taken), and functional for links into value nodes. Decision maker preferences are expressed as *utility functions*, indicating the local utility for the configurations of the variables in their domain.

Notation

The set of chance nodes is denoted by $\mathcal{U}_C$, the set of decision nodes is denoted $\mathcal{U}_D$, and the set of utility nodes is denoted by $\mathcal{U}_V$. The decision nodes have a temporal order, $D_1, \ldots, D_n$, and the chance nodes are partitioned according to when they are observed: $\mathcal{I}_0$ is the set of chance nodes observed prior to any decision, and $\mathcal{I}_i$ is the set of chance nodes observed after D_i is taken and before the decision D_{i+1} is taken. Finally, $\mathcal{I}_n$ is the set of chance nodes never observed or observed after the last decision. That is, we have a partial temporal ordering:

$$\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$$

In describing an ID, it is more convenient to think in terms of the predecessors and successors. Thus, a node can belong to different sets in relation to another node:

- **Informational predecessors**: let D_i be a decision node, then the direct predecessors of D_i are called *informational predecessors* or *informational parents*. The set of all informational predecessors of D_i is denoted $pa(D_i)$. Notice that $\mathcal{I}_0$ contains direct and indirect predecessors of D_i. The set of all possible combinations of values for direct predecessors of decision node D_i is called the *information set* for D_i. The elements of this set are denoted *information states for* D_i.

- **Conditional predecessors**: let X_i be a chance or utility node, then the direct predecessors of X_i are called *conditional predecessors* or *conditional parents*. The set of all conditional predecessors of X_i is denoted $pa(X_i)$.

- **Direct sucessors**: let X_i be a chance or decision node, then the set of all *direct successors* or *children* of X_i are denoted $ch(X_i)$.

The *universe* of the ID is $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D = \{X_1, \ldots, X_n\}$. Let us suppose that each variable X_i takes values on a finite set $\Omega_{X_i} = \{x_1, \ldots x_{|\Omega_{X_i}|}\}$. If I is a set of indexes, we shall write $\mathbf{X}_I$ for the set of variables $\{X_i | i \in I\}$, defined on $\Omega_{\mathbf{X}_I} = \times_{i \in I} \Omega_{X_i}$. The elements of Ω_{X_I} are called configurations of $\mathbf{X}_I$ and will be represented as $\mathbf{x}_I$.

In an ID, each chance node X_i have attached a conditional probability distribution $P(X_i | pa(X_i))$. In the same way, each utility node V_i have attached a utility function $U(pa(V_i))$. In general, we will talk about potentials (not necessary normalized). Let $\mathbf{X}_I$ be the set of all variables involved in a potential, then a *probability potential* denoted by ϕ is a mapping $\phi : \Omega_{\mathbf{X}_I} \rightarrow [0, 1]$. An *utility potential* denoted by ψ is a mapping $\psi : \Omega_{\mathbf{X}_I} \rightarrow \mathbb{R}$.

Oil Widcatter example

For illustrating IDs, we will consider the oil wildcatter problem [19] shown in Figure 2.1. In this problem, the decision maker must choose between drilling or not in order to find oil. The profit of drilling will depend on the amount of oil, which can be predicted by doing a test. However, this test has a cost (negative profit) and it can be noisy depending on the seismic structure. This ID has two decision variables (*Test* and *Drill*), four chance variables (*Test Results*, *Seismic Structure*, *Amount of Oil* and *Cost of Drilling*) and only one utility variable (*Profit*).

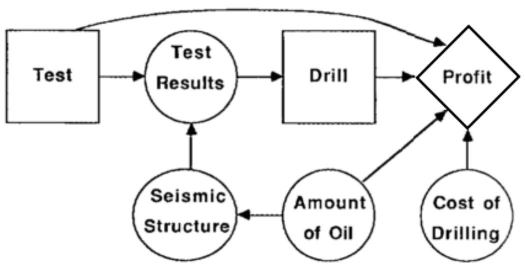


Figure 2.1: ID representing oil wildcatter's problem

The partial order in this example will be: *Test* $\prec$ {*Test Results*, *Seismic Structure*, *Amount of Oil*} $\prec$ *Drill* $\prec$ {*Cost of Drilling*}. The decision node *Drill* has only one informational predecessor (*Test Results*) and one successor (*Profit*). The chance node *Test Results* has two conditional predecessors (*Test, Seismic Structure*) and one successor (*Drill*).

No-forgetting assumption

The semantic of IDs usually assumes that the decision maker remembers past observations and decisions *(no-forgetting assumption).* Thus, at D_i we know the state of variables appearing before D_i. For example, at oil wildcatter ID, at decision *Drill*, the decision maker knows if the *Test* was performed.

2.3 Evaluation of Influence Diagrams

When evaluating an ID, it must be chosen which is the best option (or policy) for each decision variable. The evaluation is usually performed according to the *maximum expected utility principle*, which states that we should always choose an alternative that maximizes the expected utility. Here, the basics about ID evaluation and some methods (exact and approximate) are described.

The Chain Rule for Influence Diagrams

For Bayesian networks we have that the joint probability is the product of all probability potentials attached to the variables in the network. For IDs there is a similar theorem. Again, decision variables act differently from chance variables. Since a decision variable eventually will come under decision maker's control, it requires no prior probabilities.

Definition 1 (The chain rule for IDs)**.** Let ID be an influence diagram universe $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$ and $pa(X)$ the conditional predecessors of a node X. Then the joint probability of the ID is given by the expression:

$$P(\mathcal{U}_C|\mathcal{U}_D) = \prod_{X \in \mathcal{U}_C} P(X|pa(X))$$

Strategies and Expected Utilities

The relevant past π_{D_i} for a decision variable D_i is the set of its informational predecessors and the decision history (all previous decisions). Then, a policy (or decision rule) for D_i is a mapping $\delta_i : \Omega_{\pi_{D_i}} \rightarrow \Omega_{D_i}$.

A strategy is an ordered set of policies $\Delta = \{\delta_1, \ldots, \delta_n\}$ including a policy for each decision. Each strategy will have associated an expected utility.

Definition 2 (Expected Utility)**.** Let ID be an influence diagram with universe $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$, X and V a chance and utility node respectively whose conditional predecessors are $pa(X)$ and $pa(V)$. Let $\Delta = \{\delta_1, \ldots, \delta_n\}$ be an strategy for the ID. Then the expected utility of this strategy is:

$$EU(\Delta) = \prod_{X \in \mathcal{U}_C} P(X|pa(X)) \left(\sum_{V \in \mathcal{U}_V} U(pa(V)) \right)$$

An optimal strategy $\widehat{\Delta}$ returns the optimal decision for the decision maker to make at each decision. Evaluating an ID is to compute an optimal strategy $\widehat{\Delta}$ maximizing the expected utility for the decision maker and and its corresponding $MEU(ID)$.

Definition 3 (Maximum Expected Utility)**.** Let ID be an influence diagram over the universe $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$. Let the temporal order of the variables be described as $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$ and let $V = \sum_i V_i$. Then $MEU(ID)$ is:

$$MEU(ID) = \sum_{\mathcal{I}_0} \max_{D_1} \cdots \max_{D_n} \sum_{\mathcal{I}_n} \prod_{X \in \mathcal{U}_C} P(X|pa(X)) \left(\sum_{V \in \mathcal{U}_V} U(pa(V)) \right)$$

For evaluating an ID, chance variables are removed by *sum-marginalization* like in BNs. That is, adding up all its values in a potential. By contrast, decision variables are removed using *sum-marginalization.* This operation is analogous to sum- marginalization, but instead of adding up values, the maximum is taken.

2.3.1 Exact Inference

Variable Elimination

The variable elimination method (VE) for IDs [11] has many similarities with the corresponding method for BNs [25]: it starts with a set of potentials and it eliminates one variable at a time. There are however differences. First of all, all variables in the decision problem are removed in reverse order of the partial ordering imposed by the information constraints. Secondly, chance variables are removed using sum-marginalization whereas decision variables are removed using max-marginalization. That is, it first sum-marginalizes $\mathcal{I}_n$, then max-marginalizes D_n, sum-marginalizes $\mathcal{I}_{i-1}$, etc. This type of elimination order is called a strong elimination order.

Let X be the variable to be removed, Φ and Ψ the set of all probability and utility potentials respectively. Then potentials affected are selected which have X in the domain:

$$\Phi_X = \{\phi \in \Phi | X \in dom(\phi)\}$$

$$\Psi_X = \{\psi \in \Psi | X \in dom(\psi)\}$$

If X is a chance variable, it is removed using sum-marginalization:

$$\phi_X = \sum_X \prod \Phi_X$$

$$\psi_X = \sum_X \prod \Phi_X \left(\sum \Psi_X \right)$$

However, if X is a decision variable, it is removed using max-marginalization:

$$\phi_X = \sum_X \prod \Phi_X$$

$$\psi_X = \max_X \prod \Phi_X \left(\sum \Psi_X \right)$$

Finally, the set of potentials are updated. In case of probability potentials, they are substituted by the result of marginalization. In case of utility potentials, they are substituted by the result of marginalization divided by the marginalized probability potentials. The aim of the division is to normalized the potential.

$$\Phi = (\Phi \backslash \Phi_X) \cup \{\phi_X\}$$

$$\Psi = (\Psi \backslash \Psi_X) \cup \{\frac{\psi_X}{\phi_X}\}$$

Arc Reversal

The method *arc reversal* (AR) for evaluating IDs was first proposed by R.D. Shachter [21]. This method consists of a sequence of transformations to the diagram that do not modify the optimal policy or expected utility.

AR method allows evaluating *oriented* and *regular* IDs. An ID is said to be oriented if it has only one utility node. For problems with multiple utility functions it is necessary to use *super-value nodes*[23] It is said to be regular if it satisfies the following conditions:

- the directed graph has not cycles.
- the utility node has not children.
- there is a directed path that contains all of the decision nodes.

The method uses four basic operations: barren node removal, chance node removal, decision node removal and arc reversal.

- **Barren node removal**: a chance or decision node is a barren node if it is a sink, that is, it has no successors. It can be removed directly.
- **Chance node removal**: a chance node X can be removed if its only successor is the utility node. After removal, the utility inherits all conditional predecessors from node X (see Figure 2.3). Let U be the previous utility, then the new utility U^* will be:

$$U^* = \sum_X P(X|pa(X)) \cdot U$$

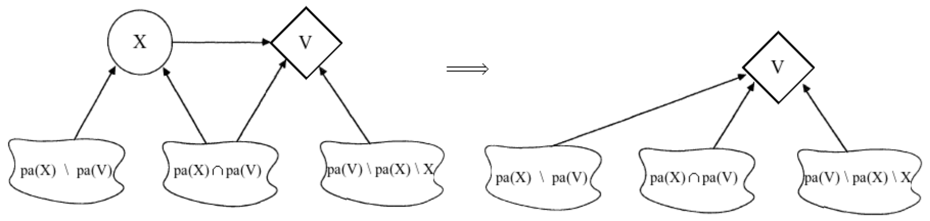


Figure 2.2: Removing a chance node

- **Decision node removal**: a decision node D can be removed if it is an utility conditional predecessor and a successor of the rest of utility informational predecessors. It is removed directly and the utility node does not inherit the informational predecessors of D (see Figure 2.3). Let U be the previous utility, then the new utility U^* will be:

$$U^* = \max_D U$$

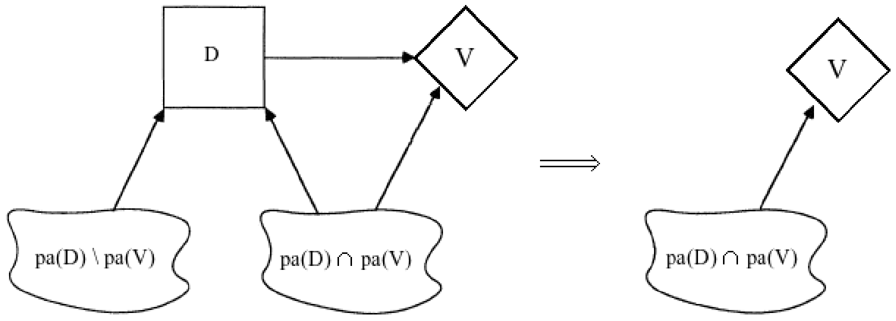


Figure 2.3: Removing a decision node

- **Arc reversal**: given two chance nodes X_i and X_j, the arc between them can be removed if there is not other directed path between them. Afterwards, both nodes inherit each other's parents. For example, let be a node A whose parents are $C, \ldots, D$ and a node B whose parents are $E, \ldots, F$. The result of reversing the arc (A,B) is shown at Figure 2.4.

  The new potentials are computed using Bayes' theorem:

$$P(B|C, \ldots, D, E, \ldots, F) = \sum_A P(A|E, \ldots, F) P(B|C, \ldots, D)$$

$$P(A|B, C, \ldots, D, E, \ldots, F) = \frac{P(A|E, \ldots, F) P(B|C, \ldots, D)}{P(B|C, \ldots, D, E, \ldots, F)}$$

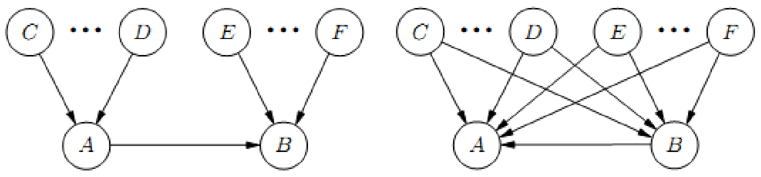


Figure 2.4: Result of reversing the arc (A,B)

Using all these basic operations, the algorithm can be described in the following way:

While it remains nodes to be removed (except utility node):

- Find a chance that could be removed. If any, remove it and change utility associated to the utility node.
- If not, find a decision node that could be removed. If any, save the utility function as a decision table and remove it using max-marginalization.
- If not, find a chance node predecessor of the utility node that has chance nodes as successors. Reverse any arc so that the node could be removed.

Table 2.1: Arc reversal algorithm

Lazy Propagation

The principles of lazy propagation were already used for making inference at BNs [15], so it can be adapted for solving bayesian decision problems [10]. The basic idea of this method is to maintain the decompositions of potentials as late as possible and to postpone computations for as long as possible.

Lazy Propagation is based on message passing in a *junction tree*, which is a representation of an BN or ID constructed by moralization and triangulation of the graph. For IDs, it is build following a strong elimination order, so it will be called *strong junction tree*. The nodes of the strong junction trees correspond to *cliques* (maximal complete subgraphs) of the triangulated graph. The cliques of the strong junction tree are connected by separators such that the so-called junction tree property holds. The junction tree property insures that whenever two cliques C_i and C_j are connected by a path, the intersection $S = C_i \cap C_j$ is a subset of every clique and separator on the path. A junction tree also has the property that each variable and its parents are contained in at least one clique. An example of an ID and a corresponding strong junction tree are shown in Figures 2.5 and 2.6 respectively.

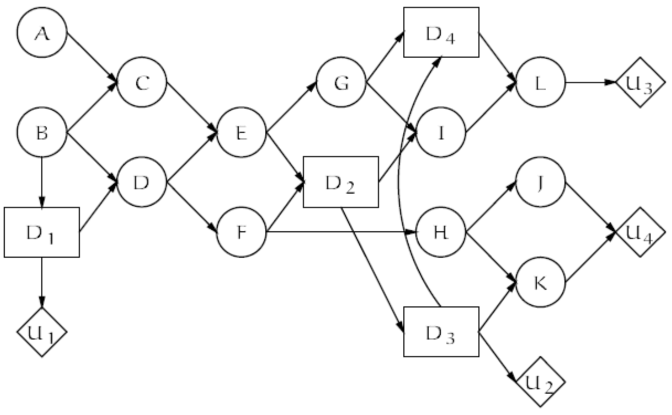


Figure 2.5: An example of an ID

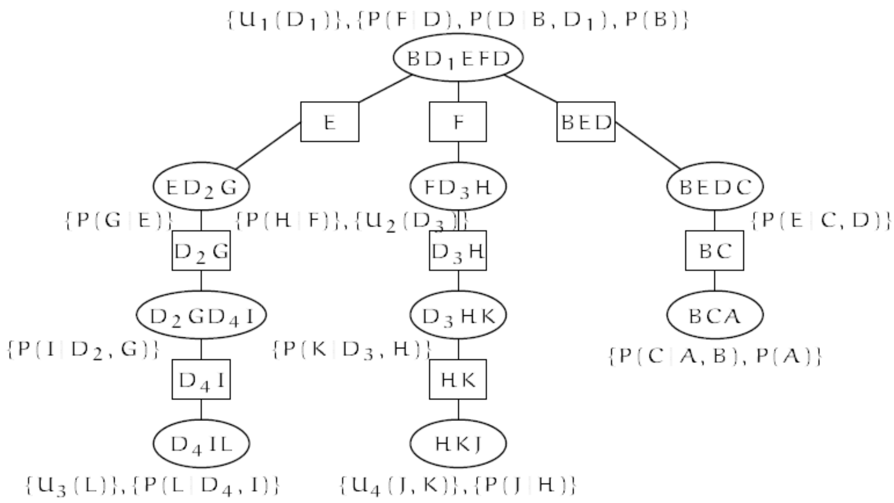


Figure 2.6: Strong junction tree for the ID in Figure 2.5

Computations in the strong junction tree are performed as a special *collect* operation from the leaves to some root of the tree. Initially, each potential in the ID is associated to only one clique containing all the variables in the domain of the potential. That is, each clique C_i has associated a list of probability and utility potentials,Φ_{C_i} and Ψ_{C_i} respectively (potentials are not combined until necessary).

Now, propagation is performed by message-passing. For that, messages are passed from the leaves towards the root by recursively invoking *CollectMessage*. When *CollectMessage* is invoked on a clique C_j from an adjacent clique C_i, clique C_j invokes *CollectMessage* on all other adjacent cliques. When these cliques have finished their *CollectMessage*, C_j absorbs messages from each of them and C_i absorbs from C_j:

Definition 4 (CollectMessage)**.** Let C_i and C_j be adjacent cliques. If *CollectMessages* is invoked on C_j from C_i then:

1. C_j invokes *CollectMessage* on all adjacent cliques except C_i.
2. The message from C_j to C_i is absorbed.

From the description of the *CollectMessage* algorithm, it is clear that information is passed between adjacent cliques by absorption. Consider two cliques C_i and C_j separated by S. Absorption from C_j to C_i over S amounts to eliminating the variables of $C_j \backslash S$ from the list of probability and utility potentials Φ and Ψ associated with C_j and the separators of $ch(C_j)$ and then associating the obtained potentials with S:

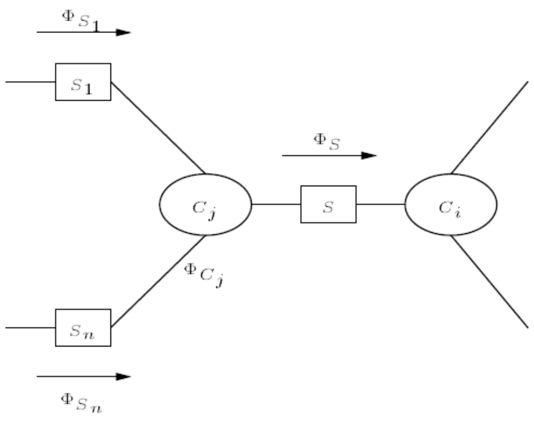


Figure 2.7: An example of an ID

Definition 5 (Absorption)**.** Let C_i and C_j be adjacent cliques and let S be the separator between C_i and C_j. If *Absorption* is invoked on C_j from C_i, then:

1. Let $\mathcal{R}_S = \Phi_{C_j} \cup \Psi_{C_j} \cup \bigcup_{S' \in ch(C_j)} \Phi^*_{S'} \cup \Psi^*_{S'}$.
2. Marginalize out all variables $\{Y \in \Omega_\rho | \rho \in \mathcal{R}_S, Y \notin S\}$. Let Φ^*_S y Ψ^*_S the set of probability and utility potentials obtained.

3. Associate $\{\Phi_S^*$ and $\Psi_S^*\}$ with S as the sets of potentials passed from C_j to C_i.

Absorption algorithm uses the algorithm for marginalization of variables from two sets of potentials:

Definition 6 (Marginalization)**.** Let Φ and Ψ be sets of probability and utility potentials, respectively. If *Marginalization* of variable Y is invoked on $\Phi \cup \Psi$, then:

1. Set $\Phi_Y = \{\phi \in \Phi | Y \in \Omega_\phi\}$ and $\Psi_Y = \{\psi \in \Psi | Y \in \Omega_\psi\}$.
2. Calculate
   - If Y is a chance variable, it is removed using sum-marginalization:
   $$\Phi_Y^* = \sum_X \prod \Phi_Y$$
   $$\Psi_Y^* = \sum_Y \prod \Phi_Y \left(\sum \Psi_Y\right)$$
   - However, if Y is a decision variable, it is removed using max-marginalization:
   $$\Phi_Y^* = \sum_Y \prod \Phi_Y$$
   $$\Psi_Y^* = \max_Y \prod \Phi_Y \left(\sum \Psi_Y\right)$$
   - Return $\Phi^* = \Phi \cup \{\Phi_Y^* \backslash \Phi_Y\}$ and $\Psi^* = \Psi \cup \{\frac{\Psi_Y^*}{\Phi_Y^*} \backslash \Psi_Y\}$

Note that this definition of absorption is asymmetric in the sense that information only flows in the direction permitted by the partial order $\prec$. The optimal policy for a decision variable can be determined from potentials on the clique that is closest to the strong root and contains the decision variable.

2.3.2 Inference using Numerical Trees

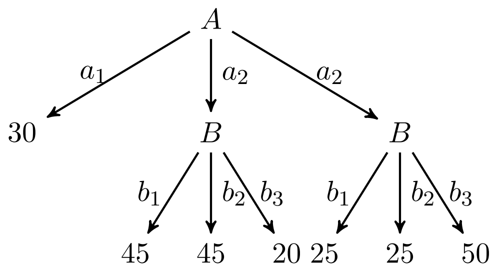
Numerical trees (NT*s*) have previously been used to represent potentials in BNs [5] and for IDs [9]. NT*s* may be used in order to calculate in an exact or approximate way. Their main advantage is that they enable the specification of *context-specific independences* [2]. Moreover a NT can be pruned in order to reduce its storage size which decreases the execution time required to calculate with them. That also will allow the approximation of the potentials. In general, NTs can be used for encoding probability and utility functions, which will be called respectively *numerical probability trees*

(NPTs) and *numerical utility trees* (NUTs).

A numerical tree NT defined over the set of variables $\mathbf{X}_I$ is a directed tree, where each internal node is labelled with a variable (random variable or decision node), each leaf node is labelled with a number (a probability or a utility value). We use L_t to denote the *label of node t.* Each internal node has an outgoing arc for each state of the variable associated with that node. Outgoing arcs from a node X_i are labelled with the same name of the associated state ($x_i \in \Omega_{X_i}$) of X_i. The *size* of a tree $\mathcal{NT}$, denoted *size($\mathcal{NT}$)*, is defined as its number of leaves. A subtree of $\mathcal{NT}$ is a terminal tree if it contains one node labelled with a variable and all the children are leaf nodes.

$U(A,B)$	b_1	b_2	b_3
a_1	30	30	30
a_2	45	45	20
a_3	25	25	50

(a)



(b)

Figure 2.8: Utility potential represented as a table (a) and as a NUT (b)

Figure 2.8 shows two different representations for an utility potential: as a table in (a) and as a NUT in (b). When $A = a_1$, the potential always will take the value 30 independently of the value of B. Therefore, less space is needed for representing it as a tree since it can be pruned. This type of independence is called *context-specific independence* [2].

Operations with Numerical Trees

The three main operations necessary to evaluate IDs are: combination, sum-marginalization and max-marginalization. The three operations can be carried out directly on the numerical tree representation. In [5] and [20], the authors show how to perform the first two operations, and how to build a numerical tree from a potential in detail. Figure 2.9 shows an example of combination of an NUT with a NPT. As it can be observed, the result is a tree of bigger size since a_1 branch cannot be pruned any more. However, tree size could be reduce if the order of the variables is inverted: when $B = b_3$, the potential always will take the value 0.

Sum-marginalization is used when a chance node is going to be removed.

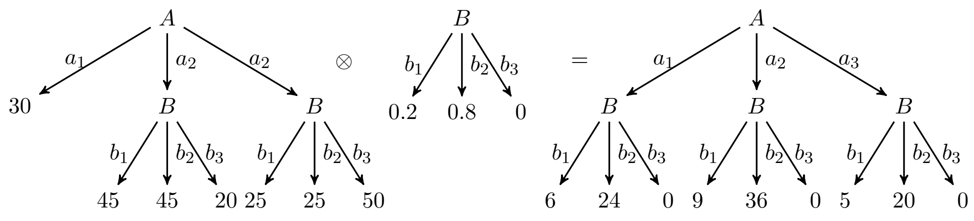


Figure 2.9: Combination of a NUT and a NPT

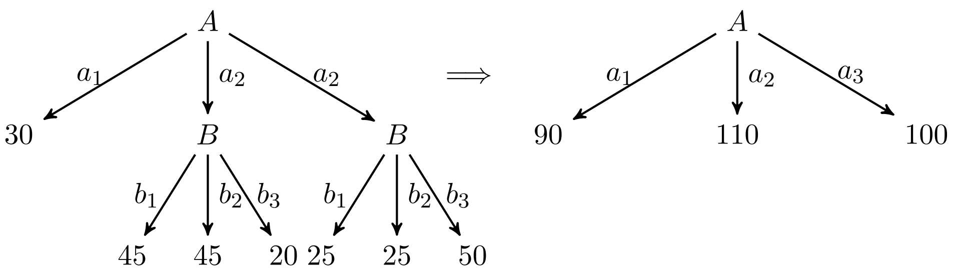


Figure 2.10: sum-marginalization of a NUT with respect to variable B

A variable is removed using sum-marginalization by adding up all its values. An example is shown in Figure 2.10.

Max-marginalization is used when a decision node is going to be removed. This operation is completely analogous to marginalization because it also deletes a variable from the numerical tree, but now instead of adding up values, the maximum is taken. Figure 2.11 shows an example of this operation.

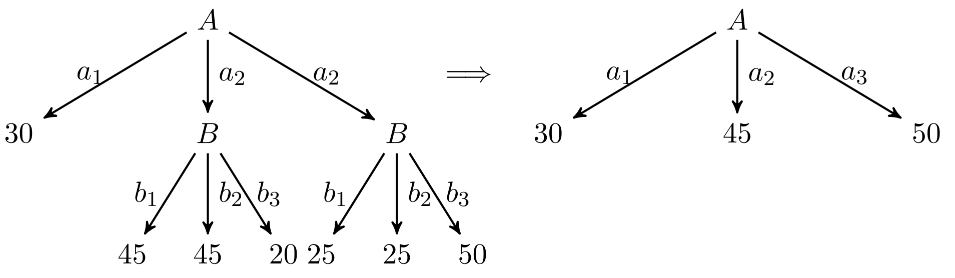


Figure 2.11: max-marginalization of a NUT with respect to variable B

Building and approximating a Numerical Tree

Let ϕ be a potential over a set of variables $\mathbf{X}_I$. Constructing a tree NT representing a potential ϕ without any other additional restriction is a trivial task: the tree will contain one branch for every configuration of $\mathbf{X}_I$, and one leaf for every value $\phi(\mathbf{x}_I)$ with $\mathbf{x}_I \in \Omega_{\mathbf{X}_I}$. However, this procedure can lead to unnecessarily big trees. For that reason it could be necessary approximating the trees by pruning it. For NPTs, the distance between a potential ϕ and its approximated tree $\mathcal{NPT}$ can be measured using *Kullback-Leibler cross entropy*:

$$D(\phi, \mathcal{NPT}) = \sum_{\mathbf{x_i} \in \Omega_{\mathbf{x}_I}} \phi(\mathbf{x}_I) log \frac{\phi(\mathbf{x}_I)}{\mathcal{NPT}(\mathbf{x}_I)} \tag{2.1}$$

In order to prune a NT, a *terminal node* can be selected (a node such that all its children are leaves) and replaced with the average of all its children. The pruning operation can be applied again to the pruned tree until a tree of an acceptable size is obtained, or while the *error* (distance) with respect to the original tree is below a given threshold (stopping condition).

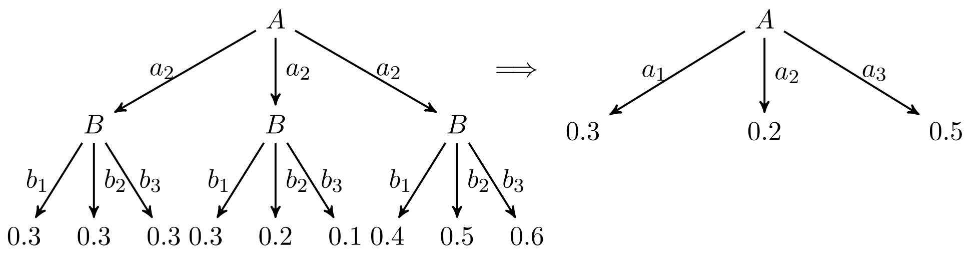


Figure 2.12: Prune of the terminal nodes of a probability tree

Figure 2.12 shows an example of pruning the 3 terminal nodes of a probability tree. The error of this approximation is computed using equation 2.1 where $\phi(\mathbf{x}_I)$ is the value in the original potential (not pruned tree) and $\mathcal{NPT}(\mathbf{x}_I)$ is the corresponding value in the approximated tree:

$$\begin{aligned} D(\phi, \mathcal{NPT}) = & \left(0.3 \cdot \log \frac{0.3}{0.3}\right) + \left(0.3 \cdot \log \frac{0.3}{0.3}\right) + \left(0.3 \cdot \log \frac{0.3}{0.3}\right) + \\ & + \left(0.3 \cdot \log \frac{0.3}{0.2}\right) + \left(0.2 \cdot \log \frac{0.2}{0.2}\right) + \left(0.1 \cdot \log \frac{0.1}{0.2}\right) + \\ & + \left(0.4 \cdot \log \frac{0.4}{0.5}\right) + \left(0.5 \cdot \log \frac{0.5}{0.5}\right) + \left(0.6 \cdot \log \frac{0.6}{0.5}\right) = 0.0725 \end{aligned}$$

It could also be necessary applying the *sort* operation to NTs. This operation tries to restructure the nodes of the numerical tree in such a way that the more informative variables are in the upper levels of the tree. In this way, if the tree is pruned, then only the less informative variables will be eliminated.

Inference using Numerical Trees

Inference algorithms for IDs such as Variable Elimination (VE) [11] can be easily adapted for working with NTs. The adaptation only needs using NTs and their related operations for computing. Prune and sort operation can be applied to each tree at the beginning and after each operation.

2.3.3 Other Approximate methods

In previous section it was analyzed the use of alternative potential representation for approximating ID evaluation: *numerical trees*. There are, however, many other approaches:

- Sampling methods: these methods sample from the joint probability distribution of all the variables in the ID. However, for IDs having many variables, the state space of all variables grows exponentially, and the sample sizes required may be too large. Most of the research aims to reduce the sample sizes [6, 3].
- Limited memory influence diagrams (LMIDs): this approach [13] can be used for obtaining an approximate solution by assuming that you have less information at hand in the future, and your decision will therefore not be necessarily be optimal. Thus, no-forgetting assumption is relaxed.
- Branch-and-bound search: the main drawback of other methods is that they can waste computation in solving decision scenarios that have zero or are unreachable from any initial state by following an optimal decision policy. This drawback can be overcome by adopting a branch-and-bound approach to solve an influence diagram [24] that uses a search tree to represent all possible decision scenarios. This approach can use upper bounds on maximum utility to prune branches of the search tree that correspond to low-quality decisions that cannot be part of an optimal policy; it can also prune branches that have zero probability.

Chapter 3

Research

3.1 Introduction

The aim of the present work is to introduce a new kind of tree for potentials of an ID: *binary trees* (BT). In Section 3.2 basic concepts about BTs are described and how they can be used for ID evaluation. A previous work [4] describes how BTs can be used for BN inference. For that reason we have focused on the use of BTs for representing utility potentials. In Section 3.3 several similarity measures for pruning trees are described.

3.2 Binary Trees

A BT is similar to a NT. It is also a directed labelled tree, where each internal node is labelled with a variable, and each leaf is labelled with a non-negative real number. It also allows to represent a potential for a set of variables $\mathbf{X}_I$. But now, each internal node has always two outgoing arcs, and a variable can appear more than once labelling the nodes in the path from the root to a leaf node. Another difference is that, for an internal node labelled with X_i, the outgoing arcs can usually be labelled with more than one state of Ω_{X_i}. We denote $L_{lb(t)}$ and $L_{rb(t)}$ the labels (two subsets of $\Omega^t_{X_i}$) of the left and right branches of the node t. We denote then with t_l and t_r the children of t. Trees for encoding probability functions will be called *binary probability trees* (BPTs) and trees for utility functions will be called *binary utility trees* (BUTs) .

An advantage of BTs is that they allow to represent finer grain context-specific independences that those which can be represented with NTs. For example, Figure 3.1 shows three different representations for an utility potential: as a table in (a), as a NUT in (b) and as BUT in (c). When $A = a_1$, the potential always will take the value 30 independently of the value of B. Therefore, less space is needed for representing it as a tree since it can be

$U(A,B)$	b_1	b_2	b_3
a_1	30	30	30
a_2	45	45	20
a_3	25	25	50

(a)

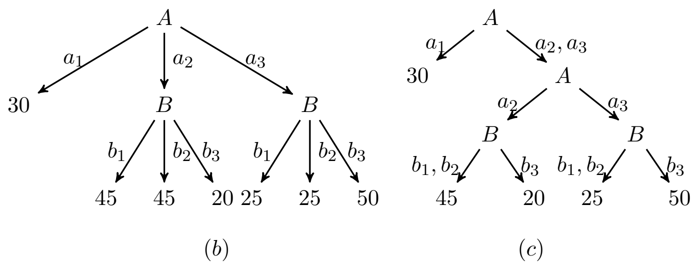


Figure 3.1: Utility potential represented as a table (a), as a NUT (b) and as BUT (c)

pruned. Besides, an additional pruning can be made for the BUT: when $A = a_2$ and $B \in \{b_1, b_2\}$, the potential always will be 45.

3.2.1 Building a Binary Utility Tree

Building a BUT is an optimization problem that consists on choosing the labels for each internal node (variables) and for each arc (states). The order for the labels will affect to the context-specific independences that are represented by the tree. In a previous work [4] a greedy algorithm was proposed in order to build a BPT from a given probability potential. The algorithm for building a BUT, which is quite similar, is described in this subsection.

The algorithm builds the BUT using a top-down approach choosing at each step a variable and two partitions of its states. The process begins with an initial $\mathcal{BUT}_0$ with only one node whose label L_t is the average of the values in the potential:

$$L_t = \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X_I}}} \psi(\mathbf{x}_I)/|\Omega_{\mathbf{x}_I}|$$

A greedy step is then applied successively until an exact BUT is obtained. At each step, a new $\mathcal{BUT}_{j+1}$ is generated from the previous one $\mathcal{BUT}_j$. This new tree is the result of expanding one of the leaf nodes t in $\mathcal{BUT}_j$ with a terminal tree (with t rooting the terminal tree, and t_l and t_r as children of t). The node t is labelled with one of the *candidate variables*. The set of available states $\Omega^t_{X_i}$ of the chosen candidate variable X_i are partitioned into two subsets, $\Omega^{t_l}_{X_i}$ and $\Omega^{t_r}_{X_i}$. Each subset labels one of the two outgoing arcs (left and right) of t. The two leaf nodes t_l and t_r in the new terminal tree are labelled with the average of ψ values of consistent with the states labelling the path from the root to the terminal node.

Trees built at every step can be considered as an approximation of the original potential. Therefore, it is required a distance to measure the goodness of the approximation represented by a $\mathcal{BUT}$ to a given potential ψ. The Euclidean distance is proposed for that purpose:

$$D(\psi, \mathcal{BUT}) = \sqrt{\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} (\psi(\mathbf{x}_i) - \mathcal{BUT}(\mathbf{x}_i))^2} \tag{3.1}$$

At each step, a variable and two state partitions must be chosen in order to maximize the *information gain*. It can be defined as the difference between the distance of two approximations.

It cannot be assured that optimal approximation is achieved using the Euclidean distance. In Section 3.3 other alternative similarity (or dissimilarity) measures for pruning will be considered.

Definition 7 (Information Gain)**.** Let ψ be the potential we are constructing and $\mathcal{BUT}_j(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_l}_{X_i})$ the tree resulting of expanding the leaf node t with the candidate variable X_i and a partition of its available states into sets $\Omega^{t_l}_{X_i}$ and $\Omega^{t_r}_{X_i}$. The *information gain* can be defined as:

$$I(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i}) \;=\; D(\psi, \mathcal{BUT}_j) \;-\; D(\psi, \mathcal{BUT}_j(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_l}_{X_i})) \tag{3.2}$$

In our experiments (Chapter 4) we did not check every possible partition of $\Omega^t_{X_i}$, because this would be a very time-consuming task. Assuming that the set of available states for X_i at node t are ordered, we will only check partitions into subsets with consecutive states. Thus, each expansion is defined by a variable and a splitting point.

3.2.2 Pruning a Binary Utility Tree

If the size of a BUT needs to be reduced, it can be pruned in order to get another one which approximates the potential. Pruning a BUT consists in replacing a terminal tree by the average value os its leaves.

Definition 8 (Pruning a terminal tree)**.** Let $\mathcal{BUT}$ be a binary utility tree encoding ψ, t the root of a terminal tree labelled with X_i, t_c and t_r its children, $\Omega^{t_l}_{X_i}$ and $\Omega^{t_r}_{X_i}$ the sets of states for left and right child respectively, $max(\psi)$ and $min(\psi)$ the maximum and minimum values in ψ, and ε a given threshold $\varepsilon \geq 0$, then the terminal tree rooted by X_i can be pruned if:

$$I(t, X_i, \Omega^{t_l}_{X_i}, \Omega^{t_r}_{X_i}) \leq \varepsilon \cdot (max(\psi) - min(\psi)) \tag{3.3}$$

The goal of pruning involves detecting leaves that can be replaced by one value without a great increment in the distance between the approximate and the exact tree.

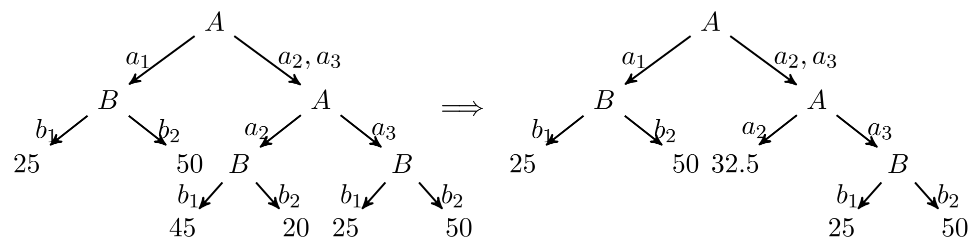


Figure 3.2: Example of pruning a terminal node in a BT

Figure 3.2 shows an example of pruning a terminal node in a BT, where node consistent with the configuration $A = \{a_2\}$ has been replaced by the average value of its children. The distance within the original potential is computed using Equation 3.1:

$$D(\psi, \mathcal{BUT}) = \sqrt{(45 - 32.5)^2 + (20 - 32.5)^2} = 17.678$$

In the completely expanded tree, the euclidean distance between the tree and the potential is 0. Therefore the information gain will be 17.678.

3.2.3 Operations with Binary Trees

Inference algorithms for IDs require five operations with potentials: those also used for BNs are *restriction, combination* and *sum-marginalization.* These operations for BTs were introduced at previous work [4]. The evaluation of IDs also requires additional operations: *max-marginalization* and *division* and the auxiliary operation called *max.* The contribution of this work is the adaptation of these additional operations for working directly with BTs.

The operation of *restriction* of a binary tree to a configuration $\mathbf{x}_J$ is trivial. If $\mathcal{BT}$ is a binary probability on $\mathbf{X}_I$ and $\mathbf{x}_J$ a configuration for the set of variables $\mathbf{X}_J$, $\mathbf{X}_J \subseteq \mathbf{X}_I$, $\mathcal{BT}^{R(\mathbf{x}_J)}$ ($\mathcal{BT}$ restricted to configuration $\mathbf{x}_J$). The algorithm for this operation consists in traversing the tree from root to leaves, replacing in $\mathcal{BT}$ every node t labelled with a variable X_k, $X_k \in \mathbf{X}_J$, with the subtree $\mathcal{BT}_k$ (children of t) consistent with the value of X_k in $\mathbf{x}_J$. The restriction operation is described at Algorithm 1.

```
input  : $t$ (root node of $\mathcal{BT}$); $X_j$ (variable to restrict); $S_{X_j}$ (set of states of $X_j$ to
         restrict)
output: The root of $\mathcal{BT}^{R(S_{X_j})}$
if $t$ is not a leaf node then
    if $L_t == X_j$ then
        Set $S^l_{X_j} = L_{lb(t)} \cap S_{X_j}$ and $S^r_{X_j} = L_{rb(t)} \cap S_{X_j}$;
        if $S^l_{X_j} == \emptyset$ then
            return Restrict($t_r$,$X_j$,$S^r_{X_j}$)
        end
        else if $S^r_{X_j} == \emptyset$ then
            return Restrict($t_l$,$X_j$,$S^l_{X_j}$)
        end
        else
            Set $L_{lb(t)} = S^l_{X_j}$, $L_{rb(t)} = S^r_{X_j}$ the new labels of the branches of $t$;
            Set Restrict($t_l$,$X_j$,$S^l_{X_j}$) as the new left child of $t$;
            Set Restrict($t_r$,$X_j$,$S^r_{X_j}$) as the new right child of $t$
        end
    end
    else
        Set Restrict($t_l$,$X_j$,$S_{X_j}$) as the new left child of $t$;
        Set Restrict($t_r$,$X_j$,$S_{X_j}$) as the new right child of $t$
    end
end
return $t$
```

Algorithm 1: Restrict

Figure 3.3 shows an example of restriction where a $\mathcal{BUT}$ has been restricted to the configuration $B = b1$. Notice that restriction can be applied to BPTs and BUTs.

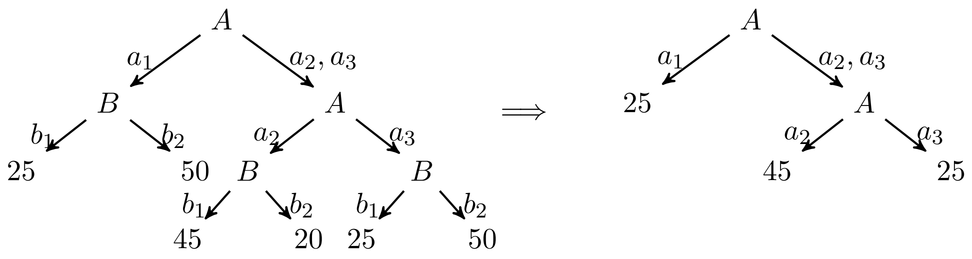


Figure 3.3: Restriction of a BUT

The *combination* of two probability trees $\mathcal{BT}_1$ and $\mathcal{BT}_2$, $\mathcal{BT}_1 \otimes \mathcal{BT}_2$, can be achieved with Algorithm 2. The inputs of the algorithm are the roots of $\mathcal{BT}_1$ and $\mathcal{BT}_2$. It returns the root node (t) of $\mathcal{BT}_1 \otimes \mathcal{BT}_2$.

```
input : $t1$ and $t2$ (root nodes of $\mathcal{BUT}_1$ and $\mathcal{BPT}_2$);
output: the root of $\mathcal{BUT} = \mathcal{BUT}_1 \otimes \mathcal{BPT}_2$
Build a new node $t$;
if $t$ is a leaf node then
    if $t2$ is a leaf node then
        $L_t = L_{t1}/L_{t_2}$
    end
    else
        Set $L_t = L_{t2}$ the label of $t$;
        Set $L_{lb(t)} = L_{lb(t2)}$ the label of left branch of $t$;
        Set $L_{rb(t)} = L_{rb(t2)}$ the label of right branch of $t$;
        Set Combine$(t1, t2_l)$ the left child of $t$;
        Set Combine$(t1, t2_r)$ the right child of $t$;
    end
end
else
    $X_i = L_{t1}$;
    Set $L_t = L_{t1}$ the label of $t$;
    Set $L_{lb(t)} = L_{lb(t1)}$ the label of left branch of $t$;
    Set $L_{rb(t)} = L_{rb(t1)}$ the label of right branch of $t$;
    Set Combine$(t1_l, \mathcal{BPT}_2^{R(X_i, L_{lb(t1)})})$ the left child of $t$;
    Set Combine$(t1_r, \mathcal{BPT}_2^{R(X_i, L_{rb(t1)})})$ the right child of $t$;
end
return $t$
```

Algorithm 2: Combine

Figure 3.4 shows an example of combination of a BUT with a BPT. During ID evaluations, there are combinations between a BUT and a BPT or between two BPTs.

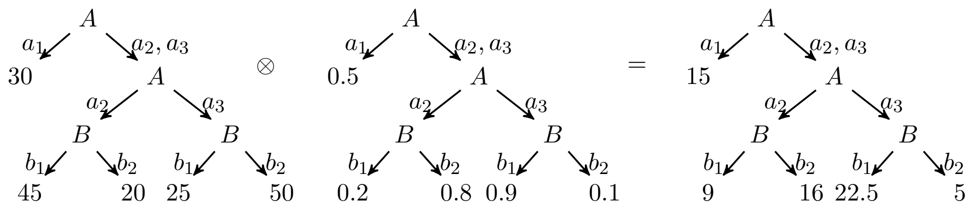


Figure 3.4: Combination of a BUT with a BPT

Sum-marginalization is used when a chance node is going to be removed. A variable is removed using sum-marginalization by adding up all its values. As it can bee seen in Algorithm 3, the removal of X_j using the operation sum-marginalization produces a new tree denoted by $\mathcal{BPT}^{\downarrow X_j}$.

```
input  : $t$ (root node of $\mathcal{BPT}$);
         $X_j$ (variable to remove)
output: the root of $\mathcal{BPT}^{\downarrow X_j}$
if $t$ is a leaf node then
    Build a new node $tn$;
    Set $L_{tn} = L_t$ the label of $tn$;
end
else
    if $L_t == X_j$ then
        $t1$ = sum-marginalize($t_l$, $X_j$);
        $t2$ = sum-marginalize($t_r$, $X_j$);
        $tn$ = sum($t1, t2$);
    end
    else
        Build a new node $tn$;
        Set $L_{tn} = L_t$;
        Set $L_{lb(tn)} = L_{lb(t)}$;
        Set $L_{lr(tn)} = L_{lr(t)}$;
        Set sum-marginalize($tl, X_j$) the left child of $tn$;
        Set sum-marginalize($tr, X_j$) the right child of $tn$;
    end
end
return $tn$;
```

Algorithm 3: sum-marginalization

Figure 3.5 shows an example of sum-marginalization. During ID evaluation, sum-marginalization is performed over BPTs.

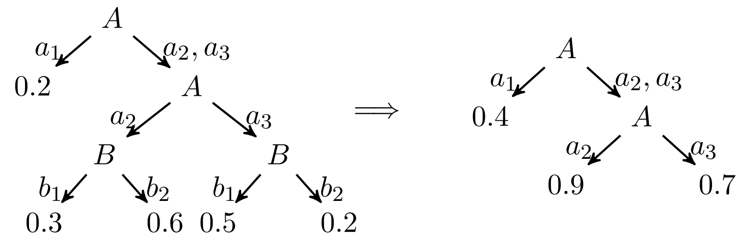


Figure 3.5: sum-marginalization of a BUT with respect to variable B

When solving IDs, chance nodes are removed using sum-marginalization (as in BNs) while decision nodes are removed with the max-marginalization. As it can bee seen in Algorithm 4, the removal of X_j using the operation max-marginalization produces a new tree denoted by $max_{X_j}\mathcal{BUT}$. This operation is similar to sum-marginalization, but instead of adding the values for X_j and a given configuration for $\mathbf{X}_I \backslash X_j$, it returns the maximum value.

```
input  : $t$ (root node of $\mathcal{BUT}$);
         $X_j$ (variable to remove)
output: the root of $max_{X_j}\mathcal{BUT}$
if $t$ is a leaf node then
    Build a new node $tn$;
    Set $L_{tn} = L_t$ the label of $tn$;
end
else
    if $L_t == X_j$ then
        $t1$ = max-marginalize($t_l$, $X_j$);
        $t2$ = max-marginalize($t_r$, $X_j$);
        $tn$ = max($t1, t2$);
    end
    else
        Build a new node $tn$;
        Set $L_{tn} = L_t$;
        Set $L_{lb(tn)} = L_{lb(t)}$;
        Set $L_{lr(tn)} = L_{lr(t)}$;
        Set max-marginalize($tl, X_j$) the left child of $tn$;
        Set max-marginalize($tr, X_j$) the right child of $tn$;
    end
end
return $tn$;
```

Algorithm 4: max-marginalization

Figure 3.6 shows the application of this operations on a tree in order to remove the variable B. The algorithm is recursively executed until a node labelled with the variable to be removed is found. When it happens, it max-marginalizes the left and right children trees and combines them using the *max* operation.

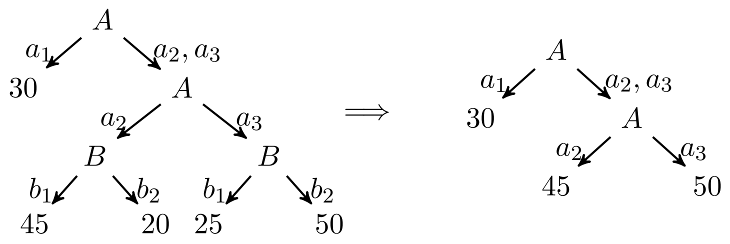


Figure 3.6: max-marginalization of a BUT with respect to variable B

The *max* operation returns a potential containing the maximum for each configuration of the variables involved (see Algorithm 5). Figure 3.7 shows an example of application of *max* operation over two BUTs.

```
input  : $t1$ and $t2$ (root nodes of $\mathcal{BUT}_1$ and $\mathcal{BUT}_2$);
output: the root of $max(\mathcal{BUT}_1, \mathcal{BUT}_2)$
Build a new node $t$;
if t1 is a leaf node then
    if t2 is a leaf node then
        if $t1 > t2$ then
            $L_t = L_{t1}$
        end
        else
            $L_t = L_{t2}$
        end
    end
    else
        Set $L_t = L_{t2}$;
        Set $L_{lb(t)} = L_{lb(t2)}$;
        Set $L_{rb(t)} = L_{rb(t2)}$;
        Set $\max(t1, t2_l)$ the left child of $t$;
        Set $\max(t1, t2_r)$ the right child of $t$;
    end
end
else
    $X_i = L_{t1}$;
    Set $L_t = L_{t1}$;
    Set $L_{lb(t)} = L_{lb(t1)}$;
    Set $L_{rb(t)} = L_{rb(t1)}$;
    Set $\max(t1_l, \mathcal{BUT}_2^{R(X_i, L_{lb(t1)})})$ the left child of $t$;
    Set $\max(t1_r, \mathcal{BUT}_2^{R(X_i, L_{rb(t1)})})$ the right child of $t$;
end
return $t$
```

Algorithm 5: max operation

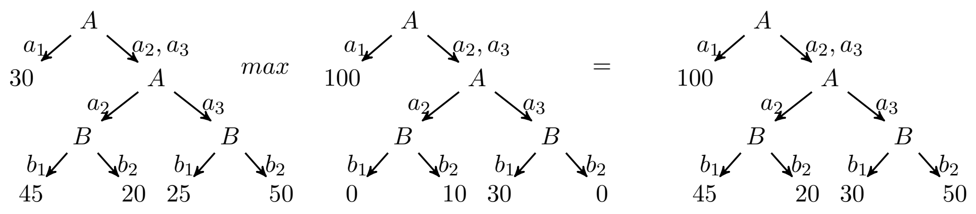


Figure 3.7: max operation between two BUTs

Another operation used for some ID inference algorithms is the division (see Algorithm 6). It is very similar to the *max* operation: instead of returning the maximum between two leaves of the same configuration, it returns the division of the values. Notice that division for ID evaluation is performed between a BUT and a BPT. An example is shown at Figure 3.8.

```
input  : $t1$ and $t2$ (root nodes of $\mathcal{BUT}_1$ and $\mathcal{BPT}_2$);
output: the root of $\mathcal{BUT} = \mathcal{BUT}_1/\mathcal{BPT}_2$
Build a new node $t$;
if $t$ is a leaf node then
    if $t2$ is a leaf node then
        $L_t = L_{t1}/L_{t_2}$
    end
    else
        Set $L_t = L_{t2}$ the label of $t$;
        Set $L_{lb(t)} = L_{lb(t2)}$ the label of left branch of $t$;
        Set $L_{rb(t)} = L_{rb(t2)}$ the label of right branch of $t$;
        Set divide$(t1, t2_l)$ the left child of $t$;
        Set divide$(t1, t2_r)$ the right child of $t$;
    end
end
else
    $X_i = L_{t1}$;
    Set $L_t = L_{t1}$ the label of $t$;
    Set $L_{lb(t)} = L_{lb(t1)}$ the label of left branch of $t$;
    Set $L_{rb(t)} = L_{rb(t1)}$ the label of right branch of $t$;
    Set divide$(t1_l, \mathcal{BPT}_2^{R(X_i, L_{lb(t1)})})$ the left child of $t$;
    Set divide$(t1_r, \mathcal{BPT}_2^{R(X_i, L_{rb(t1)})})$ the right child of $t$;
end
return $t$
```

Algorithm 6: division

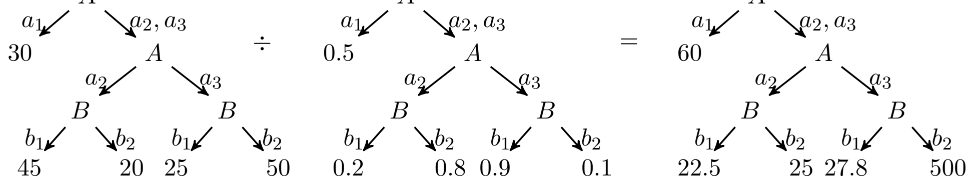


Figure 3.8: division between a BUT and a BPT

3.2.4 Influence Diagrams Inference

Inference algorithms for IDs such as Variable Elimination (VE) [11] can be easily adapted for working with BTs. The adaptations only needs using BTs and their related operations for computing. Here we propose to use VE algorithm using BTs. A pruning process can be performed to obtain smaller trees, reducing the computing time. This pruning is performed only with the initial utility potentials present in the ID. After that, the inference algorithm is the same.

3.3 Similarity Measures for Pruning

In previous section, it was proposed the Euclidean distance for measuring the similarity between an approximated BUT and the exact utility potential. However, it cannot be assured that optimal approximation is achieved using this distance. In this section several similarity (or dissimilarity) measures for pruning will be considered.

3.3.1 Proposed Similarity Measures

Minkowski distances

Given two vectors x and y, the Minkowski distances are defined with the expression $D(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{1/p}$. These are the standard metrics for geometrical problems. For $p = 2$ it is obtained the Euclidean distance, and its expression for measuring the distance between a tree $\mathcal{BUT}$ and a utility potential ψ is:

$$D_{EU}(\psi, \mathcal{BUT}) = \sqrt{\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} (\psi(\mathbf{x}_i) - \mathcal{BUT}(\mathbf{x}_i))^2} \tag{3.4}$$

The Euclidean distance can be normalized between $[0,1]$ using the following expression. This new distance will be called *Euclidean normalized*:

$$D_{NORM}(\psi, \mathcal{BUT}) = \frac{1}{1 + D_{EU}(\psi, \mathcal{BUT})} \tag{3.5}$$

It can also be normalized using the exponential function and it will be called *Euclidean exponential*:

$$D_{EXP}(\psi, \mathcal{BUT}) = e^{-D_{EU}(\psi, \mathcal{BUT})^2} \tag{3.6}$$

In the Euclidean space, iso-similarities are concentric hyper-spheres around a considered point (see Figure 3.9). Euclidean distance is translation invariant but scale variant.

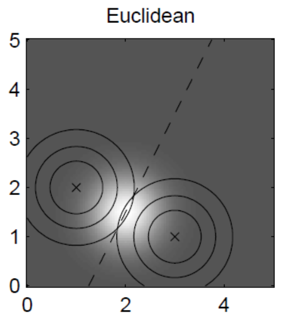


Figure 3.9: Iso-similarities in the Euclidean space

Cosine Measure

Similarity can also be defined by the cosine of the angle between two vectors [8]. The cosine distance between a tree $\mathcal{BUT}$ and a utility potential ψ is given by:

$$D_{COS}(\psi, \mathcal{BUT}) = \frac{\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} \psi(\mathbf{x}_i) \cdot \mathcal{BUT}(\mathbf{x}_i)}{\sqrt{\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} \psi(\mathbf{x}_i)^2} \cdot \sqrt{\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} \mathcal{BUT}(\mathbf{x}_i)^2}} \tag{3.7}$$

Cosine measure is defined in the interval $[0, 1]$. It is decreasing: similar points have distance 1, while the highest distance is 0. Cosine distance is translation variant but scale invariant (see Figure 3.10).

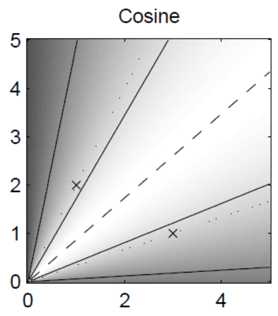


Figure 3.10: Iso-similarities in the Cosine space

Extended Jaccard coefficient

Another popular similarity measure is the Extended Jaccard coefficient [22]. Let ψ be an utility potential and a tree $\mathcal{BUT}$ approximating the potential. Then, the Extended Jaccard coefficient is defined as follows:

$$D_{JAC}(\psi, \mathcal{BUT}) = \frac{\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} \psi(\mathbf{x}_i) \cdot \mathcal{BUT}(\mathbf{x}_i)}{\left(\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} \psi(\mathbf{x}_i)^2\right) \cdot \left(\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} \mathcal{BUT}(\mathbf{x}_i)^2\right) - \sum_{\mathbf{x}_i \in \Omega_{\mathbf{X_I}}} \psi(\mathbf{x}_i) \cdot \mathcal{BUT}(\mathbf{x}_i)} \tag{3.8}$$

The Extended Jaccard coeficient is defined in the interval $[0, 1]$. It is decreasing: similar points have distance 1, while the highest distance is 0. The iso-similarities are non-concentric hyper-spheres (see Figure 3.11).

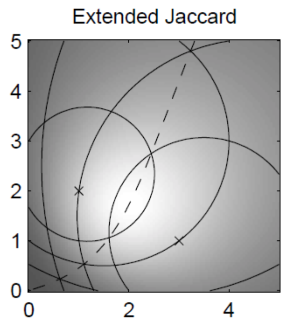


Figure 3.11: Iso-similarities in the Extended Jaccard space

Kullback Leibler divergence

Traditionaly, Kullback Leibler divergence [7] has been used for measuring the discrepancy between two probability distributions. It is given by the following expression:

$$D(\psi, \mathcal{BUT}) = \sum_{\mathbf{x_i} \in \Omega_{\mathbf{x}_I}} \psi(\mathbf{x}_I) log \frac{\psi(\mathbf{x}_I)}{\mathcal{BUT}(\mathbf{x}_I)} \tag{3.9}$$

Kullback Leibler divergence is a non-negative distance where the maximum similarity is given by the value 0. The minimum similarity is given by the value ∞.

Relative Percent difference

The relative percent difference is a measure for comparing two quantities similar to the relative error. The problem of the relative error is that when the exact value is 0, the error is ∞. This problem is solved using the relative percent difference since it is the division of the absolute error between the mean of both values. The expression for comparing an utility tree and a potential is given by the following expression:

$$D(\psi, \mathcal{BUT}) = \sum_{\mathbf{x_i} \in \Omega_{\mathbf{x}_I}} \left| \frac{\psi(\mathbf{x}_I) - \psi(\mathbf{x}_I)}{1/2(\mathcal{BUT}(\mathbf{x}_I) + \psi(\mathbf{x}_I))} \right| \tag{3.10}$$

The relative percent difference is a non negative increasing function. That is, the minimum distance is 0.

3.3.2 Adaptation of pruning condition

Depending on the measure used, the pruning conditions can change lightly. Some of the measures are normalized decreasing functions: Euclidean exponential, Euclidean normalized, Cosine and Extended Jaccard. For that reason, the maximum and minimum of the potential are not used at pruning condition. Moreover, the limit for pruning must be adapted to the decreasing condition:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq 1 - \varepsilon \tag{3.11}$$

Relative percent error is an increasing function which depends on the magnitude of the values compared. For that reason, the pruning condition is:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq \varepsilon \tag{3.12}$$

Finally, Kullback Leibler distance is decreasing and it is not normalized. Thus, the pruning condition will be the same that for Euclidean distance:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq \varepsilon \cdot (max(\psi) - min(\psi)) \tag{3.13}$$

Chapter 4

Experiments

4.1 Introduction

This chapter includes the experimental work. The main objective of the experimentation is to prove that BUTs offer better approximated solutions than NUTs when evaluating an ID. A secondary objective was to know which is the best similarity measure for pruning a BUT.

However, it is necessary to introduce first some concepts about *Multi-objective Optimization Problems* in order to be able to analyse the results. This introduction is done in Section 4.2. The set of IDs used for experimentation are detailed in Section 4.3. In Section 4.4 it is explained the procedure for the experimentation. Finally Section 4.6 shows the results and their analysis.

4.2 Multi-Objective Problems

In our problem and given an approximate potential tree, there are two objectives to be considered: minimization of error and size of the approximated potential tree. These two objectives can be controlled with the ε threshold for pruning: a low value for ε will produce large trees with a low error, while a high value for ε will produce small trees with a big error. Thus, it can be considered as a Multi-objective Optimization Problem (MOP) [12] with two objectives to be minimized. In this kind of problems, there is a set of objective functions that must be optimized. Hence, optimizing means finding such a solution with acceptable values for all the objectives.

Without additional information about the preferred objective and level of performance, it may exist several possible solutions. Moreover, the objectives considered are usually in conflict. In Multi-Objective Optimization problems, the set of acceptable solutions composes the Pareto set. This set contains all the non-dominated solutions.

In order to compare two solution sets, a quality measure is needed. In our experiments the Hyper-volume indicator was used [26]. It is an unary indicator that measures the area of the dominated portion of the space given a reference point r. It is defined in the interval $[0, 1]$, being 1 the optimal solution. For minimization problems (this is our case) the reference point is the maximum at every dimension.

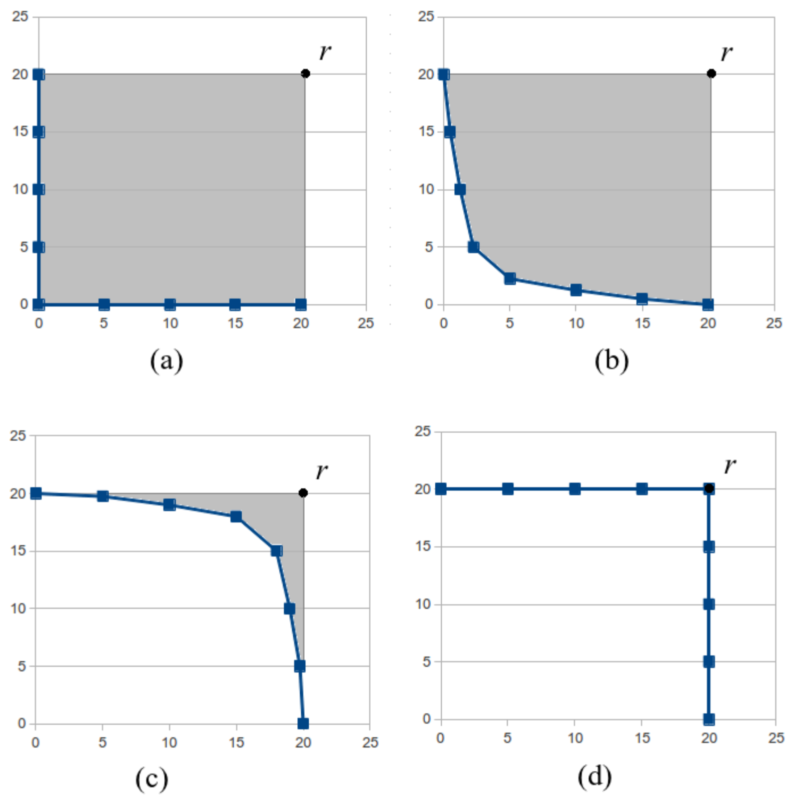


Figure 4.1: Four different solutions in a minimization problem: (a) optimal, (b) good, (c) bad, (d) worst.

Figure 4.1 shows 4 different solutions in a minimization problem with two objectives: (a) is the optimal solution, (b) is a good solution, (c) is a bad solution and (d) is the worst possible solution. The corresponding hyper-volume values are shown at Table 4.2.

	Hypervolume
(a) optimal	1
(b) good	0.8389
(c) bad	0.0720
(d) worst	0

Table 4.1: Hypervolume for solutions shown in Figure 4.1

4.3 Datasets

In order to perform the experiments, it was necessary a set of ID to evaluate. The desirables IDs were those which are complex enough to notice the improvements. However, it must to be possible to evaluate them since it was necessary to compare the approximate solutions with the exact ones.

The first ID used in the experiments was a real world ID used for the treatment of gastric NHL disease [1] with 3 decisions, 1 utility node and 17 chance nodes. This ID is shown in Figure 4.2.

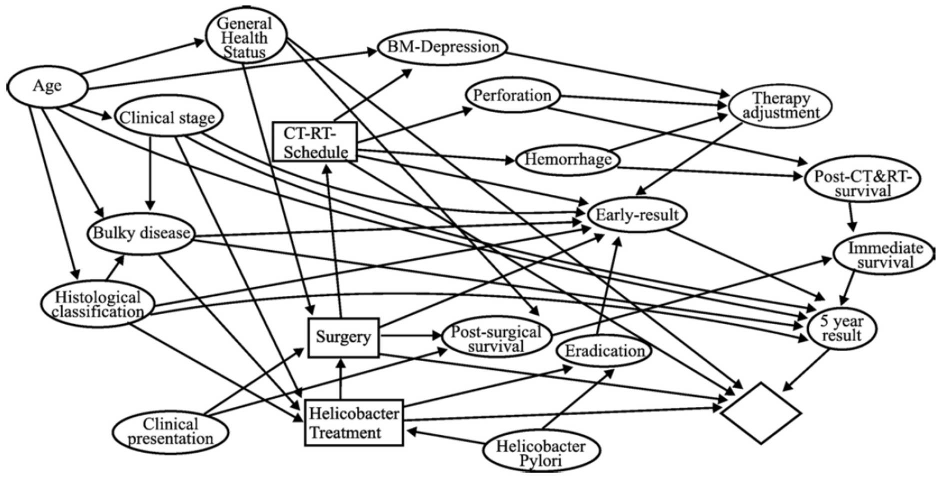


Figure 4.2: ID used for the treatment of gastric NHL disease

However, in order to obtain any conclusion, it was not enough with the NHL IDs. For that reason it was also used a set of 30 randomly generated IDs with 2 decisions, 1 utility node, and a random number of chance nodes (between 6 and 17). The utility function was created using a random biased number generator. Table 4.2 shows more details about each random ID.

chance nodes	decisions	utility nodes	links	utility parents	mean number of states	utility size
14	2	1	39	6	3.41	1600
10	2	1	21	2	3.62	12
12	2	1	31	6	2.73	648
9	2	1	22	2	3.17	10
11	2	1	25	2	3.86	10
9	2	1	23	3	2.83	27
9	2	1	24	3	4.25	100
12	2	1	33	4	2.4	36
9	2	1	25	4	2.75	48
9	2	1	28	6	2.33	144
11	2	1	30	3	4.43	120
16	2	1	33	2	3.32	12
14	2	1	37	4	2.94	24
10	2	1	25	2	1.85	4
9	2	1	24	3	1.83	8
15	2	1	40	3	1.89	8
15	2	1	39	6	2.72	648
10	2	1	26	4	2.15	24
14	2	1	40	6	3.41	6000
14	2	1	32	3	3.88	24
11	2	1	23	2	3.64	8
14	2	1	34	4	1.88	16
16	2	1	35	2	1.89	4
10	2	1	30	3	3.77	50
9	2	1	22	2	3.42	12
13	2	1	29	2	3.31	8
11	2	1	32	5	4.14	864
10	2	1	26	2	2.38	6
15	2	1	39	5	1.89	32
13	2	1	36	6	3.44	1080

Table 4.2: Characteristics of random generated IDs

4.4 Procedure

Variable Elimination (VE) was the inference algorithm employed in the test. In order to check the error produced by pruning the trees, utility trees were initially transformed with this operation. Probability trees were not pruned to focus on utility trees manipulation. The threshold ε used for pruning were ranged in the interval $[0, 1]$ (see Equation 3.3). The utility potentials analyzed to check the error were the initial ones present in the ID and the expected utility calculated for each decision. These are tested measuring their size and the *root-mean-square error* (RMSE) respect to the exact ones obtained with a 0 value for the threshold ε. The experimentation had two objectives:

- Prove that BUTs offer better approximated solutions than NUTs when evaluating an ID.
- Find out which is the best similarity measure for pruning a BUT.

For testing the improvements using BUTs, each ID was evaluated using NUTs and BUTs with different threshold values. For each evaluation, the size and the RMSE were measured. All the pairs (size, RMSE) for the same ID and kind of tree compose a solution set. For each solution set, the Pareto front and hyper-volume indicator were computed. Finally, a Willcoxon signed-rank test was performed with the hyper-volume values from random IDs. The null hypothesis was that there was not any difference in the hyper-volume obtained using NUTs or BUTs. The significance level for rejecting the null hypothesis was 5%.

In order to compare the different similarity measures explained at previous chapters, each ID was evaluated using only BTs and different measures and threshold values. Each similarity measure was compared with the Euclidean distance computing the hyper-volume values and performing a Willcoxon signed-rank test.

4.5 Implementation

All the experimentation was performed using the Elvira system [1], which is an open-source tool to construct model based decision support systems. The models supported are based on probabilistic uncertainty.

All the code necessary for the experimentation was developed in Java. In particular, it was implemented the classes related to BTs and the evaluation of IDs with the method of Variable Elimination. In addition, it was also implemented the code for computing the different similarity measures.

[1]http://leo.ugr.es/elvira/

4.6 Results

4.6.1 Improvements using Binary Trees

The experiments show that BUTs offer better approximated solutions than NUTs. The same error level will be achieved using BUTs of smaller size than the corresponding NUTs. This situation can be shown in Figure 4.3, which includes four different graphics representing RMSE (horizontal axis) against tree size (vertical axis) for initial utilities and policies for decision 0, 1 and 2 for the NHL ID. An additional advantage of BUTs is the higher number of different solutions obtained. This is due to the chance of a softer pruning operation on them: it can act only on a certain subset of states gathered in a given branch.

Table 4.3 shows the hyper-volume indicators obtained from the evaluation of NHL ID. Each row corresponds to each one of the utility potentials (initial utility and expected utilities), whereas each column corresponds to the kind of tree encoding the potential (NUT and BUT). It can be shown that the binary tree hyper-volume value (H_B) is always higher than the corresponding numerical tree hyper-volume value (H_N). Only for the initial utilities similar values are obtained. Moreover, for the last decision removed (Decision 0), the difference is more significant. Thus, better approximations are achieved using BUTs.

	H_N	H_B
Initial Utility	0.737	**0.795**
Decision 2	0.252	**0.829**
Decision 1	0.248	**0.991**
Decision 0	0	**0.915**

Table 4.3: Hyper-volume values for approximated utility trees at NHL Influence Diagram comparing NTs and BTs

The hyper-volume values for the random IDs are shown in Table 4.4. Each row corresponds to a different random ID whereas the columns indicate the kind of tree (NUT and BUT) and utility potential analyzed (initial utility and expected utilities). It can be observed that the BT hyper-volume (H_B) is usually higher than the corresponding NT hyper-volume (H_N).

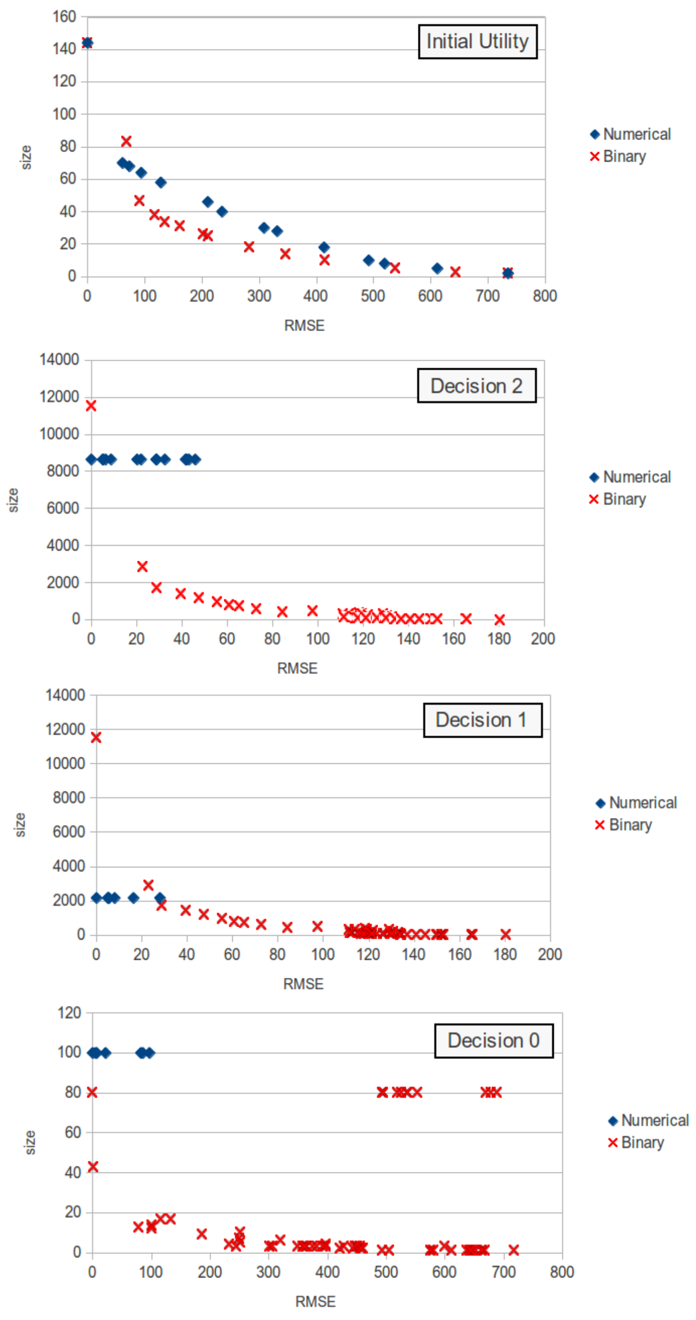


Figure 4.3: Results for approximating utility trees at NHL Influence Diagram.

The results for the Willcoxon sign-rank test comparing H_N and H_B obtained from random IDs evaluations are shown in Table 4.5. For each utility potential, it is shown the *p-value* and if the null hypothesis is rejected (NTs and BTs are not equal) with a significance level of 5%. In all the utility potentials, the null hypothesis is rejected. That is, the results for BUTs are better than those for NUTs.

Initial Utility		Decision 1		Decision 0	
H_N	H_B	H_N	H_B	H_N	H_B
0.514	**0.761**	0.854	**0.879**	0	**0.897**
0.314	**0.489**	0.231	**0.693**	0	**0.113**
0.794	**0.879**	0.302	**0.901**	0.475	**0.916**
0.22	**0.564**	0.379	**0.687**	**0.386**	0.0773
0.22	**0.565**	0.382	**0.688**	**0.385**	0.0786
0.397	**0.665**	0.164	**0.733**	0	**0.875**
0.568	**0.809**	0.751	**0.815**	0	**0.829**
0.747	**0.787**	0.82	**0.827**	**0.909**	0.806
0.508	**0.682**	0.642	**0.691**	**0.8**	0.764
0.47	**0.654**	0.569	**0.892**	0	**0.821**
0.205	**0.819**	0.501	**0.826**	0.809	**0.915**
0.319	**0.487**	0.231	**0.689**	0	**0.113**
0.577	**0.615**	**0.637**	0.571	0.735	**0.765**
0	**0**	0	**0**	0	**0**
0.461	**0.466**	0.371	**0.462**	0	**0.635**
0.462	**0.463**	**0.614**	0.559	**0.605**	0.527
0.579	**0.861**	0	**0.875**	0	**0.417**
0.584	**0.589**	0	**0.9**	0	**0.837**
0.607	**0.856**	**0.55**	0.529	0.574	**0.71**
0.57	**0.724**	0.638	**0.783**	0	**0.236**
0.233	**0.491**	0.396	**0.659**	**0.386**	0.0964
0.541	0.541	0.226	**0.656**	0	**0.735**
0	**0**	0	**0**	0	**0**
0.586	**0.719**	0	**0.743**	0	**0.921**
0.211	**0.521**	0.359	**0.681**	**0.38**	0.0632
0.236	**0.491**	0.398	**0.66**	**0.389**	0.0941
0.627	**0.83**	0.743	**0.954**	0	**0.864**
0.284	**0.359**	0.285	**0.358**	0	**0.146**
0.725	**0.725**	0.266	**0.888**	0	**0.91**
0.777	**0.901**	0.535	**0.878**	0	**0.794**

Table 4.4: Hyper-volume values for utility trees at each random ID comparing NTs and BTs

	p-value	rejected
Initial Utility	$4.23 \cdot 10^{-6}$	yes
Decision 1	$1.68 \cdot 10^{-6}$	yes
Decision 0	0.0033	yes

Table 4.5: Results for the Willcoxon sign-rank test to the results obtained with NTs and BTs

4.6.2 Comparison of Similarity Measures

In general, experiments show that it cannot be assured that the best results are always obtained using the Euclidean distance. Kullback Leibler, Cosine and Relative Percent difference also give good approximations for the utility potentials. By contrast, results using Euclidean normalized, Euclidean exponential and Extended Jaccard are not promising. The performance of the similarity measure depends on the type of ID.

Table 4.6 shows a comparison of the hyper-volumes of Euclidean distance (H_{EU}) and each proposed distance (H_{DIST}). The NHL ID was the ID evaluated. It can be observed that the hyper-volume value is always higher using Euclidean distance. However hyper-volumes obtained using Kullback Leibler, Cosine and Relative are quite similar.

	Initial Utility		Decision 2		Decision 1		Decision 0	
	H_{EU}	H_{DIST}	H_{EU}	H_{DIST}	H_{EU}	H_{DIST}	H_{EU}	H_{DIST}
Kullback Leibler	**0.792**	0.727	**0.853**	0.651	**0.991**	0.946	**0.894**	0.875
Euclidean normalized	**0.954**	0.841	**0.899**	0.419	**0.992**	0.285	**0.893**	0.245
Euclidean exponential	**0.955**	0	**0.827**	0	**0.991**	0.281	**0.894**	0.241
Cosine	**0.789**	0.571	**0.879**	0.755	**0.991**	0.927	**0.893**	0.856
Extended Jaccard	**0.953**	0	**0.829**	0	**0.991**	0.281	**0.893**	0.241
Percent	**0.794**	0.746	**0.83**	0.749	**0.991**	0.969	**0.895**	0.88

Table 4.6: Hyper-volume values for utility trees at NHL ID the use of Euclidean distance with other ones for pruning

Experiments using the set of random IDs show that results obtained using Euclidean distance are not always the best. Table 4.7 shows the hyper-volume values for experiments comparing Euclidean and Kullback Leibler distances. Each row correspond to a random ID. The hyper-volume for the Euclidean and Kullback Leibler distance are denoted respectively H_{EU} and H_{KB}.

Initial Utility		Decision 1		Decision 0	
H_{EU}	H_{KB}	H_{EU}	H_{KB}	H_{EU}	H_{KB}
0.489	**0.513**	**0.691**	0.583	0.112	**0.272**
0.849	0.823	**0.916**	0.868	**0.929**	0.9
0.566	**0.58**	0.688	**0.746**	0.0767	**0.388**
0.563	**0.579**	0.687	**0.747**	0.0766	**0.386**
0.657	**0.697**	**0.597**	0.555	**0.866**	0.785
0.803	**0.813**	**0.794**	0.789	0.767	**0.867**
0.544	**0.627**	0.823	**0.838**	0.809	**0.881**
0.679	**0.739**	0.697	**0.799**	0.796	**0.885**
0.655	**0.696**	**0.849**	0.824	**0.834**	0.778
0.815	**0.819**	**0.861**	0.843	**0.919**	0.917
0.487	**0.516**	**0.695**	0.581	0.114	**0.27**
0.617	**0.627**	**0.688**	0.682	0.766	**0.824**
0	**0.243**	0.285	**0.348**	0.181	**0.182**
0.461	**0.5**	**0.463**	0.461	**0.637**	0.634
0.461	**0.5**	0.559	**0.613**	0.525	**0.61**
0.859	0.831	**0.894**	0.863	0.435	**0.662**
0.587	**0.641**	0.848	**0.853**	0.837	**0.859**
0.85	0.822	**0.531**	0.48	**0.714**	0.458
0.721	**0.747**	0.814	**0.857**	0.392	**0.594**
0.49	**0.522**	0.659	**0.692**	0.0983	**0.386**
0.545	**0.578**	0.656	**0.825**	0.734	**0.736**
0	**0.243**	0.286	**0.346**	**0.182**	0.182
0.722	**0.766**	**0.743**	0.725	**0.926**	0.809
0.522	**0.566**	0.815	**0.833**	0.269	**0.435**
0.489	**0.524**	0.658	**0.688**	0.0961	**0.386**
0.831	0.825	**0.953**	0.793	0.866	**0.909**
0.359	**0.444**	0.36	**0.401**	0.148	**0.253**
0.727	**0.752**	**0.888**	0.836	0.911	**0.913**
0.901	0.882	**0.894**	0.745	**0.809**	0.622

Table 4.7: Hyper-volume values for utility trees at each random ID comparing the use of Euclidean and Kullback-leibler distance for pruning

Table 4.8 shows the corresponding Willcoxon test to the results shown in Table 4.7. Only for Decision 1, the null-hypothesis is not rejected. For the Initial Utility and Decision 0 it is rejected. Thus, taking into account the number of IDs where each measure is best, it can be assured that Kullback Leibler distance works better than Eculidean distance for Initial Utility and Decision 0.

	p-value	rejected	% H_{EU} wins	best
Initial Utility	0.0005	yes	20.69	KB
Decision 1	0.5172	no	51.72	equals
Decision 0	0.0267	yes	30.48	KB

Table 4.8: Results for the Willcoxon sign-rank test to the results obtained with Euclidean and Kullback Leibler distance

Initial Utility		Decision 1		Decision 0	
H_{EU}	H_{EUNORM}	H_{EU}	H_{EUNORM}	H_{EU}	H_{EUNORM}
0.49	0	**0.69**	0	**0.112**	0
0.875	0	**0.881**	0	**0.913**	0.225
0.564	0	**0.685**	0.0821	**0.0774**	0.0767
0.566	0	**0.69**	0.0824	0.077	**0.0785**
0.659	0	**0.596**	0.225	**0.863**	0.371
0.803	0	**0.796**	0.245	**0.768**	0.64
0.77	0	**0.825**	0.329	**0.809**	0.468
0.681	0.259	**0.69**	0.458	**0.765**	0.702
0.813	0.465	**0.853**	0.34	**0.817**	0.544
0.816	0	**0.825**	0.231	**0.917**	0.365
0.487	0	**0.691**	0	**0.112**	0
0.616	0	**0.572**	0.153	**0.767**	0.673
0	**0**	0	**0**	0	**0**
0.463	0	**0.465**	0	**0.635**	0
0.462	0	**0.557**	0.321	**0.526**	0.241
0.898	0.264	**0.885**	0.192	**0.429**	0
0.584	0	**0.844**	0.282	**0.838**	0.262
0.878	0	**0.531**	0.2	**0.714**	0.155
0.725	0.465	**0.781**	0.265	0.236	**0.238**
0.488	0	**0.662**	0.123	0.0955	**0.0969**
0.542	0	**0.654**	0.114	**0.736**	0.552
0	**0**	0	**0**	0	**0**
0.724	0	**0.744**	0.301	**0.93**	0.371
0.521	0	**0.687**	0.0673	0.0624	**0.0638**
0.488	0	**0.661**	0.122	0.0965	**0.0969**
0.831	0	**0.955**	0.274	**0.862**	0.217
0.358	0	**0.36**	0	**0.148**	0
0.726	0	**0.887**	0.0677	**0.91**	0.228
0.902	0	**0.877**	0.15	**0.794**	0.138

Table 4.9: Hyper-volume values for utility trees at each random ID comparing the use of Euclidean and Euclidean Normalized distance for pruning

Table 4.9 shows the hyper-volume values comparing Euclidean (H_{EU}) and Euclidean Normalized distance (H_{EUNORM}) . In the great majority of IDs, the Euclidean distance is much better. Besides, for many cases, the value 0, which is the worst one, it is obtained using Euclidean distance. Thus, it is not necessary performing a Willcoxon test. The same conclusion can be deduced for Euclidean Exponential distance, whose hyper-volume values are shown at Table 4.10.

Initial Utility		Decision 1		Decision 0	
H_{EU}	H_{EUEXP}	H_{EU}	H_{EUEXP}	H_{EU}	H_{EUEXP}
0.49	0	**0.69**	0	**0.112**	0
0.875	0	**0.881**	0	**0.913**	0.225
0.564	0	**0.685**	0.0821	**0.0774**	0.0767
0.566	0	**0.69**	0.0824	0.077	**0.0785**
0.659	0	**0.596**	0.225	**0.863**	0.371
0.803	0	**0.796**	0.245	**0.768**	0.64
0.77	0	**0.825**	0.329	**0.809**	0.468
0.681	0.259	**0.69**	0.458	**0.765**	0.702
0.813	0.465	**0.853**	0.34	**0.817**	0.544
0.816	0	**0.825**	0.231	**0.917**	0.365
0.487	0	**0.691**	0	**0.112**	0
0.616	0	**0.572**	0.153	**0.767**	0.673
0	**0**	0	**0**	0	**0**
0.463	0	**0.465**	0	**0.635**	0
0.462	0	**0.557**	0.321	**0.526**	0.241
0.898	0.264	**0.885**	0.192	**0.429**	0
0.584	0	**0.844**	0.282	**0.838**	0.262
0.878	0	**0.531**	0.2	**0.714**	0.155
0.725	0.465	**0.781**	0.265	0.236	**0.238**
0.488	0	**0.662**	0.123	0.0955	**0.0969**
0.542	0	**0.654**	0.114	**0.736**	0.552
0	**0**	0	**0**	0	**0**
0.724	0	**0.744**	0.301	**0.93**	0.371
0.521	0	**0.687**	0.0673	0.0624	**0.0638**
0.488	0	**0.661**	0.122	0.0965	**0.0969**
0.831	0	**0.955**	0.274	**0.862**	0.217
0.358	0	**0.36**	0	**0.148**	0
0.726	0	**0.887**	0.0677	**0.91**	0.228
0.902	0	**0.877**	0.15	**0.794**	0.138

Table 4.10: Hyper-volume values for utility trees at each random ID comparing the use of Euclidean and Euclidean Exponential distance for pruning

Table 4.11 shows the hyper-volume values comparing Euclidean distance (H_{EU}) and Cosine measure (H_{COS}). For most of the ID, hyper-volume values are quite similar.

Initial Utility		Decision 1		Decision 0	
H_{EU}	H_{COS}	H_{EU}	H_{COS}	H_{EU}	H_{COS}
0.486	**0.499**	**0.692**	0.603	0.112	**0.323**
0.879	0.864	**0.934**	0.856	**0.919**	0.802
0.563	**0.577**	0.686	**0.687**	0.0771	**0.386**
0.564	**0.58**	**0.689**	0.686	0.0784	**0.384**
0.664	**0.679**	**0.736**	0.616	**0.924**	0.918
0.808	0.777	**0.814**	0.769	**0.826**	0.772
0.787	**0.8**	0.826	**0.839**	0.809	**0.879**
0.683	**0.733**	**0.691**	0.645	0.808	**0.856**
0.811	0.811	**0.891**	0.849	**0.839**	0.781
0.819	0.801	**0.843**	0.783	**0.915**	0.906
0.488	**0.501**	**0.692**	0.601	0.113	**0.323**
0.617	**0.651**	0.57	**0.624**	0.819	**0.913**
0	**0.241**	0.284	**0.348**	0.182	**0.362**
0.461	**0.512**	0.462	**0.472**	0.633	**0.635**
0.46	**0.511**	0.555	**0.589**	0.529	**0.606**
0.904	0.89	**0.891**	0.8	0.429	**0.672**
0.587	**0.649**	**0.902**	0.839	0.839	**0.853**
0.878	0.855	0.571	**0.572**	**0.72**	0.335
0.723	**0.729**	0.781	**0.802**	0.236	**0.462**
0.488	**0.509**	**0.659**	0.583	0.0974	**0.389**
0.544	**0.587**	**0.659**	0.431	0.736	**0.745**
0	**0.241**	0.283	**0.345**	0.181	**0.362**
0.718	**0.736**	**0.744**	0.714	**0.929**	0.874
0.525	**0.562**	0.686	**0.754**	0.0644	**0.377**
0.491	**0.507**	**0.658**	0.581	0.0962	**0.389**
0.832	0.825	**0.955**	0.871	**0.866**	0.807
0.361	**0.43**	0.359	**0.4**	0.146	**0.254**
0.725	**0.745**	**0.948**	0.929	0.93	**0.935**
0.902	0.893	**0.886**	0.69	**0.805**	0.673

Table 4.11: Hyper-volume values for utility trees at each random ID comparing the use of Euclidean and Cosine distance for pruning

	p-value	rejected	% H_{EU} wins	best
Initial Utility	0.0017	yes	24.14	COS
Decision 1	0.0169	yes	65.52	EU
Decision 0	0.0179	yes	34.48	COS

Table 4.12: Results for the Willcoxon sign-rank test to the results obtained with Euclidean distance and Cosine measure

A priori, it cannot be assured that any distance is best that the other one. Thus, it is necessary to perform a Willcoxon test (see Table 4.11). The p-values obtained are lower than 0.05. That means that the null hypothesis is rejected. So, one measure is better than the other one.

In order to know which measure is better, it will be taken into account the number of wins of each measure and the Willcoxon test. Thus, it can be deduced that Cosine measure is best for Initial Utility and Decision 0. However, for Decision 1 works better the Euclidean distance.

Table 4.13 shows the hyper-volume values for experiments with Extended Jaccard distance. In most of the cases the Euclidean Hyper-volume is higher. In fact, in many of the IDs, an hyper-volume equal to 0 is obtained with the Extended Jaccard distance. This is the worst situation possible. The reason for that could be that the prune using Extended Jaccard distance is very aggressive. That is, the prune usually takes place in the higher nodes of the trees. Therefore trees are always very small but the error obtained is very high. Some of the hyper-volumes corresponding to the Euclidean distance are also equal to 0. The reason for that is that utility trees are so small that they are not usually pruned. So the number of solutions of the Pareto Set is very small.

Initial Utility		Decision 1		Decision 0	
H_{EU}	H_{JAC}	H_{EU}	H_{JAC}	H_{EU}	H_{JAC}
0.488	0	**0.692**	0	**0.111**	0
0.878	0	**0.901**	0	**0.917**	0.224
0.564	0	**0.686**	0.0814	**0.0782**	0.0772
0.564	0	**0.687**	0.0812	0.0765	**0.0782**
0.662	0	**0.733**	0.0845	**0.876**	0.278
0.809	0	**0.814**	0.247	**0.827**	0.258
0.785	0	**0.825**	0.329	**0.809**	0.47
0.683	0	**0.688**	0.271	**0.763**	0.226
0.81	0	**0.889**	0.338	**0.819**	0.545
0.818	0	**0.825**	0.231	**0.916**	0.37
0.488	0	**0.691**	0	**0.112**	0
0.617	0	**0.573**	0.152	**0.766**	0.67
0	**0**	0	**0**	0	**0**
0.462	0	**0.465**	0	**0.633**	0
0.465	0	**0.557**	0.324	**0.528**	0.241
0.902	0	**0.889**	0	**0.426**	0
0.584	0	**0.902**	0.283	**0.837**	0.261
0.876	0	**0.529**	0.201	**0.71**	0.159
0.724	0	**0.782**	0.267	0.238	**0.239**
0.49	0	**0.658**	0.122	0.0953	**0.0961**
0.542	0	**0.656**	0.113	**0.736**	0.471
0	**0**	0	**0**	0	**0**
0.724	0	**0.743**	0	**0.932**	0
0.523	0	**0.685**	0.0682	0.0618	**0.0635**
0.491	0	**0.658**	0.123	**0.0981**	0.0978
0.831	0	**0.954**	0.272	**0.864**	0.214
0.357	0	**0.36**	0	**0.148**	0
0.725	0	**0.888**	0.0672	**0.912**	0.12
0.902	0	**0.878**	0.149	**0.793**	0.138

Table 4.13: Hyper-volume values for utility trees at each random ID comparing the use of Euclidean and Extended Jaccard distance for pruning

Finally, table 4.14 shows the hyper-volume values for experiments with Relative percent distance. In this case, it is not clear which measure offer best results, so a Willcoxon test was performed (see Table 4.15). Taking into account this test and the number of wins of each measure, similar results were obtained for Initial Utility and Decision 1. However for Decision 0, better results were obtained with the Relative Percent distance.

Initial Utility		Decision 1		Decision 0	
H_{EU}	H_{PER}	H_{EU}	H_{PER}	H_{EU}	H_{PER}
0.486	0.484	0.691	**0.693**	0.112	**0.165**
0.877	0.859	**0.953**	0.919	**0.936**	0.909
0.566	0.53	0.793	**0.822**	0.245	**0.432**
0.561	0.529	0.793	**0.826**	0.246	**0.431**
0.665	**0.677**	0.737	**0.814**	**0.92**	0.9
0.807	0.785	**0.814**	0.809	**0.825**	0.814
0.785	**0.796**	0.824	**0.836**	0.836	**0.851**
0.683	**0.743**	0.765	**0.82**	0.845	**0.9**
0.658	**0.694**	**0.889**	0.824	**0.845**	0.844
0.818	0.783	**0.824**	0.802	0.919	**0.925**
0.487	0.48	**0.693**	0.692	0.112	**0.167**
0.616	**0.641**	**0.693**	0.668	0.767	**0.881**
0	**0**	0	**0**	0	**0**
0.465	**0.509**	0.463	**0.471**	0.634	**0.635**
0.465	**0.504**	0.608	**0.61**	0.545	**0.547**
0.862	**0.867**	**0.901**	0.858	0.433	**0.661**
0.582	**0.632**	**0.929**	0.902	0.855	**0.879**
0.851	0.829	0.529	**0.537**	**0.713**	0.558
0.721	**0.728**	0.814	**0.815**	0.29	**0.522**
0.486	**0.508**	0.66	**0.689**	0.0988	**0.388**
0.544	**0.587**	0.811	**0.854**	0.733	**0.84**
0	**0**	0	**0**	0	**0**
0.723	**0.734**	0.745	**0.75**	**0.922**	0.897
0.523	0.52	0.683	**0.726**	0.0625	**0.375**
0.49	**0.51**	0.66	**0.688**	0.0974	**0.389**
0.833	0.825	**0.955**	0.863	**0.863**	0.797
0.357	**0.36**	**0.361**	0.359	**0.146**	0.146
0.727	**0.755**	0.929	**0.931**	0.923	**0.928**
0.9	0.887	**0.879**	0.839	**0.795**	0.731

Table 4.14: Hyper-volume values for utility trees at each random ID comparing the use of Euclidean and Extended Percent difference for pruning

	p-value	rejected	% H_{EU} wins	best
Initial Utility	0.2205	no	44.83	equals
Decision 1	0.7186	no	44.83	equals
Decision 0	0.0345	yes	34.48	PER

Table 4.15: Results for the Willcoxon sign-rank test to the results obtained with Euclidean distance and Relative percent distance

Table 4.16 shows the comparisons between each measure and the Euclidean distance. Each column correspond to an utility potential analized and each row to a measure. If the result of the comparison is that similar results are obtained with both measures, the entry of the table will be *equals*. If the Euclidean distance is better, the entry will be *EU*. However, if the Euclidean is worst, the entry will be *KB, EUNORM, EUEXP, COS, JAC* or *PER*. Analizing the results, it can be deduced that most promising measures for random IDs are Kullback Leibler, Cosine and Relative Percent.

	Initial Utility	Decision 1	Decision 0
Kullback Leibler	KB	equals	KB
Euclidean normalized	EU	EU	EU
Euclidean exponential	EU	EU	EU
Cosine	Cos	EU	Cos
Extended Jaccard	EU	EU	EU
Percent	equals	equals	PER

Table 4.16: Summary of the comparison of each similarity measure with the Euclidean distance

.

Chapter 5

Conclusions, Future Work and Publications

In this work, it has been introduced a new type of utility potential representation: BUTs. The experiments showed that BUTs offer better approximated solutions than NUTs. The same error level will be achieved with a BUT of smaller size than the corresponding NUT. As regards as future directions of research, we can study the behaviour of BTs using other inference algorithms apart from VE: *Arc Reversal* [21], *Lazy propagation* [14], etc. Finally another direction of research could be the integration of restrictions with BTs. That will allow the treatment of asymmetric decision problems.

Concerning to the prune of the trees, experiments showed that similar results were obtained using Euclidean, Kullback-Leibler, Cosine and Relative Percent difference. However, further research will be make in order to know which are the features of the IDs that make more convenient one measure or another one.

Most of the content in this work conforms the content of the article *Approximate Inference in Influence Diagrams using Binary Trees* sent to the European workshop *Probabilistic Graphical Models 2012.* This article is still under review. Another article about probabilistic graphical models was published by the author of this work in a national workshop [18].

Index

Bibliography

[1] C. Bielza, J.A. Fernández del Pozo, and P.J.F. Lucas. Explaining clinical decisions by extracting regularity patterns. *Decision Support Systems*, 44(2):397–408, 2008.

[2] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the 12th International Conference on Uncertainty in AI*, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

[3] A. Cano, M. Gómez, and S. Moral. A forward–backward Monte Carlo method for solving influence diagrams. *International journal of approximate reasoning*, 42(1):119–135, 2006.

[4] A. Cano, M. Gómez-Olmedo, and S. Moral. Approximate inference in Bayesian networks using binary probability trees. *International Journal of Approximate Reasoning*, 52(1):49–62, 2011.

[5] A. Cano, S. Moral, and A. Salmerón. Penniless propagation in join trees. *International Journal of Intelligent Systems*, 15(11):1027–1059, 2000.

[6] J.M. Charnes and P.P. Shenoy. Multistage Monte Carlo method for solving influence diagrams using local computation. *Management Science*, pages 405–418, 2004.

[7] T.M. Cover, J.A. Thomas, J. Wiley, et al. *Elements of information theory*, volume 6. Wiley Online Library, 1991.

[8] I.S. Dhillon and D.S. Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1):143–175, 2001.

[9] M. Gómez and A. Cano. Applying numerical trees to evaluate asymmetric decision problems. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 196–207, 2003.

[10] F. Jensen, F.V. Jensen, and S.L. Dittmer. From influence diagrams to junction trees. In *Proceedings of the Tenth international conference on*

Uncertainty in artificial intelligence, pages 367–373. Morgan Kaufmann Publishers Inc., 1994.

[11] F.V. Jensen and T.D. Nielsen. *Bayesian networks and decision graphs.* Springer Verlag, 2007.

[12] Y. Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):397–415, 2008.

[13] S.L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, pages 1235–1251, 2001.

[14] A.L. Madsen and F.V. Jensen. Lazy evaluation of symmetric Bayesian decision problems. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 382–390. Morgan Kaufmann Publishers Inc., 1999.

[15] A.L. Madsen and F.V. Jensen. Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1-2):203–245, 2004.

[16] S.M. Olmsted. Representing and solving decision problems. *Dissertation Abstracts International Part B: Science and Engineering[DISS. ABST. INT. PT. B- SCI. & ENG.],*, 45(3), 1984.

[17] J.R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[18] Rafael Cabañas de Paz, M. Julia Flores, Jesus Martinez-Gomez. Dynamic bayesian networks for gesture recognition. *Workshop de Agentes Físicos - WAF 2011*, 2011.

[19] H. Raiffa. Decision analysis: introductory lectures on choices under uncertainty. 1968.

[20] A. Salmerón, A. Cano, and S. Moral. Importance sampling in Bayesian networks using probability trees. *Computational Satistics & Data Analysis*, 34(4):387–413, 2000.

[21] R.D. Shachter. Evaluating influence diagrams. *Operations research*, pages 871–882, 1986.

[22] Alexander Strehl, Er Strehl, and Joydeep Ghosh. Value-based customer grouping from large retail data-sets. In *In Proceedings of the SPIE Conference on Data Mining and Knowledge Discovery*, pages 33–42, 2000.

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

[24] C. Yuan, X. Wu, and E. Hansen. Solving multistage influence diagrams using branch-and-bound search. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 691–700, 2010.

[25] N.L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research 5*, 5:301–328, 1996.

[26] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Evolutionary Multi-Criterion Optimization*, pages 862–876. Springer, 2007.