Synthesis of strategies in influence diagrams

Manuel Luque and **Manuel Arias** and **Francisco J. Díez**
Dept. Artificial Intelligence, UNED
Juan del Rosal, 16
28040 Madrid, Spain
{mluque,marias,fjdiez}@dia.uned.es

Abstract

Influence diagrams (IDs) are a powerful tool for representing and solving decision problems under uncertainty. The objective of evaluating an ID is to compute the expected utility and an optimal strategy, which consists of a policy for each decision. Every policy is usually represented as a table containing a column for each decision scenario, i.e., for each configuration of the variables on which it depends. The no-forgetting assumption, which implies that the decision maker always remembers all past observations and decisions, makes the policies grow exponentially with the number of variables in the ID. For human experts it is very difficult to understand the strategy contained in huge policy tables, not only for their size, but also because the vast majority of columns correspond to suboptimal or impossible scenarios and are hence irrelevant. This makes it difficult to extract the rules of action, to debug the model, and to convince the experts that the recommendations of the ID are reasonable. In this paper, we propose a method that presents the strategy in the form of a compact tree. It has been implemented in OpenMarkov, an open-source software tool for probabilistic graphical models. This facility was essential when evaluating an influence diagram for the mediastinal staging of non-small cell lung cancer; the optimal strategy, whose biggest policy table contained more than 15,000 columns, was synthesized into a tree of only 5 leaves.

1 INTRODUCTION

Influence diagrams (IDs) are a graphical framework for representing and solving uncertain decision problems (Howard and Matheson, 1984; Bielza et al., 2010). The evaluation of an ID consists in obtaining the expected utility and an optimal strategy, formed by a policy for each decision in the model. Every policy is usually represented by a table containing a column for each decision scenario, i.e., for each configuration of the variables whose values are known when making the decision. For example, the ID in Figure 1 contains two decision nodes, drawn as squares: whether to do a test (T) and whether to apply a therapy (D). It has two chance nodes, drawn as circles that represent the presence of the disease (X) and the result of the test (Y). The domains of these variables are $\{+t, \neg t\}$, $\{+d, \neg d\}$, $\{+x, \neg x\}$, $\{+y, \neg y, np\}$ respectively, where $+y$ denotes a positive result of the test, $\neg y$ a negative result, and np means that the test has not been performed. It also contains two utility nodes, drawn as hexagons, which represent the cost and benefit of the therapy (U_1) and the cost of the test (U_2). Figure 2b shows the optimal policy for the decision D in the form of a table that contains a column for each configuration of T and Y, the variables known when making decision D, as shown in OpenMarkov, an open-source tool for probabilistic graphical models.[1] The two lower rows correspond to the two values for D: $+d$ (apply the therapy) or $\neg d$ (no therapy). They represent a probability distribution for D given each configuration of T and Y. When there is one optimal option, its probability is 1; in OpenMarkov that cell is colored in red. When there is a tie between two or more optimal options, all of them are assigned the same probability. The optimal policy for decision T is shown in Figure 2a. The optimal strategy consists of these two policy tables.

This way of representing the strategy faces several problems. First, the no-forgetting assumption, which implies that the decision maker always remembers all past observations and decisions, makes the policy tables grow exponentially with the number of variables known when making them. In our example, the policy for D in Figure 2b contains $3 \times 2 = 6$ columns. Second, many of the columns in the table may correspond to scenarios that are impossible due to structural constraints of the problem. Thus, when we decide not to do the test ($T = \neg t$), the test results

[1] See `www.openmarkov.org`.

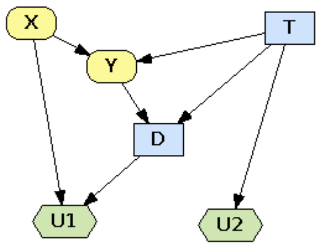
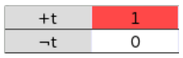
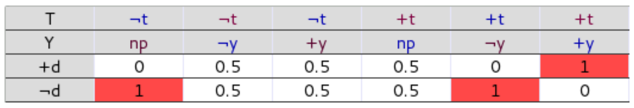


Figure 1: An ID for the decide-test problem.



+t	1
¬t	0

(a) Policy for T.



T	¬t	¬t	¬t	+t	+t	+t
Y	np	¬y	+y	np	¬y	+y
+d	0	0.5	0.5	0.5	0	1
¬d	1	0.5	0.5	0.5	1	0

(b) Policy for D.

Figure 2: Optimal policy tables for the ID in Figure 1. Different numerical parameters in the ID may lead to different optimal policies, but the structure of the tables would always be the same.

$Y = +y$ and $Y = \neg y$ are impossible; when we decide to do it, the value $Y = np$ (test not performed) is impossible. Therefore, three of the columns in Figure 2b are irrelevant, regardless of the numerical parameters of the problem (sensitivity, specificity, costs, etc.) and should be removed from the table. Third, depending on the values of the numerical parameters, some columns may correspond to suboptimal scenarios. This is the case for the column $\{T = \neg t, Y = np\}$ when the optimal decision is to do the test and for the columns $\{T = +t, Y = +y\}$ and $\{T = +t, Y = \neg t\}$ when the optimal decision is to do it. Therefore, in this example only one or two out of the 6 columns in Figure 2b are relevant for the optimal strategy.

In general, the proportion of irrelevant columns grows exponentially with the number of variables. For example, MEDIASTINET is an influence diagram for lung cancer (it is explained in further detail in Sec. 3.3.2); the policy table for the last decision in this ID contains 15,552 columns, but only 5 of them make part for the optimal strategy; even worse, the relevant columns cannot be detected by just analyzing this table, because they depend on the probabilities of its configurations and on the optimal policies for previous decisions. The problem is that standard algorithms for the evaluation of IDs only return the policy tables, without being able to extract the relevant columns nor to combine the optimal choices for different decisions into a single representation that can be easily understood by human experts.

For this reason, we have developed an algorithm that builds a strategy tree by using dynamic programming; more precisely, it extends the variable elimination algorithm described by Jensen and Nielsen (2007) and refined by Luque and Díez (2010). The version proposed in this paper builds the strategy tree at the same time it operates with the probability and utility potentials to compute the maximum expected utility.

The remainder of this paper is structured as follows. Section 2 presents the definitions of influence diagrams and their evaluation. Section 3 presents the algorithm for computing the tree representation of the strategy. Section 4 contains the conclusions and proposes some lines for future research.

2 FOUNDATIONS

2.1 INFLUENCE DIAGRAMS

In this paper a capital letter (X) denotes a variable and a lower-case letter (x) a generic value of the corresponding variable. A bold capital letter ($\mathbf{X}$) denotes a set of variables and a bold lower-case letter ($\mathbf{x}$) a configuration of them. When there is an arc $X \rightarrow Y$ in a graph, we say that X is a parent of Y and Y is a child of X. $Pa(X)$ is the set of parents of X and $pa(X)$ is a configuration of $Pa(X)$.

Influence diagrams (Howard and Matheson, 1984) are a framework for representing and solving probabilistic decision problems. Formally, an *influence diagram* (ID) consists of an acyclic directed graph having three disjoint sets of nodes: decisions ($\mathbf{V}_D$), chance nodes ($\mathbf{V}_C$) and utility nodes ($\mathbf{V}_U$). Decision nodes represent the actions that are under the direct control of the decision maker; they are represented graphically as squares or rectangles. Chance nodes represent uncertain events; they are drawn as ovals or rounded rectangles. Utility nodes quantify the decision maker's preferences and are depicted as diamonds or hexagons. In medical IDs, utility nodes represent health outcomes (quality of life, morbidity, mortality...), and sometimes economic costs. The parents of a node can only be of type chance or decision, i.e., utility nodes have no children.

An ID contains three types of arcs, depending on the type of node they go into. Arcs into chance nodes represent probabilistic dependencies. Arcs into decision nodes represent availability of information or temporal precedence between decisions; thus, an arc from a node X into a decision node D means that the value taken by X is known when making the decision. Arcs into utility nodes indicate the domain of the associated utility functions.

We assume that a directed path connects all the decision nodes, which induces a total ordering of the decisions, $\{D_1, D_2, \ldots, D_n\}$. This order partitions $\mathbf{V}_C$ into a collection of disjoint subsets, $\{\mathbf{C}_0, \mathbf{C}_1, \ldots, \mathbf{C}_n\}$, and induces a *partial order* $\prec$ in $\mathbf{V}_C \cup \mathbf{V}_D$:

$$\mathbf{C}_0 \prec \{D_1\} \prec \mathbf{C}_1 \prec \ldots \prec \{D_n\} \prec \mathbf{C}_n \,. \qquad (1)$$

The variables known to the decision maker when deciding on D_j are called *informational predecessors* of D_j and denoted *iPred*(D_j). We assume the *no-forgetting* hypothesis, which states that the decision maker remembers all previous decisions and observations. In particular, *iPred*(D_j) is the set of variables that occurs before D_j under $\prec$: *iPred*$(D_j) = \mathbf{C}_0 \cup \{D_1\} \cup \mathbf{C}_1 \cup \ldots \cup \{D_{j-1}\} \cup \mathbf{C}_{j-1}$.

The quantitative information that defines an ID is given by: (1) a conditional probability $P(X|pa(X))$ assigned to each chance node X for each configuration of its parents, $pa(X)$, and (2) a potential $\psi_U(pa(U))$ assigned to each utility node U, which maps each configuration of its parents onto a real number. (A *potential* is a real-valued function over a set of variables.) When there are several utility nodes in the ID, we assume that there is an implicit sum among them.

2.2 EVALUATION OF IDs

2.2.1 Policies and strategies

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

A *strategy* Δ for an ID is a set of policies, one for each decision, $\{P_D \mid D \in \mathbf{V}_D\}$. If every policy in the strategy Δ is deterministic, then Δ is said to be *deterministic*; otherwise Δ is *stochastic*.

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

$$P_\Delta(\mathbf{v}_C, \mathbf{v}_D) = P(\mathbf{v}_C \mid \mathbf{v}_D) \prod_{D \in \mathbf{V}_D} P_D(d|\textit{iPred}(D)) \; . \tag{2}$$

where $P(\mathbf{v}_C \mid \mathbf{v}_D)$ is the product of the conditional probabilities:

$$P(\mathbf{v}_C \mid \mathbf{v}_D) = \prod_{X \in \mathbf{V}_C} P(x|pa(X)) \; .$$

We define the *expected utility under the strategy* Δ, denoted by EU(Δ), as:

$$\text{EU}(\Delta) = \sum_{\mathbf{v}_C} \sum_{\mathbf{v}_D} P_\Delta(\mathbf{v}_C, \mathbf{v}_D)\, \psi(\mathbf{v}_C, \mathbf{v}_D). \tag{3}$$

where $\psi(\mathbf{v}_C, \mathbf{v}_D)$ is the sum of the utilities:

$$\psi(\mathbf{v}_C, \mathbf{v}_D) = \sum_{U \in \mathbf{V}_U} \psi_U(pa(X)) \; .$$

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

$$\Delta_{opt} = \arg\max_{\Delta \in \boldsymbol{\Delta}} \text{EU}(\Delta), \tag{4}$$

where $\boldsymbol{\Delta}$ is the set of all the strategies for the ID. Each policy in an optimal strategy is said to be an *optimal policy*. The *maximum expected utility* (*MEU*) is the expected utility under an optimal strategy.

The evaluation of an ID consists in finding the *MEU* and an optimal strategy. It can be proved (Cowell et al., 1999) that

$$\textit{MEU} = \sum_{\mathbf{c}_0} \max_{d_1} \ldots \max_{d_n} \sum_{\mathbf{c}_n} P(\mathbf{v}_C \mid \mathbf{v}_D)\, \psi(\mathbf{v}_C, \mathbf{v}_D) \; . \tag{5}$$

The optimal policy for decision D_i is (in the case of a tie, any of the values of D_i that maximize that expression can be arbitrarily chosen):

$$\delta_{D_i}(\text{iPred}(D_i)) = \arg\max_{d_i \in D_i} \sum_{\mathbf{c}_i} \max_{d_{i+1}} \ldots \sum_{\mathbf{c}_{n-1}} \max_{d_n} \sum_{\mathbf{c}_n} P(\mathbf{v}_C \mid \mathbf{v}_D)\, \psi(\mathbf{v}_C, \mathbf{v}_D) \; . \tag{6}$$

2.2.2 Variable elimination

One of the algorithms for evaluating IDs is variable elimination with division of potentials (Jensen and Nielsen, 2007). It operates on the set of probability potentials $\boldsymbol{\Phi}$ and the set utility potentials $\boldsymbol{\Psi}$ of the ID by eliminating the variables in the opposite order of $\prec$: first each of the variables in $\mathbf{C}_n$, then D_n, then those in $\mathbf{C}_{n-1}$, then D_{n-1}, etc., as indicated in line 5 of Algorithm 1. It returns the *MEU* and an optimal strategy, in the form of a set of policies, $\boldsymbol{\Delta}$—see also (Luque and Díez, 2010) for a more detailed explanation and justification of this algorithm.

3 SYNTHESIZING THE STRATEGY

3.1 DEFINITION OF STRATEGY TREE

In this section we show that, instead of representing the strategy as a set of policy tables, it is more efficient to use a *strategy tree*, which by definition has the following properties:

1. it has three types of nodes: chance, decision and *nil*;
2. every non-terminal node X represents a decision or a chance variable of the ID (but several chance nodes in the tree may correspond to the same node in the ID);
3. every terminal node is *nil* and its parent is a decision;
4. each decision node has exactly one child;
5. the ancestors of a decision node D in the tree are informational predecessors of D in the ID; and
6. each arc outgoing from a node X is labeled with a non-empty subset of states of X, and each state labels at most one arc.

```
1  Function variableElimination(ID):
       Result: $\{MEU, \mathbf{\Delta}\}$
2      $\mathbf{\Phi} \leftarrow$ probability potentials in the ID;
3      $\mathbf{\Psi} \leftarrow$ utility potentials in the ID;
4      $\mathbf{\Delta} \leftarrow \varnothing$ ;   // strategy (set of policies)
5      $\mathbf{V} \leftarrow$ ordered array of $\{\mathbf{V}_C \cup \mathbf{V}_D\}$;
6      while $\mathbf{V}$ is not empty do
7          $X \leftarrow$ first variable in $\mathbf{V}$;
8          // take out the potentials that depend on $X$
9          $\mathbf{\Phi}_X \leftarrow \{\phi \in \mathbf{\Phi} \mid X \in dom(\phi)\}$;
10         $\mathbf{\Phi} \leftarrow \mathbf{\Phi} \setminus \mathbf{\Phi_X}$;
11         $\mathbf{\Psi}_X \leftarrow \{\psi \in \mathbf{\Psi} \mid X \in dom(\psi)\}$;
12         $\mathbf{\Psi} \leftarrow \mathbf{\Psi} \setminus \mathbf{\Psi_X}$;
13         // compute the joint probability and the total
14         // utility for the potentials that depend on $X$
15         $\phi_{\text{joint}} \leftarrow \prod_{\phi \in \mathbf{\Phi}_X} \phi$;
16         $\psi_{\text{total}} \leftarrow \sum_{\psi \in \mathbf{\Psi}_X} \psi$;
17         if $X \in \mathbf{V_C}$ then
18             $\phi_{\text{marg}} \leftarrow \sum_x \phi_{\text{joint}}$;
19             $\phi_{\text{cond}} \leftarrow \phi_{\text{joint}} / \phi_{\text{marg}}$;
20             $\psi_{\text{new}} \leftarrow \sum_x \phi_{\text{cond}} \cdot \psi_{\text{total}}$;
21         else   // $X$ is a decision
22             // then $\phi_{\text{joint}}$ does not depend on $X$
23             $\phi_{\text{marg}} \leftarrow \text{project}_x\ \phi_{\text{joint}}$;
24             $\psi_{\text{new}} \leftarrow \max_x \psi_{\text{total}}$;
25             // find the optimal policy for this decision
26             $\delta_X \leftarrow \arg\max_x \psi_{\text{total}}$;
27             $\mathbf{\Delta} \leftarrow \mathbf{\Delta} \cup \{\delta_X\}$;
28         // store the new potentials
29         $\mathbf{\Phi} \leftarrow \mathbf{\Phi} \cup \{\phi_{\text{marg}}\}$
30         $\mathbf{\Psi} \leftarrow \mathbf{\Psi} \cup \{\psi_{\text{new}}\}$
31     $\psi_{\text{final}} \leftarrow \sum_{\psi \in \mathbf{\Psi}_X} \psi$
32     // $\psi_{\text{final}}$ does not depend on any variable;
33     // it contains just one value
34     $MEU \leftarrow$ first value of $\psi_{\text{final}}$
35     return {MEU, $\mathbf{\Delta}$}
```

Algorithm 1: Variable elimination with division of potentials.

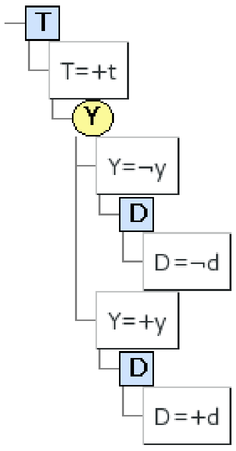


Figure 3: A strategy tree for the ID in Figure 1 as shown in OpenMarkov. Terminal nodes are not displayed because they convey no information. The size and structure of the optimal strategy depends on the numerical parameters of the ID.

Figure 3 shows a strategy tree for the ID in Figure 1.

3.2 ALGORITHM FOR BUILDING THE STRATEGY TREE

In the original variable elimination method (Algorithm 1) each probability potential assigns a real number to each configuration of its variables, $\phi(\mathbf{x}) \in \mathbb{R}$, and the same holds for utility potentials, $\psi(\mathbf{x}) \in \mathbb{R}$. However, the modified version that we present in this paper operates with *extended utility potentials*, such that $\psi^*(\mathbf{x}) = (u, s) \in \mathbb{R} \times \mathcal{S}$, where $\mathcal{S}$ is the space of strategies (in the form of trees). We denote by $u(\psi^*(\mathbf{x}))$ and $s(\psi^*(\mathbf{x}))$ the utility and the strategy of ψ^*, respectively, for configuration $\mathbf{x}$. We describe now the modifications that must be introduced in the algorithm. We do not discuss the computation of utilities because it is the same as in the original version.

3.2.1 Initialization

In line 3 of Algorithm 1 the utility potentials must be extended by assigning to each configuration $\mathbf{x}$ the strategy tree that only contains one node, *nil*. The utility does not change: $u(\psi^*(\mathbf{x})) = \psi(\mathbf{x})$.

3.2.2 Elimination of a decision

The elimination of decision D is determined by the potential $\psi^*_{\text{total}}(d, \mathbf{y})$—cf. line 24 in Algorithm 1—where $\mathbf{Y}$ is the set of variables that coexist with D in at least one potential. In the same way as the utility is computed by taking the highest utility and discarding the others, the strategy $s(\psi^*_{\text{new}}(\mathbf{y}))$ is built by making D the root of the tree and drawing one branch labeled with $x = \arg\max_{x'} u(\psi^*_{\text{total}}(x', \mathbf{y}))$. For example, when eliminating

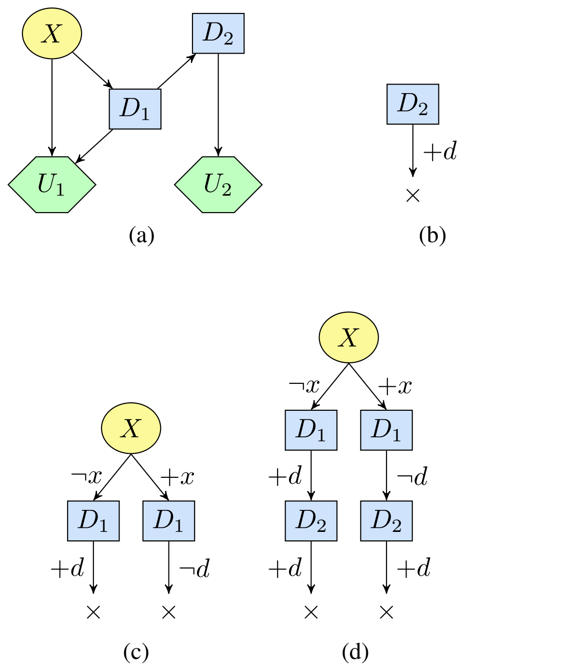


Figure 4: (a) An influence diagram with two decisions; the order is $X \prec D_1 \prec D_2$. (b) Strategy in the potential ψ_2^* obtained when eliminating D_2. (c) Strategy in the potential ψ_1^* obtained after eliminating D_1 and X. (d) Strategy in the potential $[\psi_1^* + \psi_2^*](x)$ computed at line 31 in Algorithm 1; it is built by replacing each *nil* node in (c) with the tree in (b).

D_2 for the ID in Figure 4b, $\mathbf{Y} = \varnothing$ and $s(\psi^*_{\text{new}}(\blacklozenge))$ is the tree shown in Figure 4b—"$\blacklozenge$" is the only configuration of the empty set.

In the case of a tie between two or more states of X, it is possible to label the branch with all of them instead of randomly selecting one; users should be informed that when a branch outgoing from a decision is labeled with several states it means that they are all optimal and it is indifferent to choose any of them.

Please note that in the modified version of Algorithm 1 lines 26 and 27 should be removed because now it is not necessary to build a policy table for any decision.

3.2.3 Elimination of a chance variable

The elimination of a chance variable X is performed by multiplying a conditional probability $\phi_{\text{cond}}(x|\mathbf{y})$ and an extended utility potential $\psi^*_{\text{total}}(x, \mathbf{y})$ and then marginalizing for X in order to obtain $s(\psi_{\text{new}}(\mathbf{y}))$—cf. line 20 in Algorithm 1—where $\mathbf{Y}$ is the set of variables coexisting with X in at least one potential. In the most simple case, the strategy for each configuration $\mathbf{y}$ is built by making X the root of the tree and adding a branch for each state x and copying $s(\psi^*_{\text{total}}(x, \mathbf{y}))$ in that branch. For example, when eliminating X for the ID in Figure 4b, the two decisions have already been eliminated and $\mathbf{Y} = \varnothing$; $s(\psi^*_{\text{new}}(\blacklozenge))$ is the tree shown in Figure 4c.

However, it is often possible to build a simplified tree by applying these rules:

1. When no decision has been eliminated yet, $s(\psi^*_{\text{total}}(x, \mathbf{y})) = nil$ for all x. In this case, it is more efficient to make $s(\psi_{\text{new}}(\mathbf{y})) = nil$, because it does not make sense to include in the strategy tree a variable that is not an informational predecessor of any decision. This is the reason why the strategy in Figure 3 does not contain any node for variable X (*Disease*), whose value is never known.

2. When $\phi_{\text{cond}}(x|\mathbf{y}) = 0$, the value x must not generate a branch. For example, the node Y in Figure 3 has no outgoing branch for np because when the test is done the probability of "test not performed" is 0. This way the algorithm avoids adding branches corresponding to impossible scenarios.

3. When $\phi_{\text{cond}}(x|\mathbf{y}) = 1$ for a certain value x, X must not make part of the strategy—at least for configuration $\mathbf{y}$—and we should just make $s(\psi_{\text{new}}(\mathbf{y})) = s(\psi^*_{\text{total}}(x, \mathbf{y}))$. The reason is that a variable that can only take one value in a certain scenario does not contribute any information and would be unnecessary in the strategy tree. For example, if the optimal decision for T in the ID of Figure 1 were $\neg t$ (not to do the test) the probability of $Y = np$ (test not performed) would be 1 and it would not make sense to include Y and its only branch, np, in the strategy.

4. When there are two states x_1 and x_2 such that $s(\psi^*_{\text{total}}(x_1, \mathbf{y})) = s(\psi^*_{\text{total}}(x_2, \mathbf{y}))$, both of them must be put in the same branch (instead of having two branches with the same strategy).

3.2.4 Sum of extended utility potentials

In lines 16 and 31 the modified version of Algorithm 1 has to sum extended utility potentials. Let $\psi_1^*(\mathbf{x}_1)$ and $\psi_2^*(\mathbf{x}_2)$ be two such potentials, $\mathbf{x}$ a configuration of $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$, $s_1 = s(\psi_1^*(\mathbf{x}^{\downarrow \mathbf{X}_1}))$—where $\mathbf{x}^{\downarrow \mathbf{X}_1}$ is the projection of $\mathbf{x}$ onto $\mathbf{X}_1$, i.e., the configuration of $\mathbf{X}_1$ compatible with $\mathbf{x}$—and $s_2 = s(\psi_2^*(\mathbf{x}^{\downarrow X_2}))$. The strategy $s([\psi_1 + \psi_2](\mathbf{x}))$ is the result of connecting s_1 and s_2 by adding arcs from the leaves of the one to the root of the other, as shown in Figure 4d.

3.2.5 Termination

In the traditional version of variable elimination, ψ_{final} is obtained after eliminating all the variables (line 31). Hence, we have $MEU = \psi_{\text{final}}(\blacklozenge)$. Analogously, in the modified version we have $MEU = u(\psi^*_{\text{final}}(\blacklozenge))$ and the optimal strategy is $s(\psi^*_{\text{final}}(\blacklozenge))$.

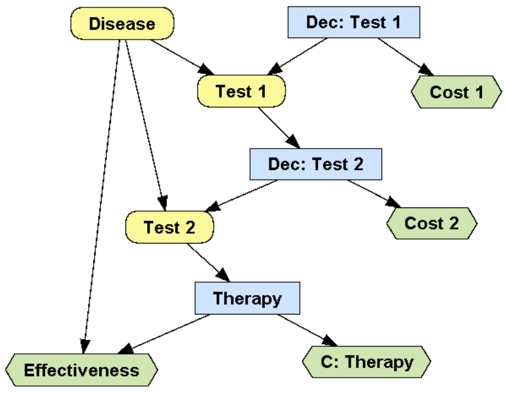


Figure 5: An ID for a hypothetical medical problem.

The interpretation of the tree returned by the algorithm is straightforward by reading each chance node as an "if" statement. For example, the strategy shown in Figure 3 can be read as follows: "do the test; if the result is positive, apply the therapy; if it is negative, do not apply it".

3.3 MORE COMPLEX EXAMPLES

3.3.1 AN ID WITH TWO TESTS

Let us extend the above example by adding a second test, such as the result of the first (when performed) is available before deciding whether to do the second, as shown in Figure 1. There are two therapies, incompatible with each other, which implies that there is only one decision about therapy, with three possible values: *therapy 1*, *therapy 2*, and *no therapy*.

Each decision about a test has two possible values (to do it or not) and the result of each test has three possible values (*positive*, *negative*, and *not performed*). Therefore, the policy table for *Therapy* has $2 \times 3 \times 2 \times 3 = 36$ columns, regardless of the numerical parameters of the model (sensitivities, specificities, costs, etc.). However, the optimal strategy does not depend on the parameters. For example, just by multiplying the three costs by a common factor—i.e., keeping the proportion among them—we may obtain 6 different strategies; four of them are shown in Figure 6. When the costs are very high, the optimal strategy is not to do any test nor to apply any therapy (Fig. 6a); in this case, the strategy tree contains just one node. In other cases the optimal strategy tree has up to 23 nodes, with 4 leaves. These trees show the optimal strategy much more clearly than the set of three policy tables, having 1, 6, and 36 columns respectively; we should note that even though this is a very small ID, the proportion of irrelevant columns in the last table ranges from 88% to 97%, depending on the numerical parameters.

3.3.2 MEDIASTINET

MEDIASTINET (Luque et al., 2016a) is an ID for the mediastinal staging of non-small cell lung cancer, which consists in determining whether there is metastasis in the mediastinum (this is represented by node *N2_N3* in Fig. 7), because it is the key factor for selecting a therapy for this type of cancer. Initially a CT scan is performed to all patients; the result is represented by the node *CT_scan*. Then the doctor can perform different diagnostic tests—*TBNA*, *PET*, *EBUS*, *EUS*, and mediastinoscopy (*MED*)— and then decides which *Treatment* to apply. Therefore, the ID contains 8 chance variables and 5 decisions; the number of columns in their respective policy tables is 2, 12, 72, 1,296, and 15,552. The number of relevant columns is 2 (100%), 3 (25%), 3 (4.2%), 5 (0.39%), and 5 (0.032%, i.e., 1 in every 3,110).

Given that the algorithm presented in this paper has been incorporated into the most recent version of OpenMarkov, we invite the reader to try it as follows: download the ID MEDIASTINET,[2] open it with OpenMarkov, compile it (with the thunder button), accepting the default values to convert it into a unicriterion problem, right-click on the last decision and have a look at the optimal policy table. Then click the S button and check that the strategy tree only contains 5 leaves, as shown in Figure 8.

4 CONCLUSIONS AND FUTURE WORK

A problem of representing the optimal strategy as a set of policy tables is that their size grows exponentially with the number of informational predecessors for each decision. The first attempt to alleviate it was to transform each policy table into a list-based representation (Fernández del Pozo et al., 2005), but this approach is unsatisfactory because those lists can still be very large and because the issue is not how to compact the whole table, but how to present the few relevant columns intuitively, which cannot not be found out by examining each single policy.

The approach presented in this paper is completely different: it represents each strategy as a tree (instead of a set of tables) and uses an extended version of the variable elimination algorithm (Jensen and Nielsen, 2007) that builds the tree at the same time as it computes the maximum expected utility. Other inference algorithms, such as arc reversal (Olmsted, 1983), can be adapted in the same way.

Our algorithm has been incorporated to OpenMarkov, an open-source tool for probabilistic graphical models. In this paper we have applied it to two toy examples and to MEDIASTINET, an influence diagram for a real medical problem (Luque et al., 2016a), whose policy table for the last decision contained more than 15,000 columns; in contrast,

[2] `www.probmodelxml.org/networks/id/ID-mediastinet-ce.pgmx`.

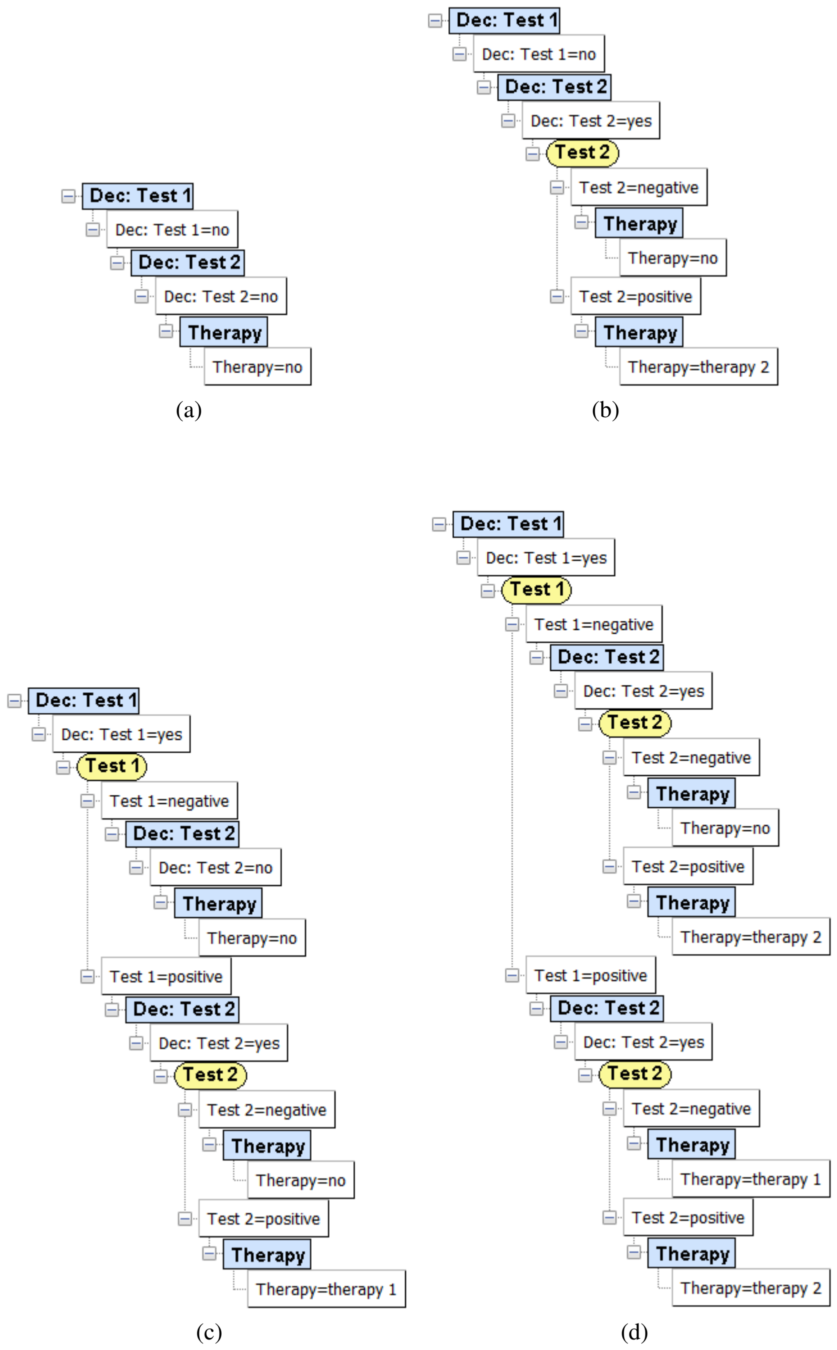


Figure 6: Optimal strategy trees for different assignments of costs for the ID in Figure 5.

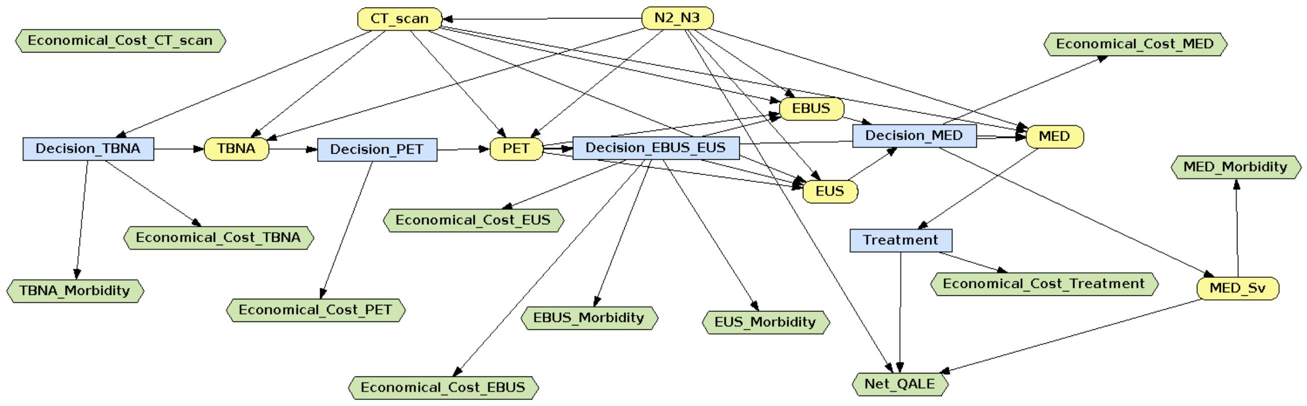


Figure 7: Influence diagram of MEDIASTINET.

the strategy tree for this ID contains only 5 leaves, which proves that only 0.03% of the columns in that table are relevant.

A limitation of the construction of the strategy tree is that in the worst case the number of leaves grows exponentially with the number of chance variables that are informational predecessors of the last decision—previous decisions do not increase the number of leaves because each decision node in the tree has just one outgoing branch—but, as we have seen in the examples, the tree is usually much smaller.

An open line for future research is how to reduce the size of the strategy tree, because there are some problems for which the resulting tree, even though much smaller than the policy tables, is still large—for instance, in the ID Arthronet.[3] In some cases reordering the chance variables between consecutive decisions might lead to significant reductions. Another approach would be to apply coalescence (Olmsted, 1983), i.e., to collapse the subtrees that appear several times in the tree. For example, in Figure 8 there are two identical subtrees having *Decision_PET* as the root; collapsing them would transform the tree into an acyclic directed graph with only 3 leaves.

Another research topic is to adapt this method to other representation frameworks for uncertain decision problems, such as decision analysis networks (DANs), an alternative to IDs for asymmetric decision problems (Díez et al., 2012), and Markov IDs (Díez et al., 2017), an extension of IDs especially designed for cost-effective analysis in health economics.

We are interested in evaluating the proposed algorithm on a large set of models. But, given that there are very few real-world non-small IDs, we will have to find a method for randomly generating IDs whose structure is similar to those we would expect to find in real-world problems. One possibility for addressing this issue would be to use parametrized templates, as in (Luque et al., 2016b).

Acknowledgments

This work has been supported by the Spanish Government under grants TIN2009-09158, PI13/02446, and TIN2016-77206-R, and co-financed by the European Regional Development Fund. We wish to thank the members of our research group for many fruitful discussions. We also thank all developers of OpenMarkov for providing a basis for our implementation.

[3] `www.probmodelxml.org/networks/id/ID-arthronet-ce.pgmx`.

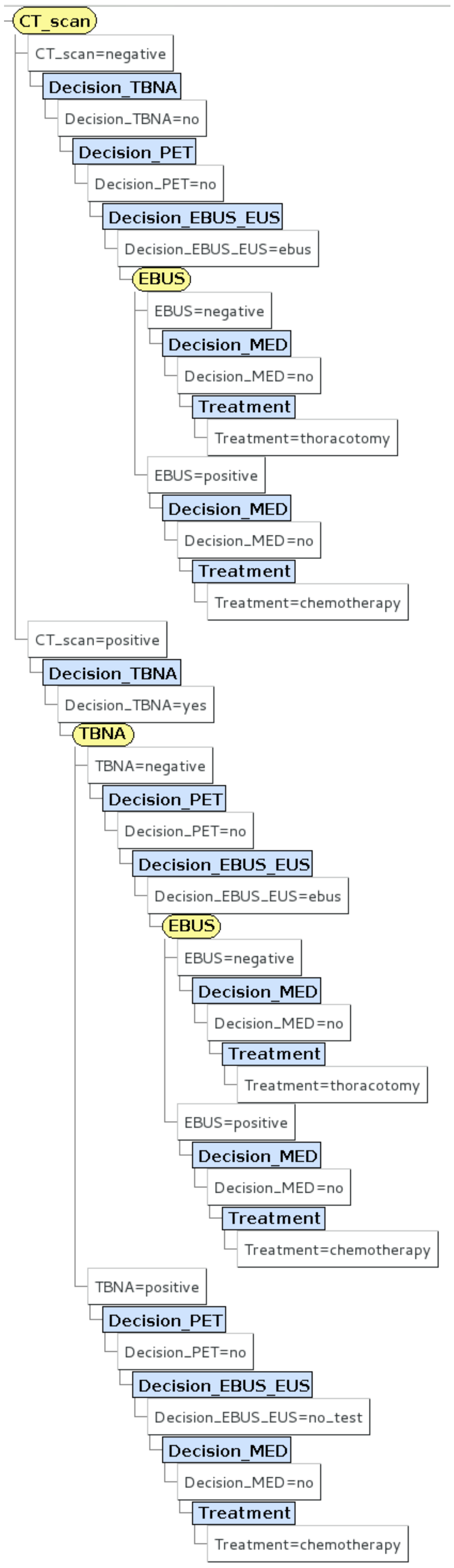


Figure 8: Optimal strategy tree for MEDIASTINET.

References

Bielza, C., Gómez, M., and Shenoy, P. P. (2010). Modeling challenges with influence diagrams: Constructing probability and utility models. *Decision Support Systems*, 49:354 – 364.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

Díez, F. J., Luque, M., and König, C. (2012). Decision analysis networks. In Cano, A., Gómez, M., and Nielsen, T. D., editors, *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM'12)*, pages 83–90, Granada, Spain.

Díez, F. J., Yebra, M., Bermejo, I., Palacios-Alonso, M. A., Arias, M., Luque, M., and Pérez-Martín, J. (2017). Markov influence diagrams: A graphical tool for cost-effectiveness analysis. *Medical Decision Making*, 37:183–195.

Fernández del Pozo, J. A., Bielza, C., and Gómez, M. (2005). A list-based compact representation for large decision tables management. *European Journal of Operational Research*, 160:638–662.

Howard, R. A. and Matheson, J. E. (1984). Influence diagrams. In Howard, R. A. and Matheson, J. E., editors, *Readings on the Principles and Applications of Decision Analysis*, pages 719–762. Strategic Decisions Group, Menlo Park, CA.

Jensen, F. V. and Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, second edition.

Luque, M. and Díez, F. J. (2010). Variable elimination for influence diagrams with super-value nodes. *International Journal of Approximate Reasoning*, 51:615 – 631.

Luque, M., Díez, F. J., and Disdier, C. (2016a). Optimal sequence of tests for the mediastinal staging of non-small cell lung cancer. *BMC Medical Informatics and Decision Making*, 16:1–14.

Luque, M., Nielsen, T. D., and Jensen, F. V. (2016b). Anytime decision-making based on unconstrained influence diagrams. *International Journal of Intelligent Systems*, 31:379–398.

Olmsted, S. M. (1983). *On Representing and Solving Decision Problems*. PhD thesis, Dept. Engineering-Economic Systems, Stanford University, CA.