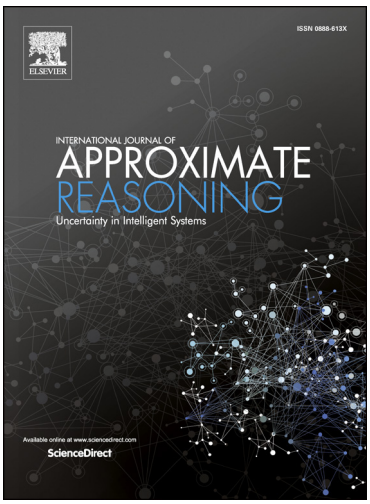
Accepted Manuscript



Decision analysis networks

Francisco Javier Díez, Manuel Luque, Iñigo Bermejo

PII: S0888-613X(17)30433-4
DOI: https://doi.org/10.1016/j.ijar.2018.02.007
Reference: IJA 8189

To appear in: *International Journal of Approximate Reasoning*

Received date: 7 July 2017
Revised date: 15 December 2017
Accepted date: 21 February 2018

Please cite this article in press as: F.J. Díez et al., Decision analysis networks, *Int. J. Approx. Reason.* (2018), https://doi.org/10.1016/j.ijar.2018.02.007

Decision analysis networks

Francisco Javier Díez, Manuel Luque, Iñigo Bermejo

Dept. Artificial Intelligence, Universidad Nacional de Educación a Distancia (UNED), Juan del Rosal 16, 28040 Madrid, Spain

Abstract

This paper presents decision analysis networks (DANs) as a new type of probabilistic graphical model. Like influence diagrams (IDs), DANs are much more compact and easier to build than decision trees and can represent conditional independencies. In fact, for every ID there is an equivalent symmetric DAN, but DANs can also represent asymmetric problems involving partial orderings of the decisions (order asymmetry), restrictions between the values of the variables (domain asymmetry), and conditional observability (information asymmetry). Symmetric DANs can be evaluated with the same algorithms as IDs. Every asymmetric DAN can be evaluated by converting it into an equivalent decision tree or, much more efficiently, by decomposing it into a tree of symmetric DANs. Given that DANs can solve symmetric problems as easily and as efficiently as IDs, and are more appropriate for asymmetric problems—which include virtually all real-world problems—DANs might replace IDs as the standard type of probabilistic graphical model for decision support and decision analysis. We also argue that DANs compare favorably with other formalisms proposed for asymmetric decision problems. In practice, DANs can be built and evaluated with OpenMarkov, a Java open-source package for probabilistic graphical models.

Keywords: Decision analysis, decision trees, influence diagrams, probabilistic graphical models, asymmetric decision problems.

Email addresses: fjdiez@dia.uned.es (Francisco Javier Díez), mluque@dia.uned.es (Manuel Luque), ibermejo@dia.uned.es (Iñigo Bermejo)

1. Introduction

The two formalisms most widely used for the representation and analysis of decision problems are decision trees (DTs) [31] and influence diagrams (IDs) [15]. DTs have the advantage of almost absolute flexibility, but also have three drawbacks: their size grows exponentially with the number of variables, they cannot represent conditional independencies, and they require in general a preprocessing of the probabilities [15, 4]; for example, medical diagnosis problems are usually stated in terms of direct probabilities, namely the prevalence of the diseases and the sensitivity and specificity of the tests, while DTs are built with inverse probabilities, i.e., the positive and negative predictive values of the tests. Even in cases with only a few chance variables, this preprocessing of probabilities is a difficult task.[1] In contrast, IDs have the advantages of being very compact, easily representing conditional independence, and using direct probabilities; but they can only represent symmetric decision problems. Following partially the definitions of Bielza et al. [4] and Jensen et al. [17], we say that there is *domain asymmetry* when the value taken by a variable restricts the values that other variables can take; there is *information asymmetry* when a variable Y is observed for some values of a variable X but not for others; and there is *order asymmetry* when the decisions can be made in different orders.

In practice, virtually all real-world problems are asymmetric, in particular all those that involve the possibility of getting additional information at a cost; for example, the variable that represents the result of a test can take some values only when the decision is to do the test. Several formalisms have been proposed for representing and solving asymmetric decision problems, but all of them have drawbacks; for example, unconstrained IDs [18] cannot represent domain asymmetry nor information asymmetry, and sequential IDs [17] may need redundant links with complex labels, as we discuss in Section 3. In this paper we present a new formalism, called *decision analysis networks* (DANs), which can represent all symmetric problems as easily as IDs and, as we argue below, can also

[1] When there are two or more chance variables between two decisions in a DT, their order can be changed without altering the results of the evaluation, but when there is a decision between two chance nodes, the order of the two cannot be changed. For this reason it is often necessary to preprocess the probabilities.

represent real-world asymmetric decision problems more naturally than other existing formalisms. We developed it when trying to solve a complex medical decision problem: the mediastinal staging of non-small cell lung cancer [26].

The DAN for that medical problem, together with other DANs for the most famous asymmetric decision problems proposed in the literature, can be found at `www.ProbModelXML.org/networks`; they are encoded in ProbModelXML, an open format for probabilistic graphical models [3]. OpenMarkov, an open-source tool for probabilistic graphical models, can be used to view, create, edit, and evaluate DANs. It is available at `www.openmarkov.org`. Its tutorial explains how to build and evaluate DANs and, when possible, compares them with equivalent IDs.

The rest of the paper is structured as follows. First, we introduce the n-test problem, which will serve to illustrate the properties of DANs and to discuss different formalisms. Then the paper consists of two main parts. The first deals with knowledge representation: Section 2 presents the definition of DANs and Section 3 compares them with other formalisms. The second deals with inference: Section 4 describes several algorithms for the evaluation of DANs and Section 5 analyzes the efficiency of those algorithms. Finally, Section 6 contains the conclusions and some proposals for future work.

Example 1. The *n-test problem* consists in deciding how to treat a patient that may suffer from a certain disease. After an initial examination of the symptoms, the doctor may order one or more of n available tests, each one having a cost. Every test can be performed once at most and its result is known immediately. The doctor has to decide which tests to perform and in which order.

In the simplest version of the problem there is only one symptom and all variables are dichotomous, i.e., the disease and the symptom are either present or absent and the result of each test is either positive or negative. The diabetes problem [10] is an instance of the two-test problem.

2. Definition of a DAN

In this section we define DANs by describing the elements that compose them and the ways of indicating the availability of information; we also discuss the meaning of the different types of links. We make several remarks about causality and availability of information, which may be useful for those who wish to apply DANs to real-world problems. However, a reader interested only

in an axiomatic definition of DANs can skip those remarks without reducing the logical consistency of the presentation.

2.1. Graph and variables of a DAN

In this paper we represent variables with capital letters (X) and their values with lower-case letters (x). A bold upper-case letter ($\mathbf{X}$) denotes a vector of variables and a bold lower-case letter ($\mathbf{x}$) represents a configuration of them, i.e., the assignment of a value to each variable in $\mathbf{X}$. When $\mathbf{Y} \subseteq \mathbf{X}$, $\mathbf{x}^{\downarrow \mathbf{Y}}$ denotes the projection $\mathbf{x}$ onto $\mathbf{Y}$, which is a configuration of $\mathbf{Y}$.

The set of nodes of a DAN, $\mathbf{V}$, can be partitioned into three disjoint subsets: chance variables ($\mathbf{C}$), decisions ($\mathbf{D}$), and utilities ($\mathbf{U}$). Chance variables represent properties or events that are not under the direct control of the decision maker, decisions correspond to actions that are under their direct control, and utilities represent payoffs (the decision maker's values). A DAN also has an acyclic directed graph such that each node represents a variable; hence we speak indifferently of nodes and variables. When the graph has a link $X \to Y$, we say that X is a parent of Y and Y is a child of X. The set of parents of a node X is denoted by $Pa(X)$, and $pa(X)$ represents a configuration of them. When there is a directed path from X to Y, we say that X is an ancestor of Y and Y is a descendant of X. In this paper we assume that utility nodes do not have children, but this requirement can easily be relaxed. We also assume that all the decisions and chance variables are discrete.

The DAN in Figure 1 contains 3 decisions, drawn as rectangles, 4 chance variables, drawn as rounded rectangles, and 3 utility nodes, drawn as hexagons.

2.2. Restrictions

A *restriction* associated to a link $X \to Y$, such that X and Y are chance or decision variables, is a pair (x, y), where x is a value of X and y is a value of Y. It means that variable Y cannot take the value y when X takes the value x. In OpenMarkov the restrictions between X and Y are represented by a table with a column for each value of X and a row for each value of Y; when the values x and y are compatible, i.e., when there is no restriction (x, y), the corresponding cell contains a 1 and is colored in green; otherwise it contains a 0 and is colored in red—see Figure 2. When all the values of Y are incompatible with x, a

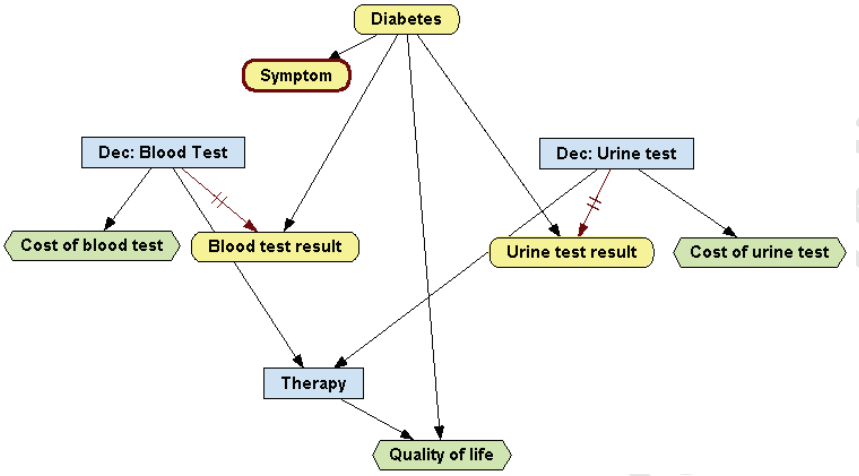


Figure 1: A DAN for the diabetes problem [10]. The decision about a test may depend on the presence of the symptom and on the result of the other test. There is no constraint on the order of the tests.

particular value of X, as in this figure, we say that there is a *total restriction* and denote it by (x, Y). It implies that in some scenarios the variable Y does not exist; we will see it more clearly when expanding the equivalent decision tree. In Figures 1 and 3 links having total restrictions are marked with a short perpendicular double line.

Dec: Blood T...	no	yes
positive	0	1
negative	0	1

Figure 2: Compatibility table for the link *Dec: Blood test* → *Blood test result* in Figure 1. It contains two restrictions, (*no*, *positive*) and (*no*, *negative*), which mean that when the test is not performed, it gives neither a positive nor a negative result. There is a total restriction because the value *no* is not compatible with any value of *Blood test result*.

When there is a restriction (x, y) but every value of X is compatible with at least one value of Y, we say that there is a *partial restriction*. For example, the DAN for the reactor problem (Fig. 3) contains a partial restriction because a bad test result prevents the construction of an advanced reactor, but every value of *Result of test* is compatible with at least one value of *Build decision*, as shown in Figure 4. This partial restriction is denoted by a short perpendicular line on the link *Result of test* ⟶ *Build decision* in Figure 3.

The meaning of a restriction (x, y) depends on whether Y is a chance variable or a decision. If it is a chance variable, the restriction (x, y)—associated to

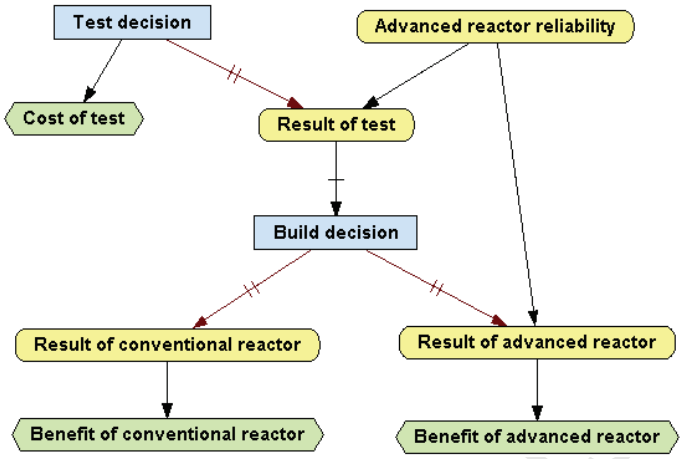


Figure 3: A DAN for the reactor problem [8, 5]. The main decision is which type of reactor to build, if any. There is a partial restriction associated to the link *Result of test* → *Build decision*, as shown in Figure 4.

Result of test	bad	good	excellent
build none	1	1	1
build conven...	1	1	1
build advanced	0	1	1

Figure 4: Compatibility table for the link *Result of test* → *Build decision* in Figure 3. The zero in the red cell means that a bad test result prevents the construction of an advanced reactor. It is a partial restriction because *bad* is compatible with other values of *Build decision*.

the link $X \to Y$, which implies that $X \in Pa(Y)$—means that $P(y \mid pa(Y)) = 0$ for all the configurations of $Pa(Y)$ in which $X = x$; we explain this in further detail in the next section. Therefore, when Y is a chance variable, there is not a significant difference between attaching a partial restriction to a link and setting to 0 the corresponding cells in the conditional probability table. However, from the point of view of knowledge engineering, it may be useful to declare a partial restriction when building the graph of the network—qualitative information—instead of just setting the probability to 0—quantitative information—because the more structural information we have about a model, the easier it will be to maintain it, avoiding inconsistencies in its numerical parameters. Additionally, restrictions can be useful when doing inference (as shown below), when performing sensitivity analysis (in order to determine which parameters can vary and which cannot), and when explaining the reasoning.

In contrast, if D is a decision, the restriction (x, d) means that when $X = x$

the decision maker (and consequently the evaluation algorithm) cannot choose the option d—see again the example in Figure 4. In summary, a partial restriction associated to a link $X \rightarrow Y$ affects the edition of the network if Y is a chance variable; if Y is a decision, the restriction affects inference.

2.3. Potentials

A potential ψ is a function that maps each configuration $\mathbf{x}$ of a set of variables $\mathbf{X}$ onto $\mathbb{R}$, i.e., $\psi(\mathbf{x})$ is a real number. Each chance node Y has an associated conditional probability potential, denoted by $\psi(y \mid pa(Y))$. When there is a restriction (x, y) and $X = x$ in the configuration $pa(Y)$, then $\psi(y \mid pa(Y)) = 0$, because y is incompatible with x. If at least one value of Y is compatible with all the values of its parents in the configuration $pa(Y)$, then $\psi(y \mid pa(Y))$ is a conditional probability distribution for Y given that configuration, as shown in Figure 5.

Dec: Blood Test	no	no	yes	yes
Diabetes	absent	present	absent	present
positive	0	0	0.02	0.96
negative	0	0	0.98	0.04

Figure 5: Probabilistic potential associated to the chance node *Blood test result*. The second and third columns are 0 due to the restrictions in Figure 2. The fourth column defines a conditional probability distribution for *Blood test result*; 0.98 is the specificity of the test. The last column defines another conditional probability distribution for the same variable; 0.96 is the sensitivity of the test.

Similarly, each utility node U has an associated potential, $u(pa(U))$.

2.4. Representing the availability of information

In DANs there are two ways to indicate when a variable becomes observed: *always-observed variables* and *revelation links*. In both cases, DANs satisfy the *no-forgetting property* [15, 27], which means that when the value of a variable is known, then it is known for all subsequent decisions.

2.4.1. Always-observed variables

A chance variable declared as *always observed* means that its value is known before making any decision. In Figure 1, the node *Symptom* is marked with a thick red border to indicate that it is always observed.

2.4.2. Revelation links

Given a link $X \to Y$, such that Y is a chance variable, we can declare that certain values of X *reveal* the value of Y; we then say that $X \to Y$ is a *revelation link*. In Figures 1 and 3 revelation links are colored in red. In general, X is a decision node, but it might also be a chance node; in this case, it means that Y is known only if a fortuitous event $X = x$ occurs and the value of X is observed. The *revelation conditions* are the values of X that reveal Y. We say that X reveals Y *unconditionally* if all values of X reveal Y; otherwise we say that X reveals Y *conditionally*. We will see below that conditional revelations induce asymmetry, while unconditional revelations do not.

When Y is an always-observed node, it does not make sense to declare a revelation condition for any link $X \to Y$. We also assume that a variable X being a descendant of a decision D cannot be always-observed because it would violate our notion of causality: if the outcome of X depends on the option chosen for D, then X cannot be known before making D. It is also contradictory to declare x as a revelation condition for Y when there is a total restriction (x, Y). In this paper we assume that all DANs satisfy these consistency properties.

2.5. Summary: The meaning of links

A link $X \to Y$ in a DAN may have five meanings:

1. *Causal influence:* For example, the links *Diabetes* $\to$ *Symptom* and *Diabetes* $\to$ *Blood test result* mean that the presence of the disease affects the probability of having the symptom and the outcomes of this test, respectively. Y must be a chance node.
2. *Functional dependence:* For example, the four links pointing at *Cost of blood test*, *Cost of urine test*, and *Quality of life* denote which variables directly affect the decision maker's values. Y must be a utility node.
3. *Temporal order:* For example, the link *Dec: Blood Test* $\to$ *Therapy* indicates which decision is made first. X and Y must be decisions.
4. *Revelation:* For example, the link *Dec: Blood Test* $\to$ *Blood test result* has an associated revelation condition: the result is known when doing the test. Y must be a chance node.

5. *Restriction:* For example, *Dec: Blood Test* $\rightarrow$ *Blood test result* has an associated restriction: when the test is not done, no result is possible. X and Y must be decision or chance nodes.

The first three meanings are the same as in IDs. Revelation links in DANs replace information links in IDs; the main difference is that in general a revelation link goes from a decision to a chance variable while an information link goes from a chance node to a posterior decision—see Sec. 3.1. Restrictions are a novelty of DANs with respect to IDs.

3. Comparison of representation formalisms

In this section we examine seven types of probabilistic graphical models (PGMs) for decision analysis, as well as another formalism recently proposed for asymmetric decision problem. First we compare DANs with IDs, which are the standard, and with the two latest types of PGMs for asymmetric decision problems, namely unconstrained influence diagrams (UIDs) and sequential influence diagrams (SIDs), and then we offer an overall review of IDs and all the formalisms for asymmetric decision problems. For a comparison of these formalisms applied to different problems, see König's master thesis [19].

3.1. Symmetric DANs vs. IDs

Definition 2. A DAN is *symmetric* if

1. it has no restrictions (domain symmetry),
2. all revelations are unconditional, i.e., if a value of X reveals Y, then all the values of X reveal it (information symmetry), and
3. a directed path connects all the decisions (order symmetry).

We will introduce in Section 4.1 an algorithm that converts a DAN into an equivalent DT. If the DAN is symmetric, so is the DT (Proposition 7).

In an ID or a symmetric DAN with n decisions $\{D_1, \ldots, D_n\}$ the set of chance variables $\mathbf{C}$ can be partitioned into $n+1$ disjoint subsets $\{\mathbf{C}_0, \ldots, \mathbf{C}_n\}$. In an ID a chance variable X belongs to $\mathbf{C}_i$ (with $0 \leq i < n$) if and only if there is an information link $X \rightarrow D_{i+1}$ and there is no link from X to any previous decision; $X \in \mathbf{C}_n$ if there is no information link $X \rightarrow D_i$ for any decision. In a symmetric DAN $\mathbf{C}_0$ contains the always-observed variables; a chance variable

X belongs to $\mathbf{C}_i$ (with $1 \leq i < n$) if and only if the decision D_i reveals X and no previous decision D_j ($j < i$) reveals it; $\mathbf{C}_n$ is the set of the variables that are not revealed by any decision.[2] Both in the ID and in the DAN, $\mathbf{C}_0$ is the set of variables observed before making the first decision, $\mathbf{C}_i$ is the set of variables observed after decision D_i and before D_{i+1}, and $\mathbf{C}_n$ is the set of unobservable variables.

Proposition 3. *For every ID there is an equivalent symmetric DAN, and vice versa.*

Proof. The transformation of an ID into a symmetric DAN is straightforward: for every chance variable X, if $X \in \mathbf{C}_0$ we declare it as always observed; if $X \in \mathbf{C}_i$, with $1 < i < n$, and there is no directed path from X to D_i, we declare that all the values of D_i reveal X. However, when the ID contains a directed path from X to D_i, it is not possible to add a revelation arc from D_i to X because it would create a cycle in the DAN. In this case we add a chance variable X' having X and D_i as its parents, with

$$P(x'|x, d) = \delta_{x,x'} , \tag{1}$$

where δ is Kronecker delta. (Variable X' can be interpreted as a test that indicates the value of X with 100% accuracy, but the result of this test is available only after the decision D_i has been made.) In the DAN, $X' \in \mathbf{C}_i$ and $X \in \mathbf{C}_n$. The combination of Equations 1 and 3 proves that the expected utility and the optimal policies for the DAN are the same as for the ID.

The transformation of a symmetric DAN into an ID is analogous: for every chance variable X in $\mathbf{C}_i$, with $i < n$, we draw a link $X \to D_i$. In this case, it is never necessary to add a dummy variable because, due to the non-forgetting property, D_i cannot be an ancestor of X. □

Given that the equations for the expected utility and for the optimal policies are identical for IDs and symmetric DANs, any algorithm for IDs can be used to evaluate symmetric DANs. For example, the arc reversal algorithm [29, 32] only differs in how to identify the nodes to be removed first: in an ID, $\mathbf{C}_n$ is the set of nodes that have no outgoing information links, while in a DAN it is the set of those that have no incoming revelation links. The variable elimination algorithm [16, 25], which only works with the set of potentials and the partial ordering of the variables, as we will see in Sec. 4.2, is identical in both cases.

[2] In a symmetric DAN it is irrelevant whether the last decision, D_n, reveals any variable or not, because this information has no effect on any decision. Therefore, in order to simplify the exposition in this section, we assume that in symmetric DANs only the decisions $\{D_1, \ldots, D_{n-1}\}$ can reveal any variables. Additionally, if an always-observed variable X reveals Y, then all the values of X reveal Y; therefore we can declare Y as always-observed and remove the revelation conditions from X to Y. Similarly, if a decision D reveals X, and X reveals Y, we may declare that D reveals Y and remove the revelation conditions from X to Y. We assume that these adjustments have been made before partitioning $\mathbf{C}$.

3.2. DANs vs. IDs for asymmetric problems

We have shown that symmetric DANs are equivalent to IDs and, therefore, they are equally suited to represent symmetric problems. In this section we argue that DANs are clearly superior to IDs for the representation of asymmetric problems.

Problems having only domain or information asymmetry can be symmetrized by using two modeling tricks [34]. First, a total restriction on a chance variable X can be modeled by adding a dummy state. For example, the one-test problem can be represented with an ID having three values for the result of the test: *positive*, *negative*, and (the dummy state) *not-performed*. This way we not only avoid domain asymmetry, but also information asymmetry: the result of the test is always known, but sometimes the "result" is *not-performed*. The drawback of this trick is that it complicates the edition of the probability tables for the variables having dummy states. It also makes the evaluation less efficient, due to the enlarged probability and utility tables, and complicates the interpretation of the results, because some policies contain configurations that can never occur, such as (*do not test*, *positive*), (*do not test*, *negative*), and (*do test*, *not-performed*). Even for this small problem it is advantageous to use a DAN, which does not need dummy states. The second trick, used for restrictions of the form (x, d) where D is a decision, consists of adding a dummy utility node U, with parents X and D, such that $u(x, d) = -\infty$ and $u(x', d') = 0$ for the other configurations of $\{X, D\}$. In the case of a total restriction on D, it is also necessary to add a dummy state for this decision. This trick complicates the graph of the ID and makes inference less efficient than when using a DAN.

Order asymmetry poses a much more serious difficulty for IDs. For example, if we try to model the n-test problem with an ID, we need n decision nodes for the tests, $\{T_1, \ldots, T_n\}$, each having $n + 1$ options; the extra option is *do no test*. We then need $n(n-1)/2$ dummy utility nodes to represent the restrictions that tests cannot be repeated, and that if the i-th decision is not to test, then all subsequent test decisions are not to test. We also need n chance variables, $\{R_1, \ldots, R_n\}$, for the results of the tests. The meaning of R_i depends on which test has been performed in the i-th place. Each R_i has $m + 1$ possible states, where m is the maximum of the number of outcomes of the tests; the extra

state is "not performed". If a test has fewer than m outcomes, some states of the R's are meaningless. Specifying the conditional probability tables for the R's is cumbersome. Tests that return numeric results cannot coexist with those having a finite set of outcomes. Additionally, the conditional probability of R_i given the tests must be repeated for every R_i, which complicates the maintenance of the model and makes sensitivity analysis virtually impossible. If a new test is added to the model, the domains of all the T's and the R's must be revised, as well as the conditional probability tables of the R's and the tables of the dummy utility nodes; the effort is comparable to building the new model from scratch.

In contrast, the DAN for the n-test problem does not need dummy states, nor dummy utility nodes, nor links between the T's; each T_i is dichotomous and represents the decision about a single test (do the i-th test or not); the states of R_i correspond to the outcomes of the i-th test, thus enabling the combination of numeric and discrete tests in the same model; each parameter is encoded only once; the size of the model is proportional to n, and adding a new test to the model is straightforward.

3.3. DANs vs. UIDs

Unconstrained influence diagrams (UIDs) were proposed as a representation framework for problems with order asymmetry [18]. Both UIDs and DANs differ from IDs in that, instead of using information links, they can declare that a variable becomes observed after making a decision. However, the representation of the flow of information is different. In a UID, a chance variable X can be declared as observable, which means that it becomes observed when all the decisions that are ancestors of X have been made, regardless of the option chosen for each decision. This prevents the representation of information asymmetry. In a DAN a chance variable can be declared as always-observed, but it is also possible to add one or several revelation conditions (x, Y) to a link $X \to Y$ indicating which values of X reveal Y; please note that X may be a chance variable, while in UIDs only decisions can reveal other variables. Another difference is that UIDs do not have restrictions and therefore cannot represent domain asymmetry.

The next propositions describe more precisely the relation between IDs, UIDs, and DANs.

Proposition 4. *For each UID having order symmetry there is an equivalent ID, and vice versa.*

The proof is analogous to that of Proposition 3.

Proposition 5. *For each DAN having domain symmetry and information symmetry there is an equivalent UID, and vice versa.*

Proof. The transformation of the DAN into a UID is straightforward: all the variables that are always-observed or revealed in the DAN are marked as observable in the UID. The reverse transformation is analogous: every observable variable in the UID that is not an ancestor of any decision is declared as always-observed in the DAN; if an observable variable X is a descendant of a decision D, an unconditional revelation link $D \to X$ is drawn. In most cases the graph of the DAN and that of the UID are identical. □

Therefore, we can say that UIDs lie halfway between IDs and DANs, because they can represent order asymmetry, but not domain nor information asymmetry. This explains why the king problem [18], used as a running example to define UIDs and illustrate their power, was designed to have only order asymmetry. Similarly, when Ahlmann-Ohlsen et al. [1] built a UID for the used car buyer problem [14, 30], they failed to mention that in its original formulation there was a constraint (restriction) on the second decision: testing the differential subsystem is possible only if the first test performed is on the transmission subsystem. Due to the inability of UIDs to represent these types of asymmetry, the review of formalisms for asymmetric problems by Bielza et al. [4, p. 233] concluded that UIDs "are not appropriate for the general class of decision problems".

3.4. DANs vs. SIDs

Sequential influence diagrams (SIDs) [17] were designed to represent the three types of asymmetry. From this point of view they made a step forward with respect to UIDs, but the reintroduction of information links complicated significantly the representation of order asymmetry. As their authors say, "an SID can basically be seen as two diagrams superimposed onto each other. One diagram encodes information precedence as well as structural and order asymmetry, whereas the other encodes functional relations for the utility nodes and probabilistic dependence relations for the chance nodes". In the SID for the dating

problem (Fig. 4a in [17]) the existence of two superimposed graphs is manifested by the two links from the node *Accept* to the node *ToDo*: the probabilistic link is drawn as a solid arrow and the structural link as a dashed arrow. Structural links are annotated with logical expressions that encode the restrictions and may involve several variables.

In principle the idea of representing restrictions as logical formulas, which was first proposed by Covaliu and Oliver [8], increases the expressiveness of the formalism. However, the complexity of the resulting graphs, with two types of links, logical expressions, clusters, and links from nodes to clusters and from clusters to nodes, makes it more difficult to implement a graphical user interface for SIDs than for DANs. Additionally, logical expressions require a language for encoding variables, their values, logical operators, nested parentheses. . . , as well as an interpreter for evaluating them at run time. Furthermore, it is difficult to check the consistency of the graph when building the model, because of the need to ensure, among other conditions, that all the cycles in the graph will disappear when evaluating it. Consistency should be re-checked when a variable is removed or when its domain changes. Omitting this build-time checks would simplify the implementation of a software package for SIDs, but would transfer the responsibility to the modeler and might cause run-time errors. These difficulties might explain why SIDs—to our knowledge—have never been implemented.

Moreover, even if there existed a software package with a graphical user interface and a consistency checking facility for SIDs, building a model for a real-world problem, explaining it to experts, and maintaining it would be significantly more difficult than with a DAN because of the above-mentioned complexity of SID graphs; see also [19].

In summary, the main advantage of DANs with respect to SIDs is that they achieve a similar expressive power—we have found no problem that be can be represented with one formalism but not with the other—much more easily. DANs have no clusters, no cycles, only node-to-node links, and at most one link between each pair of nodes. Every restriction might be encoded as a logical expression, but given that it involves only two variables, it is easier and more intuitive to represent the restriction as a table. Additionally, the only global

Table 1: Main features of several methods for representing decision problems.

	dummy states	total order	inform. links
IDs [15]	yes	yes	yes
IDPTs [34]	no	yes	yes
AIDs [28]	no	yes	yes
SVNs [33]	no	yes	yes
UIDs [18]	yes	no	no
SIDs [17]	no	no	yes
DANs	no	no	no

constraints, namely that that the graph does not contain cycles and that no always-observed variable is a descendant of a decision, can be checked during the edition of the network at no significant cost. These properties facilitate the development of software tools for DANs and the construction of models.

3.5. Overview of formalisms for asymmetric decision problems

Table 1 summarizes the main features of seven formalisms for decision analysis: influence diagrams [15], influence diagrams with probability trees (IDPTs) [34], asymmetric influence diagrams (AIDs) [28], sequential valuation networks (SVNs) [10, 33], UIDs, SIDs, and decision analysis networks (DANs)—see also the comparisons in [5, 4].

The first column indicates whether a formalism needs dummy states to "symmetrize" problems containing domain asymmetries, as explained in Section 3.2. IDs and UIDs suffer from this drawback.

The second column indicates which models require a total ordering of the decisions. Obviously, these formalisms are inadequate for problems involving order asymmetry.

Finally, the third column shows which frameworks need information links. These links are problematic when there are several possible orderings of the decisions because a variable may be known for a decision in one scenario and not in others. König [19] has built different models for several decision problems and explained why the complexity of the graphs that use information links increases significantly with n for the n-test problem.

In summary, IDPTs, AIDs, and SVNs were designed to represent domain asymmetries, while UIDs were designed only for order asymmetry. SIDs and DANs were designed for the three types of asymmetry but, as we have argued,

DANs are easier to build and interpret.

Very recently Thwaites and Smith [35] have proposed chain event graphs (CEGs) as a tool for modeling asymmetric decision problems. Given that CEGs are highly-coalesced decision trees, they differ from the models analyzed in this section in that a CEG often contains several nodes for each variable; for example, Figures 6 and 8 in their paper contain 6 and 4 nodes respectively for variable C_3, while DANs, for instance, would contain only one. For this reason DANs are more compact than CEGs and much easier to build for large problems. Furthermore, CEGs are not suitable for representing problems with several unordered decisions because the CEG must contain a path from the root to a utility node for each possible total ordering. This may be the reason why none of the examples in their paper has order asymmetry and the comparison with other formalisms does not include UIDs, which were especially designed for this type of problems. Additionally, there is, to our knowledge, no software tool for building and evaluating CEGs.

4. Evaluation of DANs

In the first part of this paper we have described the properties of DANs as a formalism for knowledge representation. In the second part we explain how to evaluate them, i.e., how to compute the expected utility and the optimal strategy, which consists in selecting the best option for each decision in each scenario; when there are several decisions to make in a scenario, the optimal strategy also determines which one must be made first. This section presents one algorithm for symmetric DANs and two for general DANs.

4.1. Equivalent decision tree

The first method for general DANs consists in evaluating an equivalent DT. Due to its time complexity, this algorithm is feasible only for small to medium DANs, but serves two purposes. First, it provides a formal semantics for DANs; the other algorithms, which are much more efficient, must return the same results as this one. Second, even though this algorithm does not need to explicitly build a decision tree, doing so may be helpful for decision analysts, who are all familiar with DTs. In fact, the conversion of IDs into DTs was a useful ex-

planation facility for teaching purposes and for interacting with experts when building decision support systems [21].

Algorithm 1 shows the pseudo-code for this method. The symbol "$\circ$" denotes the combination of a configuration of values with the value of another variable: if $\mathbf{e} = (e_1, \ldots, e_n)$, then $\mathbf{e} \circ x = (e_1, \ldots, e_n, x)$.

The algorithm proceeds by conditioning the DAN on each variable, i.e., by removing the decisions and treating every chance variable as if it were observed—the argument $\mathbf{e}$ (evidence) is a configuration of $\mathbf{E}$, the variables "observed" when evaluating a DAN. This is similar to the evaluation of Bayesian networks by recursive conditioning [9]. In our case, the main purpose of conditioning is to eliminate the asymmetries; for example, when a value of a decision D reveals Y but other values do not, this asymmetry disappears by conditioning on D. The evaluation of a DAN is performed by calling `evaluateDT`(DAN,$\blacklozenge$), where $\blacklozenge$ is the only configuration of the empty set; i.e., initially there are no conditioning variables. (In the recursive call to `evaluateDT` in line 14 of the algorithm, $(x, \blacklozenge) = (x)$.) The probability returned, $P(\blacklozenge)$, is always 1 and $U(\blacklozenge)$ is the expected utility of the DAN.

The method `instantiate()` (lines 13 and 21), which takes as arguments a DAN, a variable X, and a value x, generates a new network, DAN_x, as follows:

1. create DAN_x as a copy of the DAN;
2. if x reveals a variable Y that is not a descendant of any decision (other than X), then declare Y as observed in DAN_x; if Y is a descendant of some decisions (other than X), $\{D'_1, \ldots, D'_n\}$, draw a link $D'_i \rightarrow Y$ for $i \in \{1, \ldots, n\}$—if it did not exist—and declare that D'_i reveals Y unconditionally;
3. if there is a total restriction (x, Y), remove Y; then remove recursively all the chance and utility nodes that are children of Y;
4. if there is a restriction (x, y), remove y from the domain of Y;
5. if a chance node Y is a child of X, project the table $P(y|pa(Y))$ or $u_Y(pa(Y))$ by making $X = x$;
6. remove all the links outgoing from X; and
7. if X is a decision, remove node X.

Algorithm 1: Evaluation of an equivalent decision tree (DT).

Function `evaluateDT(`*DAN*, **e**`):`
Result: $\{P(\mathbf{e}), U(\mathbf{e})\}$
if *DAN has decision nodes* **then**
  $\mathbf{O} \leftarrow$ always-observed variables in *DAN*, except those in $\mathbf{E}$;
else
  $\mathbf{O} \leftarrow$ all the chance variables in *DAN*, except those in $\mathbf{E}$;
end
if $\mathbf{O} = \varnothing$ *and DAN has no decisions* **then**
  $P(\mathbf{e}) \leftarrow$ product of the projected probability potentials;
  $U(\mathbf{e}) \leftarrow$ sum of the projected utility potentials;
else if $\mathbf{O} \neq \varnothing$ **then**
  select $X \in \mathbf{O}$ such that X has no ancestor in $\mathbf{O}$;
  foreach x *of* X **do**
    $DAN_x \leftarrow$ `instantiate` (*DAN*, X, x);
    $\{P(\mathbf{e} \circ x), U(\mathbf{e} \circ x)\} \leftarrow$ `evaluateDT(`DAN_x, $\mathbf{e} \circ x$`)`;
  end
  $P(\mathbf{e}) \leftarrow \sum_x P(\mathbf{e} \circ x)$;
  $P(x \mid \mathbf{e}) \leftarrow P(\mathbf{e} \circ x)/P(\mathbf{e})$;
  $U(\mathbf{e}) \leftarrow \sum_x P(x \mid \mathbf{e})\, U(\mathbf{e} \circ x)$;
else if *exactly one decision* D *can be made first* **then**
  foreach d *of* D **do**
    $DAN_d \leftarrow$ `instantiate` (*DAN*, D, d);
    $\{P_d(\mathbf{e}), U_d(\mathbf{e})\} \leftarrow$ `evaluateDT(`DAN_d, **e**`)`;
  end
  $P(\mathbf{e}) \leftarrow P_d(\mathbf{e})$ for an arbitrary value d;
  $U(\mathbf{e}) \leftarrow \max_{d \in D} U_d(\mathbf{e})$;
else
  $\mathbf{D}_I \leftarrow$ *decisions that can be made first*;
  foreach D *in* $\mathbf{D}_I$ **do**
    $DAN_D \leftarrow$ `prioritize` (*DAN*, D);
    $\{P_D(\mathbf{e}), U_D(\mathbf{e})\} \leftarrow$ `evaluateDT(`DAN_D, **e**`)`;
  end
  $P(\mathbf{e}) \leftarrow P_D(\mathbf{e})$ for an arbitrary decision D;
  $U(\mathbf{e}) \leftarrow \max_{D \in \mathbf{D}_I} U_D(\mathbf{e})$;
end
return $\{P(\mathbf{e}), U(\mathbf{e})\}$

Please note that step 2 in this method preserves the property, mentioned in Section 2.4.2, that a descendant of a decision cannot be always-observed. Also note that if X is a decision, it is removed. If X is a chance node, it still appears in DAN_x but has no children. The reason for keeping X and the links pointing at it is that we need the potential $P(x|pa(X))$ to compute the probability of the evidence.

The prioritization of decision D in a DAN (cf. line 29) is implemented by drawing links from D to all the other decisions.

Algorithm 1 conditions first on the observed-variables (line 3), then on the decisions (line 19), which may make other variables observed, and finally on the unobserved variables (line 5). When this algorithm is applied to a symmetric DAN, it conditions first on the variables in $\mathbf{C}_0$, then on D_0, then on $\mathbf{C}_1$, etc., and finally on $\mathbf{C}_n$. It is the same order as when expanding a decision tree, and the reverse order of the evaluation of an influence diagram—we will come back to this point in the next section.

The algorithm always terminates because the recursive call that follows the instantiation of a chance variable X adds that variable to the evidence (line 14), and the call that follows the instantiation of a decision (line 21) removes that decision; when there are several decisions that can be made first (line 27), the number of DANs to be evaluated is finite. Therefore, the algorithm always arrives at the termination condition (line 7) in a finite number of steps.

We can illustrate Algorithm 1 with the diabetes DAN (Fig. 1). In this section D stands for *Diabetes*, S for *Symptom*, DB for the decision about the blood test, RB for its result, U_B for its cost, etc. First the algorithm is invoked with two arguments: the DAN and $\mathbf{e} = \blacklozenge$ (no evidence). In this case, $\mathbf{O} \leftarrow \{S\}$ (line 3). The instantiation of this DAN for S does not modify it because S has no descendants. Then the algorithm is recursively invoked twice: `evaluateDT`(DAN, $(+s)$) and `evaluateDT`(DAN, $(\neg s)$), as shown in Figure 6. In both cases $\mathbf{E} = \{S\}$ and $\mathbf{O} \leftarrow \varnothing$; there are two decisions that can be made first, BT and UT, and therefore `evaluateDT` must be invoked twice for each value of S: once for the DAN in which BT is made first and once for that in which UT precedes BT—see again Figure 6. The instantiation of the former for $BT = +bt$ (i.e., for the decision of doing the blood test) creates a new DAN in

which BT has been removed and BR, the result of the blood test, is an observed variable; the instantiation for $BT = \neg bt$ removes both BT and BR, as shown in Figure 6.

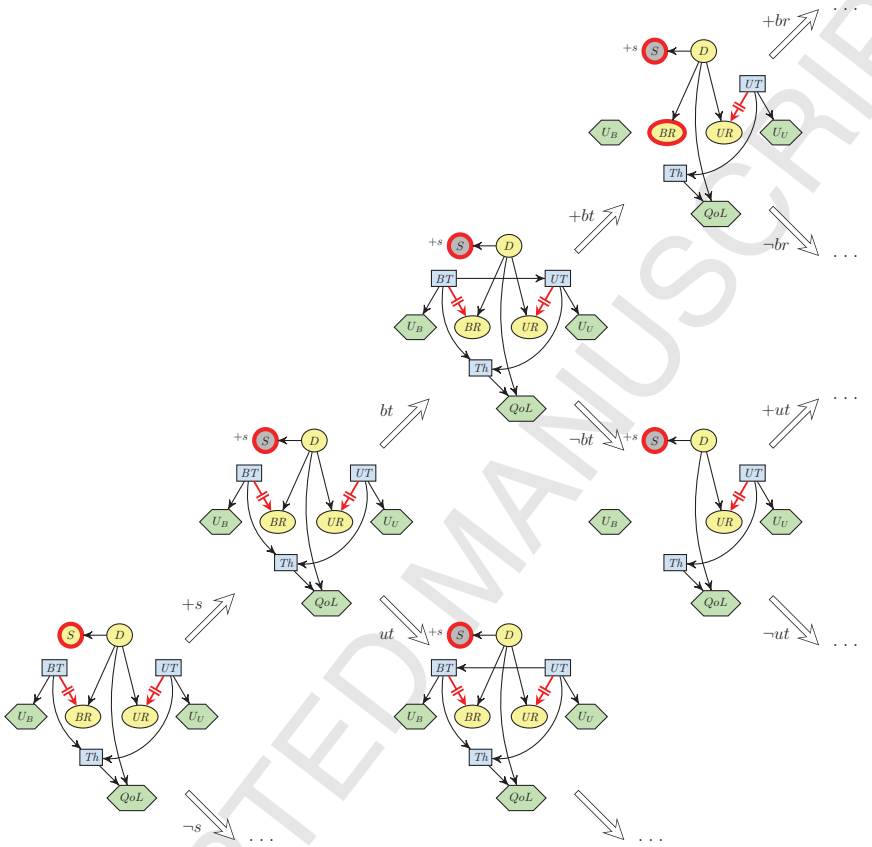


Figure 6: Evaluation of the diabetes DAN with Algorithm 1. Nodes having associated evidence are colored in gray.

This process of recursively instantiating all the decision and chance variables in the DAN is equivalent to evaluating the decision tree shown in Figure 7: there is a one-to-one correspondence between the nodes in that DT and the recursive calls to `evaluateDT`; the argument $\mathbf{e}$ in each call is the configuration of all the chance variables in the path from the root of the DT to the corresponding node. Thus, the node D in Figure 7 corresponds to a DAN in which the three decisions have been removed and $\mathbf{e} = (+ur, +br, +s)$. The instantiation of this network for $D = +d$ generates a new DAN having four chance nodes, three utility nodes, and no links. The potentials of this network have resulted from

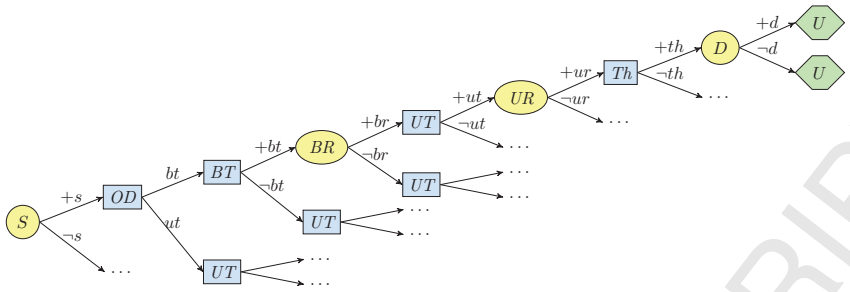


Figure 7: Decision tree for the diabetes DAN. The "meta-decision" node *OD* (order of the decisions) determines which decision to make first.

projecting the potentials of the original network: $P(d)$, $P(s|{+}d)$, $P(br|{+}d,{+}bt)$, $P(ur|{+}d,{+}bt)$, $U_B({+}bt)$, $U_U({+}ut)$, and $U_{QoL}({+}d,{+}th)$. Then `evaluateDT` is invoked with this DAN and the configuration $\mathbf{e} = ({+}d,\ {+}ur,\ {+}br,\ {+}s)$. Now $\mathbf{O} \leftarrow \varnothing$ and the network has no decisions; therefore in lines 8 and 9 we obtain, respectively,

$$P(\mathbf{e}) = P({+}d) \cdot P({+}s \mid {+}d) \cdot P({+}br \mid {+}d, {+}bt) \cdot P({+}ur \mid {+}d, {+}bt)$$

and

$$U(\mathbf{e}) = U_B({+}bt) + U_U({+}ut) + U_{QoL}({+}d, {+}th)\,.$$

Similarly, we obtain $P(\neg d, {+}ur, {+}br, {+}s)$ and $U(\neg d, {+}ur, {+}br, {+}s)$. Therefore, we can compute

$$P({+}ur, {+}br, {+}s) = P({+}d, {+}ur, {+}br, {+}s) + P(\neg d, {+}ur, {+}br, {+}s)$$

and the conditional probabilities $P({+}d \mid {+}ur, {+}br, {+}s)$ and $P(\neg d \mid {+}ur, {+}br, {+}s)$. These are the probabilities of the two branches outgoing from the node D in Figure 7. The utility for this node is computed as indicated in line 18 of the algorithm.

The utility for the node *Th* in that figure is computed at line 25; in this case, $\mathbf{e} = ({+}ur, {+}br, {+}s)$. Its probability is computed at line 22. Please note that in this line we have written $P(\mathbf{e})$ instead of $P_d(\mathbf{e})$ because the variables in $\mathbf{E}$ are not descendants of the decision D (in this example, the decision is *Th*) and therefore the probability $P_d(\mathbf{e})$ does not depend on d—see also Equation A.4 in the appendix. For the same reason we have written $P(\mathbf{e})$ instead of $P_D(\mathbf{e})$ in line 30. When building explicitly the DT, as in Figure 7, it is not necessary to write these probabilities on the branches outgoing from decision nodes.

This way we can compute the utility of every node in the DT and the probability of each branch outgoing from a chance node. The utility of the root node is the expected utility of the DAN.

We insist, however, that Algorithm 1 does not explicitly build the DT. As shown in this example, when all the decision nodes have been removed and all the chance nodes have been instantiated (line 7), the DAN has no links and consequently its size is very small. During the evaluation there are at most n DANs in the working memory, where n is the number of variables in the original DAN. Therefore, the space complexity of this algorithm is negligible. The downside is that the time complexity grows exponentially with the number of variables in the network.

It is worth noting that a decision that can reveal a variable should be made earlier than the decisions that cannot—the mathematical justification is the same as for unconstrained IDs [18]. For example, in the diabetes DAN, the decision *Therapy* does not reveal any variable. Therefore, even if the links *Dec: Blood Test* $\rightarrow$ *Therapy* and *Dec: Urine Test*$\rightarrow$ *Therapy* were not present, the node *OD* in Figure 7 (and the corresponding node in Fig. 6) should have only two outgoing branches, *bt* and *ut*, because *Therapy* should always be made after *Dec: Blood Test* and *Dec: Urine Test*. It is then possible to fine-tune the algorithm as follows: in lines 19 and 27, if the variables revealed by D_i are a proper subset of those revealed by D_j, then D_i must not be the first decision.

Finally, we point out that this algorithm may generate different DTs for one DAN depending on the variable X selected in line 11, but these trees only differ in the order of the chance variables placed between two consecutive decisions. Therefore, they are equivalent in the sense that there is a one-to-one correspondence between the decision nodes (for each pair of equivalent trees), and the optimal policies and the maximum expected utility are the same for all these trees.

4.2. Evaluation of symmetric DANs

We prove now that, as we anticipated in Section 3.1, symmetric DANs can be evaluated with the same algorithms as IDs. First we prove that the expansion of a symmetric DAN—like that of an ID—produces a symmetric DT.

Definition 6. A DT is *symmetric* if

1. every path from the root to a leaf has the same variables and in the same order, and
2. every node representing the variable X has an outgoing branch for each value x.

We mentioned above that Algorithm 1 does not need to explicitly build the DT, but it can do so. The following proposition refers to this possibility.

Proposition 7. *When a DAN is symmetric, Algorithm 1 generates a symmetric DT (provided that when there are several choices for X at line 11 it always chooses the same node).*

Proof. Order symmetry and information symmetry in the DAN guarantee that every path from the root to a leaf in the DT contains the same variables and in the same order. Domain symmetry guarantees that every node X in the tree has an outgoing branch for each value x. □

Proposition 8. *When a DAN is symmetric, the utility returned by Algorithm 1 is*

$$U = \sum_{\mathbf{c}_0} \max_{d_1} \sum_{\mathbf{c}_1} \dots \max_{d_n} \sum_{\mathbf{c}_n} P(\mathbf{c} \mid \mathbf{d}) \cdot U(\mathbf{c}, \mathbf{d}) \quad (2)$$

$$= \sum_{\mathbf{c}_0} \max_{d_1} \sum_{\mathbf{c}_1} \dots \max_{d_n} \sum_{\mathbf{c}_n} \prod_{X \in \mathbf{C}} P(x \mid pa(X)) \cdot \sum_j u_j(pa(U_j)) \,, \quad (3)$$

where the $\mathbf{C}_i$ sets are those defined in Section 3.1. (Obviously, in this equation the configurations in the potentials must be coherent with those in the sum and max operators; for example, $pa(X) = (\mathbf{c}_0, d_1, \dots, d_n, \mathbf{c}_n)^{\downarrow Pa(X)}$.)

Proof. As mentioned above, this algorithm conditions first on the variables in $\mathbf{C}_0$, then on D_1, etc. For each scenario $(\mathbf{c}_0, d_1, \dots, d_n, \mathbf{c}_n)$, the probability and the utility are the same as those computed by Algorithm 1 (lines 8 and 9). Each return from a recursive call is equivalent to the elimination of a variable, from right to left, in Equation 2 (cf. lines 16, 18, and 25). □

This proposition is relevant because the right-hand side of Equation 2 is the same as for influence diagrams and, consequently, symmetric DANs can be evaluated with the same algorithms as IDs. In particular, variable elimination with division of potentials [16, 25] is similar to Algorithm 1 but much more efficient (in time, not in space) because when eliminating a variable it only operates with the potentials that depend on that variable, while Algorithm 1 needs to compute the full joint probability of each scenario.

In the appendix we introduce a modified version of the variable elimination algorithm that we will use to evaluate general DANs by decomposing them into symmetric DANs.

4.3. Decomposition into symmetric DANs

The algorithm in Section 4.1 evaluates a DAN by recursively decomposing it into a set of DANs that have no decisions and no links. One way of optimizing this algorithm is to stop the decomposition when the DAN to be evaluated is symmetric. For example, during the evaluation of the diabetes DAN, recursive conditioning might have stopped at the node *UR* depicted in Figure 7 (properly speaking, at the corresponding node in Figure 6) because the DAN at that node was symmetric. Another way of benefiting from symmetry in this example is to observe that the node S does not induce asymmetry; in fact, the two sub-trees at the branches $+s$ and $\neg s$ in Figure 6 contain the same DANs, and the corresponding sub-trees in Figure 7 only differ in their numerical values. Therefore, it is not necessary to have two instantiations of the original DAN, one for $+s$ and another one for $\neg$s; it suffices that the algorithm returns the probability and the utility of the DANs conditioned on S.

These considerations lead to Algorithm 2, which has several similarities with the evaluation of SIDs [17]. The method `instantiate()` is the same as in Algorithm 1—see Sec. 4.1. The first difference with respect to Algorithm 1 is that when the DAN is symmetric (line 5) recursion stops and the DAN is evaluated with the algorithm `conditionalSymmetricEvaluation` described in the appendix. Another difference is that the algorithm only instantiates the chance variables that induce asymmetry (lines 3 and 8 to 11). For this reason Algorithm 2 takes one extra argument, $\mathbf{V}$, containing some observable variables that have not been instantiated because they do not induce any asymmetry. Finally, Algorithm 1 returns $P(\mathbf{e})$ and $U(\mathbf{e})$, which are real numbers (scalars) because $\mathbf{e}$ is a single configuration, while Algorithm 2 returns $P(\mathbf{v}, \mathbf{e})$ and $U(\mathbf{v}, \mathbf{e})$, which are potentials that depend on $\mathbf{V}$.

In the evaluation of the diabetes DAN, the algorithm is first invoked with this network, $\mathbf{V} = \varnothing$, and $\mathbf{e} = \blacklozenge$. The observed variables are $\mathbf{O} \leftarrow \{S\}$; as S does not induce asymmetry, $\mathbf{O}_s \leftarrow \{S\}$ and $\mathbf{O}_a \leftarrow \varnothing$. The algorithm arrives at line 32 and makes $\mathbf{D}_I \leftarrow \{BT, UT\}$. Line 35 invokes `evaluateDSD(`$DAN_{BT}$`,{`$S$`},`$\blacklozenge$`)`, where DAN_{BT} contains a link from BT to UT that prioritizes BT. Given that D induces asymmetry, the DAN must be instantiated for $+bt$ and $\neg bt$ (line 21). The decomposition continues until all the DANs to be evaluated are symmetric,

Algorithm 2: Decomposition into symmetric DANs (DSD).

Function `evaluateDSD(`*DAN*, $\mathbf{V}$, $\mathbf{e}$`):`
Result: $\{P(\mathbf{v},\mathbf{e}), U(\mathbf{v},\mathbf{e})\}$
$\mathbf{O} \leftarrow$ always-observed variables in *DAN*, except those in $\mathbf{E} \cup \mathbf{V}$;
$\mathbf{O}_a \leftarrow$ variables in $\mathbf{O}$ that induce asymmetry;
$\mathbf{O}_s \leftarrow$ variables in $\mathbf{O}$ that do not induce asymmetry (i.e., $\mathbf{O}_s = \mathbf{O} \setminus \mathbf{O}_a$);
if *DAN is symmetric* **then**
  $\{P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}), U(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e})\} \leftarrow$
    `conditionalSymmetricEvaluation` (*DAN*, $\mathbf{V} \cup \mathbf{O}_s$, $\mathbf{e}$);
else if $\mathbf{O}_a \neq \varnothing$ **then**
  select $X \in \mathbf{O}_a$ such that X has no ancestor in $\mathbf{O}_a$;
  foreach *x of X* **do**
    $DAN_x \leftarrow$ `instantiate` (*DAN*, X, x);
    $\{P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e} \circ x), U(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e} \circ x)\} \leftarrow$
      `evaluateDSD(`DAN_x, $\mathbf{V} \cup \mathbf{O}_s$, $\mathbf{e} \circ x$`)`;
  end
  $P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}) \leftarrow \sum_x P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e} \circ x)$;
  $P(x \mid \mathbf{v} \circ \mathbf{o}_s, \mathbf{e}) \leftarrow P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e} \circ x)/P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e})$;
  $U(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}) \leftarrow \sum_x P(x \mid \mathbf{v} \circ \mathbf{o}_s, \mathbf{e})\, U(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e} \circ x)$;
else if *exactly one decision D can be made first* **then**
  if *D induces asymmetry* **then**
    foreach *d in D* **do**
      $DAN_d \leftarrow$ `instantiate` (*DAN*, D, d);
      $\{P(\mathbf{v} \circ \mathbf{o}_s \circ d, \mathbf{e}), U(\mathbf{v} \circ \mathbf{o}_s \circ d, \mathbf{e})\} \leftarrow$
        `evaluateDSD(`DAN_d, $\mathbf{V} \cup \mathbf{O}_s$, $\mathbf{e}$`)`;
    end
  else
    $\{P_d(\mathbf{v} \circ \mathbf{o}_s \circ d, \mathbf{e}), U_d(\mathbf{v} \circ \mathbf{o}_s \circ d, \mathbf{e})\} \leftarrow$
      `evaluateDSD(`*DAN*, $\mathbf{V} \cup \mathbf{O}_s \cup \{D\}$, $\mathbf{e}$`)`
  end
  $P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}) \leftarrow P(\mathbf{v} \circ \mathbf{o}_s \circ d, \mathbf{e})$ for an arbitrary value d;
  $U(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}) \leftarrow \max_{d \in D} U(\mathbf{v} \circ \mathbf{o}_s \circ d, \mathbf{e})$;
else
  $\mathbf{D}_I \leftarrow$ *decisions that can be made first*;
  foreach *D in* $\mathbf{D}_I$ **do**
    $DAN_D \leftarrow$ `prioritize` (*DAN*, D);
    $\{P_D(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}), U_D(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e})\} \leftarrow$ `evaluateDSD(`DAN_D, $\mathbf{v} \circ \mathbf{o}_s$, $\mathbf{e}$`)`;
  end
  $P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}) \leftarrow P_D(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e})$ for an arbitrary decision D;
  $U(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}) \leftarrow \max_{D \in \mathbf{D}_I} U_D(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e})$;
end
$P(\mathbf{v}, \mathbf{e}) \leftarrow \sum_{\mathbf{o}_s} P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e}))$;
$P(\mathbf{o}_s \mid \mathbf{v}, \mathbf{e}) \leftarrow P(\mathbf{v} \circ \mathbf{o}_s, \mathbf{e})/P(\mathbf{v}, \mathbf{e})$;
$U(\mathbf{v}, \mathbf{e}) \leftarrow \sum_{\mathbf{o}_s} P(\mathbf{o}_s \mid \mathbf{v}, \mathbf{e})\, U(\mathbf{o}_s, \mathbf{e})$;
return $\{P(\mathbf{v}, \mathbf{e}), U(\mathbf{v}, \mathbf{e})\}$

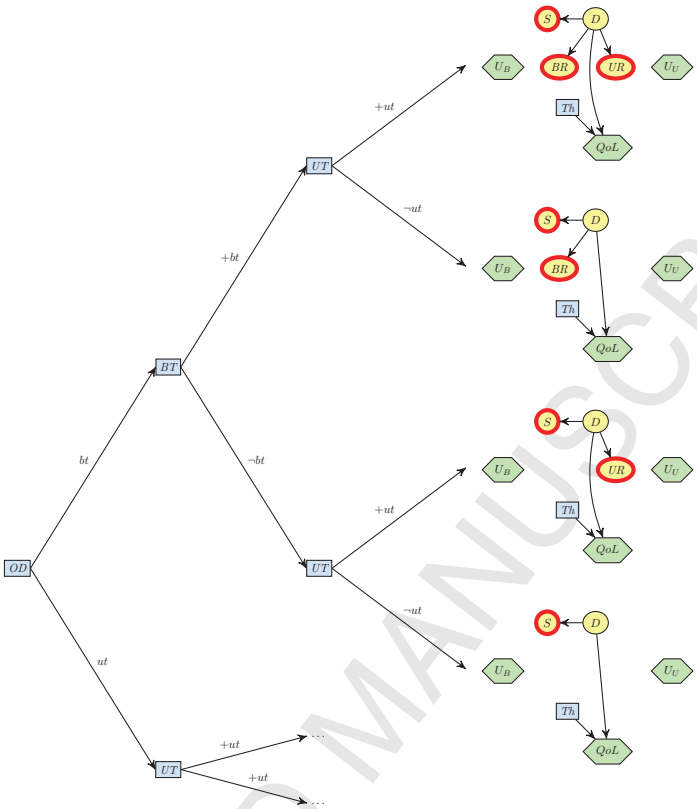


Figure 8: Decomposition of the diabetes DAN into symmetric DANs (Algorithm 2).

as shown in Figure 8, where each node represents a call to `evaluateDSD`. For example, the upper-left DAN in that figure represents a call in which $\mathbf{V} = \{S, BR, UR\}$ and $\mathbf{e} = \blacklozenge$, which is evaluated with the method `conditionalSymmetricEvaluation` (line 7).

This algorithm yields the same result as Algorithm 1 but is much faster because it avoids many redundant computations.

5. Efficiency of the algorithms

In the case of symmetric DANs, the time complexity of Algorithm 1 grows exponentially with the number of variables in the DAN, but its space complexity is negligible. The worst-case time and space complexity of Algorithm 2, which reduces to `conditionalSymmetricEvaluation` in the case of symmetric DANs, also grows exponentially; but in practice the complexity is usually much lower,

just as in the case of IDs—remember that `conditionalSymmetricEvaluation` is a variant of variable elimination [16, 25].

In the case of asymmetric DANs, restrictions reduce the time complexity of Algorithm 1 and the time and space complexity of Algorithm 2, whereas partial orderings increase their complexity: when there are n decisions and no relation of precedence among them, it is necessary to evaluate $n!$ possible orderings instead of only one. Stirling's approximation, which says that $n! \approx \sqrt{2\pi n}(n/e)^n$, explains the rapid growth of the computation time for the n-test problem.

The next subsection describes some experiments aimed at measuring the efficiency of these algorithms for different DANs.

5.1. Experiments

The main difficulty for evaluating these algorithms is the scarcity of DANs (unlike the case of Bayesian networks and IDs, for which there are excellent repositories). We have tested them on the five DANs we have built for several toy problems proposed in the literature: the used car buyer problem [14], the reactor problem [8], the dating problem [28], the king problem [18], and the diabetes problem [10]. We have also tried them on the n-test problem for several values of n, as well as on Mediastinet, a DAN for the mediastinal staging of non-small cell lung cancer, which is an adaptation of an ID for the same problem [26]. The main difference is that in this DAN there is not a total ordering of the 5 possible tests; there remains, however, a partial ordering for medical reasons. Another difference is that all the dummy states in the ID have been removed when building the DAN, except for the result of a test that affects the sensitivity and specificity of three subsequent tests. All these networks are available at `www.ProbModelXML.org/networks`.

The algorithms were implemented in OpenMarkov and ran on a PC with an Intel Core i7-2670QM CPU @ 2.20 GHz.

5.2. Results

Table 5.2 shows that the time necessary to evaluate small DANs—including the five toy examples and the 2- and 3-test problems—is negligible. For networks that have 5 or more tests, including Mediastinet, the decomposition into symmetric DANs (Algorithm 2, DSD) is significantly faster than the evaluation

Table 2: Time complexity of DT (evaluation of an equivalent decision tree—Algorithm 1) and DSD (decomposition into symmetric DANS—Algorithm 2), and the ratio between them.

	DT (ms)	**DSD** (ms)	**DT/DSD**
Reactor [8]	84	158	0.53
Used car buyer[14]	141	173	0.82
Dating[28]	154	150	1.03
King [18]	8,547	523	16.34
Diabetes (2 tests) [10]	214	171	2.37
3 tests	595	398	1.25
4 tests	3,205	1,399	2.29
5 tests	22,425	6,033	3.72
6 tests	377,025	43,242	8.72
7 tests	8,577,422	529,329	16.20
Mediastinet [26]	35,623	2,596	13.72

of an equivalent decision tree (Algorithm 1, DT). In particular, the former evaluated the 7-test DAN in less than 9 minutes, while the latter needed 2 hours and 23 minutes.

6. Conclusion

In this paper we have introduced DANs as a new formalism for decision analysis. We have compared DANs with other types of probabilistic graphical models, especially with IDs (the standard) and UIDs, SIDs and CEGs (the most recent proposals). We showed that for each ID there is an equivalent DAN, which can be evaluated with the same algorithms, but the converse is not true: for some problems that can be easily represented with DANs it is unfeasible to build an ID. For this reason DANs might favorably replace IDs as the standard type of probabilistic graphical model for decision support and decision analysis.

DANs having only order asymmetry are equivalent to UIDs, but they can also represent domain and information asymmetry. In principle the expressive power of DANs is lower than that of decision trees, SIDs or CEGs; additionally, IDs can be augmented with almost arbitrary constraints (restrictions) [13, 12], while every restriction in a DAN is associated with a link, i.e., with only two variables. However after examining 25 books that describe DTs, most of them about medical decision making, and all the literature on asymmetric decision problems, we have not found any problem that cannot be modeled with a DAN. Except in the case of extremely small problems, which can be represented with

DTs, building a DAN is never more complex than building a model of a different type; on the contrary, it is usually easier or much easier.

Another limitation of DANs is the no-forgetting property (cf. Sec. 2.4), also present in all the models analyzed in this paper, including DTs. In the rare cases in which this property does not hold it would be more appropriate to use limited-memory IDs (LIMIDs) [23].

OpenMarkov is an open-source software tool for editing DANs; several examples built with it are available at `www.ProbModelXML.org/networks`. It can transform every DAN into an equivalent DT (with Algorithm 1) or evaluate it much more efficiently (with Algorithm 2) and then show the optimal strategy in the form of a strategy tree, just as in the case of IDs [24]. (In fact, we developed the idea of strategy trees for DANs and then realized that it could also be very useful for IDs.) In the near future OpenMarkov will support cost-effectiveness analysis with DANs (based on [2]) and the same types of sensitivity analyses and the same explanation facilities that are already available as for IDs [20, 22, 21]—see OpenMarkov's tutorial. We will also implement Markov DANs, in analogy with Markov IDs [11].

An open line for future research is to develop approximate and stochastic algorithms for DANs with continuous variables and for those that cannot be evaluated with exact algorithms. A point of departure might be the algorithms in [4, 6, 7].

Acknowledgments

Caroline König, during her master thesis research, implemented the edition of DANs in OpenMarkov and compared DANs with other models for several decision problems [19]. Her work contributed to fine-tuning the definition of DANs. Jorge Pérez-Martín collaborated in adapting the above algorithms to version 0.3.0 of OpenMarkov and co-authored the chapter about DANs in the tutorial. We also thank the anonymous reviewers for many useful comments and all the developers of OpenMarkov for their contributions to the project.

This work has been supported by the Spanish Government under grants TIN2006-11152, TIN2009-09158, PI13/02446, and TIN2016-77206-R, and co-financed by the European Regional Development Fund. I.B. received a doctoral grant from the Universidad Nacional de Educación a Distancia (UNED)

and a Marie Curie grant from the European Union (FP7-PEOPLE-2012-IAPP, no. 324401).

AppendixA. Conditional evaluation of symmetric DANs

In Section 4.2 we have explained how to compute the expected utility of a symmetric DAN using the algorithm called variable elimination with division of potentials [16, 25]. In this appendix we present a variant of this algorithm that, given a DAN and some evidence $\mathbf{x}$, returns the probability $P(\mathbf{x})$ and the conditional expected utility $U(\mathbf{x})$, which we define here.

In a symmetric DAN, the joint probability of the chance variables conditioned on the decisions is

$$P(\mathbf{c} \mid \mathbf{d}) = \prod_{C \in \mathbf{C}} P(c \mid pa(C)) \,. \tag{A.1}$$

Let $\mathbf{C}$ be a subset of the always-observed variables in a symmetric DAN, $\mathbf{X} \subseteq \mathbf{C}_0$, and $\mathbf{x}$ a configuration of $\mathbf{X}$. We denote by $\mathbf{C}^{\setminus \mathbf{x}}$ the other chance variables ($\mathbf{C}^{\setminus \mathbf{x}} = \mathbf{C} \setminus \mathbf{X}$) and by $\mathbf{c}^{\setminus \mathbf{x}}$ a configuration of them. We then have

$$P(\mathbf{x} \mid \mathbf{d}) = \sum_{\mathbf{c}^{\setminus \mathbf{x}}} P(\mathbf{c}^{\setminus \mathbf{x}}, \mathbf{x} \mid \mathbf{d}) \,. \tag{A.2}$$

We may also define

$$P(\mathbf{x}) = \sum_{\mathbf{a}} \prod_{C \in \mathbf{X} \cup \mathbf{A}} P(c \mid pa(C)) \,, \tag{A.3}$$

where $\mathbf{A}$ is the set of ancestors of $\mathbf{X}$ in the DAN. $P(\mathbf{x})$ does not depend on $\mathbf{D}$ because the always-observed variables are not descendants of any decision; therefore, in this expression $Pa(C)$ does not contain a decision for any C. It is possible to prove, by eliminating first the barren nodes in Equation A.2, that

$$\forall \mathbf{d},\, P(\mathbf{x}) = P(\mathbf{x} \mid \mathbf{d}) \,. \tag{A.4}$$

Lemma 9. *In the above context,*

$$P(\mathbf{x}) = 0 \;\; \Leftrightarrow \;\; \forall \mathbf{c}^{\setminus \mathbf{x}}, \forall \mathbf{d},\, P(\mathbf{c}^{\setminus \mathbf{x}}, \mathbf{x} \mid \mathbf{d}) = 0 \,. \tag{A.5}$$

Proof. If follows from Equations A.2 and A.4. □

By analogy with Equation 2, we introduce the following definition:

Definition 10. The *conditional expected utility* of a symmetric DAN given $\mathbf{x}$ is

$$U(\mathbf{x}) = \begin{cases} 0 & \text{if } P(\mathbf{x}) = 0 \\ \sum_{\mathbf{c}_0^{\setminus \mathbf{X}}} \max_{d_1} \sum_{\mathbf{c}_1} \dots \max_{d_n} \sum_{\mathbf{c}_n} P(\mathbf{c}^{\setminus \mathbf{X}} \mid \mathbf{x}, \mathbf{d}) \cdot U(\mathbf{c}, \mathbf{d}) & \text{otherwise}. \end{cases} \tag{A.6}$$

When $P(\mathbf{x}) \neq 0$,

$$U(\mathbf{x}) = \frac{1}{P(\mathbf{x})} \sum_{\mathbf{c}_0^{\setminus \mathbf{X}}} \max_{d_1} \sum_{\mathbf{c}_1} \dots \max_{d_n} \sum_{\mathbf{c}_n} P(\mathbf{x}, \mathbf{c}^{\setminus \mathbf{X}} \mid \mathbf{d}) \cdot U(\mathbf{c}, \mathbf{d}) \tag{A.7}$$

$$= \frac{1}{P(\mathbf{x})} \sum_{\mathbf{c}_0^{\setminus \mathbf{X}}} \max_{d_1} \sum_{\mathbf{c}_1} \dots \max_{d_n} \sum_{\mathbf{c}_n} \prod_{C \in \mathbf{C}} P(c \mid pa(C)) \cdot \sum_j u_j(pa(U_j)) \,. \tag{A.8}$$

Proposition 11. *For every symmetric DAN and every subset* $\mathbf{X}$ *of its always-observed variables, the expected utility is*

$$U = \sum_{\mathbf{x}} P(\mathbf{x}) \cdot U(\mathbf{x}) \,. \tag{A.9}$$

Proof. By comparison with Equation 2, it suffices to prove that for every $\mathbf{x}$

$$P(\mathbf{x}) \cdot U(\mathbf{x}) = \sum_{\mathbf{c}_0^{\setminus \mathbf{X}}} \max_{d_1} \sum_{\mathbf{c}_1} \dots \max_{d_n} \sum_{\mathbf{c}_n} \prod_{C \in \mathbf{C}} P(\mathbf{x}, \mathbf{c}^{\setminus \mathbf{X}} \mid \mathbf{d}) \cdot U(\mathbf{c}, \mathbf{d}) \,.$$

When $P(\mathbf{x}) = 0$, this equality follows from Lemma 9. When $P(\mathbf{x}) > 0$, it follows from Equation A.7. □

A special case that deserves a comment is when $\mathbf{X} = \varnothing$. This set has only one configuration, $\mathbf{x} = \blacklozenge$, whose probability is $P(\blacklozenge) = 1$. In this case $\mathbf{c}_0^{\setminus \mathbf{X}} = \mathbf{c}_0$ and, because of Equation A.6, $U(\mathbf{x}) = U(\blacklozenge) = U$.

The marginalizations and maximizations in Equation A.8 can be performed with the variable elimination algorithm [16, 25], and $P(\mathbf{x})$ can be computed on a Bayesian network having only the variables in $\mathbf{X}$ and its ancestors (cf. Eq. A.3). A more efficient approach, which computes $P(\mathbf{x})$ and $U(\mathbf{x})$ at the same time, is to apply variable elimination with division of potentials [16, 25]. This algorithm maintains two separate lists, one of probability potentials and another one of utility potentials. After eliminating the variables in $\mathbf{C}_\mathbf{0}^{\setminus \mathbf{X}} \cup \mathbf{C}_1 \cup \ldots \cup \mathbf{C}_n \cup \{D_1, \ldots, D_n\}$—i.e., all except those in $\mathbf{X}$—the product of the remaining probability potentials is $P(\mathbf{x})$. If this probability is zero, we make $U(\mathbf{x}) = 0$, in accordance with Definition 10; otherwise, $U(\mathbf{x})$ is the sum of the remaining utility potentials.

Finally, we can split $\mathbf{X}$ into two sets, $\mathbf{V}$ and $\mathbf{E}$, and then compute $P(\mathbf{v}, \mathbf{e})$ and $U(\mathbf{v}, \mathbf{e})$ for a particular configuration of $\mathbf{E}$ (the evidence $\mathbf{e}$) and all the configurations of $\mathbf{V}$ (the conditioning variables). This way the variable elimination algorithm can implement the method `conditionalSymmetricEvaluation` used in Algorithm 2, line 5, by projecting each probability and utility potential in accordance with $\mathbf{e}$ and then eliminating all the variables except those in $\mathbf{V}$.

References

[1] K. S. Ahlmann-Ohlsen, F. V. Jensen, T. D. Nielsen, O. Pedersen, and M. Vomlelová. A comparison of two approaches for solving unconstrained influence diagrams. *International Journal of Approximate Reasoning*, 50:153–173, 2009.

[2] M. Arias and F. J. Díez. Cost-effectiveness analysis with influence diagrams. *Methods of Information in Medicine*, 54:353–358, 2015.

[3] M. Arias, F. J. Díez, M. A. Palacios-Alonso, and I. Bermejo. ProbModelXML. A format for encoding probabilistic graphical models. In A. Cano, M. Gómez, and T. D. Nielsen, editors, *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM'12)*, pages 11–18, Granada, Spain, 2012.

[4] C. Bielza, M. Gómez, and P. P. Shenoy. A review of representation issues and modelling challenges with influence diagrams. *Omega*, 39:227–241, 2011.

[5] C. Bielza and P. P. Shenoy. A comparison of graphical techniques for asymmetric decision problems. *Management Science*, 45:1552–1569, 1999.

[6] J. M. Charnes and P. P. Shenoy. Multistage Monte Carlo method for solving influence diagrams using local computation. *Management Science*, 50:405–418, 2004.

[7] B. R. Cobb and P. P. Shenoy. Decision making with hybrid influence diagrams using mixtures of truncated exponentials. *European Journal of Operational Research*, 186:261–275, 2008.

[8] Z. Covaliu and R. M. Oliver. Representation and solution of decision problems using sequential decision diagrams. *Management Science*, 41:1860–1881, 1995.

[9] A. Darwiche. Recursive conditioning. *Artificial Intelligence*, 126:5–41, 2001.

[10] R. Demirer and P. P. Shenoy. Sequential valuation networks for asymmetric decision problems. *European Journal of Operational Research*, 169:286–309, 2006.

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

[12] M. Gómez. Real-world applications of influence diagrams. In J. A. Gámez, S. Moral, and A. Salmerón, editors, *Advances in Bayesian Networks*, volume 146 of *Studies in Fuzziness and Soft Computing*, pages 161–180, Berlin, 2004. Springer.

[13] M. Gómez and A. Cano. Applying numerical trees to evaluate asymmetric decision problems. In T. D. Nielsen and N. L. Zhang, editors, *Proceedings of the Seventh European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2003)*, Lecture Notes in Computer Science, pages 196–207, Aalborg, Denmark, 2003. Springer.

[14] R. A. Howard. The used car buyer. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 689–718. Strategic Decisions Group, Menlo Park, CA, 1984.

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

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

[17] F. V. Jensen, T. D. Nielsen, and P. P. Shenoy. Sequential influence diagrams: A unified asymmetry framework. *International Journal of Approximate Reasoning*, 42:101–118, 2006.

[18] F. V. Jensen and M. Vomlelová. Unconstrained influence diagrams. In A. Darwiche and N. Friedman, editors, *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI'02)*, pages 234–241, San Francisco, CA, 2002. Morgan Kaufmann.

[19] C. König. Representing asymmetric decision problems with decision analysis networks. Master's thesis, Dept. Artificial Intelligence, UNED, Madrid, Spain, 2012.

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

[21] C. Lacave, M. Luque, and F. J. Díez. Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, 37:952–965, 2007.

[22] C. Lacave, A. Oniśko, and F. J. Díez. Use of Elvira's explanation facilities for debugging probabilistic expert systems. *Knowledge-Based Systems*, 19:730–738, 2006.

[23] S. L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1235–1251, 2001.

[24] M. Luque, M. Arias, and F. J. Díez. Synthesis of strategies in influence diagrams. In *Proceedings of the Thirty-third Conference on Uncertainty in Artificial Intelligence (UAI'17)*, pages 1–9, Corvallis, OR, 2017. AUAI Press.

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

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

[27] T. D. Nielsen and F. V. Jensen. Welldefined decision scenarios. In K. Laskey and H. Prade, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 502–511, San Francisco, CA, 1999. Morgan Kaufmann.

[28] T. D. Nielsen and F. V. Jensen. Representing and solving asymmetric Bayesian decision problems. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI'00)*, pages 416–425, San Francisco, CA, 2000. Morgan Kaufmann.

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

[30] R. Qi, N. L. Zhang, and D. Poole. Solving asymmetric decision problems with influence diagrams. In R. L. de Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 491–497, San Francisco, CA, 1994. Morgan Kaufmann.

[31] H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory*. John Wiley & Sons, Cambridge, MA, 1961.

[32] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[33] P. P. Shenoy. Valuation network representation and solution of asymmetric decision problems. *European Journal of Operational Research*, 121:579–608, 2000.

[34] J. E. Smith, S. Holtzman, and J. E. Matheson. Structuring conditional relationships in influence diagrams. *Operations Research*, 41:280–297, 1993.

[35] P. A. Thwaites and J. Q. Smith. A new method for tackling asymmetric decision problems. *International Journal of Approximate Reasoning*, 88:624–639, 2017.