Decision making with decision event graphs

Robert G. Cowell
Faculty of Actuarial Science and Insurance, Cass Business School, City University, London, 106 Bunhill Row, London EC1Y 8TZ, UK, rgc@city.ac.uk

Peter Thwaites, James Q. Smith
Statistics Department, University of Warwick, Coventry CV4 7AL, UK, Peter.Thwaites@warwick.ac.uk
J.Q.Smith@warwick.ac.uk

We introduce a new modelling representation, the Decision Event Graph (DEG), for asymmetric multistage decision problems. The DEG explicitly encodes conditional independences and has additional significant advantages over other representations of asymmetric decision problems. The colouring of edges makes it possible to identify conditional independences on decision trees, and these coloured trees serve as a basis for the construction of the DEG. We provide an efficient backward-induction algorithm for finding optimal decision rules on DEGs, and work through an example showing the efficacy of these graphs. Simplifications of the topology of a DEG admit analogues to the sufficiency principle and barren node deletion steps used with influence diagrams.

Key words: **Asymmetric decision problem; Backwards induction; Decision tree; Extensive form; Influence diagram; Local computation; Optimal policy.**

1. Introduction

The decision tree is one of the oldest representations of a decision problem (Raiffa 1968, Lindley 1985, Smith and Thwaites 2008a). It is flexible and expressive enough to represent asymmetries within both the decision space and outcome space, doing this through the topological structure of the tree. The topology of the tree can also be used to encode the natural ordering, or unfolding, of decisions and events. By exploiting the topology optimal decision rules may be found using backward induction.

However, a drawback of decision trees is that the representation can be unwieldy even for relatively simple problems and can obscure conditional independence relationships. The goal of this

paper is to introduce a new graphical representation of multistage decision problems together with a local computation scheme over the graph for finding optimal decision policies. This new representation we call the *Decision Event Graph* (DEG). This compact representation is particularly suited to decision problems whose state spaces do not admit a product form. As well as representing a decision problem, the DEG also provides a framework for computing decision rules. As the DEG has fewer edges and nodes than a decision tree, the computation of such rules is necessarily more efficient using this new graph. In addition, unlike the standard decision tree, the DEG allows for the explicit depiction of conditional independence relationships.

A variety of alternative graphical representations have been developed to alleviate the complexity associated with decision trees, and allow for local computation. The most commonly used and influential of these is the *influence diagram* (ID) (Howard and Matheson 1984, Olmsted 1983, Shachter 1986, Jensen and Nielsen 2007, Smith and Thwaites 2008b, Howard and Matheson 2005, Boutilier 2005, Pearl 2005). Modifications to ID solution techniques have been made since their introduction in, for example, (Smith 1989a, Cowell 1994, Jensen et al. 1994). The principal drawback of the ID representation is that it is not suited to asymmetric decision problems where different actions can result in different choices in the future (Covaliu and Oliver 1995) or different events give rise to very different unfoldings of the future. Attempts have been made (Smith and Matheson 1993) to adapt IDs for use with such problems, and Call and Miller (1990) developed techniques which used both decision trees and IDs. The Sequential Decision Diagram, a representation similar to an ID but with context-specific information added to the edges, was introduced by Covaliu and Oliver (1995), and Shenoy (1996) proposed the Valuation Network. A good discussion of the pros and cons of the various techniques for tackling asymmetric problems is given by Bielza and Shenoy (1999).

The specification of asymmetric decision problems in terms of IDs and related representations typically requires the introduction of dummy states and/or additional values into probability and utility tables in order to symmetrize the problem for both representation and analysis purposes. In addition, finding optimal strategies tends to generate even larger tables, for example by operating

on a secondary structure such as a junction tree (Cowell et al. 1999, Jensen and Nielsen 2007) or by modifying the ID with arc reversal and barren node removal operations (Shachter 1986). These factors both increase computational complexity and hinder interpretation.

In contrast the DEG introduced in this paper, by explicitly retaining the asymmetries, does not require additional dummy states or values, giving a more streamlined specification. Furthermore when using a DEG no secondary structures need be created, and the DEG can be used directly as a framework on which to calculate an optimal policy rapidly.

In this paper we concentrate of the relationship between DEGs and their corresponding decision trees. DEGs are closely related to *Chain Event Graphs* (CEGs) (Smith and Anderson 2008), which are themselves functions of *event trees* (or *probability trees*). Indeed, if a DEG consists solely of chance nodes, (so that is it has neither utility nor decision nodes), and if the edges only have probabilities associated with them, then the DEG reduces to a CEG.

A DEG is a function of a decision tree, although it is not necessary in all problems to first construct a decision tree before creating the DEG. There are a number of different types of decision tree, dependent on the method of construction. Commonly used types include *Extensive form* (EF), *Normal form* and *Causal* trees (Smith and Thwaites (2008a)). But regardless of type, all trees consist of a root node from which directed paths emanate consisting of edges and vertices, each path terminating in a leaf node. Vertices may be *chance*, *decision* or *utility* nodes, and edges may have associated probabilities and / or utilities. Edges are labelled by events if the edge emanates from a chance node or by possible actions open to the decision maker (DM) if the edge emanates from a decision node.

Many authors (eg. Jensen and Nielsen (2007)) implicitly or explicitly assume that decision trees are always EF although this is not strictly necessary – see for example Smith and Thwaites (2008a), Smith (2010). In an EF decision tree, variables appear in an order consistent with that in which they are observed, so a DM will know the outcomes of all random and decision variables encountered on any path from the root to a decision vertex before (s)he makes the decision at that vertex. Note

that in all the standard types of decision tree the order in which the decision variables appears is the order in which the DM has to make decisions.

In this paper we define a general DEG, but concentrate in particular on Extensive form DEGs. We note however that as with IDs there can be gains from constructing DEGs in non-EF order. *Causal* DEGs where variables appear in the order in which they happen, often give a better picture of the conditional independence structure of a problem, and if produced early enough in a problem analysis often admit the discovery of a simpler equivalent representation which can be solved (using a new EF DEG) more efficiently and more transparently. In Section 6 we exploit this flexibility in DEG-representation to show how the topology of a DEG can be simplified under certain conditions.

In common with most decision analysis practitioners, we make assumptions about our DM. We assume firstly that the only uncertainties (s)he has regarding the problem are uncertainties about outcomes of chance variables, and not uncertainties regarding the structure of the problem (as described by the topology of the tree). Secondly we assume that (s)he will always make decisions in order to maximise her/his expected utility. Lastly we assume (s)he is *parsimonious* in that (s)he will disregard information which she knows cannot help her to increase her/his expected utility.

The plan of the remainder of the paper is as follows. The next section discusses decision trees and conditional independences thereon. DEGs are formally defined in Section 3, and their conditional independence properties are related to those of the tree. We then present the algorithm on the DEG for finding an optimal decision policy. This is followed by a worked example that exhibits several asymmetries. Some theoretical results are then presented on conditions that allow the further simplification of a DEG structure before finding the optimal decision policy.

2. Decision trees and conditional independence

The DEG is a close relative of the decision tree, and we devote this section to a brief discussion of this more well-known graph, before proceeding to a definition of a DEG in Section 3.

Decision trees are often constructed so that the leaf nodes are utility nodes to which the entire

utility of the root-to-leaf path is attached. But there are particular forms of the utility function which allow the analyst to use a different construction. A utility function can be *decomposable* (Keeney and Raiffa (1976)), if for example it is *additive* $\left(u(x) = \sum_i u_i(x_i) \right)$. In this case some of the utility attached to the leaf nodes can be reassigned to edges on the root-to-leaf paths. It is usually the case that those components of the utility reassigned to edges are *costs* (non-positive) and those that remain attached to the leaf nodes are *rewards* (non-negative), but this is not a necessary condition. It is **not** common practice to create any utility nodes between the root node and the leaf nodes, intermediate utilities being assigned to edges leaving chance or (more usually) decision nodes.

Each way of adding utilities to the tree has its own merit. Analysis of trees where the entire utility associated with a root-to-leaf path is assigned to a leaf node is straightforward. Associating utilities with edges often allows for a more transparent description of the process, and can speed analysis by keeping the evaluation of policies modular. Both forms are included in the specification in this section.

2.1. Specification of decision trees

A general decision tree $\mathcal{T}$ is a directed, rooted tree, with vertex set $V(\mathcal{T}) = V_D(\mathcal{T}) \cup V_C(\mathcal{T}) \cup V_U(\mathcal{T})$ and edge set $E(\mathcal{T}) = E_D(\mathcal{T}) \cup E_C(\mathcal{T})$. The root-to-leaf paths $\{\lambda\}$ of $\mathcal{T}$ label the different possible unfoldings of the problem.

Each vertex $v \in V_C(\mathcal{T})$ serves as an index of a random variable $X(v)$ whose values describe the next stage of possible developments of the unfolding process. The state space $\mathbb{X}(v)$ of $X(v)$ is identified with the set of directed edges $e(v, v') \in E_C(\mathcal{T})$ emanating from v $(v' \in V(\mathcal{T}))$. For each $X(v)$ $(v \in V_C(\mathcal{T}))$ we let

$$\Pi(v) \equiv \{P(X(v) = e(v, v')) \mid e(v, v') \in \mathbb{X}(v)\}.$$

Each edge $e(v, v') \in E_C(\mathcal{T})$ also has a *utility* $(v, v')[\mathfrak{u}]$ associated with it (which may be zero).

Each vertex $v \in V_D(\mathcal{T})$ serves as an index of a decision variable, whose states describe the possible choices available to the DM at this point in the process. The outcome space of this variable is

identified with the set of directed edges $e(v,v') \in E_D(\mathcal{T})$ emanating from v ($v' \in V(\mathcal{T})$). Each edge $e(v,v') \in E_D(\mathcal{T})$ has a *utility* $(v,v')[\mathfrak{u}]$ associated with it (which may be zero). Similarly each vertex $v \in V_U(\mathcal{T})$ is a leaf-vertex, and has a *utility* associated with it.

If the sets of edges identified by the outcome spaces of the variables v^1, v^2 (both members of $V_C(T)$ or both members of $V_D(T)$) label the same set of events conditioned on different histories, then there exists a bijection $\psi(v^1, v^2)$ between them which maps the edge leaving v^1 labelling a specific event onto the edge leaving v^2 labelling this same event.

2.2. Coloured decision trees

The *coloured decision tree* is a logical extension of a decision tree for identifying conditional independence structure. We lengthen each branch of $\mathcal{T}$ by the addition of a single edge emanating from each $v \in V_U(\mathcal{T})$, and transfer to this *terminal* edge the utility attached to the original leaf node. We then introduce three further properties — *coloured* edges, *stages* and *positions*. These are defined formally in Definition 1, but the process by which a coloured decision tree is produced is illustrated in full detail in the example which immediately succeeds the definition.

Definition 1 *The* stages, colouring *and* positions *of a decision tree are defined as follows:*

1. *Two chance nodes $v^1, v^2 \in V_C(\mathcal{T})$ are in the same* stage u *if there is a bijection $\psi(v^1, v^2)$ between $\mathbb{X}(v^1)$ and $\mathbb{X}(v^2)$ such that if $\psi : e(v^1, v^{1\prime}) \mapsto e(v^2, v^{2\prime})$ then $P(X(v^1) = e(v^1, v^{1\prime})) = P(X(v^2) = e(v^2, v^{2\prime}))$, and $(v^1, v^{1\prime})[\mathfrak{u}] = (v^2, v^{2\prime})[\mathfrak{u}]$.*

The edges $e(v^1, v^{1\prime})$ and $e(v^2, v^{2\prime})$ have the same colour *if v^1 and v^2 are in the same stage, and $e(v^1, v^{1\prime})$ maps to $e(v^2, v^{2\prime})$ under this bijection $\psi(v^1, v^2)$.*

2. *Two decision nodes $v^1, v^2 \in V_D(\mathcal{T})$ are in the same* proto-stage *if there is a bijection $\psi(v^1, v^2)$ between their outcome spaces such that if $\psi : e(v^1, v^{1\prime}) \mapsto e(v^2, v^{2\prime})$ then $(v^1, v^{1\prime})[\mathfrak{u}] = (v^2, v^{2\prime})[\mathfrak{u}]$.*

The edges $e(v^1, v^{1\prime})$ and $e(v^2, v^{2\prime})$ have the same colour *if v^1 and v^2 are in the same proto-stage, and $e(v^1, v^{1\prime})$ maps to $e(v^2, v^{2\prime})$ under this bijection $\psi(v^1, v^2)$.*

Two decision nodes $v^1, v^2 \in V_D(\mathcal{T})$ are in the same stage u *if a DM acting in accordance with our assumptions would employ a decision rule which was not a function of whether the process had reached v^1 or v^2.*

3. *Two utility nodes* $v^1, v^2 \in V_U(\mathcal{T})$ *are in the same* stage u *if their emanating edges carry the same utility.*

The edges emanating from v^1, v^2 *then have the same* colour.

4. *Two vertices* $v^1, v^2 \in V(\mathcal{T})$ *are in the same* position w *if for each subpath emanating from* v^1*, the ordered sequence of colours is the same as that for some subpath emanating from* v^2*.*

The set of stages of the tree is labelled $L(\mathcal{T})$*, and the set of positions is labelled* $K(\mathcal{T})$*.*

Example

Consider the decision tree in Figure 1. Here the DM must make an initial choice between two actions a and b, neither of which has an immediate cost. (S)he may need to make a second choice, between two actions c and d, the latter action having an immediate cost of 10 units. The development of the problem depends at various points on random variables (associated with chance nodes) whose distributions depend on the history of the problem up to that point. Each possible unfolding of the problem has its own associated reward.

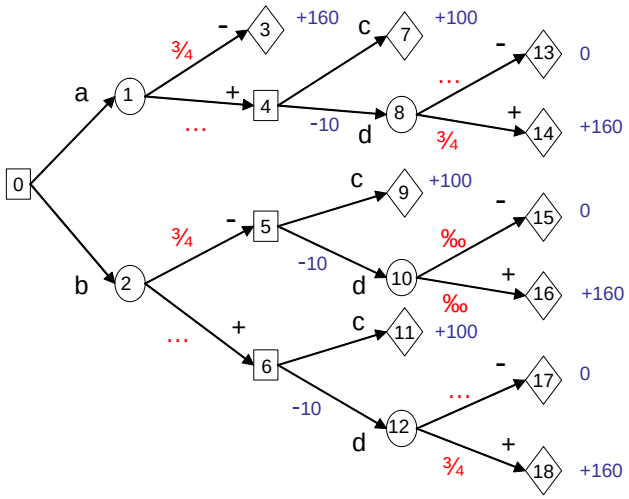

Figure 1 **Unmodified decision tree**

Using Definition 1, we see that

- chance nodes 1 and 2 are in the same stage, but not the same position,

- decision nodes 4, 5 and 6 are in the same proto-stage,
- decision nodes 4 and 6 are in the same stage and in the same position,
- chance nodes 8 and 12 are in the same stage and in the same position,
- utility nodes 3, 14, 16 and 18 are in the same stage and in the same position,
- as are utility nodes 7, 9 & 11 and utility nodes 13, 15 & 17.

The resultant coloured decision tree is given in Figure 2

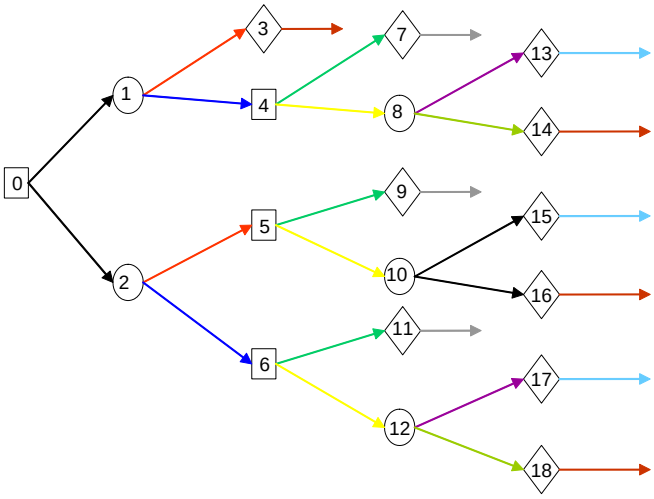

Figure 2 Coloured decision tree

It is useful here to consider what is meant by $v^1, v^2 \in V_D(\mathcal{T})$ being in the same stage in a little more detail. If a DM acts in accordance with our assumptions and employs a decision rule which is not a function of whether the process has reached v^1 or v^2, then either the difference between the vertices is *inconsequential* or the vertices are *indistinguishable* to her/him. By *inconsequential* we mean that the consequences (in terms of further costs and rewards) of choosing the same action at each vertex are identical. The difference between two vertices in $V_D(\mathcal{T})$ is *inconsequential* if they are in the same position. By *indistinguishable* to the DM we mean that (s)he has no information available to her/him to distinguish between whether the process has unfolded so as to reach the vertex v^1 or so as to reach the vertex v^2.

This definition applies to general decision trees. But in an EF tree no pair of vertices $v^1, v^2 \in V_D(\mathcal{T})$ are ever *indistinguishable* to the DM as (s)he always knows the outcomes of all variables encountered prior to making a decision. Hence in an EF tree no pair of decision nodes are ever in the same stage but not the same position, and we can ignore this part of the definition. This is **not** true of other types of decision tree.

Clearly any collection of chance nodes in the same position are in the same stage, and any utility nodes in the same stage are necessarily in the same position.

2.3. Conditional independence structure

This coloured decision tree provides us with a formal framework for the DEG representation and propagation rules given in Section 3 and Section 4. It also allows us to read the conditional independence structure of our problem from the tree, a hitherto almost impossible task.

Dawid (2001) discusses in detail the concepts of conditional independence and irrelevance in the fields of Probability Theory, Statistics, Possibility Theory and Belief Functions. We show how they can be interpreted for coloured decision trees. So consider a set of variables defined on our tree whose outcome spaces can be identified with the set of subpaths connecting two positions or stages, and which may be neither wholly random variables nor wholly decision variables.

For $w \in K(\mathcal{T})$, let $Z(w)$ be the variable whose outcome space $\mathbb{Z}(w)$ can be identified with the set of subpaths joining v_0 to some $v \in w$. In general each of these subpaths will consist of a collection of chance and decision edges.

For $u \in L(\mathcal{T})$, let $Z(u)$ be the variable whose outcome space $\mathbb{Z}(u)$ can be identified with the set of subpaths joining v_0 to some $v \in u$.

Let $Y(w)$ be the variable whose outcome space $\mathbb{Y}(w)$ can be identified with the set of subpaths emanating from some $v \in w$ and ending in a *terminal* edge. In general each of these subpaths will consist of a collection of chance and decision edges, plus the one terminal edge.

Also, for $v \in V_C(\mathcal{T}),\ v \in u \in L(\mathcal{T})$, let $X(u)$ be the random variable whose values describe the next possible developments in the unfolding process. The state space $\mathbb{X}(u)$ of $X(u)$ can be identified

with the set of directed edges $e(v,v') \in E_C(\mathcal{T})$ emanating from each $v \in u$. For each $X(u)$ we let $\Pi(u) \equiv \{P(X(u) = e(v,v') \mid v \in u\}$.

Lastly, let $\Lambda(w)$ be the event which is the union of all root-to-leaf paths of the tree passing through some $v \in w$; and $\Lambda(u)$ be the event which is the union of all root-to-leaf paths passing through some $v \in u$.

There are two basic collections of conditional independence statements that we can read off an EF tree:

(A) For any $w \in K(\mathcal{T})$, we can write

$$Y(w) \amalg Z(w) \mid \Lambda(w)$$

which can be read as:

1. *the probability and utility distributions on the edges emanating from any chance node encountered on any subpath $y(w) \in \mathbb{Y}(w)$ are not dependent on the vertex $v \in w$ reached*;

2. *the utility distribution on the edges emanating from any decision node encountered on any subpath $y(w)$, and the* **consequences** *(in terms of further costs and rewards) of any decision made at that decision node are not dependent on the vertex $v \in w$ reached*;

3. *the utility associated with the edge leaving the utility node on any subpath $y(w)$ is not dependent on the vertex $v \in w$ reached.*

So, if we know that a unit has reached some position w, then we do not need to know how our unit reached w (ie. along which $v_0 \to v \in w$ subpath) in order to make predictions about or decisions concerning the future of the process (ie. along which subpath emanating from $v \in w$ our unit is going to pass).

(B) For any $v \in V_C(\mathcal{T}),\ v \in u \in L(\mathcal{T})$, we can write

$$X(u) \amalg Z(u) \mid \Lambda(u)$$

For a general tree we can also define $X(u)$ for decision stages, but as in an EF tree there are no decision stages which are not positions, we can effectively restrict our analysis to stages consisting

of chance nodes, and interpret this statement as follows: *The probability and utility distributions on the edges emanating from each $v \in u$ are are not dependent on the vertex $v \in u$ reached.*

So, if we know that a unit has reached some stage u (consisting of chance nodes), then we do not need to know how our unit reached u (ie. along which $v_0 \rightarrow v \in u$ subpath) in order to predict how the process is going to unfold in the immediate future (ie. by which edge emanating from $v \in u$ our unit leaves by).

3. Decision Event Graphs

A DEG is a connected directed acyclic graph which is a function of a decision tree. It differs from the tree in having a single sink node, with every edge lying on some directed path from the root node to the sink node. Additionally, intermediate nodes in the DEG may have multiple incoming edges, whereas in the tree these nodes have only one incoming edge. As with trees, the graphical structure is annotated with the numerical specification of the decision problem.

Definition 2 *The Decision Event Graph $\mathcal{D}$ (a function of a tree $\mathcal{T}$ with positions $K(\mathcal{T})$ and stages $L(\mathcal{T})$) is the coloured directed graph with vertex set $V(\mathcal{D})$ and edge set $E(\mathcal{D})$, defined by:*

1. $V(\mathcal{D}) = K(\mathcal{T}) \cup \{w_\infty\}$.

The set $V(\mathcal{D}) \setminus \{w_\infty\}$ is partitioned into decision, chance *and* utility *positions. The subsets of these positions are labelled $V_D(\mathcal{D})$, $V_C(\mathcal{D})$ and $V_U(\mathcal{D})$ respectively.*

2. *(a) For $w, w' \in V(\mathcal{D}) \setminus \{w_\infty\}$, there exists a directed edge $e(w, w') \in E(\mathcal{D})$ iff there are vertices $v, v' \in V(\mathcal{T})$ such that $v \in w \in K(\mathcal{T})$, $v' \in w' \in K(\mathcal{T})$ and there is an edge $e(v, v') \in E(\mathcal{T})$.*

(b) For each $w \in V_U(\mathcal{D})$ there exists a single directed edge $e(w, w_\infty) \in E(\mathcal{D})$.

The set $E(\mathcal{D})$ is partitioned into decision, chance *and* terminal *edges, labelled by whether they emanate from* decision, chance *or* utility *positions. The subsets of these edges are labelled $E_D(\mathcal{D})$, $E_C(\mathcal{D})$ and $E_U(\mathcal{D})$ respectively.*

3. *If $v^1 \in w^1 \in K(\mathcal{T})$, $v^2 \in w^2 \in K(\mathcal{T})$, and v^1, v^2 are members of the same stage $u \in L(\mathcal{T})$, then we say that w^1, w^2 are in the same stage u and assign the same colour to these positions. We label the set of stages of $\mathcal{D}$ by $L(\mathcal{D})$.*

4. *(a) If* $v \in w \in K(\mathcal{T})$, $v' \in w' \in K(\mathcal{T})$ *and there is an edge* $e(v, v') \in E(\mathcal{T})$, *then the edge* $e(w, w') \in E(\mathcal{D})$ *has the same colour as the edge* $e(v, v')$.

(b) If $v \in w \in K(\mathcal{T})$, $w \in V_U(\mathcal{D})$, *then the edge* $e(w, w_\infty) \in E(\mathcal{D})$ *has the same colour as the edge emanating from* $v \in V_U(\mathcal{T})$.

Example cont.

Figure 3 shows the DEG produced from the coloured decision tree in Figure 2 using Definition 2.

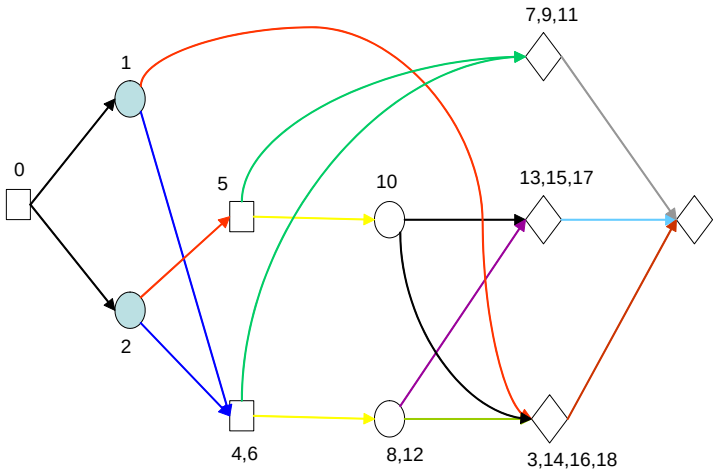

Figure 3 Unannotated DEG

As noted in the introduction, the DEG-representation of a problem may have very different topologies depending on which type of tree it has been derived from. The EF DEG in particular is a good description of how the problem appears to the DM and provides an ideal framework for calculating optimal decisions. Note that in an EF DEG there are no decision nodes which are in the same stage but not the same position.

3.1. Conditional independence structure

We can read the conditional independence structure of a problem from a DEG even more readily than from a coloured extended tree. As the vertices of the DEG are *positions*, this structure is embedded in the topology of the DEG. As for the EF tree, there are two basic collections of

conditional independence statements that we can read off an EF DEG.

Consider variables defined on our DEG, analogous to those we have defined on our tree, which have outcome spaces identified with the sets of subpaths connecting two positions or stages.

For $w \in V(\mathcal{D}) \setminus \{w_0, w_\infty\}$, let $Z(w)$ be the variable having outcome space $\mathbb{Z}(w)$ identified with the set of subpaths joining w_0 to w. For $w \in u \in L(\mathcal{D})$, let $Z(u)$ be the variable having outcome space $\mathbb{Z}(u)$ identified with the union over all $w \in u$ of the sets of subpaths joining w_0 to w. Let $Y(w)$ be the variable having outcome space $\mathbb{Y}(w)$ identified with the set of subpaths joining w to w_∞. Also, for $w \in V_C(\mathcal{D}),\ w \in u \in L(\mathcal{D})$, let $X(u)$ be the random variable whose values describe the next possible developments in the unfolding process. The state space $\mathbb{X}(u)$ of $X(u)$ can be identified with the set of directed edges $e(w, w') \in E_C(\mathcal{D})$ emanating from each $w \in u$. For each $X(u)$ we let $\Pi(u) \equiv \{P(X(u) = e(w, w') \mid w \in u\}$. Lastly, let $\Lambda(w)$ be the event which is the union of all $w_0 \to w_\infty$ paths passing through w; and $\Lambda(u)$ be the event which is the union of all $w_0 \to w_\infty$ paths passing through some $w \in u$. Then:

(A) For any $w \in V(\mathcal{D}) \setminus \{w_0, w_\infty\}$, we can write

$$Y(w) \amalg Z(w) \mid \Lambda(w)$$

which can be read as:

1. *the probability and utility distributions on the edges emanating from any chance node encountered on any subpath $y(w) \in \mathbb{Y}(w)$ are independent of the subpath taken from w_0 to w;*

2. *the utility distribution on the edges emanating from any decision node encountered on any subpath $y(w)$, and the* **consequences** *of any decision made at that decision node are independent of the subpath taken from w_0 to w;*

3. *the utility associated with the edge leaving the utility node on any subpath $y(w)$ is independent of the subpath taken from w_0 to w.*

If there are no decision nodes on any $w \to w_\infty$ subpath, then this expression takes its standard meaning for CEGs (Smith et al. (2009)) of

$$\forall\ y(w) \in \mathbb{Y}(w),\ z(w) \in \mathbb{Z}(w), \quad P(y(w) \mid z(w), \Lambda(w)) = P(y(w) \mid \Lambda(w))$$

This can be expressed as a conditional probability statement even though if there are decision nodes on any $w_0 \to w$ subpath, the joint probability $P(z(w), \Lambda(w))$ is undefined.

So, if we know that a unit has reached some position w, then we do not need to know how our unit reached w in order to make predictions about or decisions concerning the future of the process.

(B) For any $w \in V_C(\mathcal{D}),\ w \in u \in L(\mathcal{D})$, we can write

$$X(u) \amalg Z(u) \mid \Lambda(u)$$

For a general DEG we can also define $X(u)$ for decision stages, but as in an EF DEG there are no decision stages which are not positions, we can again restrict our analysis to stages consisting of chance nodes, and interpret this statement as follows: *The probability and utility distributions on the edges emanating from each $w \in u$ are independent of both the position w reached, and the subpath taken from w_0 to w.*

As we are considering only stages made up of chance nodes, we can write

$$\forall\ x(u) \in \mathbb{X}(u),\ z(u) \in \mathbb{Z}(u), \quad P(x(u) \mid z(u), \Lambda(u)) = P(x(u) \mid \Lambda(u))$$

Again this is a valid expression even if there are decision nodes on $w_0 \to w \in u$ subpaths.

So, if we know that a unit has reached some stage u (consisting of chance nodes) in our EF DEG, then we do not need to know how our unit reached u in order to predict how the process is going to unfold in the immediate future.

4. An optimal decision rule algorithm for Decision Event Graphs

Finding the optimal decision rule on an EF DEG is similar to that of finding it on an EF decision tree. The computation proceeds backwards from the sink node to the root node. First one finds a topological ordering of all the positions of the DEG $\mathcal{D}$. Then one traverses the positions in the reverse of the ordering, starting from the sink until the root is reached, carrying out local calculations that depend on the type of the position. The sink node is given a utility value of zero. Thereafter the position could be of chance, decision or utility type. If the position is a utility node, assign it the utility value of the edge connecting it to the sink node. Alternatively, if the position

is a chance node, then assign a utility value to that position equal to the sum over all child nodes of the product of the probability associated with the edge and the sum of the utility of the child node and the utility associated with the edge (the latter has a default zero if not present). The final possibility is that the position is a decision node, in which case we assign a utility value to that position equal to the maximum value over all child nodes, of the sum of the utility of the child node and the utility value associated with the edge connecting the decision node and the child node. Mark all child edges that do not achieve this maximum value as being sub-optimal.

At the end of the local message passing, the root node will contain the maximum expected utility, and the optimal decision rule will consist of the subset of edges that have not been marked as being sub-optimal. The propagation algorithm is illustrated in the pseudocode below, Table 1. For the pseudocode we use the following notation. V_C (abbreviated from $V_C(\mathcal{D})$), V_D and V_U represent respectively the sets of chance, decision and utility nodes. The utility part of a position w will be denoted by $w[\mathfrak{u}]$. Similarly the probability part of an edge $e(w, w')$ from position w to position w' will be denoted by $(w, w')[\mathfrak{p}]$ and the utility part by $(w, w')[\mathfrak{u}]$. The set of child nodes of a position w is denoted by $ch(w)$.

Table 1 Pseudocode for the backward induction algorithm for finding optimal decision sequence

- Find a topological ordering of the positions. Without loss of generality call this $w_0, w_1, \ldots, w_n$, so that w_0 is the root node, and w_n is the sink node $(= w_\infty)$.
- Initialize the utility value $w_n[\mathfrak{u}]$ of the sink node w_n to zero.
- Iterate: for $i = n-1$ step minus 1 until $i = 0$ do:
  - If $w_i \in V_U$ then $w_i[\mathfrak{u}] = (w_i, w_n)[\mathfrak{u}]$
  - If $w_i \in V_C$ then $w_i[\mathfrak{u}] = \sum_{w' \in ch(w_i)} (w_i, w')[\mathfrak{p}] * (w'[\mathfrak{u}] + (w_i, w')[\mathfrak{u}])$
  - If $w_i \in V_D$ then $w_i[\mathfrak{u}] = \max_{w' \in ch(w_i)} (w'[\mathfrak{u}] + (w_i, w')[\mathfrak{u}])$.

    Mark the sub-optimal edges.

Notice that in *specifying* the DEG, probability and utility values are associated with the *edges*,

and that in the backward induction these are unchanged. Instead the algorithm associates utility values to the receiving nodes.

4.1. A proof of the algorithm

The basic strategy of the proof is to "unwind" the DEG into a decision tree, and show equivalence of local operations on the decision tree to those on the DEG.

Let $\mathcal{T}$ denote the underlying EF decision tree of the DEG. Suppose that the i-th position w_i of the DEG has m_i distinct paths from the root node to w_i; then in $\mathcal{T}$ there will be m_i vertices $\{v_{ij} : j = 1, \ldots, m_i\}$ that are coalesced into w_i when forming the DEG from $\mathcal{T}$. Call the number m_i the *multiplicity* of w_i. Note that the root node has multiplicity 1, and that the multiplicity of the sink node in the DEG counts the number of leaf nodes on the tree. In the tree $\mathcal{T}$ each of the m_i decision subtrees rooted at the m_i vertices in the set $\{v_{ij} : j = 1, \ldots, m_i\}$ are isomorphic.

Now consider solving the decision problem on the decision tree $\mathcal{T}$. This proceeds by backward induction (Raiffa and Schlaifer 1961) in the same way as in Table 1. Any topological ordering of the vertices of the tree will do, so let us take the following ordering.

$$v_{0,1}, v_{1,1}, v_{1,2} \ldots, v_{1,m_1}, v_{2,1}, \ldots, v_{2,m_2}, \ldots, v_{n,1} \ldots, v_{n,m_n}.$$

We modify the propagation algorithm to that shown in Table 2 for the decision tree $\mathcal{T}$.

Now because the decision subtrees rooted at the m_i vertices $\{v_{ij} : j = 1, \ldots, m_i\}$ are isomorphic, it follows that $v_{i,j}[\mathfrak{u}] = v_{i,k}[\mathfrak{u}]$ for each $j, k \in \{1, \ldots, m_i\}$. By a simple mathematical induction argument, it also follows that $w_i[\mathfrak{u}] = v_{i,j}[\mathfrak{u}]$, where $w_i[\mathfrak{u}]$ is found from the local propagation algorithm on the DEG. The induction argument is as follows. Clearly the statement is true for $i = n$, that is, $w_n[\mathfrak{u}] = v_{n,j}[\mathfrak{u}]$. Assume that it is also true for $i = I+1, I+2, \ldots, n$. Then in the tree we carry out:

- If $w_I \in V_C$ then:

  For $j = 1$ step 1 until m_I do $v_{I,j}[\mathfrak{u}] = \sum_{v' \in ch(v_{I,j})} (v_{I,j}, v')[\mathfrak{p}] * (v'[\mathfrak{u}] + (v_{I,j}, v')[\mathfrak{u}])$

- If $w_I \in V_D$ then:

  For $j = 1$ step 1 until m_I do $v_{I,j}[\mathfrak{u}] = \max_{v' \in ch(v_{I,j})} (v'[\mathfrak{u}] + (v_{I,j}, v')[\mathfrak{u}])$.

Table 2 Backward induction algorithm for the decision tree

- Use the topological ordering $v_{0,1}, v_{1,1}, v_{1,2} \ldots, v_{1,m_1}, v_{2,1}, \ldots, v_{2,m_2}, \ldots, v_{n,1} \ldots, v_{n,m_n}$.
- Initialize the utility values $\{v_{n,j}[\mathfrak{u}] \mid j = 1 \ldots m_n\}$ of all leaf nodes to zero.
- Iterate: for $i = n-1$ step minus 1 until $i = 0$ do:
  - —If $w_i \in V_U$ then
    - ∗ For $j = 1$ step 1 until m_i do $v_{i,j}[\mathfrak{u}] = (v_{i,j}, v_{n,k})[\mathfrak{u}]$ where $v_{n,k}$ is the child leaf node of $v_{i,j}$ in the tree.
  - —If $w_i \in V_C$ then
    - ∗ For $j = 1$ step 1 until m_i do $v_{i,j}[\mathfrak{u}] = \sum_{v' \in ch(v_{i,j})} (v_{i,j}, v')[\mathfrak{p}] * (v'[\mathfrak{u}] + (v_{i,j}, v')[\mathfrak{u}])$
  - —If $w_i \in V_D$ then
    - ∗ For $j = 1$ step 1 until m_i do $v_{i,j}[\mathfrak{u}] = \max_{v' \in ch(v_{i,j})} (v'[\mathfrak{u}] + (v_{i,j}, v')[\mathfrak{u}])$.
    - ∗ Mark the sub-optimal edges.

But by the topological ordering, each $v' \in ch(v_{I,j})$ is some $v_{k,l}$ vertex for some $k \geq I+1$ and $l \in \{1, \ldots m_k\}$, hence the utility associated with that node is by the induction assumption equal to that on the w_k node in the DEG propagation algorithm. Additionally by construction the edges emanating from w_I in the DEG are in a one-to-one equivalence with the edges (and have the same probabilities if w_I is a chance node) emanating from any $v_{I,j}$ in the tree.

Hence each j loop in the above algorithm snippet is equivalent to

- If $w_I \in V_C$ then:

  For $j = 1$ step 1 until m_i do $v_{I,j}[\mathfrak{u}] = \sum_{w' \in ch(w_I)} (w_I, w')[\mathfrak{p}] * (w'[\mathfrak{u}] + (w_i, w')[\mathfrak{u}])$
- If $w_I \in V_D$ then:

  For $j = 1$ step 1 until m_i do $v_{I,j}[\mathfrak{u}] = \max_{w' \in ch(w_I)} (w'[\mathfrak{u}] + (w_I, w')[\mathfrak{u}])$.

But this is identical to the corresponding $i = I$-th loop in the DEG propagation algorithm, and so $w_I[u] = v_{I,j}[u]$ for any $j \in \{1, \ldots m_I\}$ which proves the induction argument, and thus proves the correctness of the propagation algorithm for DEGs in Table 1. □

5. An example

We here present a complex problem involving forensic analysis of samples taken from a crime scene, which illustrates the advantages of the DEG in the representation and analysis of asymmetric decision problems. The story is as follows:

The police have a suspect whom they believe broke a window and committed a burglary. They have the option of asking the forensic service to examine the suspect's clothing for fragments of glass from the broken window. If the suspect is indeed the culprit (an event the police give a 0.8 *probability to), then the probabilities that a fragment of glass will be found on his coat or pants are* 0.75 *and* 0.4 *respectively, these events being independent. If the suspect is not the culprit then the probability of a match is zero. The probability of getting a conviction with a forensic match is* 0.9*, and without a match is* 0.15*.*

The costs for analysing the coat and pants are \$1000 *and* \$800*, or if they are analysed together* \$1700*. The cost of taking the suspect to court is* \$2600*.*

The police utility has components of financial cost x *and* probability of conviction y*, and takes the form*

$$U = 25 - \frac{x}{200} + 80y$$

At various possible points in this procedure the police have to decide from a collection of possible actions — we denote these by

$a_\vee$	release the suspect
$a^\wedge$	go to court
a_c	test the coat
a_p	test the pants
a_2	test both together

There are 22 different decision / outcome possibilities, each with its own utility. These are listed in Table 3.

The decision tree representation of this problem contains 11 decision nodes, 5 chance nodes, 37 edges and 22 utility nodes. Finding an optimal decision rule requires a different calculation (a weighted average or a maximum value) at each of the sixteen chance and decision nodes.

The DEG can be created from the information provided, utilising the symmetries of the problem. There is no need to draw the decision tree first, or to calculate U for each scenario as in Table 3. In the DEG there are 22 root-to-sink paths corresponding to the 22 different decision / outcome possibilities, but there are now only 5 decision nodes and 4 utility nodes. Some preprocessing of probabilities does need to be done, but this is exactly the same as required for the decision tree.

These processed probabilities are added to the edges leaving the chance nodes, and the utilities of the 22 possibilities can be *additively* decomposed as *costs* on the edges leaving decision nodes and *rewards* on the edges leaving utility nodes. The resultant DEG is given in Figure 4. This is an EF DEG, hence there are no decision stages which are not positions. There are also here no chance stages which are not positions. When a DEG obeys both of these conditions we call it *simple* (see Thwaites et al. (2008) for the analogous *simple* CEG), and we can suppress the colouring of edges and vertices. We have found that a high proportion of problems yield EF DEGs which are simple.

The DEG representation has two immediate advantages over the tree — it is much more compact; and the symmetries of the problem (obscured in the tree representation) become apparent. Note also that by decomposing the utilities of the root-to-leaf paths and adding them to appropriate edges, we see exactly what they represent — the *cost* of analysing the coat is 5 units ($\$1000 \div 200$) etc. The *reward* of taking the suspect to court when there is no positive forensic evidence against him is 24 units, as opposed to the 25 units for releasing him. The *reward* for taking him to court

Table 3 The 22 paths and their utilities

	x	y	U
$a_\vee$	0	0	25
$a^\wedge$	2600	0.15	24
$a_c + a_\vee$	1000	0	20
$a_c + a^\wedge$	3600	0.90	79
$a_c - a_\vee$	1000	0	20
$a_c - a^\wedge$	3600	0.15	19
$a_c - a_p + a_\vee$	1800	0	16
$a_c - a_p + a^\wedge$	4400	0.90	75
$a_c - a_p - a^\wedge$	4400	0.15	15
$a_c - a_p - a_\vee$	1800	0	16
$a_p + a_\vee$	800	0	21
$a_p + a^\wedge$	3400	0.90	80
$a_p - a_\vee$	800	0	21
$a_p - a^\wedge$	3400	0.15	20
$a_p - a_c + a_\vee$	1800	0	16
$a_p - a_c + a^\wedge$	4400	0.90	75
$a_p - a_c - a^\wedge$	4400	0.15	15
$a_p - a_c - a_\vee$	1800	0	16
$a_2 - a_\vee$	1700	0	16.5
$a_2 - a^\wedge$	4300	0.15	15.5
$a_2 + a_\vee$	1700	0	16.5
$a_2 + a^\wedge$	4300	0.90	75.5

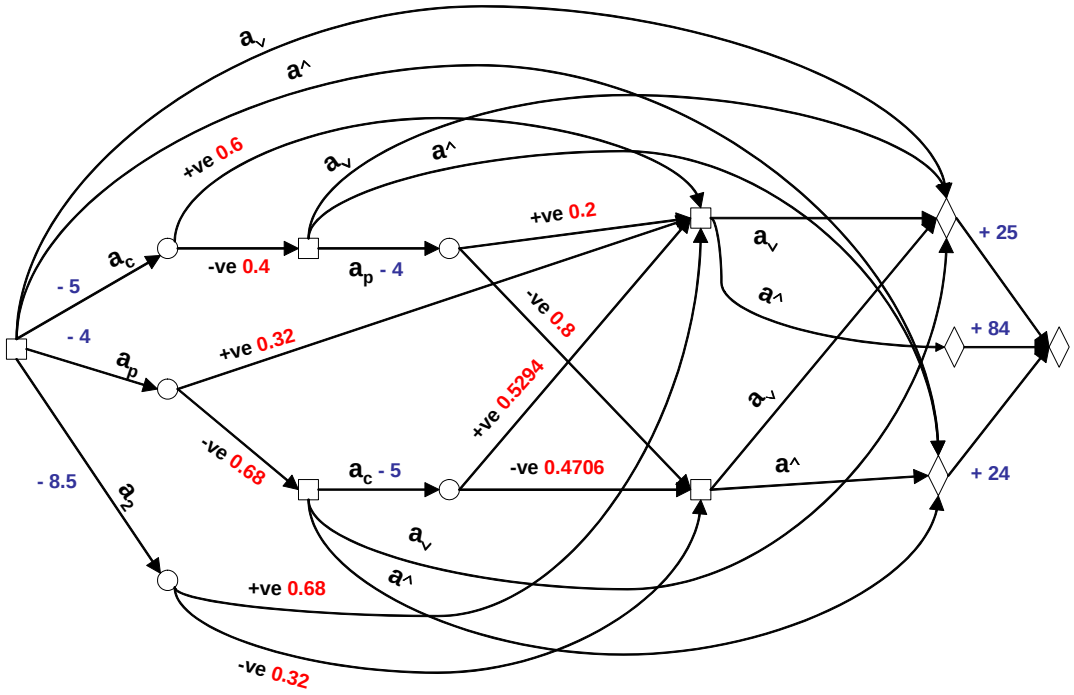

Figure 4 **Decision event graph for the forensic example**

when there is positive forensic evidence is 84 units.

Note that all edges representing a $+ve$ test result converge in a single decision node. We can read this position (using the expression $Y(w) \amalg Z(w) \mid \Lambda(w)$ from Section 3) as — *the future outcome of the investigation is independent of the test choices made, given that one of the tests has a* $+ve$ *result.* Similarly, edges representing a second $-ve$ test result (including a $-ve$ result when test both together) converge in a single decision node, indicating that *the future outcome of the investigation is independent of the test choices made, given that both items of clothing have tested* $-ve$. All edges representing *release* converge in a single utility node, indicating that *the reward for releasing the suspect is independent of any other decisions made, and of any test results obtained.* All edges representing *go to court* converge in a single utility node, **except one**, indicating that *the reward for going to court is independent of any other decisions made, so long as there has not been a* $+ve$ *test result.*

The optimal rule is calculated by working backwards from the sink-node — here there are ten chance and decision nodes compared to the sixteen in the underlying tree. In Figure 5 suboptimal paths are marked and the optimal rule has been indicated in bold. The optimal rule, which gains a *reward* of 58.52 units, is to test the coat — if a positive result is obtained take the suspect to court, if not test the pants — if a positive result is obtained take the suspect to court, otherwise release him. Finding this rule and calculating the expected utility are clearly quicker than when performing the same calculations directly on the framework of the tree.

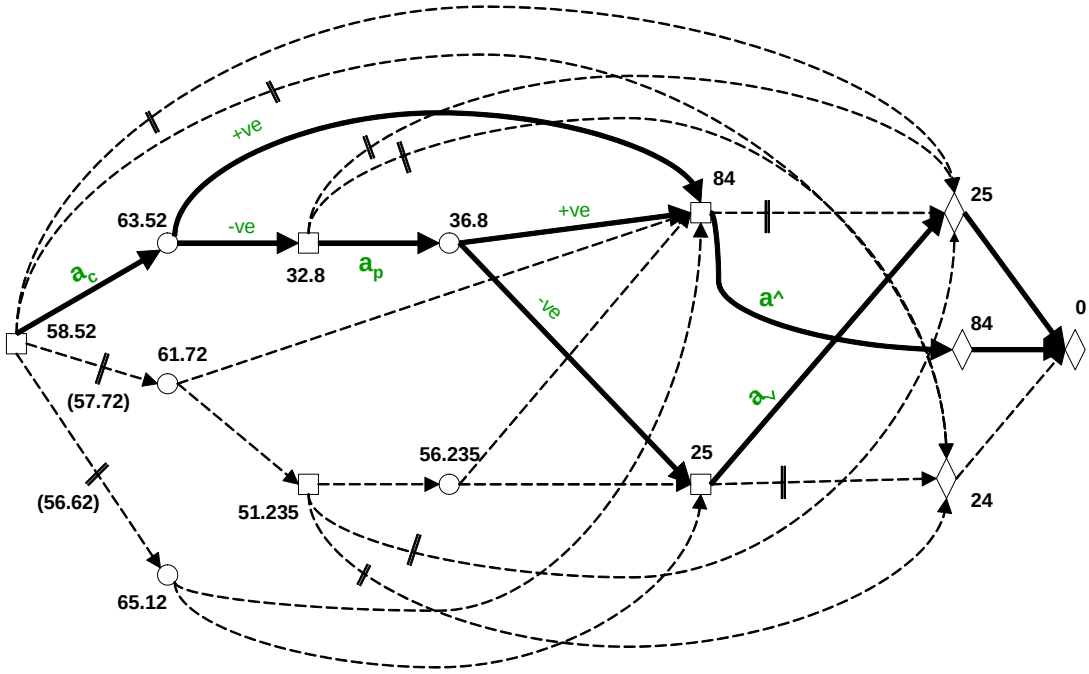

Figure 5 Decision event graph for the forensic example, showing optimal paths

The flexibility of the DEG is a direct consequence of the fact that the conditional independence structure of the problem is stored in the topology of the graph. It gives the DEG additional advantages over the tree. Thus suppose we need to examine the impact of changes in the utility function, and consider the cost of testing the coat rising to 7 units ($1400). Using the tree we would have to revise all branches utilising the edges leaving the root-node labelled a_c and a_p, and revise our optimal policy calculations over 13 of the 16 nodes. In the DEG the only calculations affected

are those connected with the nodes associated with testing the pants, testing the pants and getting a $-ve$ result, and the root-node. There is a similar gain in efficiency over the tree if we change, say, the probability of getting a positive result from testing the pants. This type of flexibility, exhibited by IDs for a much more restrictive class of decision problems, is a characteristic feature of DEGs.

6. Simplifying results

There are a number of highly useful results which allow us to increase the efficiency of our algorithm for finding optimal decision rules, and to simplify the specification of a decision process so that it expresses only attributes that might have a bearing on the choice of an optimal action. These allow the preprocessing of an elicited DEG, transforming it to one with a simpler topology which embodies all the information from the structure of the problem needed to discover an optimal policy. This is often a valuable step to make since the transformation allows us to deduce that certain quantitative information which we might elicit will have no impact on the consequences of our decisions. So we learn from the DEG which information need not be collected. This type of preprocessing is commonly performed when a decision problem is represented as an ID. However the analogous transformations of DEGs are more powerful and apply to a much wider range of decision problems. We illustrate some of these transformations below.

6.1. Path reduction

If every path from some position $w_1 \in V(\mathcal{D})$ passes through some subsequent position w_2, and every edge of every subpath $\mu(w_1, w_2)$ has zero associated utility, then the set of subpaths $\{\mu(w_1, w_2)\}$ can be replaced by a single edge $e(w_1, w_2)$ with zero associated utility. This highly useful result is expressed in the following lemma, for which we introduce some extra notation.

Let $S(w_1, w_2)$ be the set of subpaths connecting w_1 and w_2; $\Lambda(w_a, w_b)$ be the event which is the union of all $w_0 \to w_\infty$ paths passing through the positions w_a and w_b; $\Lambda(w_a, e)$ be the event which is the union of all $w_0 \to w_\infty$ paths passing through the position w_a and the edge e; $w \in \mu$ be a position lying on the subpath μ; and $e \in \mu$ be an edge lying on the subpath μ.

Lemma 1 *For a DEG $\mathcal{D}$, if every path from a position $w_1 \in V(\mathcal{D})$ $(\neq w_0)$ passes through some*

subsequent position $w_2 \in V(\mathcal{D})$ $(\neq w_\infty)$*, and every subpath* $\mu(w_1, w_2)$ *has zero associated utility, then there is a DEG* $\hat{\mathcal{D}}$ *where*

$$V(\hat{\mathcal{D}}) = V(\mathcal{D}) \setminus \{w \in \mu \mid w \neq w_1, w_2,\ \Lambda(w_1, w) \equiv \Lambda(w),\ \mu \in S(w_1, w_2)\}$$

$$E(\hat{\mathcal{D}}) = E(\mathcal{D}) \setminus \{e \in \mu \mid \Lambda(w_1, e) \equiv \Lambda(e),\ \mu \in S(w_1, w_2)\} \cup \hat{e}(w_1, w_2)$$

where $\hat{e}(w_1, w_2)$ *has zero associated utility, and such that an optimal decision rule for* $\hat{\mathcal{D}}$ *is an optimal decision rule for* $\mathcal{D}$*.*

This Lemma (for proof see the Appendix) is particularly useful when our DEG does not have utilities decomposed onto edges, but also has applications when there are utilities attached to edges. It has an analogue in *barren node deletion* (Shachter (1986)). If a problem can be represented by both a DEG and an ID, then the DEG can always be constructed so that the edges corresponding to the outcomes of a collection of barren nodes on the ID form collections of subpaths of the $w_0 \to w_\infty$ paths of the DEG. Moreover these subpaths correspond to vectors of outcomes of vertices in the ID which have no directed paths from them to the utility node, so these outcomes have no effect on the value of the utility function and consequently the utilities attached to each of these subpaths is zero.

There is a similar result when every path from a position $w_1 \in V(\mathcal{D})$ $(\neq w_0)$ passes through some subsequent position $w_2 \in V(\mathcal{D})$ $(\neq w_\infty)$, but the subpaths $\mu(w_1, w_2)$ have different associated utilities. The graph rooted in w_1 and having w_2 as a sink is itself a DEG. Optimal decision analysis can be performed separately on this sub-DEG and nested in the analysis for the full DEG.

These ideas can be used to reduce the length and complexity of collections of subpaths between non-adjacent nodes in isolation from other parts of the DEG. Once one collection has been tackled, other collections can be investigated, and this piecemeal approach suits asymmetric problems, as some parts of the graph may simplify more readily than others.

6.2. Parsimony

Definition 3 *For an EF DEG* $\mathcal{D}$*, a* parsimonious *decision maker is one who, when choosing an action at a decision node* $w \in V_D(\mathcal{D})$*, disregards all information on* $Z(w)$ *other than* $\Lambda(w)$ *(for all*

$w \in V_D(\mathcal{D})$*)*.

In an Extensive form ID we can uniquely specify the variables preceding and succeeding any decision variable D. Labelling these by $Z(D)$, $Y(D)$ respectively, a parsimonious DM using this EF ID is one who, if when arriving at a decision variable D and finding that

$$U \amalg R(D) \mid (D, Q(D)) \tag{1}$$

for some partition $Z(D) = R(D) \cup Q(D)$, ignores $R(D)$ (Smith (1996)). $R(D)$ are the *redundant* parents and $Q(D)$ are the *sufficient* parents of D.

It is reasonable to specify that the EF ID has had any barren nodes removed, and for such a graph, statement (1) is equivalent to

$$Y(D) \amalg R(D) \mid (D, Q(D)) \tag{2}$$

But if a DM ignores $R(D)$ then it is *irrelevant* for D (if $Q(D)$ is known). This can therefore be expressed as

$$D \amalg R(D) \mid Q(D) \tag{3}$$

But in any irrelevance system, $\cdot \amalg \cdot \mid \cdot$ must satisfy the property that $X \amalg (Y, Z) \mid W \Leftrightarrow \{X \amalg Y \mid (Z, W),\ \ X \amalg Z \mid W\}$ (Smith (1989b) Dawid (2001)), so expressions (2) and (3) both hold if and only if

$$(D, Y(D)) \amalg R(D) \mid Q(D) \equiv (D, Y(D)) \amalg Z(D) \mid Q(D) \tag{4}$$

Consider now an EF DEG of our problem, where the DM has to make a decision associated with D on each $w_0 \to w_\infty$ path. Then there exists a set $W \subset V_D(\mathcal{D})$ of decision positions associated with the variable D. An outcome of $Q(D)$ (the *sufficient* parents of D) then corresponds to $\Lambda(w)$ for some $w \in W$; $Z(D)$ corresponds to $Z(w)$ when conditioned on this $\Lambda(w)$; and an outcome of D together with an outcome of $Y(D)$ corresponds to a path $Y(w)$ when conditioned on $\Lambda(w)$. So the analogous expression to (4) for EF DEGs is

$$Y(w) \amalg Z(w) \mid \Lambda(w)$$

A DM using an EF DEG is parsimonious by disregarding information in $Z(w)$ other than $\Lambda(w)$. This allows the DM to simplify the topology of the DEG in an analogous manner to that in which a parsimonious DM can simplify the topology of an ID (Smith 1996), which in turn allows the DM to deduce information about the problem which it is unnecessary to elicit.

7. Discussion

In this paper we have introduced a new graphical representation of multistage decision problems, called the *Decision Event Graph* (DEG), which is suited to asymmetric decision problems. We have shown how the structure may be used to find optimal decision policies by a simple local backward induction algorithm.

The focus of this paper has been on the relationship between DEGs and decision trees. The relationship between DEGs and IDs is too substantial a topic to do any more than touch upon here, but there are a couple of points which can be made quickly – ID-representations of asymmetric problems are at best a compromise, and storing the numerical specification of such problems is inefficient when compared with a DEG-representation. These can both be seen if we consider the example from Section 5. In order to create an ID for this problem we would have to impose a product space structure and accept that the inherent asymmetries would produce a considerable number of zeroes in our tables. If we do this then the ID in Figure 6 is probably the simplest of a number which could be drawn. Using this graph numerical specification would require 74 cells, whereas with the DEG we need at most 24 cells. Note also that although the ID may appear simpler than the DEG, it does not depict sample information which is explicit in the DEG. With both the DEG and the ID there are tweaks which could be made to the graph and / or the storage tables which reduce the storage costs, albeit with some loss to the integrity of the representation. There is a similar increased efficiency for the DEG algorithm over the analogous ID algorithms.

We return to the relationship between DEGs and IDs in a forthcoming paper in which we also develop the ideas from Section 6. Software for finding optimal decision rules on EF DEGs is currently being coded and will be freely available shortly.

In their retrospective article on IDs, (Howard and Matheson 2005) write:

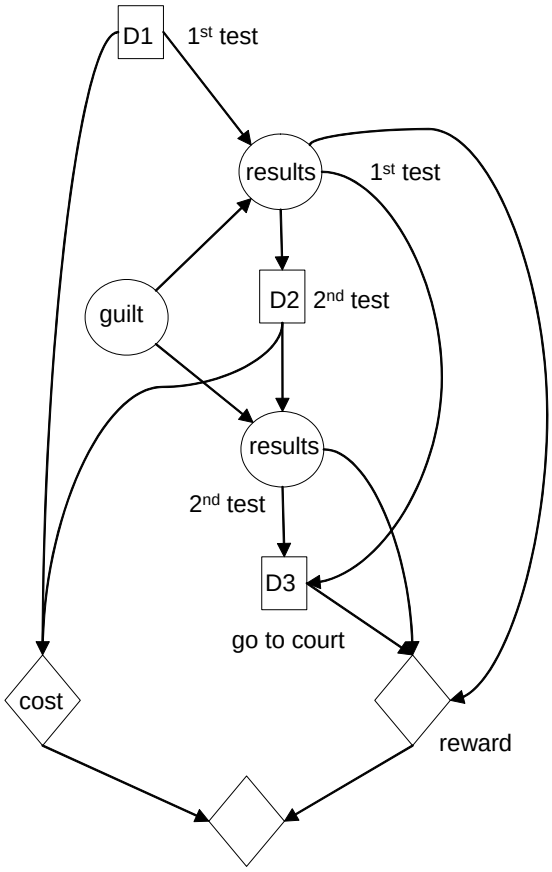

Figure 6 Influence diagram for the forensic example

We believe that the power and acceptance of the influence diagram comes from a graphical method that is interpretable by both professions and lay-people, and that can contain in-depth technical information to completely characterize any decision. The diagram provides a common language for representing both uncertain relationships and decision problems in a wide variety of applications.

Because of their greater generality, we hope that a similar acceptance will develop in the future for decision event graphs.

Appendix. Proof of Lemma 1

Consider a set $\{w\} \subset V(\hat{\mathcal{D}})$ which partitions the $w_0 \to w_\infty$ paths of $\hat{\mathcal{D}}$, and which contains the position w_1. By construction of $\hat{\mathcal{D}}$ all the positions in this set also exist in $\mathcal{D}$, and obey one of three properties:

(1) $\Lambda(w, w_2) = \phi$

(2) $w \prec w_2$ but $\Lambda(w_1, w) = \phi$

(3) $w = w_1$

For cases (1), (2) there exist sub-DEGs in $\mathcal{D}$, $\hat{\mathcal{D}}$ rooted in w, whose topology, edge labels, edge probabilities and edge utilities are identical in each graph. So in cases (1), (2), $w[\mathfrak{u}]_{\hat{\mathcal{D}}} = w[\mathfrak{u}]_{\mathcal{D}}$.

For case (3), by construction, $w_1[\mathfrak{u}]_{\hat{\mathcal{D}}} = w_1[\mathfrak{u}]_{\mathcal{D}} = w_2[\mathfrak{u}]_{\mathcal{D}}$.

The subgraphs of $\mathcal{D}, \hat{\mathcal{D}}$ rooted in w_0 and with the elements of $\{w\}$ as leaves have identical topology, edge labels, edge probabilities and edge utilities.

Hence by backward induction, $w_0[\mathfrak{u}]_{\hat{\mathcal{D}}} = w_0[\mathfrak{u}]_{\mathcal{D}}$.

The optimal decision rule is found by moving forward (downstream) from w_0. If in $\mathcal{D}$ this does not take us to w_1, then it will not in $\hat{\mathcal{D}}$, and the optimal decision rule is unchanged.

If in $\mathcal{D}$ the rule takes us to w_1, then it also takes us to w_2 with probability 1, and any decisions made at w_1 or between w_1 and w_2 are irrelevant as all $\mu(w_1, w_2)$ subpaths have zero associated utility. In $\hat{\mathcal{D}}$ the rule takes us to w_1, w_2, and downstream of w_2 is unchanged from $\mathcal{D}$.

□

Acknowledgments

This research has been partly funded by the UK Engineering and Physical Sciences Research Council as part of the project *Chain Event Graphs: Semantics and Inference* (grant no. EP/F036752/1).

References

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

C. Boutilier. The influence of influence diagrams on artificial intelligence. *Decision Analysis*, 2(4):229–231, 2005.

H. J. Call and W. A. Miller. A comparison of approaches and implementations for automating decision analysis. *Reliability Engineering and System Safety*, 30:115–162, 1990.

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

R. G. Cowell. Decision networks: a new formulation for multistage decision problems. *Research Report* **132**, Department of Statistical Science, University College London, London, United Kingdon, 1994.

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

A. P. Dawid. Separoids: A mathematical framework for conditional independence and irrelevance. *Annals of Mathematics and Artificial Intelligence*, 32:335–372, 2001.

R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings in the Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, California, 1984.

R. A. Howard and J. E. Matheson. Influence diagram retrospective. *Decision Analysis*, 2(3):144–147, 2005.

F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In R. L. de Mantaras and D. Poole, editors, *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 367–373, San Francisco, California, 1994. Morgan Kaufmann.

F. V. Jensen and T. D. Nielsen. *Bayesian networks and decision graphs*. Springer, 2007. 2nd edition.

R. L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and Value trade-offs*. Wiley, New York, 1976.

D. V. Lindley. *Making Decisions*. John Wiley and Sons, Chichester, United Kingdom, 1985.

S. M. Olmsted. *On Representing and Solving Decision Problems*. Ph.D. Thesis, Department of Engineering–Economic Systems, Stanford University, Stanford, California, 1983.

J. Pearl. Influence diagrams—historical and personal perspectives. *Decision Analysis*, 2(4):232–234, 2005.

H. Raiffa. *Decision Analysis*. Addison–Wesley, Reading, Massachusetts, 1968.

H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory*. MIT Press, Cambridge, Massachusetts, 1961.

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

P. P. Shenoy. Representing and solving asymmetric decision problems using valuation networks. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 99–108. Springer–Verlag, New York, 1996.

J. E. Smith and S. H. J. E. Matheson. Structuring conditional relationships in influence diagrams. *Operations Research*, 41(2):280–297, 1993.

J. Q. Smith. *Bayesian Decision Analysis: Principles and Practice*. CUP, 2010. (to appear).

J. Q. Smith. Influence diagrams for Bayesian decision analysis. *European Journal of Operational Research*, 40:363–376, 1989a.

J. Q. Smith. Influence diagrams for statistical modelling. *Annals of Statistics*, 7:654–672, 1989b.

J. Q. Smith. Plausible Bayesian Games. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 5*, pages 387–402. Oxford, 1996.

J. Q. Smith and P. E. Anderson. Conditional independence and chain event graphs. *Artificial Intelligence*, 172:42–68, 2008.

J. Q. Smith and P. A. Thwaites. Decision trees. In *Encyclopedia of Quantitative Risk Analysis and assessment*, volume 2, pages 462–470. Wiley, 2008a.

J. Q. Smith and P. A. Thwaites. Influence diagrams. In *Encyclopedia of Quantitative Risk Analysis and assessment*, volume 2, pages 897–909. Wiley, 2008b.

J. Q. Smith, E. Riccomagno, and P. Thwaites. Causal analysis with chain event graphs. Submitted to *Artificial Intelligence*, 2009.

P. Thwaites, J. Q. Smith, and R. G. Cowell. Propagation using chain event graphs. In D. McAllester and P. Myllymäki, editors, *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, July 2008*, pages 546–553, 2008.