**Cost-effectiveness Analysis with Influence Diagrams**

M. Arias; F. J. Diez  
Department of Artificial Intelligence, UNED, Madrid, Spain

**Keywords**  
Cost-benefit analysis, cost-effectiveness analysis, decision trees, influence diagrams

**Summary**  
**Background:** Cost-effectiveness analysis (CEA) is used increasingly in medicine to determine whether the health benefit of an intervention is worth the economic cost. Decision trees, the standard decision modeling technique for non-temporal domains, can only perform CEA for very small problems.  
**Objective:** To develop a method for CEA in problems involving several dozen variables.  
**Methods:** We explain how to build influence diagrams (IDs) that explicitly represent cost and effectiveness. We propose an algorithm for evaluating cost-effectiveness IDs directly, i.e., without expanding an equivalent decision tree.  
**Conclusion:** IDs can perform CEA on large problems that cannot be analyzed with decision trees.

**Results:** The evaluation of an ID returns a set of intervals for the willingness to pay — separated by cost-effectiveness thresholds — and, for each interval, the cost, the effectiveness, and the optimal intervention. The algorithm that evaluates the ID directly is in general much more efficient than the brute-force method, which is in turn more efficient than the expansion of an equivalent decision tree. Using OpenMarkov, an open-source software tool that implements this algorithm, we have been able to perform CEAs on several IDs whose equivalent decision trees contain millions of branches.

**1. Introduction**

In medicine, many new interventions become available every year for the diagnosis, prevention, and treatment of different diseases. Given that economic resources are limited, it is necessary to assess whether the health benefit of an intervention is worth its cost. The more common approach to economic evaluation is cost-effectiveness analysis (CEA) [1, 2].

In this context, the net monetary benefit [3] of an intervention $I$ is

$$NMB_I(\lambda) = \lambda \cdot e - c \quad (1)$$

where $e$ is the effectiveness and $c$ the cost. The parameter $\lambda$, usually called willingness to pay, cost-effectiveness threshold, or ceiling ratio, converts effectiveness into a monetary scale. It takes values on the set of positive real numbers, i.e., on the interval $(0, +\infty)$. It is measured in effectiveness units divided by cost units; for example, in dollars per death avoided or euros per quality-adjusted life year (QALY) [4]. As the willingness to pay is different for each decision maker, CEA must consider all its possible values. The result of the analysis is usually a set of intervals for $\lambda$, each one having an optimal intervention.

When the consequences of the interventions are not deterministic, it is necessary to model the probability of each outcome. Decision trees are the tool used most frequently for this task, especially in medicine [5]. Their main drawback is that their size grows exponentially with the number of variables. In the medical literature, trees usually have 3 or 4 variables and between 6 and 10 leaf nodes. A tree of 5 variables typically contains around 20 leaf nodes, which implies that building, debugging, and analyzing it would require a significant effort. Given that the maximum size of decision trees in practice is of the order of 50 nodes, they can only solve problems of at most 6 or 7 variables.

Influence diagrams (IDs) [5, 7], in contrast, have the advantages of being very compact, representing conditional independencies, and using direct probabilities, i.e., the probability of the effect conditioned on the cause. They have been used to solve complex medical problems. For example, Arthronet [8], an ID for knee arthroplasty, contains 23 variables and Mediastinet [9, 10], for lung cancer, 27.

---

a By “intervention” we mean either a single action, such as applying a therapy, or a whole strategy, such as “Do the test $T$; if it is positive, apply drug $D$ every 8 hours; if it is negative, repeat the test after 24 hours and then ...”.

b A variable represents, for instance, the presence of a disease, the result of a test, or a decision about a treatment. In general a variable gives rise to several nodes in the decision tree — see [6].

c These IDs, encoded in the ProbModelXML format, are publicly available at www.probmodelxml.org/networks.
The standard method for CEA only allows the decision tree to have one decision node, which must be the root [11, 12]. This requirement may lead to huge decision trees even for problems involving a very small number of variables [6, 12]. Using embedded decision nodes it is possible to build much more compact decision trees, which can be evaluated with a new algorithm proposed recently [12], but the limit of 6 or 7 variables remains.

IDs that encode cost and effectiveness can represent much larger problems. However, standard algorithms, such as arc reversal [13, 14] and variable elimination [15], can only evaluate unicriterion IDs. An approximate solution would be to combine cost and effectiveness into a single criterion, the NMB (Equation 1), and evaluate the ID for several values of \( \lambda \). The problem is that the more precision we require, the more values of \( \lambda \) we must examine; finding the exact solution would need infinite evaluations. Another approach would be to expand the ID into an equivalent decision tree [7] and evaluate it as proposed in [6]. The problem is that the size of the tree grows exponentially with the number of variables. When trying to convert the unicriterion versions of Arthronet and Mediastinet into decision trees, our computer ran out of memory, even though arc reversal and variable elimination were able to evaluate those IDs in a fraction of a second. The decision trees for the cost-effectiveness versions of those diagrams would run out of memory even faster, as we will discuss in Section 4.2.

In this paper we propose a new method for evaluating cost-effectiveness IDs, which combines arc reversal with the same idea used in [6] to evaluate cost-effectiveness decision trees: given that the values of \( \lambda \) are infinite and it is impossible to evaluate sequentially an infinite number of IDs, our algorithm groups the IDs having the same cost, effectiveness, and optimal intervention into a finite set of intervals, whose boundaries are determined dynamically during the evaluation of the ID. This algorithm is in general much more efficient than the brute-force method, which is in turn more efficient than the expansion of an equivalent decision tree. After implementing the algorithm in OpenMarkov, an open-source software package for probabilistic graphical models (www.openmarkov.org), we have been able to evaluate several cost-effectiveness IDs, including Arthronet and Mediastinet, which contain dozens of variables.

The rest of this paper is structured as follows: Section 2 reviews the basic concepts of CEA and IDs, Section 3 presents the new algorithm, Section 4 compares IDs with decision trees from the points of view of knowledge representation and computational complexity, Section 5 analyzes some related work, and Section 6 summarizes the conclusions. The appendix contains some formal definitions, the pseudo-code of the algorithms, a proof of correctness, and a detailed example.

2. Background

2.1 Deterministic CEA

When we have a set of interventions such that their cost and effectiveness are known with certainty, we can perform a deterministic CEA, which returns the intervention that maximizes the NMB (Equation 1) for each value of \( \lambda \).

The result of a CEA can be expressed as a cost-effectiveness partition (CEP), which consists of a set of intervals such that the cost, effectiveness, and optimal intervention is the same for all the \( \lambda \)s inside an interval, as shown in Figure 1. See Section A.2 in the Appendix for a formal definition of CEP.

The standard algorithm for deterministic CEA consists of finding a subset of interventions that dominates the other interventions for every value of \( \lambda \) [16]. A slightly more efficient algorithm is presented in [6] and in the Appendix (Sec. A.5).

2.2 Decision Trees

A decision tree is a model for decision analysis of problems involving probabilistic outcomes [17]. It has three types of nodes: chance, decision, and utility. Utility nodes represent the decision maker’s preferences. All the leaves are utility nodes and, conversely, all utility nodes are leaves.

A unicriterion decision tree can be evaluated with the roll-back algorithm, which proceeds from the leaves to the root: the utility of a chance node is the average of the utilities of its branches and the utility of a decision node is the maximum of their utilities.

It is possible to build a bi-criteria decision tree such that each utility node, instead of representing a single value, represents cost and effectiveness separately. If the only decision node is the root, the tree can be evaluated by a modified version of the roll-back algorithm that computes the cost and effectiveness of each node separately and then performs a deterministic CEA at the root node [1, 18]. However, when the tree contains embedded decision nodes this method does not work [12]. A solution proposed recently is to use a more sophisticated version of the roll-back algorithm that works with CEPs instead of scalar utilities: each chance node is evaluated by averaging the CEPs of its branches, and each decision node is evaluated by performing a deterministic CEA for each interval of \( \lambda \) [6]. This method is the basis for the evaluation of IDs proposed in this paper.

2.3 Unicriterion Influence Diagrams

An ID is a probabilistic graphical model that consists of: an acyclic directed graph containing three types of nodes – chance, decision, and utility –, a probability distribution, and one or several utility functions. Every node represents a variable, so we will speak of nodes and variables indifferently. In this paper, an uppercase letter

Figure 1 A cost-effectiveness partition (CEP) of four thresholds (five intervals). All the \( \lambda \)s inside the \( i \)-th interval have the same cost \( (c_i) \), effectiveness \( (e_i) \), and optimal intervention \( (I_i) \).
(X) denotes a variable or a node, and a lowercase letter (x) a value of the corresponding variable. A bold uppercase letter denotes a set of variables, \( X = \{X_1, \ldots, X_n\} \), and a bold lowercase letter a configuration of the corresponding set of variables: \( x = (x_1, \ldots, x_n) \).

In a graph, if there is an arc from node \( X \) to node \( Y \), we say that \( X \) is a parent of \( Y \); the set of parents of \( Y \) is denoted by \( Pa(Y) \), and \( pa(Y) \) denotes a configuration of \( Pa(Y) \). For every chance node \( C \) the ID contains a family of conditional probability distributions of the form \( P(c|pa(C)) \). A unicriterion ID also contains a utility function \( U_i(pa(U_i)) \) for each utility node \( U_i \).

The meaning of an arc in an ID depends on the type of nodes that it links. An arc \( X \rightarrow C \), where \( C \) is a chance node, denotes a probabilistic dependence of \( C \) on \( X \); in practice, it usually means that \( X \) is a cause of \( C \). An arc from a decision \( D_i \) to a decision \( D_j \) (in general, any directed path from \( D_i \) to \( D_j \)) means that \( D_i \) is made before \( D_j \). Every ID must contain a directed path passing through all the decision nodes, which determines a total ordering of the decisions. An arc from a chance node \( C \) to a decision node \( D_i \) means that the value of variable \( C \) is known when making decision \( D_i \). IDs assume the non-forgetting hypothesis, which means that a variable \( C \) known for a decision is also known for any posterior decision. An arc from a variable \( X \) to a utility node \( U_i \) means that the utility depends on \( X \).

The maximum expected utility (MEU) of an ID whose chance and decision variables are all discrete is shown in Equation 2 (Figure 2), where \( C \) denotes the set of variables known after making \( D_i \) and before making \( D_{i+1} \). Due to the non-forgetting hypothesis, the variables known when making \( D_i \) are \( C_0 \cap \{D_i\} \cup C_1 \cup \ldots \cup \{D_{i-1}\} \cup C_{i-1} \). The optimal policy for \( D_i \) consists of selecting, for each configuration of the variables, the value \( d_i \) that maximizes the expected utility.

\[
MEU = \sum_{c_0} \max_{d_i} \sum_{c_1} \ldots \sum_{c_{i-1}} \max_{d_{i-1}} \sum_{c_n} \prod_{c \in \text{V}_C} P(c|pa(C)) \cdot \sum_{j} U_j(pa(U_j))
\]  

### Table 1

<table>
<thead>
<tr>
<th>Therapy</th>
<th>Cost</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>€0</td>
<td>1.2 QALY, 10.0 QALY</td>
</tr>
<tr>
<td>therapy 1</td>
<td>€20000</td>
<td>4.0 QALY, 9.9 QALY</td>
</tr>
<tr>
<td>therapy 2</td>
<td>€70000</td>
<td>6.5 QALY, 9.3 QALY</td>
</tr>
</tbody>
</table>

Arc reversal [13, 14] is the standard algorithm for evaluating unicriterion IDs; in this context, arc and link are synonymous. Reversing an arc \( X \rightarrow Y \) (where \( X \) and \( Y \) are chance variables) consists of replacing it with \( Y \rightarrow X \), computing new conditional probabilities for \( X \) and \( Y \), and adding some links if necessary, as described in the Appendix (Sec. A.1). The algorithm proceeds by removing a node in each iteration. When no node can be removed immediately, there is always a sequence of arc reversals that make one node removable. Section 4.2 discusses the complexity of this and other algorithms for IDs.

### 3. Cost-effectiveness Analysis with IDs

This section describes our method for performing CEAs with IDs.

#### 3.1 Construction of the ID

The construction of an ID for CEA is almost identical to the unicriterion case; the only difference is that some utility nodes represent effectiveness and the others represent economic costs.

**Example 1**

For a disease whose prevalence is 0.14, there are two possible therapies. The effectiveness of each therapy depends on whether the disease is present or not, as shown in Table 1. There is a test with a sensitivity of 90%, a specificity of 93%, and a cost of €150. Is the test cost-effective?

The ID for this example is shown in Figure 3. Decisions are drawn as rectangles, chance nodes as rounded rectangles, and utility nodes as hexagons. The node \( \text{Dec:Test} \) represents the decision whether or not to do the test; it has two values: do test and do not test. The node Test represents the result of the test; its values are positive, negative, and not performed.

#### 3.2 Evaluation of Cost-effectiveness IDs

In this section we offer an overall description of the algorithm. The Appendix contains the pseudo-code (Sec. A.5) and a detailed numerical example (Sec. A.6).

##### 3.2.1 Initial Step: CEP Potential for the Utility Node

The first step of the evaluation consists of combining all the utility nodes into a single node, \( U \), whose parents are all the chance and decision nodes having at least one utility node in the original ID; we denote this set by \( X \). The utility for node \( U \) is a CEP potential, i.e., a function \( U(x) \) that assigns a CEP to each configuration \( x \). In this initial ID, every CEP \( U(x) \) consists of a single interval, because no threshold has been generated so far; the cost is the sum of the costs in the original ID for configuration \( x \) and the effectiveness is the sum of the effectiveness values. An example is shown in the Appendix, Section A.6.1.

##### 3.2.2 Arc Reversal and Node Removal

After this initial step, the arc reversal algorithm proceeds by removing all the chance and decision nodes, as in the evaluation of unicriterion IDs [13, 14]; the only differ-
When it is not possible to remove any node immediately, there is always a sequence of arc reversals that make at least one node removable. Reversing an arc in a cost-effectiveness ID is exactly the same as in the unicriterion case, explained in Section A.1 of the Appendix.

After removing all the chance and decision nodes, the potential of U contains only one CEP, which is the solution of the CEA – see Table 6 in the Appendix as an example.

4. Comparison with Decision Trees

4.1 Construction of the Model: Influence Diagrams vs. Decision Trees

The main advantage of IDs is that they are much easier to build than decision trees, mainly because they are much more compact: an ID contains only one node for each variable in the problem, while in a tree the number of branches grows exponentially with the number of variables. For example, the ID in Figure 3 contains 7 nodes while the equivalent tree contains 32 – see [6]. A symmetric decision tree containing \( n \) binary variables has \( 2^n \) leaf nodes. (A tree is said to be symmetric if every path from the root to a leaf node contains the same variables and in the same order, and all the nodes representing a variable have the same number of outgoing branches.) The number of leaves may be smaller due to asymmetries, but it may also be bigger if there are multi-valued variables. For this reason we said in the introduction that decision trees are unsuitable for problems involving more than 6 or 7 variables.

Compared with decision trees, IDs do not only require much fewer parameters: these parameters are much easier to obtain because each conditional probability only involves one node (the effect) and its parents in the graph (its causes). These probabilities can be obtained from objective data, such as databases or epidemiological studies, or from subjective estimates. In contrast, the distribution probability for the branches of a node in a tree is usually conditioned on several of its causes and effects; when the tree contains more than a few variables, obtaining these conditional probabilities is extremely difficult if not impossible. This explains why many medical problems for which it was unfeasible to build a decision tree directly have been solved with IDs [5].

In cost-effectiveness analysis, decision trees face one additional problem: the standard evaluation algorithm crashes when the tree contains embedded decision nodes; some software packages, such as TreeAge and WinDM, often give wrong results, without warning the user [12]. Building trees whose root is the only decision node, as proposed by Kuntz and Weinstein [11], guarantees a correct solution at the price of increasing dramatically the size of the tree even for very small problems [6]. A new algorithm for dealing with embedded decision nodes allows the modeler to build much more compact trees [6], but in practice they cannot have more than around 50 leaf nodes.

In contrast, IDs can perform cost-effectiveness analysis on much larger problems. Mediastinet, an ID for lung cancer, contains 8 chance variables and 5 decisions [9, 10]. Arthronet, for total knee arthroplasty, has 11 chance variables and 4 decisions [8]. The equivalent decision trees would contain thousands of leaves even when using embedded decision nodes; building those trees directly – i.e., without generating them automatically from an ID – would have been absolutely impossible, not only for their size, but also for the difficulty of
computing the conditional probabilities of their branches (indirect probabilities) given the parameters of the model, such as prevalence, sensitivities, and specificities (direct probabilities).

4.2 Complexity of the Evaluation Algorithms

4.2.1 Influence Diagrams vs. Decision Trees

The evaluation of unicriterion IDs, closely related to the evaluation of Bayesian networks, is NP-hard [19], which makes it unlikely that there exist polynomial complexity algorithms to evaluate them. Both the brute-force method, which consists of the direct application of Equation 2 (Figure 2), and the transformation of the ID into an equivalent decision tree have exponential complexity in time and space. However, expanding the tree is more inefficient than brute-force because it requires more space to store the tree and more time to compute the conditional probabilities of its branches.

Arc reversal and other exact algorithms for IDs, such as variable elimination [15] and strong junction trees [20], also have worst-case exponential complexity, both in time and space, but in many cases they are much more efficient. In fact, the property that most affects the complexity of these algorithms is not the number of nodes, but the treewidth of the graph (after removing the information arcs), a concept that has been extensively studied in Bayesian networks [21–24]. Given that the treewidth of an ID is usually much smaller than the number of variables – as is also the case for Bayesian networks – these algorithms are in general much more efficient than the brute-force method and, consequently, much more efficient than the expansion of an equivalent tree.

In the case of cost-effectiveness analysis, the complexity of CEPs operations is proportional to the number of thresholds (see Algorithms 2 and 3 in the Appendix), hence the difference is even more favorable for IDs, which perform fewer CEP operations than decision trees.

4.2.2 Evaluation of Cost-effectiveness IDs

A deterministic CEA (cf. Sec. 2.1) involving \( n \) interventions may generate up to \( n - 1 \) thresholds. The combination of CEPs generates new CEPs that may inherit the thresholds of all of them. Therefore, the number of thresholds might grow exponentially with the number of variables. However, in practice this number depends on the numerical parameters (the probabilities and utilities) of the model. Thus, when a subset of \( k \) interventions dominates the others, only \( k - 1 \) thresholds are generated. In many cases \( k \) is smaller than \( n \); frequently \( k = 1 \), which generates no new threshold. Additionally, a significant number of thresholds are usually discarded when removing a decision variable (see line 14 in Algorithm 3 in the Appendix). For these reasons, the number of thresholds often remains within manageable limits. For example, the evaluation of the ID in Figure 3 might generate up to 12 thresholds, but the maximum number during the evaluation is 3, and the final CEP contains only 2 – see Table 6 in the Appendix.

In order to test empirically the performance of our algorithm, we have designed two ID structures, inspired on medical problems, whose size can grow arbitrarily; the details can be found in the Appendix (Sec. A.7, Figs. 7 and 8). Using a desktop computer, we were able to evaluate in a few seconds some IDs containing 10 chance nodes and 10 decisions, or 14 chance nodes and 7 decisions. Expanding and evaluating the equivalent decision trees, which would have millions of leaves, would have taken more time and much more memory space. In fact, the limit on the problem size does not stem from the time we are willing to wait – because the evaluation is very fast and because cost-effectiveness analyses are usually performed off-line – but from the working memory of our computer. These experiments have also shown that the size limit for cost-effectiveness analysis is only slightly lower than for unicriterion problems, as explained in the Appendix (Figs. 9 and 10).

5. Related Work

To the best of our knowledge, our algorithm is the first method for evaluating cost-effectiveness IDs. Previously the only technique for analyzing multi-attributed IDs was the method by Nielsen et al. [25]: they select a particular configuration of \( \alpha \) – the vector that represents the decision maker’s preferences – and evaluate the corresponding unicriterion ID; then they obtain a set of inequalities, which delimit the region of \( \mathbb{R}^k \) in which \( \alpha \) can vary without leading to a different optimal strategy. This approach is clearly insufficient for CEA, especially in medicine, because it does not return the intervals for \( \lambda \) nor the optimal strategy for each interval. In contrast, the algorithm proposed in this paper limits itself to bicriteria IDs, but is able to solve them completely.

6. Conclusion and Future Work

In this paper we have explained how to build IDs for cost-effectiveness analysis and proposed an algorithm that can solve many medical problems for which it would be absolutely impossible to build a decision tree directly. In particular, our research group has built cost-effectiveness IDs for two complex medical problems whose equivalent decision trees would contain thousands of leaf nodes [8–10]. Building those trees directly – instead of generating them from the equivalent IDs – would be have been unfeasible not only for the size of the trees, but also for the difficulty of computing the conditional probabilities of their branches.

Cost-effectiveness IDs can be evaluated by combining the arc reversal algorithm [13, 14] with a method for performing CEA in decision trees [6]. This method can be combined with other algorithms for unicriterion IDs, such as variable elimination [15] and strong junction trees [20], because all of them perform essentially the same operations. An open-source Java implementation of this algorithm, together with the IDs Mediastinet and Arthronet mentioned above, is publicly available at www.openmarkov.org/cea.
In our experiments we have been able to evaluate in a few seconds some cost-effectiveness IDs containing up to 21 chance or decision nodes. Evaluating the equivalent decision trees, which contain millions of leaf nodes even when using embedded decision nodes, would have taken much more time and our computer would have run out of memory for the biggest IDs.

An open line for future research is to summarize into a small policy tree the result of evaluating a cost-effectiveness ID, by adapting the method proposed in [9]. Another line is to adapt our CEA algorithms to the evaluation of decision analysis networks (DANs) [26], which present several advantages over IDs, especially in the case of asymmetric decision problems.

Acknowledgments

We thank the anonymous reviewers for many valuable comments and suggestions. This work has been supported by the Spanish Ministry of Science and Technology under grants TIN2006–11152 and TIN200909158, by FONCICYT grant 85195, and by the Fondo de Investigaciones Sanitarias (Health Research Fund) of the Spanish Government under grant PI13/02446.

References