# Solving influence diagrams using belief networks and transformations

Soeradj Kanhai - 1229141
Rob Ooms - 1174150

3-4-2015

## 1 Introduction

Many practical problems in operations research are characterized by a large number of interrelated uncertain quantities and alternatives. To analytically address these problems its decision makers continuously prefer using ad hoc procedures that let them main their way of thinking in spite of the availability of decision analysis which allows for analytic decision making based on a normative axiomatic framework.

Influence diagrams have been designed as a knowledge representation bridging the gap between analysis and formulation. Influence diagrams are directed, acyclic graphs consistent out of different types of nodes. They illustrate the relationships between variables and represent a complete probabilistic description of the problem. In this paper we give a detailed description of influence diagrams in Section 2. Sections 3 explains two different algorithms to find solve influence diagram problems. One by transforming the influence diagram to get its maximum expected utility, that is, to make the best possible decision and the other by using belief networks to get the maximum expected utility. This paper is finally concluded by discussing and comparing the advantages and disadvantages of each of these algorithms and.

## 2 Influence diagrams

An influence diagram is a belief network augmented with decision nodes and a value node. It is a directed acyclic graph (DAG) $G = (N, E)$, with $N$ nodes and $E$ edges. We recognize three types of nodes *chance, decision* and *value* nodes. Every node $i$ has a variable $X_i$ and is associated a set $\Omega_i$.

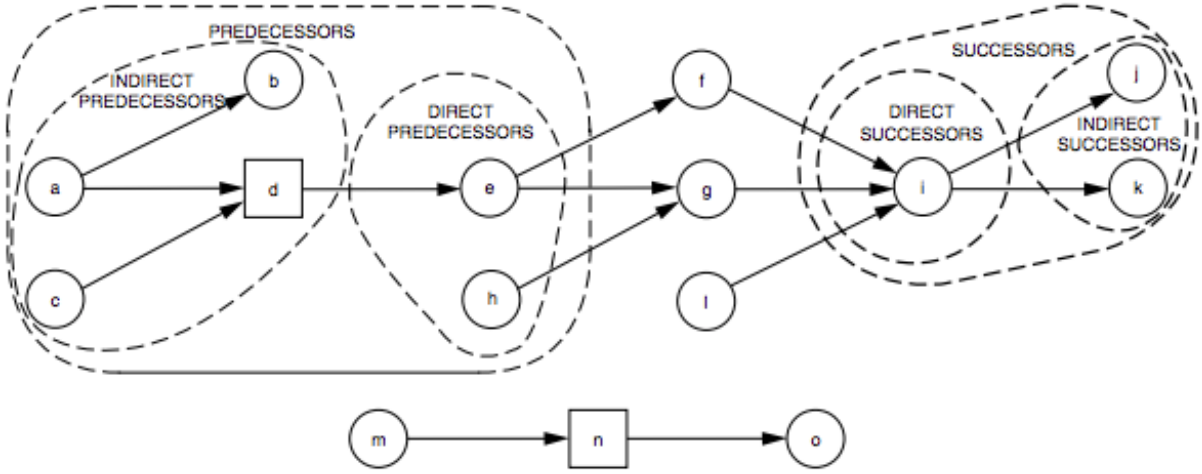
Each chance node $C$ is associated with a random value for which there is a underlying joint probability distribution for all of the random variables. Its set, $\Omega_i$, indicates the domain for the random variable $X_i$. Decision nodes $D$ represent a choice among a set of alternatives and value nodes $V$ are those nodes whose value the influence diagram is tasked to determine; our expected utility. The expected utility is a function of the values of the conditional predecessors of the value node: $U : \omega_{C(v)} \rightarrow \omega_v$. In section 2.1 we explain what condition predecessors are. Our influence diagrams can have $\geq 0$ chance or decision nodes, but we have at most 1 value node in our diagram. Similar to our different types of nodes we also recognize different types of edges: *conditional* edges go into either chance or values nodes, while *informational* edges go into decision nodes. Our conditional edges represent probabilistic dependence.

### 2.1 Predecessors and successors

In influence diagrams there is also the set of possible additional influences that are not explicitly show on the diagram. With respect to any given node $i$ we therefore recognize the following influences.

Each node $i$ has *predecessors*; the set of all nodes having a path leading *to* node $i$, and *successors*; the set of all nodes having a path leading *from* node $i$. We distinguish between *direct* and *indirect* predecessors and successors. *Direct predecessors* $P(i)$ is the set of nodes having an influence edge connected *directly* to node $i$ and *indirect predecessors* $IP(i)$ is the set of all nodes that are a predecessors of node $i$

with the exception of the nodes that form its direct predecessors. Our total set of predecessors is therefore defined as $P(i) \cup IP(i)$. Likewise, we have *direct successors* $S(i)$ of node $i$; the set of nodes having an influence arrow connected directly from node $i$, and the *indirect successors* $IS(i)$; the set of all nodes that are a successors of node $i$ with the exception of the nodes that form its direct successors. Similar to the total set of predecessors our total set of successors is defined as $S(i) \cup IS(i)$. We illustrate these sets in Figure **??**.

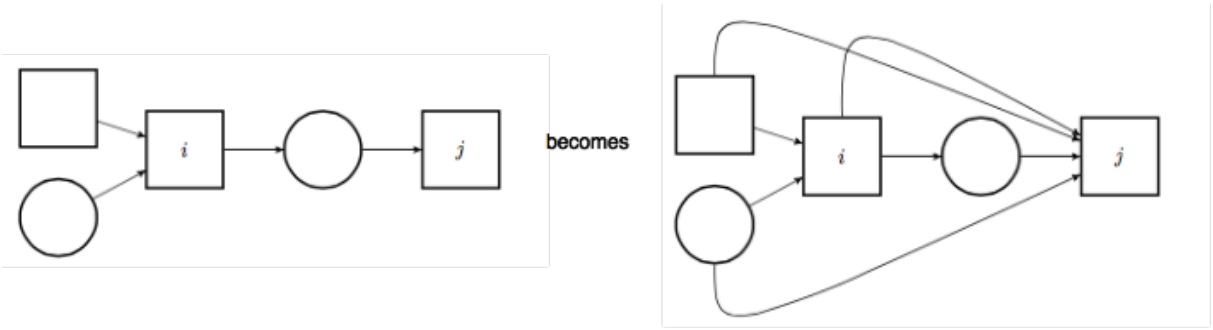



Figuur 1: Direct and indirect pre- and successors sets defined for note $g$.

Due to our different kinds of edges we can now also distinguish between the set of direct predecessors for a chance or value node $i$, *conditional predecessors C(i)*, and the set of direct predecessors for a decision node $i$, *informational predecessors I(i)*.

## 2.2 Proper, oriented and regular influence diagrams

We distinguish between our influence diagrams depending on their structure. *Proper* influence diagrams are those which unambiguously represent a single decision maker's point of view of the world. *Oriented* diagrams are those with a value node. Finally, *regular* influence diagrams are acyclic, directed graphs with, if present, no successors to our value node and a directed path that contains all decision nodes. To satisfy this last condition of the regular influence diagram we require a total ordering of all the decisions. This is possible by adding "no forgettingëdges to our diagram. We add these edges when a decision node $i$ precedes node $j$ in a regular influence diagram. As a consequence of this property note $i$ and all of its informational predecessors are the informational predecessors of node $j$. This is illustrated in ?Figure 2.




Figuur 2: Adding "no forgettingëdges to our influence diagram.

# 3 Algorithms

## 3.1 Algorithm 1

Here we discuss the algorithms as proposed by Ross D. Shachter in **??**. Using this algorithm we reduce our influence diagram to a single value node with *value-preserving reductions*. Once our influence diagram consists out of just a single value node we know the optimal decision policy. After each value-preserving reduction a node can be removed from the set of nodes of which our influence diagram consists. Together with all incoming and outgoing edges of that particular node. These reductions do not change the maximal expected value of our diagram. We recognise three kinds of reductions:

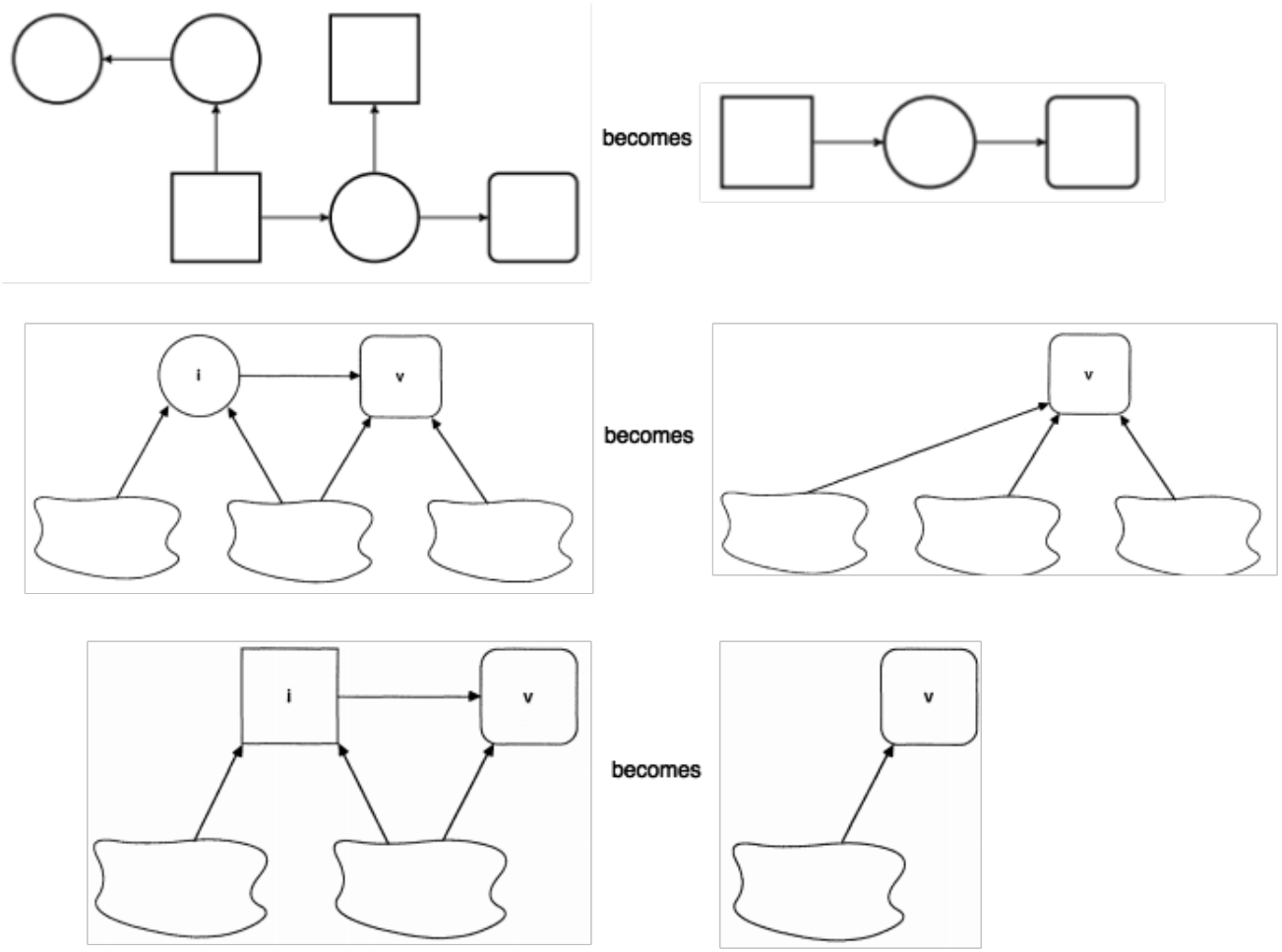
**Barren node removal:** a chance and decision node without outgoing edges can be removed safely as their removal does not affect any other nodes. If the removal of a barren node results in other barren nodes these may also be removed;

**Chance node removal:** a change node may be removed only if it is preceded by a value. Now all of its conditional predecessors are inherited by the value node making it impossible for barren nodes to be created during this reduction;

**Decision node removal:** provided that all barren nodes are removed and that our decision node $i$ is a conditional predecessor of the value node with all other conditional predecessors of the value node being informational predecessors of our decision node $i$; this node may be removed. Removing this decision node happens by maximizing the expected utility with the proviso of the values of its predecessors. Every optimal alternative is remembered as the optimal policy. As for any unobserved conditional predecessors at the time of the decision may be removed using conditional expectation.

In Figure 3.1 we list for each of our kinds of reductions an example. Additionally, we can also use *Edge reversal*. If there is an edge $(i, j)$ from node $i$ to $j$ we can place it with edge $(j, i)$ provided that there is not path from $j$ to $i$ to inhibit that any cycles are created. Here nodes $i$ inherits the conditional predecessors of node $j$ and vice versa.

We list the pseudo code for this evaluation algorithm in Algorithm 1. Every iteration in this algorithm ensures that a node is removed. Our stopping state is therefore any regular influence diagram.




Figuur 3: From top-to-bottom we have an example reduction for barren, chance and decision node removal.

**Algorithm 1** The algorithm for evaluating any regular influence diagram

```
1: procedure IDEVAL
Require: oriented, regular influence diagram
Require: "no forgetting" arcs are added
2:     eliminate all barren nodes
3:     while C(v) ≠ ∅ do
4:         if there exists i ∈ C ∩ C(v) st S(i) = {v} then
5:             remove chance node i
6:         else if there exists i ∈ D ∩ C(v) st C(v) ⊂ I(i) ∪ {i} then
7:             remove decision node i
8:             eliminate barren nodes
9:         else
10:            find i ∈ C ∩ C(v) st D ∩ S(i) = ∅
11:            while C ∩ S(i) ≠ ∅ do
12:                find j ∈ C ∩ S(i) st there is no other directed (i, j)-path
13:                reverse arc(i, j)
14:            end while
15:            remove chance node i
16:        end if
17:    end while
18: end procedure
```

### 3.1.1 Algorithm to evaluate any regular influence diagram

## 3.2 Algorithm 2

Here we discuss the algorithms as proposed by Gregory F. Cooper in [2]. We will first explain how to transform an influence diagram to a belief network and then how to solve the influence diagrom using this belief network.

### 3.2.1 Transform an Influence Diagram to a Belief Network

The transformation from influence diagram to belief network is done in two steps. In the first step all decision nodes are converted to changes. To do this we first need the total order of the different decisions. For simplicity we label the decisions $D_1, ..., D_n$ based on this order. Now we create a list which consists of triplets. The first part is the decision node $D_i$, the second part is a set of the incoming information into $D_i$, represented as $\Pi_{D_i}$, and third are the chance nodes in $\Pi_{D_i}$, written as $\Pi'_{D_i}$. After a triplet is created, all incoming arcs into $D_i$ are deleted.

The second part consists of creating probabilities of the possible outcomes of the value node in the diagram. This is done by normalising the outcomes of the value function, $v(\Pi_v)$, into the range between zero and one. $\Pi_v$ are the arcs coming into the value node. This gives us the following formula:

$$P(V = T|\Pi_v) = \frac{v(\Pi_v) - \min_{\Pi_v} v(\Pi_v)}{\max_{\Pi_v} v(\Pi_v) - \min_{\Pi_v} v(\Pi_v)} \tag{1}$$

### 3.2.2 Solve an Influence Diagram using its Belief Network representation

First we take the case with only one decision. The influence diagram is solved by taking the decision which result in the maximum expected value (MEV) of the value node, where we know evidence E:

$$MEV(D_i, E) = \max_{D_i} \sum_{\Pi'_v} v(\Pi_v) P(\Pi'_v|D_i, E) \tag{2}$$

To solve a case with multiple decisions we apply the above formula for each decision. After a decision is made, the decision is added to the evidence and we pick the next decision. The order in which we make the decision is determined by the total order, the same we used to create the triplets.

Two possible optimisation's for this algorithm are the use of dynamic programming and the removal of nodes who have no direct arcs to decisions.

# 4 Discussion

In this work we looked at two papers, both aimed at transforming an influence diagram to a belief network and solving these. Both approaches have a lot in common regarding how to handle decision nodes. The main difference is in the aggressive deleting of nodes in [1], whereas [2] leaves most of the graph intact.

Influence diagrams and belief networks are a good way to model how certain decisions influence a outcome. Another advantage of the approach of [1] is that the influence diagram is simplified step by step. This information can be used to construct more simple diagrams next time.

However the shortcoming of the influence diagrams in these two papers are that there is only one value node, whereas in real life applications there are multiple outcome values to a decision. If for example a company makes an influence diagram to decide how to do their marketing, a possible outcome value is the profit. But also other possible outcomes values must be considered, for example profit can already be split into short-term and long-term and we also have the image of a company.

In total influence diagrams and belief networks are use full to model decisions, however the model proposed in the two papers are not rich enough for more advanced real life applications.

# Referenties


[1] R. D. Shachter, *Evaluating Influence Diagrams*, Operations research, Vol. 34 No. 6 (Nov. - Dec., 1986), pp. 871-882

[2] G. F. Cooper, *A Method for Using Belief Networks as Influence Diagrams*