Evaluating Influence Diagrams

Where we've been and where we're going

Mark Crowley
Department of Computer Science
University of British Columbia
crowley@cs.ubc.ca

August 31, 2004

Abstract

In this paper we will survey the history of Influence Diagrams from their origin in Decision Theory to modern AI uses. We will compare the various methods that have been used to find an optimal strategy for a given influence diagram. These methods have various advantages under different assumptions and there is a progression to the present of richer, more general solutions. We also look at an abstract algorithm used as an informal tool and determine whether it is equivalent to other formal methods in the literature.

1 Introduction

This paper will look at the development of influence diagrams from their beginnings in decision analysis to their current important place in many areas of computer science including artificial intelligence. We will layout the different methods used to optimize decisions using influence diagrams by computing them directly or via conversions to other models such as decision graphs and bayesian networks. The latter type in particular will be looked at in depth and it will be contrasted against the performance of various algorithms. We will also explain an intuitive algorithm used informally by some researchers and analyze its advantages or similarities with methods described in the literature. The main contribution of this paper is to identify trends through all these solutions over time and help focus work on open questions and to find new directions being pointed to by the existing literature.

2 Influence Diagrams

Influence diagrams (IDs) were proposed by Howard and Matheson [HM03] as a tool to simplify modelling and analysis of decision trees. Decision trees represent each decision or chance variable as a new level in a tree. The leaves of the tree are utilities that express which ending configurations are more desirable. Solving a decision problem requires finding the optimal path through this tree that maximizes expected utility.

Now instead of representing each level of that tree we represent each variable as single nodes in a graph. An *influence diagram* is a directed acyclic graph N containing nodes representing the variables of the decision problem, $V = X \cup D \cup U$. Each variable has its own domain of values, $dom(v)$. The set of parents of a node v_i is denoted π_{v_i}. These variables are of three types, see Figure 1:

- *chance* nodes, $x_i \in X$, represent random variables in the analysis. They have an associated conditional probability table (CPT) representing a distribution $P(x_i|\pi_{x_i})$.
- *decision* nodes, $d_i \in D$, represent decisions to be made. We use *decision rules* δ_i, to represent the mapping of each permutation of its parents to exactly one decision, $\delta_i(\pi_{d_i}) \in dom(d_i)$
- *utility* nodes, denoted $u_i \in U$, represent functions that map each permutation of its parents to exactly one utility value $val_{u_i}(\pi_{u_i}) \in dom(u_i)$. No other variables are allowed to depend on a utility directly so utility nodes do not have children.

An *optimal decision rule* δ_i^* is one that maximizes $E_{\delta_i}[val_{u_i}(\pi_i)]$, the expected value of the utility nodes it effects. The goal in decision analysis is to find an *optimal policy* Δ, which is the set of all optimal decision rules $\Delta = \{\delta_1^*, \ldots, \delta_k^*\}$, one for each utility node. The expected value of a utility node given Δ is denoted $E_\Delta[val_{u_i}(\pi_{u_i})]$. Thus, we are seeking:

$$\Delta = \{\delta_i^* \in D | \exists u_i \in U, \ \ argmax_{\delta_i} E_{\delta_i}[val_{u_i}(\pi_i)]\} \tag{1}$$

and

$$E_\Delta[N] = \sum_{u_i \in U} E_\Delta[val_{u_i}(\pi_{u_i})] \tag{2}$$

Finding these is what we mean by *evaluating an influence diagram*.

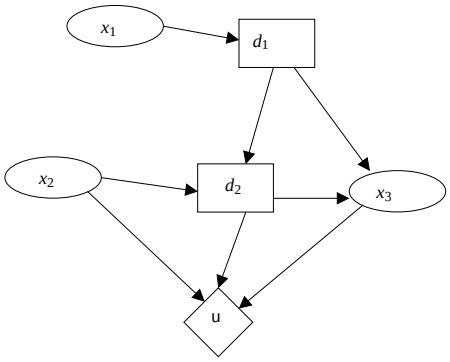

Figure 1: A simple Influence Diagram

Definition 1 *An influence diagram is* regular *if there is a definite ordering of the decision nodes such that there is a path from each decision to the next decision. Further, the ID is* no-forgetting *if each decision has access to all of the information the previous decisions had. This means that every element of $d_i \cup \pi_{d_i}$ is a parent of every decision following d_i.*

IDs have several well known advantages over decision trees. They simplify modelling by allowing the analyst to specify single nodes that represent entire probability distributions over nearly arbitrary relationships with other variables. We still limit ourselves to regularity as defined above and no loops but this still provides a level of expression not possible with trees. In order to express conditional independence relationships that are natural in an ID we would need to resort to the use of complicated information sets amongst siblings in a tree. This may help computationally but not in terms of constructing models. We also must consider that as the number of variables in the problem grows, the size of a symmetric tree grows exponentially, which is not true in the case of influence diagrams and their CPTs since they need only represent the probabilities associated with their parents in the graph.

So influence diagrams have much to offer if they can be evaluated efficiently.

3 Using Bayesian Networks to Evaluate Influence Diagrams

This approach was first put forward by Cooper [Coo88] who showed primarily how to reduce influence diagrams to bayesian networks. Shachter and Peot [SP92] soon after showed an improved algorithm that was much more efficient. Later, Zhang [Zha98] introduced a modified method that greatly reduced the number of nodes being considered in each part of the evaluation. Recently Xiang [XY01] has proposed an algorithm that claims to be as efficient as Zhang's but simpler. We will briefly present the approaches of these algorithms.

3.1 Cooper's Reduction

Cooper [Coo88] converted the ID problem to a BN problem in the following way. An influence diagram is essentially very similar to a bayesian network already, all that is required is to ensure that all nodes have proper probability distributions associated with them to allow us to perform inference. We thus proceed by essentially converting all node types to chance nodes. Decision nodes are converted to chance nodes with an even distribution:

$$\forall \alpha_{d_i} \in dom(d_i), P(\alpha_{d_i} | \pi_{d_i}) = \frac{1}{|dom(d_i)|}$$

We also need to assign some probability to utility nodes to represent their payoff. Cooper assigned the probability of the new binary-valued utility nodes given its parents

in the BN a probability proportional to the value of the utility function in the ID for the same parents[1]:

$$P(u=1|\pi_u) = \frac{val_u(\pi_u)}{max_{\pi_u} val_u(\pi_u)}$$

$$P(u=0|\pi_u) = 1 - \frac{val_u(\pi_u)}{max_{\pi_u} val_u(\pi_u)}$$

And obviously no change is needed for the chance nodes from the ID itself.

With the construction complete they then showed that the problem of finding an optimal decision in the ID reduces to a problem in the BN[2] :

$$E_{\delta_i}[val_u(\pi_u)] = P(u=1|\pi_u, d_i)$$

Their suggested method of solving then is to maximize these δ_i for each permutation of π_u. This can require an exponential number of inference steps. Therefore later work based themselves on this reduction method but went further.

3.2 Shachter and Peot's Algorithm

This method [SP92] took advantage of the independence structure of the BN and the regularity constraint that says there is a defined order of decision nodes. They noted that instead of maximizing $P(u=1|\pi_u, d_i)$ they could instead maximize $P(\pi_u, d_i|u=1)$. This is preferable since the computations can now be done locally using standard BN techniques of belief propagation. We can recursively optimize each δ_i starting from the last continue optimizing δ_{i-1} until we reach the beginning. This method seems very intuitive and is actually nearly equivalent to Shachter's original technique from [Sha86] except that now the bayesian network is handling all of the graph manipulations which previously had to be specified separately.

3.3 Zhang's Algorithm

This method was based on previous advances made by Zhang and Poole regarding the notion of *stepwise decomposable* IDs [ZP92] and bayesian inference techniques [ZP94]. The intuition for this method is that in a regular, no-forgetting ID the parents of each decision divide the graph into two independent partitions, see Figure 3.3. They show that if we consider the last decision in the ordering d_k, called the *tail decision node*, then it partitions the network in an interesting way.

Call the partition containing the tail decision node T . This set of nodes is separated from nodes previous to those in π_{d_i}. It also only includes nodes in π_{d_k} and V_2 that can effect the utilities in V_2, denoted $u \in U_i$. The optimal decision rule for d_i can then be computed as follows:

$$\delta_i^*(\pi_{d_i}) = arg\ max_{\delta_i} \sum_{u \in U_2} P_T(u=1|\pi_{d_i}, d_i) \tag{3}$$

[1] In this and the following section the algorithms limit the ID to only one utility node, therefore subscripts on u's are not necessary at this point

[2] we include d_i simply for emphasize but it is not necessary since $d_i \in \pi_u$

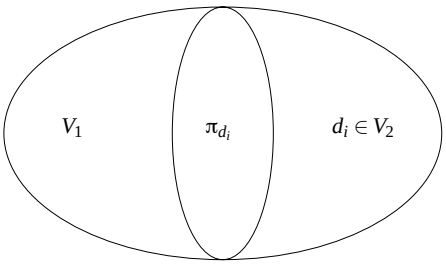

Figure 2: A simple Influence Diagram

Note that $P_T(\cdot)$ indicates that the probability is computed relative to nodes only in T since all other nodes earlier in the graph are irrelevant to this decision. This greatly reduces the computational task for computing the optimal tail decision. Previous methods always included the entire graph in their calculations even though they had no impact. The algorithm then proceeds to recursively divide the BN into a body and a tail, optimizing the tail decision, setting that decision to have decision rule δ_i and repeating on the remaining body. During this process more nodes can be pruned from the reduced tail to improve computation further.

It turns out that after we consider this we can see that the requirement of no-forgetting can now be dropped as long as regularity is maintained. That is, a decision d_i need not have incoming arcs from all $d_1 \ldots d_{i-1}$ previous to it as long as there is still a route going through $d_1 \ldots d_i$ in order. This means that previous decisions can now be 'hidden' from us with chance nodes as shown in Figure 3. The information is not really hidden since d_i is still conditionally dependent on d_{i-1} as long as c_{i-1} is not observed so the decision will still influence future ones. Using Zhang's algorithm d_i can still be optimized. So this opens up influence diagram evaluation to less restricted forms.

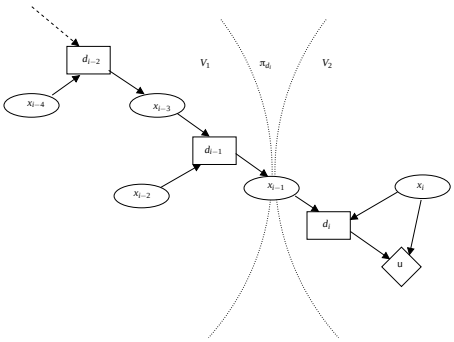

Figure 3: A influence diagram with no-forgetting rule dropped

3.4 Other algorithms

Xiang and Ye [XY01] have recently published an algorithm using BNs that claims to be as efficient as other described here and simpler than Zhang's in particular. Further analysis is required to determine if these claims are significant as the paper does not state any conclusions directly. Much work has also occurred around evaluating influence diagrams without using BNs but using other methods such as Shenoy's valuation based networks [She92] and some recent work by Dechter using BNs but with bucket elimination [Dec00].

4 An Abstract Algorithm

The following method has been proposed informally by David Poole and follows from several intuitive observations. Assume we have influence diagrams as described already which are represented as bayesian networks in the standard way as described by Cooper [Coo88]. Also assume that we have the variable elimination (VE) algorithm [ZP94]. Using VE we can specify to eliminate a node and a *factor* will be produced to distribute away that node's probability to effected nodes in the graph. With these tools in hand we can simply specify a method for evaluating influence diagrams.

We define U_d to be the set of all $u \in U$ such that $d \in an(u)$. This is the set of utility nodes that d needs to maximize. We note that as long as the parents of these utility nodes, π_{U_d}, are all accessible to d then d can be optimized easily, as in Figure 4. Thus the algorithm eliminates any nodes that are parents of a node in U_d but not d.

for all $d \in D$ **do** {chosen in reverse order}
  while $\pi_{U_d} \not\subset (\pi_d \cup d)$ **do**
    Let x be any node in $(\pi_{U_d} - (\pi_d \cup d))$
    eliminate x using Variable Elimination algorithm
    add all nodes in π_x to π_{U_d}
  end while
  for all instantions of variables in π_d **do**
    $\delta^*(\pi_d) = arg\ max_d(P(u = 1|\pi_d, d))$ {choose value for d that maximizes U_d}
  end for
  make d a chance node by setting its CPT to δ^*
end for

We see in

Consider the ID N shown in Figure 4. In this situation we can clearly see that the decision d has all the necessary information to decide optimally.

- For each permutation of π_d choose each decision $\alpha \in dom(d)$ such that $\alpha = arg\ max_\alpha(val_u(\pi_u, \alpha))$
- After this we will have δ^* so we assign that to d's CPT to turn it into another chance node
- Then we eliminate d_k which updates the table of u to take this decision rule into account

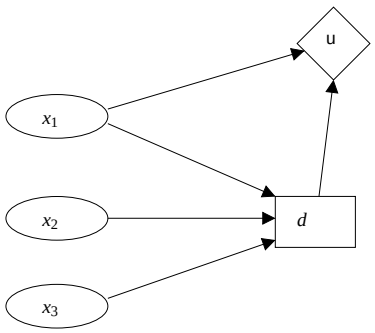

Figure 4: Base case of reduction using the algorithm

- Now we can eliminate x_1, x_2, x_3 to give us the utility node which is really just $E_{\delta^*}[N]$

With this as our base case the rest of the method involves being able to reduce more complicated IDs to ones containing the form in Figure 4. In Figure 5 we can see another simple case. Here the only node inhibiting optimizing d is x_4 since d does not observe it. With VE its easy to fix this, we simply eliminate x_4 which will update the CPTs for u, x_1, d_k.

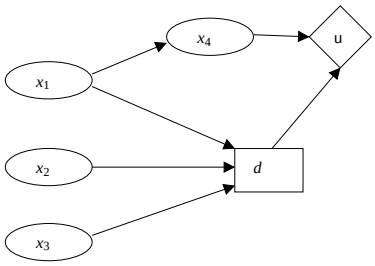

Figure 5: Another simple case of reduction using the algorithm

Let us now consider the ID shown previously for Zhang's algorithm, Figure 3. Can our intuitive algorithm evaluate this situation? If we consider the steps shown in Figure 6 it would seem the answer is yes. Any information arc that would come from d_{i-1} would be irrelevant since x_{i-1} already provides information to us and it does not effect the actual utility. Also, since we are converting d_i to a chance node once it is optimized, the no-forgetting constraint does not apply and d_i can be eliminated using VE without complication.

It is not surprising this method performs as well as Zhang's algorithm. The definition of stepwise-decomposability partitions nodes in the graph using the same fundamental rules that govern conditional independence on which VE is based. The VE algorithm for marginalizing the probabilities inherently does computations locally and

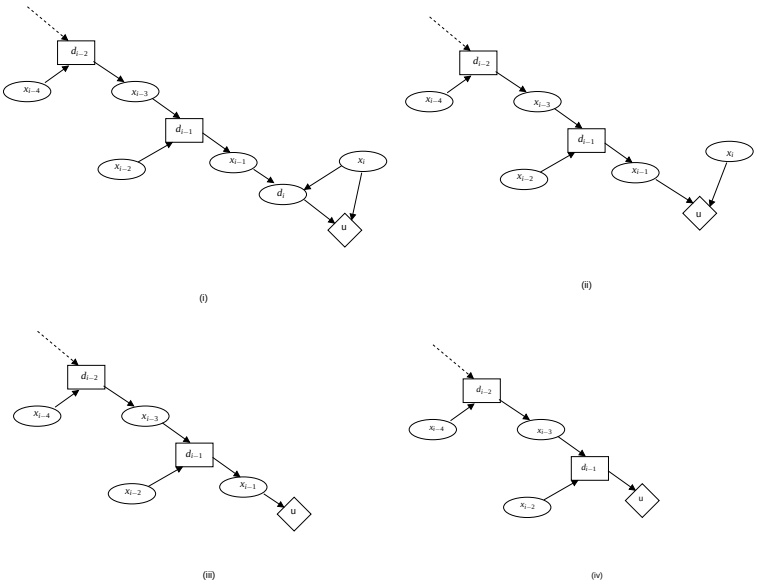

Figure 6: Evaluating a 'forgetful' ID. (i)tail decision node optimized (ii)-(iv) removing other chance nodes before next decision node

need not rely on nodes further back in the graph that are irrelevant given the parents of a node. Zhang's algorithm sets up this framework independent of VE however and thus is general enough to use any bayesian inference technique that works. Our intuitive algorithm is basically a special case of Zhang's where we have hardcoded the use of the inference technique and thus much of the other complexity of the algorithm is taken care of already.

5 Conclusion

We have shown a brief overview of the vast and still growing field of influence diagram research. We have discussed their basis in decision analysis and shown some of the major algorithms for evaluating them in order to come to an optimal policy that maximizes utility. Many of these algorithms utilize bayesian networks and we have also shown an informal algorithm from intuition used by researchers that is a special case of the algorithm by Zhang [Zha98]. There are many open areas of research being actively pursued including ways to further improve evaluation efficiency and extend influence diagrams to ever more general realms.

References

[Coo88] G.F. Cooper. A method for using belief networks as influence diagrams. In *Procceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 55–63, 1988.

[Dec00] Rina Dechter. A new perspective on algorithms for optimizing policies under uncertainty. *American Association for Artificial Intelligence*, 2000.

[HM03] R. A. Howard and J. E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762. Strategic Decisions Group, 2003.

[Sha86] R.D. Shachter. Evalutating influence diagrams. *Operations Research*, 1986.

[She92] P.P. Shenoy. Valuation-based systems for bayesian decision analysis. *Operations Research*, pages 463–484, 1992.

[SP92] R.D. Shachter and Peot. Decision making using probabilistic inference methods. In *Procceedings of the Eigth Conference on Uncertainty in Artificial Intelligence*, pages 276–283, 1992.

[XY01] Y. Xiang and C. Ye. A simple method to evaluate influence diagrams. In *Third International Conference on Cognitive Science*, 2001.

[Zha98] Nevin Zhang. Probabilistic inference in influence diagrams. In *Proceedings of the Fourteenth Conference on Unvertainty in Artificial Intelligence*, pages 514–522, 1998.

[ZP92] Nevin Zhang and David Poole. Stepwise decomposable influence diagrams. In *Proceedings of the Fourth International Conference on Knowledge Representation and Reasoning*, pages 141–152, 1992.

[ZP94] Nevin Zhang and David Poole. A simple approach to bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pages 171–178, 1994.