Approximate representation of optimal strategies from influence diagrams

Finn Verner Jensen
Department of Computer Science
Aalborg University
9220 Aalborg, Denmark
fvj@cs.aau.dk

Abstract

There are three phases in the life of a decision problem, specification, solution, and representation of solution. The specification and solution phases are off-line, while the represention of solution often shall serve an on-line situation with rather tough constraints on time and space. One of the advantages of influence diagrams (IDs) is that for small decision problems, the distinction between phases does not confront the decision maker with a problem; when the problem has been properly specified, the solution algorithms are so efficient that the ID can also be used as an on-line representation of the solution. If the solution algorithm cannot meet the on-line requirements, you will construct an alternative structure for representing the optimal strategy, for example a look-up table or a strategy tree. We report on ongoing work with situations where the solution algorithm is too space and time consuming, and where the policy functions for the decisions have so large domains that they cannot be represented directly in a strategy tree. The approach is to have separate ID representations for each decision variable. In each representation the actual information is fully exploited, however the representation of policies for future decisions are approximations. We call the approximation *information abstraction.* It consists in introducing a dummy structure connecting the past with the decision. We study how to specify, implement and learn information abstraction.

1 Introduction

There are several algorithms for solving IDs (Shachter, 1986),(Shenoy, 1992), (Jensen et al., 1994), but the principle behind them all is dynamic programming starting with the last decision. That is, first an optimal policy for the last decision is determined. Next, this policy is represented somehow, and the optimal policy for the second last decision is determined by using the policy for the last decision for forecasting the future. To illustrate this process, consider the ID in Figure 1. We can represent the optimal policy for the last decision as a conditional probability table (CPT), and the ID from Figure 1 is transformed to the one in Figure 2.

When the solution process is over, you may represent the optimal strategy as a set of CPTs representing the policies, and you will have the Bayesian network in Figure 3.

This process is off-line, and it may require very much space and time. However, as long as the task is tractable the solution phase is not really a problem.

Note that if the ID contains only one decision, then the information variables are instantiated when a decision is to be taken, and the task is reduced to propagation in a Bayesian network. That is, you need not specify the parents of the decision variable, and you have a very compact and efficient representation of the optimal policy.

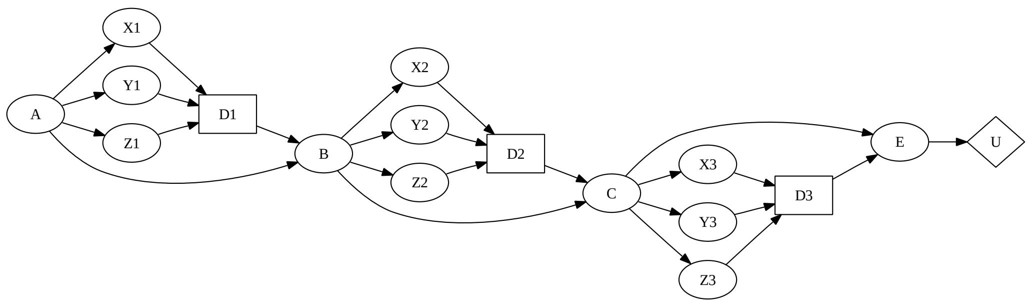

Figure 1: We shall refer to this ID throughout as an illustrating example.

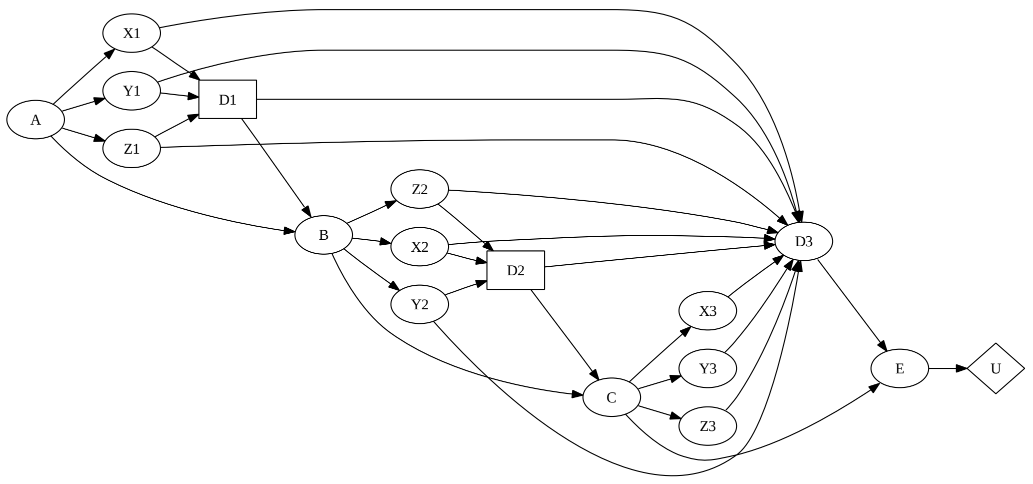

Figure 2: The node for the last decision is substituted with a chance node representation of a policy.

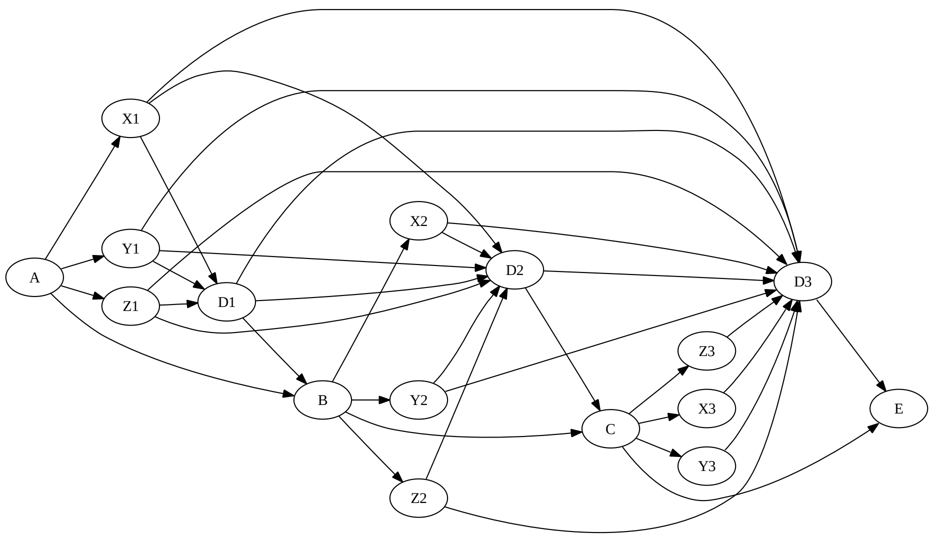

Figure 3: A representation of a strategy. Actually, if we are not interested in the expected utilities, we only need the conditional probabilities for the D-nodes

2 On-line use of a solved influence diagram

We look at situations where the influence diagram is too complex for on-line use. In particular, we consider the situation where the domains for the policies are too large. We shall assume that this is the case for the ID in Figure 1.

Now, consider the last decision, $D3$. When you decide on $D3$, you know the states of $X1, Y1, Z1, D1, X2, Y2, Z2, D2, X3, Y3$, and $Z3$. Then the model in Figure 4 can be used. The information is entered as evidence, and the expected utilities for $D3$ are easily calculated.

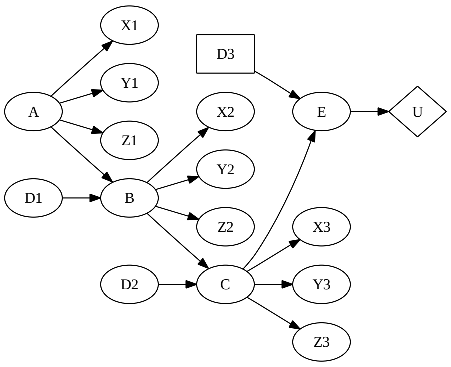

Figure 4: An ID for on-line representation of the last decision.

On the other hand, when deciding $D2$, you need to calculate the expected utility of each option, and in order to do that you need to know the policy δ_{D3} for the future decision. Representing δ_{D3} as in Figure 2 is intractable. You need an alternative representation, and you have to settle with an approximate prepresentation. What is crucial for an approximate representation is that it for each configuration over *past*($D2$) reflects the order of expected utilities for $D2$. (If X is a decision variable, then *past*(X) denotes the set of variables known at the time of deciding on X).

3 Information abstraction

An approximation approach is *information abstraction*: introduce extra structure connecting the information with the decision. There are several schemes for information abstraction. For example, *past*($D2$) way be represented by a *history variable*. An immediate representation would be a chance node H with *past*($D2$) as parents, but if you were faced with intractably learge policy domains, then the CPT for H will also be intractable. Another representation can be a *history belt* (see Figure 5).

Domain knowledge can help to determine a good way of introducing history variables. We shall later discuss ways of learning history variables from the initial ID specification.

We propose another scheme for information abstraction, *conditional decomposition of domains*. Let δ_D be a policy for a decision variable D. A way of decomposing the domain of δ_D would be to assume that the policy has the form:

$$\textbf{if } \mathrm{f(X)} \textbf{ then } \mathrm{g(Y)} \textbf{ else } \mathrm{h(Z)},$$

where X, Y, Z are subsets of the domain, and f is a Boolean valued function. The function f may be an alert function. For example, an unmanned vehicle will focus on fullfilling its mission unless an alert tells it to return for more fuel. In a game scenario, you may try to meet your own goals. However, if your opponent is close to achieving hers, you must focus on blocking her way.

If you from domain knowledge know how δ_D can be decomposed, you can exploite it in the solution phase, and you may not need an approximate representation after all. Using the language *sequential influence diagrams* (SIDs) (Jensen et al, 2006), a specification would look like the one in Figure 6, where $X = \{E, F\}, Y = \{C, E\}, Z = \{F, G\}$.

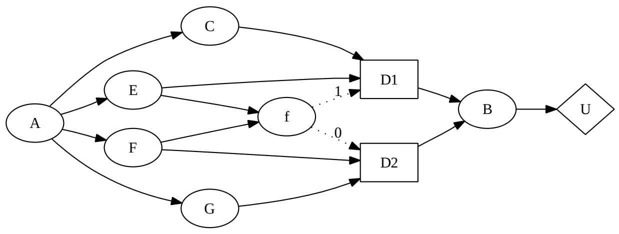

Figure 6: An SID representing a decision with a policy of the form **if** $f(E, F)$ **then** $g(C, E)$ **else** $h(E, G)$.

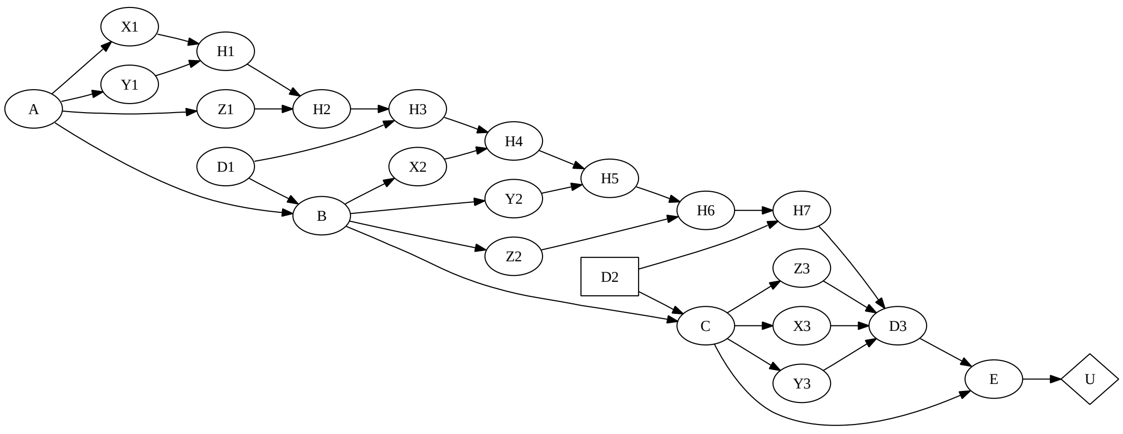

Figure 5: An ID for representing δ_{D2}. δ_{D3} is approximated through a belt of history variables ending with $H7$, wich is assumed to be observed.

The specification in Figure 6 can be translated to two IDs; one for determining $\delta_{D1}(C, E)$ with $f = 1$ inserted as evidence, and one for δ_{D2} with $f = 0$ inserted.

In an ID with several decisions, the decomposed policy from Figure 6 can be represented as in Figure 7.

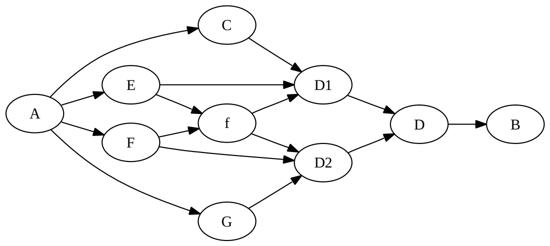

Figure 7: A BN representation of the decomposed policy from Figure 6. The nodes $D1$ and $D2$ have an extra state "no decision".

As indicated above, conditional decomposition may come up naturally from the domain. We will later discuss how artificial decompositions may be learned from the ID specification.

3.1 Overestimation of information

The abstractions presented above underestimate the information available when the decision actually is taken. The information is used to estimate the distribution of the parents of the relevant utility functions, and with abstraction of information, this estimate will be less precise.

You may also overestimate the information. For example, consider the ID in Figure 8.

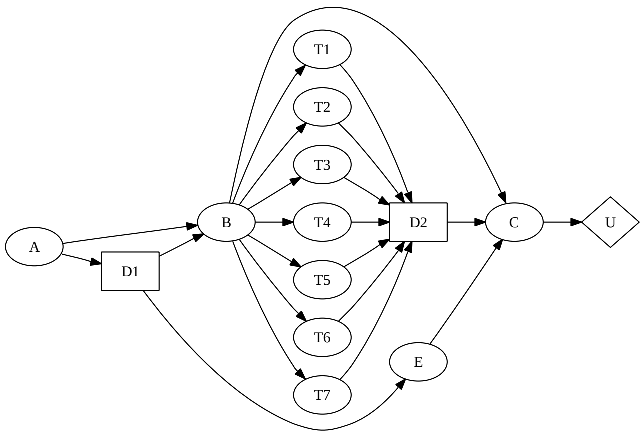

Figure 8: An ID with seven different tests for the same variable.

When deciding $D1$, it may be reasonable to say that when you in the future decide $D2$ you will know the state of B, and a good approximating ID for calcultating the policy for $D1$ will be the one in Figure 9.

We shall not pursue overestimation of information further in this paper.

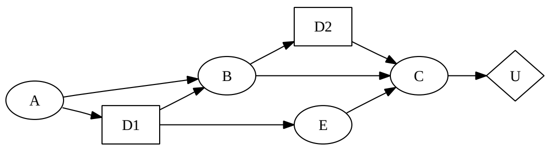

Figure 9: An ID for calculating a policy for $D1$ in Figure 8.

4 Space issues

Just to get an impression of the space issues, consider the example ID in Figure 1. Let the decision nodes and the observed nodes have 5 states, and let the background nodes A, B, C and E have 25 states. Then the maximal clique in a minimal clique junction tree is of size $6,3 \cdot 10^9$. The maximal clique size in a junction tree for the representation in Figure 3 is $2,5 \cdot 10^8$.

If we in the representation with history variables (Figure 1) let $H1$ have ten states and increase the number of states by two up to $H7$ with 22 states, then a junction tree for the model in Figure 1 has size $3,5 \cdot 10^6$, and the maximal clique has size $1,5 \cdot 10^6$. If we instead start with six states for $H1$, increase by one up to $H7$ with 12 states then the junction tree size is 430,000 and the maximal clique has size 187,000.

If we approximate δ_{D3} with the policy "**if** $f(Z1, Z2, Z3)$ **then** $g(X1, X2, X3, D1, D2)$ **else** $h(Y1, Y2, Y3, D1, D2)$", then the junction tree has maximal clique size of $3,9 \cdot 10^6$ and if we use "**if** $f(Z1, Z2, Z3)$ **then** $g(X1, X2, X3)$ **else** $h(Y1, Y2, Y3)$", then the maximal clique size is 165,000.

5 Learning information abstraction

Consider Figure 4, and assume that we wish to learn a representation of δ_{D3} as used in Figure 5. In order to do so you can establish a sample by exploiting a representation proposed by (Cooper, 1988). The decision node is substituted by a chance node with even priors, and the utility node is substituded by a chance node with two states (1 and 0). The conditional probability table $P(U = 1|pa(U))$ represents normalized utilities. For the resulting network it holds that $P(D|U = 1, e)$ is proportional to the expected ulitities for D given the evidence e. Now, sample from the network with $U = 1$ inserted. By removing the unobserved variables (and U) from the table, you have a sample representing $P(D|pa(D))$, which is proportional to the expected utilities of D given $pa(D)$.

5.1 An example

Take the simple example in Figure 10. The parameters in the model are so that the policy for D is characterized by the three functions $(f(Z), g(X), h(Y))$ in the following way:

```
**if** $Z = y$ **then**
    (**if** $X = y$ **then** $D = a_1$ **else** $D = a_2$)
**else** (**if** $Y = y$ **then** $D = a_3$ **else** $D = a_4$).
```

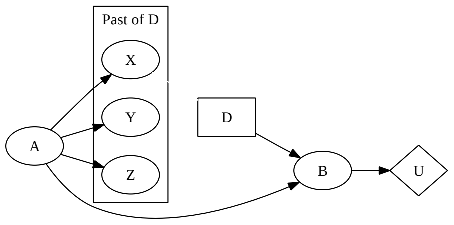

Figure 10: A simple ID to illustrate learning of information abstraction.

The model is transformed to the Bayesian network in Figure 11, where $P(D|X, Y, Z, U = 1)$ is proportional to $EU(D|X, Y, D)$, and therefore $\delta_D(X, Y, Z) = argmax_D P(D|X, Y, Z, U = 1)$.

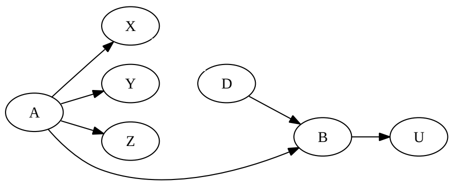

Figure 11: The Bayesian network for sampling. $U = 1$ is inserted before sampling

We sampled 10.000 cases from the model in Figure 11 with $U = 1$ inserted, and we used the EM algorithm (Lauritzen, 1995) to learn the unknown parameters $P(f|Z), P(DX|f, X)$, and $P(DY|f, Y)$ in Figure 12.

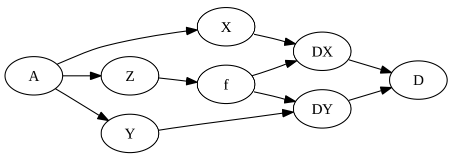

Figure 12: A decomposition model. The CPTs for f, DX, and DY are unknown

The CPT for D reflects that if $f = 1$ then $D = DX$, and if $f = 0$ then $D = DY$. The learning resulted in CPTs for $(f, DX$, and $DY)$ which are very close to the functions (f, g, h). Finally, we modify the CPTs to give probability 1 to the state of maximal probability, and we ended up with the correct policy.

If you do not know the form of the decomposition, then you have to experiment with different structures. For this example, we tried to learn a policy characterized by the three functions $(f(X), g(Z), h(Y))$. The resulting structure had 5 out of 8 decisions correct.

In the model in Figure 13 we have introduced the history variables $H1$ with three states, and $H2$ with four states. The learning procedure resulted in parameters such that the correct decision in all eight cases have maximal probability. To modify the tables to give probability 1 to the decision with maximal probability, you can use various tuning methods (see for example (Jensen and Nielsen, 2006)).

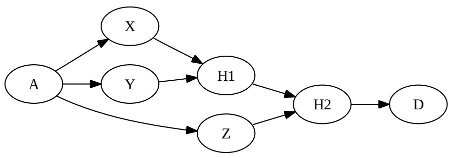

Figure 13: A model with history variables. The CPTs for $H1, H2$, and D are unknown.

6 Discussion and future work

The size of domains of decision policies is a major obstacle in practical use of decision theory for decision problems involving a sequence of decisions. We have in this paper analysed the problem and we have proposed some schemes for addressing the problem. First of all we need a larger set of schemes for information abstraction, and we need experience with underestimation as well as overestimation of information. Issues to study will be complexity as well as precision. We have in this paper presented a method for learning parameters when the structure for information abstraction is given. Learning structure is more intricate. As neither the number of latent variables nor the number of states of the variables are known, the only method known to us is trial and error. We need a systematic way of passing through possible structures.

An alternative method for addressing intractably large decision domains is LIMIDs (Nilsson and Lauritzen, 2001). The approach is to remove some variables from the domains. For the ID in Figure 1, a LIMID structure could be that the decision maker only knows the current information and the previous decision. This is illustrated in Figure 14, where the policies are represented as CPTs for $D1, D2$ and $D3$.

Nilsson and Lauritzen propose an iterative procedure for determining approximate optimal policies: start with an arbitrary set of policies and solve the three single decision IDs; use the calculated policies as CPTs and solve the new single decision ID. Continue so untill no policy is changed.

Following the approach presented in this paper, we would solve the last decision through sampling and use of the EM algorithm to determine a policy for $D3$; then use this policy to determine a policy for $D2$ through sampling and EM, and eventually solve the ID for $D1$. It is an interesting issue for further research to compare these two approaches.

References

Gregory F. Cooper (1988). A method for using belief networks as influence diagrams. *Fourth Workshop on Uncertainty in Artificial Intelligence*: 55–63.

Finn V. Jensen, Thomas D. Nielsen and Prakash P Shenoy (2006). Sequential influence diagrams: A unified asymmetry framework. *International Journal of Approximate Reasoning*, 42(1–2): 101-118.

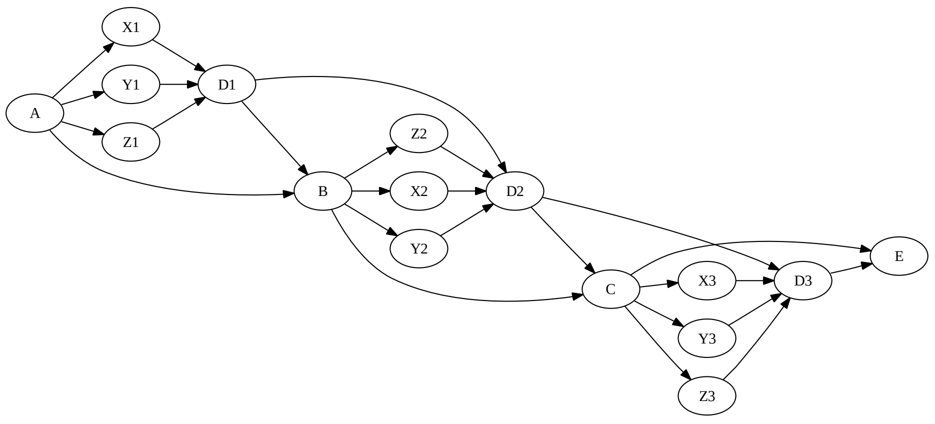

Figure 14: A LIMID structure representing decisions where only the current information and the previous decision are included in the decision domain

Finn V. Jensen, Thomas D. Nielsen (2007). *Bayesian Networks and Decision Graphs. Springer, New York.*

Frank Jensen, Finn V. Jensen and Søren L. Dittmer (1994). From Influence Diagrams to Junction trees. *Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann: 367–374.

Steffen L. Lauritzen (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19: 191–201.

Dennis Nilsson and Steffen L. Lauritzen (2001). Representing and solving decision problems with limited information. *Management Science*, 47: 1235-1251.

Ross Shachter (1986). Evaluating influence diagrams. *Operations Research*, 34(6): 871–882.

Prakash P. Shenoy (1992). Valuation Based Systems for Bayesian Decision Analysis. *Operations Research*, 40(3): 463–484.