On the impact of causal independence

Irina Rish and **Rina Dechter**
Department of Information and Computer Science
University of California, Irvine
{*irinar,dechter*}*@ics.uci.edu*

Abstract

Reasoning in Bayesian networks is exponential in a graph parameter called *induced-width* (also known as tree-width and max-clique size). In this paper, we investigate how a property called *causal independence* can improve this performance. We show that the "effective" induced-width of algorithms exploiting this property can be significantly reduced: it can be as small as the induced width of the unmoralized network's graph, and will never exceed the induced-width of the network's moral graph. For example, for poly-trees, causal independence reduces complexity from exponential to linear in the family size.

Our analysis is presented for belief updating first, and is then extended to three other tasks: finding a most probable explanation (MPE), finding the maximum aposteriori hypothesis (MAP) and finding the maximum expected utility (MEU). We show that, while causal independence can considerably reduce the complexity of belief updating, MAP and MEU, it may have no effect on MPE. For the first three tasks, we present variable-elimination algorithm that exploite causal independence.

Introduction

Bayesian networks is a widely used framework for reasoning under uncertainty. Although probabilistic inference is generally NP-hard (Cooper 1990), structure-exploiting algorithms such as join-tree propagation (Lauritzen & Spiegelhalter 1988; Jensen, Lauritzen, & Olesen 1990; Shafer & Shenoy 1990) and variable elimination (Zhang & Poole 1996; Dechter 1996) are available. They are all time and space exponential in the induced width (the maximal clique size) of the network's moral graph. Unfortunately, the induced width is frequently large, especially if the network's maximum family size (m) is large. Moreover, in networks with large families, the input conditional probability tables (CPTs), which are exponential in the family size, are too large to be specified explicitly.

One way to cope with this later problem is to identify structural properties of the probabilistic dependencies *within each family* that simplify the CPT's specification and may result in savings. In this paper, we focus on a property known as *causal independence* (Heckerman & Breese 1994; Srinivas 1993; Zhang & Poole 1996), where multiple causes contribute *independently* to a common effect. Examples of causally independent probabilistic relationships are *noisy-OR* and *noisy-MAX*, used as simplifying assumptions in large practical systems, such as CPCS and QMR-DT networks for medical diagnosis (Pradhan *et al.* 1994).

Our work builds upon the two main approaches of, respectively, (Heckerman & Breese 1994) and (Olesen *et al.* 1989), and (Zhang & Poole 1996), which demonstrated some computational benefits of causal independence. The first approach transforms each family into a binary tree Bayesian network using hidden variables resulting in a probabilistic network whose family size is bounded by two. The second approach presents an algorithm (called VE1 in (Zhang & Poole 1996)) that can be viewed as applying a network transformation implicitly during the execution of a variable elimination algorithm.

The algorithm presented here for belief-updating, called ci-elim-bel, combines and extends both approaches by explicitly creating a graph transformation and by subsequently applying a variable elimination algorithm on the transformed graph. The most important feature of the algorithm is that it can be shown to have complexity guarantees that explicitly tie the algorithm's performance to the induced-width of "decomposition graph", called the *effective induced width*. This explicit connection leads to heuristic orderings which result in better performance and allows identifying and modifying undesirable behavior in previous approaches.

We show that exploiting causal-independence can reduce the *effective induced-width* to be as small as the induced-width of the network's *unmoralized* graph, and to never exceed the induced-width of the network's moral graph. In particular, for each given network, the anticipated computational benefits due to causal independence can be evaluated in advance, and contrasted with those of the general purpose algorithm. For example, for poly-trees with arbitrary large family size

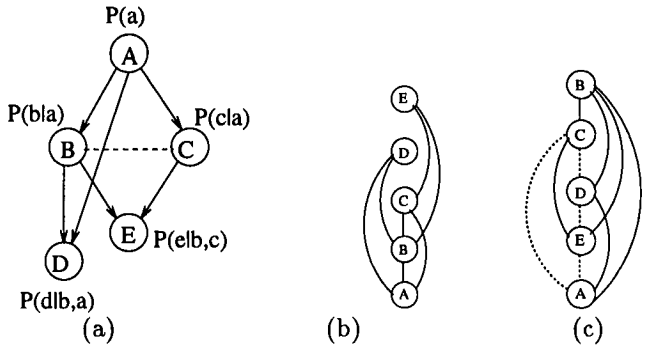

Figure 1: (a) A belief network, (b) its induced graphs along $o = (a, b, c, d, e)$, and (c) it induced graph along $o = (a, e, d, c, b)$.

m the *effective induced width* can be reduced from m to 2.

Finally we investigate the impact of causal independence on tasks such as finding the most probable explanation (MPE), finding the maximum aposteriori hypothesis (MAP) and finding the of maximum expected utility (MEU). We observe that while causal independence can reduce the complexity of belief updating and maximizing expected utility considerably, it is much less effective for tasks such as MPE.

Definitions and preliminaries

Let $X = \{x_1, ..., x_N\}$ be a set of random variables over a domain of size d. A *belief network* is a pair (G, P), where G is a directed acyclic graph on X, and $P = \{P(x_i|pa_i)|i = 1, ..., N\}$ is the set of conditional probabilities defined for each x_i and its *parents* pa_i (a node y is called a *parent* of x if there is a directed edge from y to x). The belief network represents a joint probability distribution over X having the product form

$$P(x_1, ..., x_N) = \prod_{i=1}^{N} P(x_i|pa_i).$$

We call a node and its parents a *family*. The *moral graph* G_M of a belief network (G, P) is obtained by connecting the parents of each node in G and dropping directionality of edges. An example of a belief network is shown in Figure 1a, where the dashed line denotes the edge added by moralization. An *evidence* e is an instantiated subset of variables. The following queries are defined on belief networks: 1. *belief updating*, i.e. finding the posterior probability $P(Y|e)$ of *query* nodes $Y \in X$ given evidence e; 2. finding *most probable explanation* (MPE), i.e. finding a maximum probability assignment to unobserved variables given evidence; 3. finding *maximum aposteriory hypothesis* (MAP), i.e. finding a maximum probability assignment to a set of *hypothesis* nodes, given evidence; 4. given a *utility* function, finding the *maximum expected utility* (MEU), namely, finding assignment to a set of *decision* nodes that maximizes the expected *utility* function specified on the network's nodes.

The most common probabilistic inference algorithms include join-tree algorithm (Lauritzen & Spiegelhalter 1988; Jensen, Lauritzen, & Olesen 1990; Shafer & Shenoy 1990) and variable elimination (D'Ambrosio 1994; Zhang & Poole 1996; Dechter 1996).

We briefly review the *bucket elimination* algorithm *elim-bel* (Dechter 1996) for belief updating. By definition, the query $P(x_1|e)$ is computed as follows:

$$P(x_1|e) = \alpha P(x_1, e) = \alpha \sum_{x_2} \cdots \sum_{x_N} \prod_i P(x_i|pa_i),$$

where α is a normalizing constant. By distributivity law,

$$\sum_{x_2} \cdots \sum_{x_N} \prod_i P(x_i|pa_i) = F_1 \sum_{x_2} F_2 \cdots \sum_{x_N} F_N,$$

where each $F_i = \prod_x P(x|pa(x))$ is the product of all probabilistic components such that either $x = x_i$, or $x_i \in pa(x)$. The set of all such components is initially placed into the *bucket* of x_i (denoted $bucket_i$). Algorithm *elim-bel* processes the bucket of each variable, from N to 1, by summing out the bucket's variables, thus computing the summation above from right to left. For each x_i, it multiplies the components of $bucket_i$, sums out x_i, and puts the resulting function in the bucket of its highest-index variable. If $x_i = a$, x_i is instantiated in each of the bucket's components, and the resulting functions are placed into appropriate buckets. Similar bucket elimination algorithms can be derived for finding MPE, MAP, and MEU (Dechter 1996).

The complexity of the bucket elimination algorithms is exponential in the *induced width* w_o^* (Dechter & Pearl 1987) of the network's moral graph along the processed ordering o. Given a graph G, the *width* of x_i along o is the number of x_i's earlier neighbors in o. The *width of the graph* along o, w_o, is the largest width along o. The *induced graph* of G along o is obtained by connecting the preceding neighbors of each x_i, going from x_N to x_1. The *induced width* along o, w_o^*, is the width of the induced graph along o, while the *induced width* w^* is the minimum induced width along any ordering. For example, Figures 1b and 1c show the induced graphs of the moral graph in Figure 1a along the orderings $o = (a, b, c, d, e)$ and $o' = (a, e, d, c, b)$, respectively. We get $w_o^* = 2$ and $w_{o'}^* = 4$. It can be shown that the induced width of node x_i is identical to the number of arguments of a function computed in $bucket_i$. Therefore,

Theorem 1: *(Dechter 1996) The complexity of algorithm elim-bel is* $O(Nd^{w_o^*+1})$, *where* w_o^* *is the induced width of the moral graph along ordering* o.

Although finding an ordering having smallest induced width is NP-hard (Arnborg, Corneil, & Proskurowski

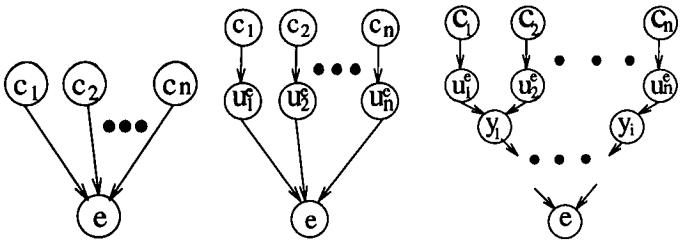

Figure 2: (a) Causally independent belief network; (b) its dependency graph and (c) its decomposition network

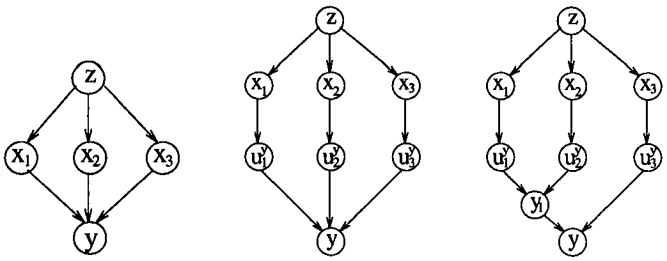

Figure 3: (a) A belief network, (b) its dependency graph, and (c) its decomposition graph

1987), good heuristic orderings are available (Bertele & Brioschi 1972; Dechter 1992; Becker & Geiger 1996).

Causal independence

Causal independence assumes that several causes contribute independently to a common effect. Specifically, a probabilistic relation between a set of causes $c_1, ..., c_n$ and an effect e (Figure 2a) can be decomposed into a noisy transformation of each cause c_i into a *hidden* variable u_i, and a deterministic function $e = u_1 * ... * u_n$, where $*$ is a binary operator. A graph depicting this relation (*dependency graph*) is shown in Figure 2b. Formally,

Definition 1: (Heckerman & Breese 1994; Zhang & Poole 1996) Let $c_1, ..., c_m$ the parents of e in a Bayesian network. The variables $c_1, ..., c_m$ are said to be *causally independent* w.r.t. e if there exists a set of random variables $u_1^e, ..., u_m^e$, a set of probabilities $P(u_i^e|c_i)$, and a binary commutative and associative operator $*$ such that 1. for each i, u_i^e is independent on any of c_j and u_j^e, $i \neq j$, given c_i, i.e.
$P(u_i^e|c_1, ..., c_n, u_1^e, ..., u_n^e) = P(u_i^e|c_i)$, and
2. $e = u_1^e * \ldots * u_n^e$.

A CPT of such a causally independent family can therefore be specified by m components $P(u_i^e|c_i)$ and can be computed by:

$$P(e|c_1^e, ...c_m^e) = \sum_{\{u_1^e ... u_m^e | e = u_1^e * ... * u_m^e\}} \prod_{i=1}^{m} P(u_i^e|c_i^e).$$

This summation can be computed linearly in m using pairwise decomposition, such as

$$\sum_{\{u_1^e, y_1 | e = u_1^e * y_1\}} P(u_1^e|c_1^e) \sum_{\{u_2^e, y_2 | y_1 = u_2^e * y_2\}} P(u_2^e|c_2^e) \ldots$$

$$\ldots \sum_{\{u_{m-1}^e, u_m^e | y_{m-2} = u_{m-1}^e * u_m^e\}} P(u_{m-1}^e|c_{m-1}^e) P(u_m^e|c_m^e),$$

where y_j are *hidden* variables introduced for keeping the intermediate results. Such pairwise summation corresponds to a traversal of a binary computation tree (Figure 2c), where each (non-leaf) node y represents the result of an operation $y = y_l * y_k$ on its children y_l and y_k.

Given a belief network (G, P), replacing each causally independent family in G by such computation tree (i.e., replacing Figure 2a by Figure 2c) yields a *decomposition graph* G_D. This transformation introduces *hidden* nodes to G, which include both the nodes u_i^e due to the definition of causal independence, and the nodes y_j of each computation tree. We assume that the probabilities in P are specified as $P = \{P(u_j^x|c_j^x)|c_j^x \in pa(x)\}$. A *decomposition network* of a belief network (G, P) is defined by tuple (G_D, P, C), where G_D is a decomposition graph, and C is the set of constraints $y = y_l * y_k$ introduced by the computation trees. The nodes of the input belief network are called the *input* nodes.

A decomposition network closely resembles the network transformations in (Heckerman & Breese 1994; Olesen *et al.* 1989), with the exception that in our approach the new CPTs for each hidden variable y_i are not compiled explicitly.

Belief updating

We next present an elimination algorithm *ci-elim-bel* for belief updating when causal independence is assumed. The algorithm's derivation is demonstrated on the following example.

Example 1: Consider the *noisy-OR* belief network in Figure 3a. Causal independence is explicitely depicted by graph in Figure 3b. The query $P(x_3|x_1 = 0)$ can be computed as follows:

$$\sum_{z,y,x_2,x_1=0} P(z)P(x_1|z)P(x_2|z)P(x_3|z)P(y|x_1,x_2,x_3) =$$

$$\sum_{z,y,x_2,x_1=0} P(z)P(x_1|z)P(x_2|z)P(x_3|z) \times$$

$$\times \sum_{\{u_1^y, u_2^y, u_3^y | y = u_1^y \vee u_2^y \vee u_3^y\}} P(u_1^y|x_1)P(u_2^y|x_2)P(u_3^y|x_3).$$

Assuming the decomposition network in Figure 3c, decomposing $\sum_{\{u_1^y, u_2^y, u_3^y | y = u_1^y \vee u_2^y \vee u_3^y\}}$ accordingly, and

pushing the summation over x_1 and x_2 as far to the right as possible, yields:

$$\sum_z P(z)P(x_3|z)\sum_y \sum_{\{y_1,u_3^y|y=y_1\vee u_3^y\}} P(u_3^y|x_3)\times$$

$$\sum_{\{u_1^y,u_2^y|y_1=u_1^y\vee u_2^y\}} \sum_{x_2} P(x_2|z)P(u_2^y|x_2)\sum_{x_1=0} P(x_1|z)P(u_1^y|x_1).$$

The variables can be summed out from right to left along the ordering $o = (x_3, z, y, \{y_1, u_3^y\}, \{u_1^y, u_2^y\}, x_2, x_1)$.

In general, the summations can be rearranged in any *legal* order, namely, as long as each pair of hidden variables $\{y_l, y_k\}$ is summed out before its parent y in the decomposition tree, where $y = y_l * y_k$. Note that some summation are performed over pairs of hidden variables, thus the variable orderings allow such pairs, and the notion of width and induced width of such orderings can be defined in the obvious way.

Algorithm *ci-elim-bel* presented in Figure 4 computes $P(x_1|e)$. It assumes as an input a decomposition network (G_D, P, C), a legal ordering $o = (Z_1, ..., Z_n)$, where $Z_1 = \{x_1\}$, and evidence e. As noted above, Z_i may be either a single variable, or a pair of sibling hidden variables in the decomposition network. Each Z_i is associated with its bucket ($bucket_i$) relative to o. First, the components of P, C and e are partitioned into buckets. Each component is placed into the bucket of its highest-index variable. Then the buckets are processed along the reverse ordering, from Z_n to Z_1. If $bucket_i$ contains evidence $x = a$, we instantiate x in all components and place each into an appropriate lower bucket. Otherwise, we compute the product F of the bucket's components and sum out Z_i's variable(s). If Z_i is a single variable x, we compute $\sum_x F$, otherwise, if Z_i is a pair (y_l, y_k), (in which case a constraint $y = y_l * y_k$ is in the bucket) we compute $\sum_{\{y_l,y_k|y=y_l*y_k\}} F$, and add the resulting component into the bucket of its highest variable. Finally, $P(x_1|e)$ is computed in $bucket_1$ as the normalized product of all its components.

We next analyze the computational complexity of *ci-elim-bel*. Recall that G_M denotes the moral graph of a belief network (G, P), and G_D denotes a decomposition graph. The largest family size in a belief network will be denoted by m. Since a computation tree of each expression $e = u_1 * ... * u_m$ contains $O(m)$ nodes,

Theorem 2: [transformation complexity] *The complexity of transforming a causally independent belief network into a decomposition network is* $O(Nm)$. □

Since the size of G_D is $O(Nm)$, from theorem 1 it follows that

Theorem 3: [ci-elim-bel complexity] *Given a decomposition network* (G_D, P, C) *of a causally independent belief network* (G, P), *and a legal ordering* $o = (Z_1, ..., Z_n)$ *of* G_D, *the complexity of ci-elim-bel is* $O(Nmd^{w_o^*+1})$, *where* w_o^* *is the induced width of* G_D *along* o. □

Algorithm ci-elim-bel

Input: A decomposition network,
a legal ordering $o = (Z_1, ..., Z_n)$,
and evidence e.
Output: $P(x_1|e)$.
1. **For** $i = n$ **to** $i = 1$, /* make buckets */
   For each $x \in Z_i$, put in $bucket_i$
   all network's components
   whose highest ordered variable is x.
2. **For** $i = n$ **downto** 1 **do** /* elimination */
   /* Assume enumeration $\lambda_1, ..., \lambda_m$ of all probabilistic components in $bucket_i$. */
- **If** $x = a$ is in the bucket, /* observation */
  replace x by a in each λ_i and put the result in appropriate lower bucket.
- **else**
  if $Z_i = \{x\}$, /* input variable */
  $\lambda^{Z_i} \leftarrow \sum_x \Pi_j \lambda_j$.
  else /* $Z_i = \{y_l, y_k\}$, $y = y_l * y_k$. */
  $\lambda^{Z_i} \leftarrow \sum_{\{y_l,y_k|y=y_l*y_k\}} \Pi_j \lambda_j$.
  Put λ^{Z_i} into the highest bucket that mentions λ^{Z_i}'s variable.
3. **Return** $Bel(x_1) = \alpha\lambda^{x_1}$,
   where α is a normalizing constant.

Figure 4: Algorithm ci-elim-bel

The induced width of a decomposition graph G_D, $w^*(G_D)$, will be also called the *effective induced width*. Since the complexity of *elim-bel* is exponential in the induced width of the network's moral graph, $w^*(G_M)$, the interesting question is how $w^*(G_D)$ compares to $w^*(G_M)$. We will show that in many cases $w^*(G_D)$ is much smaller than $w^*(G_M)$, and it can get as low as $w^*(G)$, the induced width of the original (unmoralized) graph G. In such cases *ci-elim-bel* can be exponentially faster than *elim-bel*.

Theorem 4: [poly-trees] *Algorithm ci-elim-bel requires* $O(Nmd^3)$ *time and space on a causally-independent poly-tree.*

Proof: A decomposition network of a poly-tree having N nodes and families of size not larger than m is also a poly-tree having $O(Nm)$ nodes. Since there always exists an ordering of a poly-tree having the induced width not larger than 2, the complexity of processing this decomposition network is $O(Nmd^3)$. □

Theorem 4 generalizes Pearl's result for noisy-or poly-trees (Pearl 1988). Remember, that the standard poly-tree algorithm is exponential in size of largest parent set , i.e. $O(Nd^m)$.

An example of a network with loops that has similar property is shown in Figure 3. The network has a

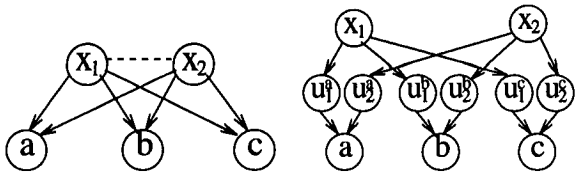

Figure 5: (a) A 2-3 network and (b) its decomposition network

"source" node z and "sink" node y, and a middle layer of nodes $x_1,...,x_n$, such that there is an arc from z to each x_i, and from each x_i to y. The standard CPT representation and inference algorithms require $O(d^n)$ time and space, while *ci-elim-bel* is $O(nd^3)$. For more details see (Rish & Dechter 1998).

In general, causal independence is exploited if *parents are eliminated before their children*, since eliminating a child node first reconstructs the full CPT. However, in some cases, it is better to ignore causal independence.

Example 2: Consider the network in Figure 5a, and its decomposition network in Figure 5b. Eliminating each parent x_i connects $u_i^a, u_i^b, and u_i^c$ to each other, therefore, eliminating next the pair $\{u_1^a, u_2^a\}$ connects a to the rest of the hidden variables, yielding $w^* = 5$. However, the induced width of the original moral graph (Figure 5a) along o =(a, b, c, x_1, x_2) is just 2. To attain the same w^* in the decomposition network, we need to eliminate the hidden variables first, which in fact reconstructs the CPTs.

The network in Figure 5a is an example of a *k-n-network*, which is a two-layer network with k nodes in the top layer, each connected to (some of) n nodes in the bottom layer. An example of *k-n-network* is Binary Node 2-layer Noisy-OR (BN2O) network used in QMR-DT (Pradhan *et al.* 1994).

The example above suggests that eliminating parents (e.g., x_1 and x_2) before their children and before their hidden nodes in a *k-n-network* yields $O((k+n)d^{2n})$ complexity, while the complexity of eliminating the hidden nodes first (thus reconstructing the CPTs) is $O((k+n)d^k)$. Indeed, we can show that

Theorem 5: *The complexity of ci-elim-bel on causally independent k-n-networks is* $O((k+n)d^{min\{k,2n\}})$. □

Since the complexity of *elim-bel* is $O((k+n)d^k)$ on such network, exploiting causal independence can greatly improve this performance when n is fixed and k is increasing. However, when $k < 2n$, we better ignore causal independence.

In (Rish & Dechter 1998), we describe a simple ordering procedure guaranteeing that *ci-elim-bel*'s performance is at least as good as the performance of *elim-bel*. This procedure ensures that, when necessary, causal independence is ignored in a family by eliminating hidden variables before the other family members. On the other hand, algorithm VE1 (Zhang & Poole 1996) which is similar to *ci-elim-bel*, but imposes certain restrictions on the ordering of the hidden nodes, can be, as a result, *exponentially worse* than *elim-bel* along same ordering of the input nodes.

In summary, causal independence always allows reducing the family size (note that the maximum number of parents in a decomposition network is 2). However, it is not always clear how the introduction of hidden variables affects the effective induced width. The following proposition weakly ties the effective induced width to the complexity of unmoralized graph:

Proposition 1: *Given a causally independent network* (G, P), *an ordering* o, *and a decomposition graph* G_D, *there exist an extension of* o *to an ordering* o' *of the nodes in* G_D, *such that the induced width of* G_D *along* o', *restricted only to the input nodes, is not larger than the induced width of the (unmoralized)* G *along* o. □

In many cases, the induced width of G_D coincides with its induced width restricted to the input nodes, which equals $w_o^*(G)$ by theorem 1. In summary,

Theorem 6 : *The complexity of ci-elim-bel is* $O(exp(w^*(G_D)))$, *where* $w^*(G) \leq w^*(G_D) \leq w^*(G_M)$.

Due to space restrictions, most proofs in this section are ommitted.

Optimization tasks: finding MPE, MAP, and MEU

Next, we investigate the impact of causal independence on the tasks of finding MPE, MAP, and MEU. Surprisingly, it is not always possible to take advantage of causal independence in optimization problems. The technical barrier is that permuting maximization and summation is not allowed, i.e.

$$\max_x \sum_y f(x,y) \neq \sum_y \max_x f(x,y).$$

This restricts the order of computations so that sometimes CPT decomposition cannot be exploited.

Consider the task of finding MPE.

$$MPE = \max_{x_1,...,x_N} \prod_i P(x_i|pa_i) = \max_{x_1} F_1 \ldots \max_{x_N} F_N,$$

where $F_i = \prod_x P(x|pa(x))$ is the product of all probabilistic components such that either $x = x_i$, or $x_i \in pa(x)$. The bucket elimination algorithm *elim-mpe* sequentially eliminates x_i from right to left. However, the decomposition introduced by the causally independent CPTs within each F_j cannot be exploited. For example, given a causally independent family in Figure 2a, having 3 parents c_1, c_2 and c_3,

$$MPE = \max_{c_1,c_2,c_3,e} P(c_1)P(c_2)P(c_3)P(e|c_1,c_2,c_3) =$$

$$= \max_{c_1} P(c_1) \max_{c_2} P(c_2) \max_{c_3} P(c_3) \max_{e}$$

$$\sum_{\{y_1, u_1^e | e = y_1 * u_1^e\}} P(u_1^e|c_1) \sum_{\{u_2^e, u_3^e | y_1 = u_2^e * u_3^e\}} P(u_2^e|c_2) P(u_3^e|c_3).$$

Maximization and summation cannot be interchanged here. Therefore, the hidden variables have to be summed out before maximizing over c_1, c_2, c_3. This reconstructs the CPT on the whole family, i.e. causal independence has no effect.

Nevertheless, the two other optimization tasks of computing MAP and MEU are able to take advantage of causal independence, because both tasks involve also summation over a subset of the variables. By definition,

$$MAP = \max_{x_1, \ldots, x_m} \sum_{x_{m+1}, \ldots, x_N} \prod_i P(x_i|pa_i),$$

where $x_1, \ldots, x_k$ are the *hypothesis* variables. For example, given the same network in Figure 2a and hypothesis e,

$$MAP = \max_{e} \sum_{c_1, c_2, c_3} P(c_1) P(c_2) P(c_3) P(e|c_1, c_2, c_3).$$

Decomposing the causally independent $P(e|c_1, c_2, c_3)$ and rearranging the summation order gives:

$$MPE = \max_{e} \sum_{c_1} P(c_1) \sum_{c_2} P(c_2) \sum_{c_3} P(c_3)$$

$$\sum_{\{y_1, u_1^e | e = y_1 * u_1^e\}} P(u_1^e|c_1) \sum_{\{u_2^e, u_3^e | y_1 = u_2^e * u_3^e\}} P(u_2^e|c_2) P(u_3^e|c_3) =$$

$$= \max_{e} \sum_{\{y_1, u_1^e | e = y_1 * u_1^e\}} \sum_{\{u_2^e, u_3^e | y_1 = u_2^e * u_3^e\}}$$

$$\sum_{c_1} P(c_1) P(u_1^e|c_1) \sum_{c_2} P(c_2) P(u_2^e|c_2) \sum_{c_3} P(c_3) P(u_3^e|c_3).$$

As we can see, the summations over the parents c_i and the corresponding hidden variables can be rearranged in this example. In general, *a decomposition of CPTs due to causal independence can be exploited when (at least some) parents in the causally independent family are not included in the hypothesis.*

Algorithm *ci-elim-map* for computing MAP in causally independent networks is shown in Figure 6. It is similar to *ci-elim-bel*, except that a decomposition network *does not transform* the families of child nodes which are included in the hypothesis. Those families will be moralized in the decomposition graph. The hypothesis variables are eliminated by maximiziation over $\prod_j \lambda_j$, where λ_j are the probabilistic components in the corresponding bucket. The hypothesis variables $H \in X$, are placed first in the ordering.

Similarly to MAP, MEU is computed by summing over a subset of the variables and maximizing over the rest. By definition,

$$MEU = \max_{x_1, \ldots, x_m} \sum_{x_{m+1}, \ldots, x_N} \prod_i P(x_i|pa_i) U(x_1, \ldots, x_N),$$

Algorithm ci-elim-map

Input: A decomposition network,
a hypothesis $H \in X$, an evidence e, and
a legal ordering $o = (Z_1, \ldots, Z_n)$, where
$Z_1, \ldots, Z_m$ are hypothesis variables.

Output: An assignment $h = \arg\max_a P(H = a|e)$.

1. **For** $i = n$ **to** $i = 1$, /* make buckets */
   For each $x \in Z_i$, put into $bucket_i$
   all network's components
   whose highest ordered variable is x.
2. /* Elimination */
   /* Assume enumeration $\lambda_1, \ldots, \lambda_k$ of all probabilistic components in $bucket_i$. */
2.1. **For** $i = n$ **downto** $m + 1$ **do** /* sum-out */
   - **If** $x = a$ is in the bucket, /* observation */
     replace x by a in each λ_i and put
     the result in appropriate lower bucket.
   - **else**
     if $Z_i = \{x\}$ /* input variable */
     $\lambda^{Z_i} \leftarrow \sum_x \prod_j \lambda_j$.
     else /* $Z_i = \{y_l, y_k\}$, $y = y_l * y_k$. */
     $\lambda^{Z_i} \leftarrow \sum_{\{y_l, y_k | y = y_l * y_k\}} \prod_j \lambda_j$.
     Put λ^{Z_i} into the highest bucket
     that mentions λ^{Z_i}'s variable.

2.2. **For** $i = m$ **downto** 1 **do**
   $\lambda^{Z_i} \leftarrow \max_x \prod_j \lambda_j$.
3. **For** $i = 1$ **to** $i = m$, /* find assignment h */
   $h_i \leftarrow \arg\max_a \prod_j \lambda_j(x = a, \ldots)$
4. **Return** assignment h.

Figure 6: Algorithm ci-elim-map

where $x_1, \ldots, x_m$ are *decision* variables (usually denoted $d_1, \ldots, d_m$), $x_{m+1}, \ldots, x_N$ are *chance* variables, and $U(x_1, \ldots, x_N)$ is a *utility function*. The utility is often assumed to be *decomposable*: $U(x_1, \ldots, x_N) = \sum r(x_i)$, where $r(x_i)$ are individual utilities, or *rewards*. A belief network with decision nodes and utility function is also called an *influence diagram*.

A bucket elimination algorithm *elim-meu* for computing the MEU was presented in (Dechter 1996). It assumes that *the decision variables have no parents*, and puts them first in the ordering.

Assume that c_1 in example of Figure 2a is a decision variable, and that the utility is decomposable, $U(d, e, c_2, c_3) = r(d) + r(e) + r(c_2) + r(c_3)$. Then

$$MEU = \max_{d} \sum_{e, c_2, c_3} P(c_2) P(c_3) P(e|d, c_2, c_3) U(d, e, c_2, c_3).$$

Decomposition of $P(e|d, c_2, c_3)$ into pairwise sums yields:

$$\max_d \sum_e \sum_{e, c_2, c_3} P(c_2) P(c_3) \sum_{\{y_1, u_1^e | e = y_1 * u_1^e\}} P(u_1^e|d) \times$$

$$\times \sum_{\{u_2^e,u_3^e|y_1=u_2^e*u_3^e\}} P(u_2^e|c_2)P(u_3^e|c_3)U(d,e,c_2,c_3).$$

Pushing the summations over c_2 and c_3 to the right yields

$$\max_d \sum_e \sum_{\{y_1,u_1^e|e=y_1*u_1^e\}} P(u_1^e|d) \sum_{\{u_2^e,u_3^e|y_1=u_2^e*u_3^e\}}$$
$$\sum_{c_2} P(c_2)P(u_2^e|c_2) \sum_{c_3} P(c_3)P(u_3^e|c_3)U(d,e,c_2,c_3).$$

Using decomposability of the utility, the last summation (over c_3) can be expressed as

$$\sum_{c_3} P(c_3)P(u_3^e|c_3)[r(d)+r(e)+r(c_2)+r(c_3)] =$$
$$= [\sum_{c_3} P(c_3)P(u_3^e|c_3)](r(d)+r(e)+r(c_2))+$$
$$+\sum_{c_3} P(c_3)P(u_3^e|c_3)r(c_3) =$$
$$= [\sum_{c_3} P(c_3)P(u_3^e|c_3)][r(d)+r(e)+r(c_2)+$$
$$+\frac{\sum_{c_3} P(c_3)P(u_3^e|c_3)r(c_3)}{\sum_{c_3} P(c_3)P(u_3^e|c_3)}] =$$
$$= \lambda^{c_3}(u_3^e)[r(d)+r(e)+r(c_2)+\mu^{c_3}(u_3^e)],$$

where

$$\lambda^{c_3}(u_3^e) = \sum_{c_3} P(c_3)P(u_3^e|c_3),$$
$$\mu^{c_3}(u_3^e) = \frac{\theta_{c_3}(u_3^e)}{\lambda^{c_3}(u_3^e)}, \text{and}$$
$$\theta^{c_3} = \sum_{c_3} P(c_3)P(u_3^e|c_3)r(c_3).$$

Summing over c_3 is the operation performed in the the bucket of c_3. Thus, i n each bucket, we will have probability components, denoted λ_i, and utility components μ_j. Initially, the probabilistic components are those specified in the belief network, and the utility components are the individual rewards. Then, in each bucket we compute a new pair of λ and μ, as demonstrated above, and place them into the appropriate lower buckets. In our example, the both new components $\lambda^{c_3}(u_3^e)$ and $\mu^{c_3}(u_3^e)$ will be placed into the bucket of $\{u_2^e, u_3^e\}$.

Algorithm *ci-elim-meu* is shown in Figure 7. Since causal independence is not defined for decision nodes, the decomposition network transforms only the families of chance nodes. Similarly to *ci-elim-bel* and and *ci-elim-map*, *ci-elim-meu* first partitions into buckets all the network components (including the utility components). Then it processes chance nodes, from last to first, computing the new λ and μ (see step 2.1 of the algorithm). The only difference between *ci-elim-meu* and *elim-meu* (Dechter 1996) is that the summation operation in buckets of hidden variables is different. Finally, the buckets of the decision nodes are processed by maximization and an optimal assignment to those nodes is generated.

Similarly to *ci-elim-bel*,

Algorithm ci-elim-meu

Input: A decomposition network, a legal ordering $o = (Z_1, ..., Z_n)$, $Z_1 = \{d_1\}$,..., $Z_m = \{d_m\}$, an evidence e.

Output: An assignment $d^1, ..., d^m$ to decision variables that maximizes the expected utility.

1. **Initialization:**
   For $i = n$ **to** $i = 1$ /* make buckets */
   For each $x \in Z_i$, put into $bucket_i$ all network's components whose highest ordered variable is x.
2. /* Elimination */
/* λ_j and μ_k denote probabilistic and utility components, correspondingly, in $bucket_i$. */
2.1. **For** $i = n$ **to** $m + 1$ **do** /* chance nodes */
   - **If** $x = a$ is in the bucket, /* observation */ replace x by a in each λ_i and put the result in appropriate lower bucket.
   - **Else if** $Z_i = \{x\}$,
     $\lambda_i \leftarrow \sum_x \Pi_j \lambda_j(S_j)$.
     $\theta_i \leftarrow \sum_x \Pi_j \lambda_j(S_j) \sum_k \mu_k(S_k)$.
   - **Else if** $Z_i = \{y_l, y_k\}$ **and** $y = y_l * y_k$,
     $\lambda_i \leftarrow \sum_{\{y_l,y_k|y=y_l*y_k\}} \Pi_j \lambda_j(S_j)$
     $\theta_i \leftarrow \sum_{\{y_l,y_k|y=y_l*y_k\}} \Pi_j \lambda_j(S_j) \sum_k \mu_k(S_k)$.
     Put λ_i and $\mu_i = \theta_i/\lambda_i$ into the buckets of their highest-index variables.

2.2. **For** $i = m$ **to** 1 **do**
   /* decision nodes $Z_i = \{d\}$ */
   $\theta_i \leftarrow \Pi_j \lambda_j(S_j) \sum_k \mu_k(S_k)$
   $\gamma_i \leftarrow \max_d \theta_i$
   $d^i = \arg\max_d \theta_i$
   Put γ_i in the bucket of its highest variable.
3. Return an optimal set of functions $(d_1, ...d_m)$ recorded in the decision buckets, and the maximum expected utility V_e.

Figure 7: Algorithm ci-elim-meu

Theorem 7: *The complexity of ci-elim-map and ci-elim-meu is* $O(Nmd^{w_o^*+1})$, *where* w_o^* *is the induced width of a decomposition network along its ordering* o. □

The decomposition graph can be inspected to determine the benefits of causal independence before running the algorithm. In general, as for belief updating, the induced width of the decomposition network can be as small as the induced width of the original unmoralized graph, and not larger than the induced width of the network's moral graph.

Related work

Algorithm *ci-elim-bel* bridges the gap between the *network transformation* approach (Heckerman & Breese 1994; Olesen *et al.* 1989)) and elimination algorithm VE1 (Zhang & Poole 1996).

The first approach transforms each family in a network into a binary tree using hidden variables. The result is a belief network whose family size is bounded by two. The second presents a variable elimination algorithm VE1, that can be viewed as applying a network transformation implicitly during the execution of a variable elimination algorithm.

It can be shown that VE1 is equivalent to *ci-elimbel* with some ordering restrictions, which imply that hidden variables cannot be eliminated before the input variables in a family. As a result, in some cases, such as k-n-networks, VE1 can be exponentially worse than a regular elimination algorithm.

The algorithms presented in this paper can be further improved for special cases, such as noisy-OR. For more details see (Rish & Dechter 1998).

Conclusion

In this paper we augment bucket-elimination algorithms for probabilistic inference with features that exploit causal independence. The complexity of the modified algorithms for tasks such as belief updating, finding the maximum aposteriori hypothesis and finding the maximum expected utility is exponential in the *induced width* of the network's *decomposition graph* which never exceeds the induced width of the network's moral graph and can be sometime as small as the induced-width of the unmoralized network's graph. Consequently, exploiting causal independence can only reduce complexity, sometimes by an exponential factor (e.g., poly-trees).

We investigate the degree to which causal independence helps for each of the tasks mentioned, namely, for belief updating, finding MPE, MAP, and MEU, showing that the task of finding the most probable explanation (MPE) may not be helped at all by causal independence.

References

Arnborg, S.; Corneil, D.; and Proskurowski, A. 1987. Complexity of finding embedding in a k-tree. *Journal of SIAM, Algebraic Discrete Methods* 8(2):177–184.

Becker, A., and Geiger, D. 1996. A sufficiently fast algorithm for finding close to optimal junction trees. In *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence*, 81–89.

Bertele, U., and Brioschi, F. 1972. *Nonserial Dynamic Programming*. New York: Academic Press.

Cooper, G. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence* 42(2–3):393–405.

D'Ambrosio, B. 1994. Symbolic probabilistic inference in large bn2o networks. In *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*, 128–135.

Dechter, R., and Pearl, J. 1987. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence* 34:1–38.

Dechter, R. 1992. Constraint networks. In *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, 2nd edition.

Dechter, R. 1996. Bucket elimination: A unifying framework for probabilistic inference. In *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence*, 211–219.

Heckerman, D., and Breese, J. 1994. A new look at causal independence. In *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*, 286–292.

Jensen, F.; Lauritzen, S.; and Olesen, K. 1990. Bayesian updating in recursive graphical models by local computations. *Computational Statistics Quarterly* 4:269–282.

Lauritzen, S., and Spiegelhalter, D. 1988. Local computations with probabilities on graphical structures and their applications to expert systems. *Journal of the Royal Statistical Society, Series B* 50(2):157–224.

Olesen, K.; Kjaerulff, U.; Jensen, F.; Falck, B.; Andreassen, S.; and Andersen, S. 1989. A munin network for the median nerve - a case study on loops. *Applied Artificial Intelligence* 3:384–403.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, California: Morgan Kaufmann Publishers.

Pradhan, M.; Provan, G.; Middleton, B.; and Henrion, M. 1994. Knowledge engineering for large belief networks. In *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*.

Rish, I., and Dechter, R. 1998. On the impact of causal independence. Technical report, Information and Computer Science, UCI.

Shafer, G., and Shenoy, P. 1990. Probability propagation. *Annals of Mathematics and Artificial Intelligence* 2:327–352.

Srinivas, S. 1993. A generalization of noisy-or model. In *Proc. Ninth Conf. on Uncertainty in Artificial Intelligence*, 208–215.

Zhang, N., and Poole, D. 1996. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence research* 5:301–328.