# A Discussion about Variable Elimination Algorithm and Extended Variable Elimination Algorithm in Bayesian Networks

J.K.F. (Jordi) Ozir and Y. (Yunpeng) Zhang

June 21, 2015

**Abstract**

In general the computation of probabilities in a *Bayesian Network (BN)* is NP-hard. Thus it is in general not possible to efficiently compute the probabilities in a BN. The computation can be de done more efficiently if we apply specific algorithms to specific network structures. The objective of this paper is to discuss two algorithms which address this task: *Variable Elimination Algorithm* and *Extend Variable Elimination Algorithm.*

## 1 Introduction

Computation of probabilities in *Bayesian Network (BN)* is NP-hard. Thus it is in general computationally inefficient to compute the probabilities in a BN. This has been proven by Cooper in [1]. The computation can be made more efficient if a specific algorithm is applied to a specific network structure. In this paper we will discuss two of these algorithms: These are *Variable Elimination Algorithm (VEA)* and *Extend Variable Elimination Algorithm (EVEA)*, which are described in [9] and [10], respectively. VEA allows to do the computation more efficiently by using a factorization of joint probabilities distributions. EVEA extends this by also taking *Casual Independence (CI)* into account.

We will now discuss how the paper is structured. We will discuss in Chapter 1.1 preliminary information that we use throughout this paper. Then, we will briefly describe in Chapter 1.2 other algorithms that address the same problem as VEA and EVEA. Thereafter, we will elaborate in Chapter 2 VEA and EVEA. We will discuss the experiments and results in Chapter 3. Finally, we will conclude this paper in Chapter 4.

### 1.1 Preliminary Information

We will now describe preliminary information that we use to explain the two algorithms in the chapter hereafter.

**Definition 1** ***(Bayesian Network)*** We define a BN as an ordered pair $\mathcal{N} = (G, P))$. Where $G = (V, A)$ is an acyclic directed graph. The set of nodes (or variables) in $G$ is represented by $V$. The set of arcs in $G$ is defined by $A$. Where $P$ is a joint probability distribution such that the following holds:

$$P(X_V) = \prod_{v \in V} P(X_v | X_{\pi(v)}).$$

Where $\pi(v)$ are the set of parents of a node $v$ in $G$. We see in Figure 1 an example of a BN. We will use it in order to explain VEA. It is factorized as follows:

$$P(x_1, \ldots, x_7) = P(x_1)P(x_6|x_1)P(x_2|x_1)P(x_3|x_2)P(x_4|x_2)P(x_5|x_3, x_4)P(x_7|x_5, x_6).$$

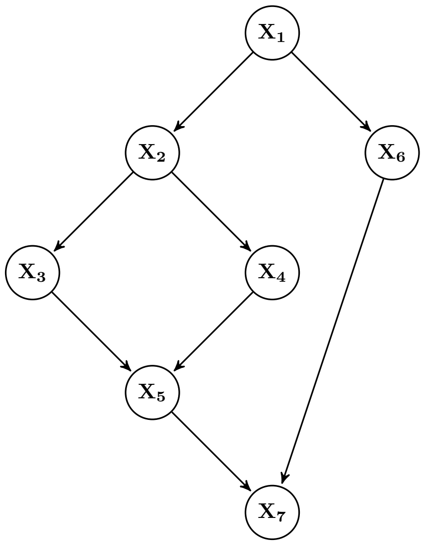

Figure 1: Example BN

The probability distribution for a variable $X \in V$ is defined as $P(X = x)$. We formulate a joint probability formally as $P(X_1, \ldots, X_n)$, and particularly, when $X_1, \cdots, X_n$ are independent variables then:

$$P(X_1, \ldots, X_n) = \prod_{X \in V} P(X), \text{ for all } i = 1, \ldots, n.$$

We are able to compute a probability of a single variable by marginalizing out the redundant variables out of the joint probability distribution. We do this as follows:

$$P(x_1) = \sum_{Y \in V} P(x_1, Y).$$

We define a conditional probability as:

$$P(X|Y) = \frac{P(X, Y)}{P(Y)}.$$

We read this as the probability of $X$ when information of $Y$ is provided.

## 1.2 Related Work

CI refers to the situation where multiple *causes* contribute independently to a common *effect*. A well-known example is the noisy OR-gate model[4]. Heckerman[5] was the first to formalize the general concept of CI. The formalization was later refined by Heckerman and Breese[6].

Kim and Pearl[7] showed how the use of noisy OR-gate can speed up inference in a special kind of BNs known as polytrees; D'Ambrosio[2][3] showed the same for two level BNs with binary variables. For general BNs, Olesen *et al*[8]. and Heckerman[5] proposed two ways of using causal independencies to transform the network structures. Which is also involved in the following chapters.

# 2 Variable Elimination Algorithm (VEA)

We will now describe VEA which was proposed by Poole and Zhang in [9]. We will start with describing the theoretical foundation and then we explain how the algorithm can be applied on a BN by means of an example.

**Definition 2 (*leaf node*)** We define a leaf node in a BN $\mathcal{N}$ as a node which does not have children.

**Definition 3 (*barren node*)** We consider a node *barren* with respect to a query when it is a leaf node and it is not in $X \cup Y$.

We formulate Proposition 1 based on Definition 1 and 2.

**Proposition 1** Provided with a BN $\mathcal{N}$ and a leaf node $v$. We define a $\mathcal{N}'$ as the BN from which $v$ has been removed. Therefore we have the following equality:

$$P_{\mathcal{N}}\left(X|Y = Y_0\right) = P_{N'}\left(X|Y = Y_0\right).$$

**Definition 4 (*moral graph*)** We define a moral graph $m\left(G\right)$ of a directed graph $G = (V, A)$ as the graph from which the directions are removed and common parents are connected.

**Definition 5 (m-*separation)*** Given the following sets of nodes $X$, $Y$ and $Z \in m\left(G\right)$. Given that $X$ and $Z$ are separated by $Y$. We say that $X$ and $Z$ are $m$-separated by $Y$ in $G$.

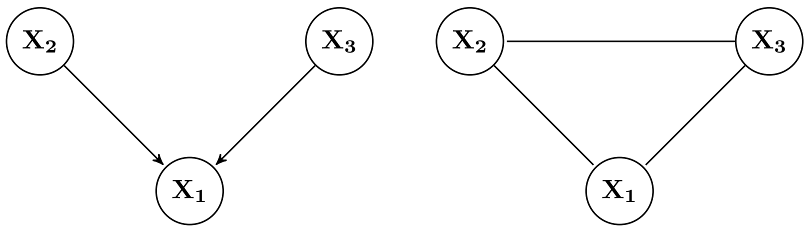

Figure 2: We see on the left a BN. We see on the right its moralized counterpart.

We are able to define Proposition 2 based on Definition 3 and 4.

**Proposition 2.** Given a BN $\mathcal{N}$ and a query $P_{\mathcal{N}}(X|Y = Y_0)$. We define $N^{'}$ as the BN from which all the nodes that are $m$-separated from $X$ by $Y$ are removed in $\mathcal{N}$. This will result in the following equality:

$$P_{\mathcal{N}}(X|Y = Y_0) = P_{N'}(X|Y = Y_0).$$

We will show by means of an example how Proposition 1 and Proposition 2 can be applied in order to reduce the structure of a BN such that the computation of a probability can be done more efficiently. We see in Figure 1 the initial BN. We will remove $X_7$ in Figure 3. The node has been coloured. Proposition 1 allows use to remove this node because it has no children and is thus a leaf node.

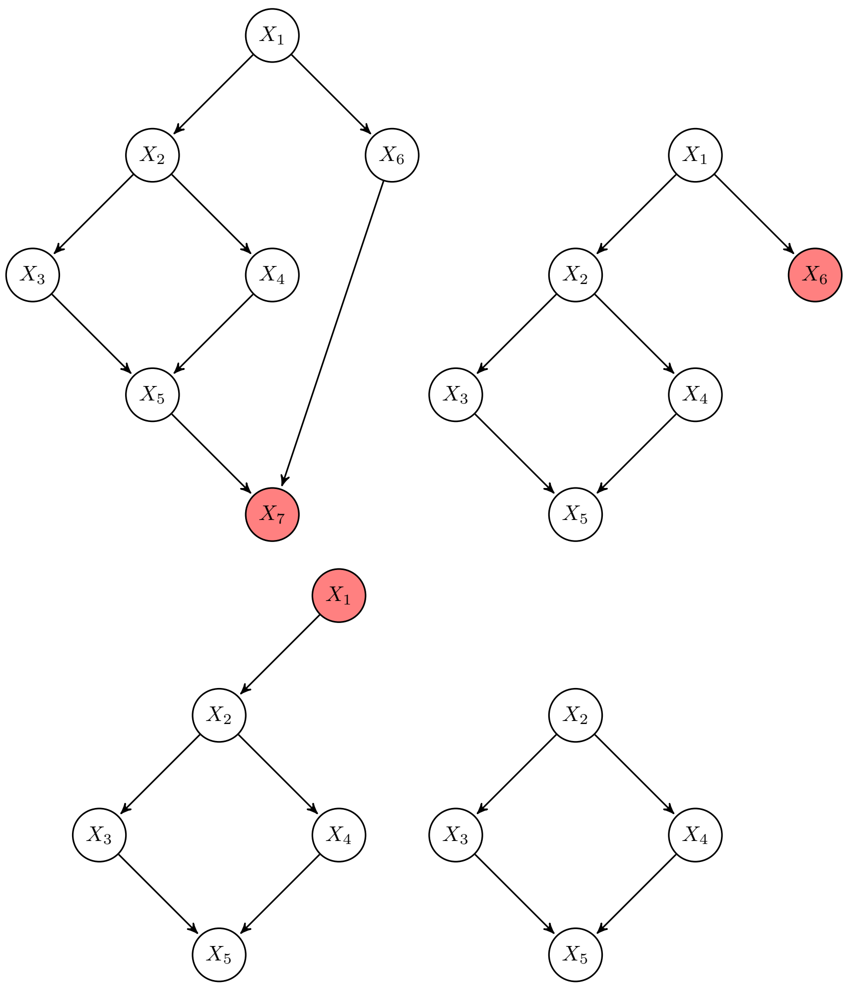

Figure 3: A visual demonstration of simplefying the network structure.

We apply again Proposition 1 on $X_6$ because after removing $X_7$ it has become a leaf node. In the next figure we apply Proposition 2 on node $X_1$. The figure thereafter shows the final BN.

We use the BN in Figure 1 as an example to compute the probability of $X_1$. We naively compute this my summing $X_1$ out as follows:

$$P(x_1) = \sum_{X_i \in V/x_1} P\left(x_1, X_2, X_3, X_4, X_5, X_6, X_7\right)$$

In order to obtain $P(x_1)$ we use the whole network. If we assume that the probabilities distributions are binary valued it would imply that we would have to store $2^n$. Thus in this particular situation we store 128 numbers. This could be done more efficient by using factorisations of potentials. We will provide a definition of a potential and two variants of it.

**Definition 6** ***(potential)*** A potential is a non-normalized variant of probability distribution.

**Definition 7** ***(joint potential)*** We define a joint potential $P_S\left(V\right)$ as a multiplication of the potentials in set $S$ defined over a set of variables $V$.

**Definition 8** ***(marginal potential)*** We define a marginal potential $P_S\left(X\right)$ by marginalizing out the variables outside of $X$ from the joint potential $P_S(V)$.

We have introduced the fundamental theory which explains when a node is allowed to be removed and what is needed to understand VEA.

Efficiently computing the probabilities of a BN consists of three stages. The objective of the first stage is to find an *elimination ordering*. That is an ordering of the variables that need to be removed. The second stage defines how the third stage will do the calculations and in which order. The third stage does the calculations. It takes as input a set of potentials. Then a variable is summed out and the result is returned.

VEA is an example of the second stage that we described above. We will demonstrate how a variable is marginalized by means of a set of potentials. We are interested in marginalizing $x_1$ out the network from Figure 1. We do this as follows:

$$\psi_{x_1}\left(x_2, x_6\right) = \sum_{x_1} P(x_1)P(x_2|x_1)P(x_6|x_1).$$

We are left with the following set of probability distributions which constitutes the BN: $\{P(x_3|x_2),\ P(x_4|x_2),\ P(x_5|x_3, x_4),\ P(x_7|x_5, x_6),\ \psi_{x_1}\left(x_2, x_6\right)\}$.
We will apply VEA in order to compute the query: $P(x_7, x_6)$. Given the BN in Figure 1 and are provided with the following the following elimination order $(x_2, x_3, x_4, x_5, x_6)$. Then we will have the following initial situation:

| $x_2$: | $x_3$: | $x_4$: | $x_5$: | $x_6$: |
|---|---|---|---|---|
| $P(x_2\|x_1)$ | $P(x_5\|x_3, x_4)$ |  | $P(x_7\|x_5, x_6)$ | $P(x_6\|x_1)$ |
| $P(x_3\|x_2)$ |  |  |  |  |
| $P(x_4\|x_2)$ |  |  |  |  |

We define $\psi_{x_2}(x_3, x_4) = \sum_{x_2} P(x_2|x_1)P(x_3|x_2)P(x_4|x_2)$ and reduce to:

| $x_3$: | $x_4$: | $x_5$: | $x_6$: |
|---|---|---|---|
| $P(x_5|x_3, x_4)$ | | $P(x_7|x_5, x_6)$ | $P(x_6|x_1)$ |
| $\psi_{x_2}(x_3, x_4)$ | | | |

Then we define $\psi_{x_3}(x_4, x_5) = \sum_{x_3} P(x_5|x_3, x_4)\psi_{x_2}(x_3, x_4)$ and reduce to:

| $x_4$: | $x_5$: | $x_6$: |
|---|---|---|
| $\psi_{x_3}(x_4, x_5)$ | $P(x_7|x_5, x_6)$ | $P(x_6|x_1)$ |

We define $\psi_{x_4}(x_5) = \sum_{x_4} \psi_{x_3}(x_4, x_5)$ and we obtain:

| $x_5$: | $x_6$: |
|---|---|
| $P(x_7|x_5, x_6)$ | $P(x_6|x_1)$ |
| $\psi_{x_4}(x_5)$ | |

We define $\psi_{x_5}(x_6, x_7) = \sum_{x_4} P(x_7|x_5, x_6)\psi_{x_4}(x_5)$.

| $x_6$: |
|---|
| $P(x_6|x_1)$ |
| $\psi_{x_5}(x_6, x_7)$ |

$\psi_{x_6}(x_7) = \sum_{x_6} P(x_7|x_5, x_6)\psi_{x_5}(x_6, x_7)$.

**Algorithm 1** Procedure VEA$(S, P(X, Y = Y_0), W)$

```
Input:
    • $S$: a set of potentials
    • $P(X, Y = Y_0)$: a standard query
    • $W$: elimination ordering
Output: $P_S(X, Y = Y_0)$
  Set $Y$ to $Y_0$ in the potentials $S$, resulting in $S_1$.
  Associate each potential $\psi$ of $S_1$ with the variable that appears earliest in $W$
  among all variables of $\psi$.
  while $W$ is not empty do
      Remove the first variable on $W$. Denote this variable by $v$.
      Call procedure do third stage. To multiply all the potentials associated $v$
  together and to sum out $v$ from the product, resulting $\psi_v$.
      Associate $\psi_v$ with the variable that appears earliest in $W$ among all the
  variables of $\psi_v$.
  return The potential from the removal of the last variable in $W$ which is
  $P_S(X, Y = Y_0)$
```

## 2.1 Extended Variable Elimination Algorithm (EVEA)

Now that we have explained the VEA. We will describe an extension on this algorithm the EVEA. We will start with explaining CI. In case a BN is not restricted to how a node depends on its parents, it would result in an exponential number of conditional probabilities for each node. We are able to efficiently solve this problem by means of CI. We define an effect $e$ which has $m$ parents:

$c_1, c_2, \cdots, c_m$, and $c_1, c_2, \cdots, c_m$ are said to be *causal independent* w.r.t. $e$ if there exists random variables $\xi_1, \xi_2, \cdots, \xi_m$ that have the same set of possible values as $e$ such that:

1. For each $i, \xi_i$ probabilistically depends on $c_i$ and is conditionally independent of all other $c_j$'s and all other $\xi_j$'s given $c_i$;

2. There exists a commutative and associative binary operator $*$ over the frame of $e$ such that $e = \xi_1 * \xi_2 * \cdots * \xi_m$.

There are several CI, i.e. *noisy OR-gates*, *noisy MAX-gates*, *noisy AND-gates*, *noisy adders* , etc.

**Definition 9** ***(convergent variable)*** In a BN a convergent variable collects and combines different independent parents of itself, and non-convergent variable is called *regular variable*

The VEA which we described earlier emphasizes on the factorization of the joint probability distributions. The EVEA extends the VEA by means of CI. Thus we are able factorize the form $P(e|c_1, \cdots, c_m)$ as follows:

**Proposition 3** Let $e$ be a node in a BN and let $c_1, c_2, \cdots, c_m$ be the parents of $e$. If $c_1, c_2, \cdots, c_m$ are causally independent w.r.t. $e$, then the conditional probability $P(e|c_1, \cdots, c_m)$ can be obtained from the conditional probabilities $P(\xi_i|c_i)$ as follows:

$$P(e = \alpha|c_1, \cdots, c_m) = \sum_{\alpha_1 * \cdots \alpha_m = \alpha} P(\xi_1 = \alpha_1|c_1) \cdots P(\xi_m = \alpha_m|c_m),$$

where for each value $\alpha$ of $e$. The symbol $*$ indicates the base combination operator of $e$.

Furthermore, Zhang and Poole introduced an operator for combining factors which contains convergent variables. Consider two factors $f$ and $g$, let $e_1, \cdots, e_k$ be the convergent variables that appears in both $f$ and $g$, let $A$ be the list of regular variables in both $f$ and $g$, let $B$ be the list of variables that appear only in $f$, and $C$ be the list of variables that appear only in $g$. Suppose $*_i$ is the base combination operator of $e_i$. Then, the *combination* $f \otimes g$ *of* $f$ *and* $g$ is a function of variables $e_1, \cdots, e_k$ and of the variables in $A$, $B$ and $C$:

$$\begin{aligned}
&f \otimes g(e_1 = \alpha_1, \cdots, e_k = \alpha_k, A, B, C) \\
&= \sum_{\alpha_{11} *_1 \alpha_{12} = \alpha_1} \cdots \sum_{\alpha_{k1} *_k \alpha_{k2} = \alpha_k} \\
&\ f(e_1 = \alpha_{11}, \cdots, e_k = \alpha_{k1}, A, B) g(e_1 = \alpha_{12}, \cdots, e_k = \alpha_{k2}, A, C)
\end{aligned}$$

And the combination operator $\otimes$ has several properties:

**Proposition 4** If $f$ and $g$ do not share any convergent variables, then $f \otimes g$ is simply the multiplication of $f$ and $g$. And the operator $\otimes$ is both commutative and associative.

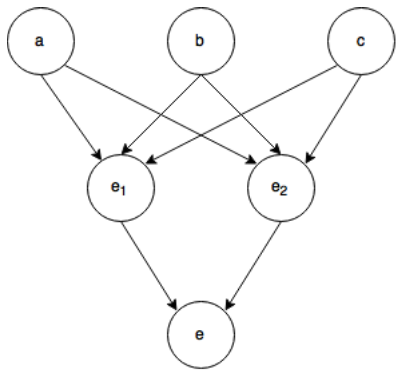

Figure 4: A BN

**Proposition 5** Suppose $f$ and $g$ are factors and variable $z$ appears in $f$ and not in $g$, then:

$$\sum_z (fg) = (\sum_z f)g$$
$$\sum_z (f \otimes g) = (\sum_z f) \otimes g$$

With the combination operator $\otimes$, one could factorize $P(e|c_1, \cdots, c_m)$ as:

$$P(e|c_1, \cdots, c_m) = \otimes_{i=1}^{m} f_i(e, c_i)$$

For example, in the BN shown in Figure 4, the joint probability $P(a, b, c, e_1, e_2, e_3)$ could be factorized into following list of factors:

$$P(a), P(b), P(c), P(e_1|a, b, c), P(e_2|a, b, c), P(e_3|e_1, e_2)$$

Now we can see that all the $e_i$'s are convergent variables, so we could further factorize their conditional probabilities as:

$$P(e_1|a, b, c) = f_{11}(e_1, a) \otimes f_{12}(e_1, b) \otimes f_{13}(e_1, c)$$
$$P(e_2|a, b, c) = f_{21}(e_2, a) \otimes f_{22}(e_2, b) \otimes f_{23}(e_2, c)$$
$$P(e_3|e_1, e_2) = f_{31}(e_3, e_1) \otimes f_{32}(e_3, e_2)$$

here for instance $f_{11}(e_1, a)$ is the contributing factor of $a$ to $e_1$. Hence the factors in the factorization of a joint probability distribution could be divided into *heterogeneous factors* with others before and *homogeneous factors.*

**Definition 9** ***(heterogeneous factor)*** A heterogeneous factor is the factor that needs to be combined with the other heterogeneous factors by the operator $\otimes$ before it can be combined with other factors by multiplication.

**Definition 10** ***(homogeneous factor)*** In contrast a heterogeneous factor, the factor that directly could be combined by multiplication is called *homogeneous factor.*

The $f_{ij}$'s are heterogeneous factors since they need to combine with other $f_{ik}$'s then it could be combined with other factors by multiplication. While $P(a)$, $P(b)$, $P(c)$ in this case are homogeneous factors since they only need to be combined with each other by multiplication.

To support the propositions above, the concept of *deputation* is introduced.

**Definition 11** ***(deputation)*** Deputation is a transformation that one can apply to a BN so that the factorization of the joint probability distribution in the BN could respect the propositions above. Let $\epsilon$ be a convergent variable. To dipute $\epsilon$ is to make a copy $\epsilon'$ of $\epsilon$, make the parents of $\epsilon$ be parents of $\epsilon'$, replace $\epsilon$ with $\epsilon'$ in the contributing factors of $\epsilon$, make $\epsilon'$ the only parent of $\epsilon$, and set the conditional probability $P(\epsilon|\epsilon')$ as:

$$P(\epsilon|\epsilon') = \begin{cases} 1 & \text{if } \epsilon = \epsilon' \\ 0 & \text{otherwise} \end{cases}$$

And $\epsilon'$ is the deputy of $\epsilon$, after the deputation, variable $\epsilon$ becomes a *new regular variable*, and $\epsilon'$ becomes a *convergent variable*. The variables that are regular variables before deputation are denoted as *old regular variables*. What's more, $P(\epsilon|\epsilon')$ is a homogeneous factor.

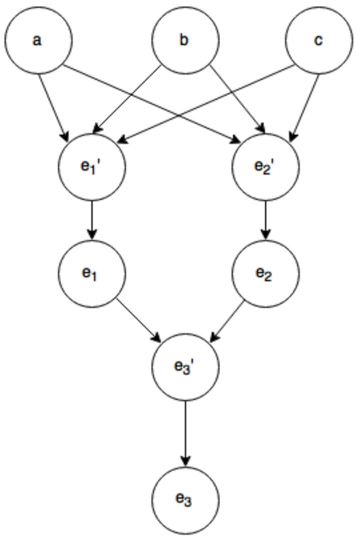

Figure 5: The BN in Figure 4 After the Deputation

So a deputation BN is obtained from a BN by deputing all the convergent variables. Figure 5 shows the deputation of BN in Figure 4. The joint probability

$$P(a, b, c, e_1, e_1', e_2, e_2', e_3, e_3'),$$

into homogeneous factors:

$$P(a), P(b), P(c), P(e_1|e_1'), P(e_2|e_2'), P(e_3|e_3'),$$

and heterogeneous factors

$f_{11}(e'_1, a), f_{12}(e'_1, b), f_{13}(e'_1, c), f_{21}(e'_2, a), f_{22}(e'_2, b), f_{23}(e'_2, c), f_{31}(e'_3, e_1), f_{32}(e'_3, e_2).$

This factorization has three important properties. That are:

1. each heterogeneous factor contains one and only one convergent variable.
2. each convergent variable $e'$ appears in one and only on homogeneous factor.
3. except for the deputing functions, none of the homogeneous factors contain any convergent variables.

More importantly, deputation does not change the answer to a query, which is
**Proposition 6** The posterior probability $P(X|Y = Y_0)$ is the same in a BN as in its deputation.

With the concepts and propositions that we have described above we are able to extend the sum-out procedure in the `Variable Elimination` algorithm to `sum-out1`. We see in Algorithm 3 a high level description of `sum-out1`. Algorithm 3 describes in pseudo-code how to compute $P(X|Y = Y_0)$ which is called `EVEA`.

**Algorithm 2** Procedure `sum-out1`$(\mathcal{F}_1, \mathcal{F}_2, z)$

```
Input:
    • $\mathcal{F}_1$: A list of homogeneous factors,
    • $\mathcal{F}_2$: A list of heterogeneous factors,
    • $z$ a variable
Output: A list of heterogeneous factors and a list of homogeneous factors.
  1. Remove from $\mathcal{F}_1$ all the factors that contain $z$, multiply them resulting in,
  say, $f$. If there are no such factors, set $f = nil$
  2. Remove from $\mathcal{F}_2$ all the factors that contain $z$, combine them by using $\otimes$
  resulting in, say, $g$. If there are no such factors, set $g = nil$.
  3.
  if $g = nil$ then,
      add the new (homogeneous) factor $\sum_z f$ to $\mathcal{F}_1$.
  else
      add the new (hetergeneous) factor $\sum_z fg$ to $\mathcal{F}_2$
  4. return ($\mathcal{F}_1$ ,$\mathcal{F}_2$)
```

## 3 Experiments and Results

In the experiment section two CPCS networks are used to test the performance of the algorithms mentioned above. The first CPCS network has 364 nodes, while the other one consists of 422 nodes and 867 arcs. In both networks each non-root variable is a convergent variable with base combination operator MAX. Besides `VEA` and `EVEA`, two other `VEA`-based approaches are also tested. The

**Algorithm 3** Procedure EVEA($\mathcal{F}_1$, $\mathcal{F}_1$, $X$, $Y$, $Y_0$, $\rho$)

```
Input:
  • $\mathcal{F}_1$: The list of homogeneous factors in the deputation BN
  • $\mathcal{F}_2$: The list of heterogeneous factors in the deputation BN
  • $X$: A list of query variables
  • $Y$: A list of observed variables
  • $Y_0$: The corresponding list of observed values
  • $\rho$: A legitimate elimination ordering, in which the deputy variable should
    be summed out before its corresponding new regular variable
Output: $h(X)/\sum_X h(X)$
  1. Set the observed variables in all factors to their observed values
  2.
  while $\rho$ is not empty do
      Remove the first variable $z$ from $\rho$
      $(\mathcal{F}_1, \mathcal{F}_2) =$ sum-out1($\mathcal{F}_1, \mathcal{F}_2, z$)
  3. $h =$ multiplication of all factors in $\mathcal{F}_1$ $\times$ combination(by $\otimes$) of all factors
  in $\mathcal{F}_2$
  4. return $h(X)/\sum_X h(X)$
```

first one is denoted by "PD", which is first applying parent-divorcing method invented by Olesen et al. to transform the BN then using VEA. The other one "TT" firstly transforms BN by temporal transformation by Heckerman and then uses VEA.

To simplify the comparison, the execution time is limited within 10 seconds and the computation resource is limited within 10 MB. The authors test different situations with 5, 10 and 15 observations, and 50 randomly generated queries are involved in the experiment. They found that CTP based approaches could not handle the two CPCS networks under the constraints above, while VEA-based approaches can.

The statistics are shown in Figure 6. In the charts, the curve "5ve1", for instance, displays the time statistics for EVEA on queries with five observations. Points on the X-axis represent CPU times in seconds. For any time point, the corresponding point on the Y-axis represents the number of five observation queries that were each answered within the time by EVEA. And obviously EVEA could answer all the queries within 10 second, much better than PD and TT methods, since in some cases they could not answer the queries.

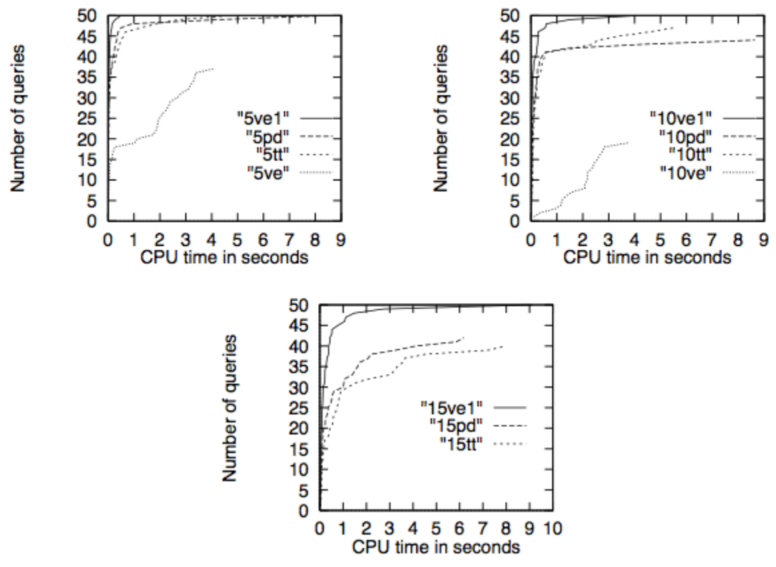

Figure 6: Comparisons in the 364-node BN.
[10]

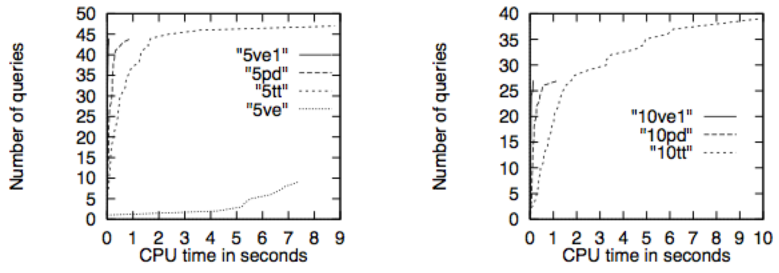

Figure 7: Comparisons in the 422-node BN.
[10]

In the experiment with 422-nodes network the same constraints are involved, and the results in shown in Figure 7. One can see that `EVEA` is much faster than the others, but not all the queries are answered, this is because of the memory limitation, if we change the memory limitation to 20 or 40 MB, then all the queries could be solved within several seconds. What's more, from the time consuming PD is better than TT, but in Figure 7 we find that TT could answer most of the queries, more than other methods, which is also due to the memory limitation. But still we can say that `EVEA` is better than other 3 `VEA`-based methods.

## 4 Conclusions

We will now conclude this paper. Cooper has proved in [1] that computation of probabilities in a BN is NP-hard. In order to perform this computation more efficient we could use specific algorithms on specific network structures. We have discussed VE and EVEA. VE uses factorisations of joint probability distributions in order to compute probabilities of a BN more efficient. EVEA extends VE by also using CI. We have seen that using EVEA allows us to compute the probabilities in a BN more efficient then the other algorithms with which it was compared.

## References

[1] G.F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, pages 393–405, 1990.

[2] Bruce D'Ambrosio. Local expression languages for probabilistic dependence. *International Journal of Approximate Reasoning*, 11(1):55–81, 1994.

[3] Bruce D'Ambrosio. Symbolic probabilistic inference in large bn20 networks. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 128–135. Morgan Kaufmann Publishers Inc., 1994.

[4] I. J. Good. A causal calculus (i). *The British Journal for the Philosophy of Science*, pages 305–318, 1961.

[5] David Heckerman. Causal independence for knowledge acquisition and inference. In *Proceedings of the Ninth international conference on Uncertainty in artificial intelligence*, pages 122–127. Morgan Kaufmann Publishers Inc., 1993.

[6] David Heckerman and John S Breese. A new look at causal independence. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 286–292. Morgan Kaufmann Publishers Inc., 1994.

[7] Jin Kim and Judea Pearl. A computational model for causal and diagnostic reasoning in inference systems. 1983.

[8] Kristian G Olesen, Uffe Kjaerulff, Frank Jensen, Finn V Jensen, Bjoern Falck, Steen Andreassen, and Stig K Andersen. A munin network for the median nerve-a case study on loops. *Applied Artificial Intelligence an International Journal*, 3(2-3):385–403, 1989.

[9] Nevin L Zhang and David Poole. A simple approach to bayesian network computations. In *Proc. of the Tenth Canadian Conference on Artificial Intelligence*, 1994.

[10] N.L. Zhang and D. Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, pages 301–328, 1996.