Structure Learning in Bayesian Networks with Parent Divorcing

Ulrich von Waldow (waldow@in.tum.de)
Technische Universität München, Arcisstr. 21
80333 Munich, Germany

Florian Röhrbein (florian.roehrbein@in.tum.de)
Technische Universität München, Arcisstr. 21
80333 Munich, Germany

Abstract

Bayesian networks (BNs) are an essential tool for the modeling of cognitive processes. They represent probabilistic knowledge in an intuitive way and allow to draw inferences based on current evidence and built-in hypotheses. In this paper, a structure learning scheme for BNs will be examined that is based on so-called Child-friendly Parent Divorcing (CfPD). This algorithm groups together nodes with similar properties by adding a new node to the existing network. The updating of all affected probabilities is formulated as an optimization problem. The resulting procedure reduces the size of the conditional probability tables (CPT) significantly and hence improves the efficiency of the network, making it suitable for larger networks typically encountered in cognitive modelling.

Keywords: Bayesian network; learning; inference; adding nodes; probability computation

Introduction

A BN is a directed acyclic graph (DAG) with a probability distribution $\mathcal{P}$, which is expressed as CPT on each node or on each variable, respectively. We assume binary nodes, thus a value can be $true$ or $false$. The CPTs have a size of 2^k, where k is the number of parents of a node. Let p be a node of the BN and let O be the set of parents of p. The aim is to reduce the size of the CPT of node p, by splitting the set of parent nodes O into smaller sets O_1, O_2 with $O_1 \cup O_2 = O$. Therefore, we introduce a new node, x, as a new parent of p and reset the connections of the network. The conclusion is a smaller CPT of node p. Of course there are several rules and net topologies, that need to be taken into account. In addition, the probabilities of node p change and new probabilities for x must be computed.

This paper describes the procedure of Child-friendly Parent Divorcing (Röhrbein, Eggert, & Körner, 2009) and the behavior as well as the variation of the considered BN. The findings are explained and illustrated in a small example. The effect of the reducing size of the total number of entries in the CPTs is shown in a simulation, where the CfPD is processed on five random BNs. Furthermore, a cognition scheme is shown, that models a BN as scenario for recognizing objects by providing evidence about their location and shape. Therefore, the nodes of the network are arranged in different levels.

Child-Friendly Parent Divorcing

Standard Parent Divorcing

The underlying technique for the CfPD learning algorithm is based on a design tool that is explained in (Olesen, Kjaerluff, Jensen, & Jensen, 1989). In the paper *Parent Divorcing* is defined as a technique to design more efficient Bayesian models. By introducing a new node as a divisive node, the number of incoming edges of a selected node is reduced. This influences the number of entries of the CPT of the selected node, therefore, the computational efficiency is increased. Figure 1 shows a simple example of standard Parent Divorcing. On the left side, node p has a CPT with size 2^m. After inserting node x, the node p in the resulting graph on the right-hand side has a CPT of size $2^{k+1} < 2^m$ (Neapolitan, 2003).

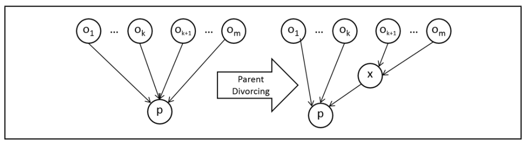


Figure 1: Simple Example of Parent Divorcing according to (Olesen et al., 1989). The left side represents the initial graph. Inserting x as an intermediate node splits the parent nodes into two sets of nodes.

Modification

In general the following statement holds for a BN: Let k be the number of parents of a node x, then the size of the CPT of x is 2^k. Moreover let n be the total number of nodes x_i with $i \in n$, then if every node x_i does not have more then k parents, the total network needs less then $n \cdot 2^k$ entries.

The Parent Divorcing mentioned above is now modified in such a way, that the resulting sets of parent nodes are not arbitrary, but similar in some way, or, that they have a feature in common respectively. Hence the aim is to find a type of similarity between those nodes that should be divorced into a set of groups (henceforth called the *to-be-divorced parents*). Here the structure, or rather the connectivity come into consideration, precisely the number of common child nodes.
Let $D = (\mathcal{V}, E)$ be a DAG with $\mathcal{V}$ as the set of nodes and E as the set of directed edges. The CfPD aims to find a set of nodes $O = \{o_1 ... o_m\}$ with $O \subseteq \mathcal{V}$ that have at least one child node $p_1 \in \mathcal{V}$ with $p_1 \notin O$ in common. These nodes should then be examined if there is another node $p_2 \in \mathcal{V}$ and $p_2 \notin O$ that has a set of parents Q with $Q \subset O$.

In summary: p_1 has the set of parents $O = \{o_1, ..., o_m\}$ and p_2 has the set of parents $Q \subset O$. Therefore O is the set of the to-be-divorced parents. Next, a node x is inserted in $\mathcal{V}$, which divides the to-be-divorced parents into two sets, which are $O_1 = \{o_1, ..., o_m\} \backslash Q$ and $O_2 = Q$. Furthermore, $Par(p_1) =$

$O_1 \cap x$ and $Par(p_2) = O_2$ hold. Figure 4 shows this example with $O_1 = \{o_1, o_2\}$ and $O_2 = \{o_3, o_4\}$.

Functionality and Rules

The section above provides an introduction and a small example of the CfPD. This section will explain the algorithm in detail. The algorithm is a combination of structure and parameter learning. In the following, let $D = (\mathcal{V}, E)$ be a DAG with $\mathcal{V}$ as the set of nodes or random variables, respectively with $|\mathcal{V}| = n$ and E as the set of directed edges with $|E| \leq \frac{n \cdot (n-1)}{2}$. In addition let $\mathcal{P}$ be a joint probability distribution of the variables in $\mathcal{V}$, then $B = (D, \mathcal{P})$ is a BN (Neapolitan, 2003). Summing up, the algorithm can be grouped into five main steps:

1. **Scanning** the network, to find out if CfPD could be executed.
2. **Selecting** a node that meets the expectations.
3. **Adding** a new node and updating the connections, which means adding and deleting some links.
4. **Computing** the parameters of the new node and the modified nodes.
5. **Handling** of already learned nodes.

These five steps are explained in detail in the following section.

```
DoCihldFriendlyParentDivorcing(bnet)
    //$n :=$ number of nodes of Bayesian network $bnet$
1:  if ($scanning(bnet) = true$){
2:     forall $i \in n${
3:        $p_1 := \mathbf{max}(Par(i))$
4:        if ($\forall j \in Child(Par(p_1)) \backslash p_1 : |Par(j) \cap Par(p_1)| < 2$){
5:           $i := i \backslash p_1 \rightarrow$ return to line 3.
6:        }
7:        elseif ($\forall j \in Child(Par(p_1)) : |\mathbf{max}(Par(j))| \geq 2$)
8:              $p_2 := rand(\mathbf{max}(Par(j))$
9:        }
10:       else {$p_2 := \mathbf{max}(Par(j))$}
11:    }
12:    $O := Par(p_1)$
13:    $O' := Par(p_2)$
14:    $O_2 := O \cap O'$
15:    $O_1 := O \backslash O_2$
16:    $delete(edges) : O_2 \rightarrow p_1$
17:    $insert(x) : O_2 \rightarrow x \wedge x \rightarrow p_1$
18:    $computing(P(x)) \wedge computing(P(p_1))$
19: }
20: else {Child-friendly Parent Divorcing is not possible}
```

Figure 2: Pseudocode of Child-firendly Parent Divorcing.

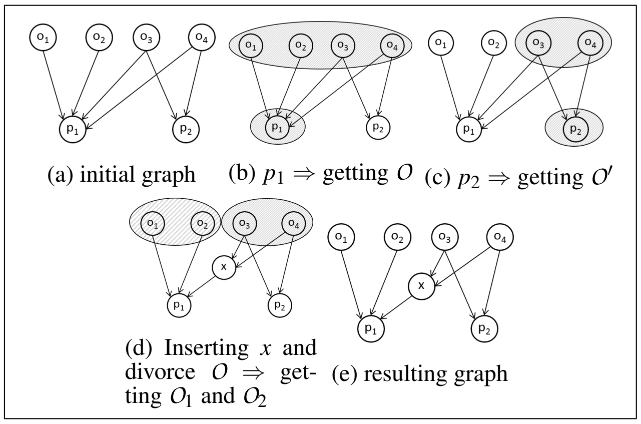
```
scanning(bnet)
1: forall ($i \in n$){
2:    if ($\exists i : Par(i) \geq 3 \wedge \exists j \in Child(Par(i)) \wedge j \neq i :$
              $Par(i) \cap Par(j) >= 2$){
3:       return $true$
4:    }
5:    else {return $false$}
6: }
```

Figure 3: Pseudocode of the scanning step.

Scanning: This step scans the graph D to check if it satisfies the expectations so that CfPD could be used. Figure 3 shows the pseudocode of this step. The following expectations must be met: (1) There must be a node $p_1 \in \mathcal{V}$ that has at least three parent nodes (line 2), otherwise a divorce will be inefficient. Assume $Par(p_1) = O \subseteq \mathcal{V}$ with $|O| = m$ and $m \geq 3$. (2) There must be a set of nodes $O' \subseteq O$ with $|O'| = k$ and $k \geq 2$ that have another child node $p_2 \in \mathcal{V}$ in common. (3) If p_1 is the only node that the to-be-divorced parents have in common, the algorithm should not be processed because the provided feature of having another child node in common is not satisfied. If one of these points is not satisfied, CfPD is not possible or should not be executed (for point 3).

Selecting If the scanning step returns true, there may be several nodes that meet the requirements. The pseudocode in figure 3 handles these conditions. In regards to step 1. of the scanning step, we are looking for a node whose number of parents are above a certain threshold $t \geq 3$. If there is more than one node $\{p_1, ..., p_k\}$ that comes into question because their number of parents is higher than or equal to the threshold (thus $Par(p_i) \geq t$ with $i \in 1, ..., k$) we should select the node with the maximum number of parents, hence $max\{|Par(p_i)|\}$ (line 3). This is because the node has the largest CPT, which we are trying to reduce. Probably there are two or more nodes with the same, maximum number of parent nodes, we will call this set $\mathcal{M}$, so $\forall p_m \in \mathcal{M} : |Par(p_m)| = \max(|Par(p_i)|)$. In that case the algorithm should choose that node which meets the requirements that the intersection of the parent nodes of another node $p_j \neq p_m$ and the parent nodes of p_m are maximized, hence $max\{|Par(p_m) \cap Par(p_j)|\}$, where $p_m \in \mathcal{M}$. Again there may be several nodes that fullfil these conditions. Then the algorithm should choose randomly (line 7). We also could go almost deeper in selecting the best fitting node, but there always may be more than one node with the same features. So we decided to choose randomly at that point.
After selecting the node that will be used, a second decision has to be made. Lets say we select p_1 as the node whose parents are the to-be-divorced parents (set O). Then we have to choose a second node $p_j \neq p_1$ with at least two parents, that are in the set of the to-be-divorced parents: $Par(p_j) \cap O \geq 2$. If there are none except p_1, we decide not to split the set O. Otherwise we choose the node p_j, such that $\forall p_j \neq p_1 : max(|Par(p_j)|)$. Thus at least two of the sets of the to-be-divorced parent nodes have a child, excluding p_1, in common. Of course it can occur that not only one node p_j meets these expectations. If this is the case, the algorithm should randomly select again.

Adding The third step is adding a divorcing node, which will be named x, and reordering the connections of the links. That means adding new directed edges to x and one outcoming edge from x. This will be executed in such a way that the to-be-divorced parents are split into two sets. Assume we chose p_1 and p_2 in the step above. Then $Par(p_1) = \{o_1, ..., o_m\} = O$ is the set of the to-be-divorced parents and $Par(p_2) = O'$. The intersection of O and O' identifies the splitting of the set O. We split the set O in two parts O_1 and O_2, such that the following statement holds: $O_1 = \{o_1,...,o_k\} \in O = O \backslash O \cap O'$ and $O_2 = \{o_{k+1},...,o_m\} = O \cap O'$. Figure 4 shows an example with $m = 4$.



(a) initial graph (b) $p_1 \Rightarrow$ getting O (c) $p_2 \Rightarrow$ getting O' (d) Inserting x and divorce $O \Rightarrow$ getting O_1 and O_2 (e) resulting graph

Figure 4: A simple example of Child-friendly Parent Divorcing

Computing By adding a new node to a BN, some changes must be made to the entries of the CPTs. The CPT of node x must be set up and the CPT of p_1 must be adjusted. If you look to the example in figure 4a, p_1 has $2^4 = 16$ entries in its CPT. Compared to 4e, where the number of entries in the CPT of p_1 is $2^3 = 8$, this is twice as much. However CfPD should not change any probabilities, more precisely it should not change the joint probability distribution $\mathcal{P}$ of B. Referring to the small example in figure 4, we compute the joint probability distribution of p_1 in the resulting graph. Let A be the initial graph and B be the resulting graph after adding a new node. Hence the equation

$$P_A(p_1, p_2, o_1, o_2, o_3, o_4) = P_B(p_1, p_2, \mathbf{x}, o_1, o_2, o_3, o_4) \quad (1)$$

must hold. We can now use the chain rule (Neapolitan, 2003) to compute the joint probability distribution:

$$\begin{aligned}
&P_A(p_1, p_2, o_1, o_2, o_3, o_4) = \\
&\quad = P(p_1|o_1, o_2, o_3, o_4) \cdot \\
&\quad\quad \cdot P(p_2|o_3, o_4) \cdot P(o_1) \cdot P(o_2) \cdot P(o_3) \cdot P(o_4) \\
&P_B(p_1, p_2, x, o_1, o_2, o_3, o_4) = \\
&\quad = \sum_x P(p_1|x, o_1, o_2) \cdot P(x|o_3, o_4) \cdot \\
&\quad\quad \cdot P(p_2|o_3, o_4) \cdot P(o_1) \cdot P(o_2) \cdot P(o_3) \cdot P(o_4)
\end{aligned}$$

Now inserting these two equations into (1) results in:

$$\begin{aligned}
P(p_1|o_1, o_2, o_3, o_4) &= \sum_x P(p_1|x, o_1, o_2) \cdot P(x|o_3, o_4) \\
&= P(p_1|x = T, o_1, o_2) \cdot P(x = T|o_3, o_4) + \\
&\quad + P(p_1|x = F, o_1, o_2) \cdot P(x = F|o_3, o_4)
\end{aligned}$$

This equation lead us to the following general formula:

$$\boxed{\begin{aligned}
&P(p_1|o_1, \ldots, o_m) = \\
&= P(p_1|x = T, o_1, \ldots, o_k) \cdot P(x = T|o_{k+1}, \ldots, o_m) + \\
&+ P(p_1|x = F, o_1, \ldots o_k) \cdot P(x = F|o_{k+1}, \ldots, o_m)
\end{aligned}} \quad (2)$$

Additionally, we can fix the conditional probability of p_1 with a little trick, by assuming a relationship. The motivation is, according to (Röhrbein et al., 2009), that the link between x and p_1 reflects a kind of taxonomic relationship. Variable x represents a subclass for p_1 and therefore the property $P(p_1 = T)$ is 1, if the value of the newly-inserted variable x is $true$ holds. Thus

$$P(p_1 = T|x = T, o_1, ..., o_k) = 1 \quad (3)$$

$$P(p_1 = F|x = T, o_1, ..., o_k) = 0 \quad (4)$$

Using the equation (3) in (2) leads to:

$$\boxed{\begin{aligned}
P(p_1|o_1, ..., o_m) = 1 - [1 - P(p_1|x = F, o_1, ..., o_k)] \cdot \\
\cdot [1 - P(x = T|o_{k+1}, ..., o_m)]
\end{aligned}} \quad (5)$$

The equations (2) and (5) will subsequently be implemented as an optimization problem to ensure that the probability of occurrence of node p_1 in the resulting graph is the same as in the initial graph.

Handling If new knowledge of a scenario or of a BN, respectively emerges, a new node and new links must be inserted. Thus it is essential for the algorithm to specify which node is a divorcing node and to set the connections appropriatly. An example is shown on the graph in figure 4. Assume there is new knowledge about p_1 and p_2 expressed in a variable o_{m+1}. In the initial case, the new node is inserted and links are set to p_1 and p_2. However, if the new knowledge emerges after divorcing the parent nodes (i.e. after inserting x), the algorithm must identify the new knowledge o_{m+1} as part of the to-be-divorced set and consequently set the links to x and p_2.

Scenarios

There are several net topologies that need to be examined. As a result there are different scenarios of reordering and inserting new nodes, depending on the net topology. Figure 5 shows four different types of scenarios (topologies).

a) In this case, it would not be efficient to introduce a new node, x, because the size of the set of the parents that can be divorced is 1 (i.e. o_{k+1} is the only parent node of the set of the to-be-divorced parents that has two children nodes in common). The insertion of a new node x does not reduce the size of the CPT of node p_1. Thus a divorce is redundant. The algorithm only divides a set with at least two considered nodes.

b) This case describes the classical case of CfPD. The nodes p_1 and p_2 have a set of parent nodes in common:

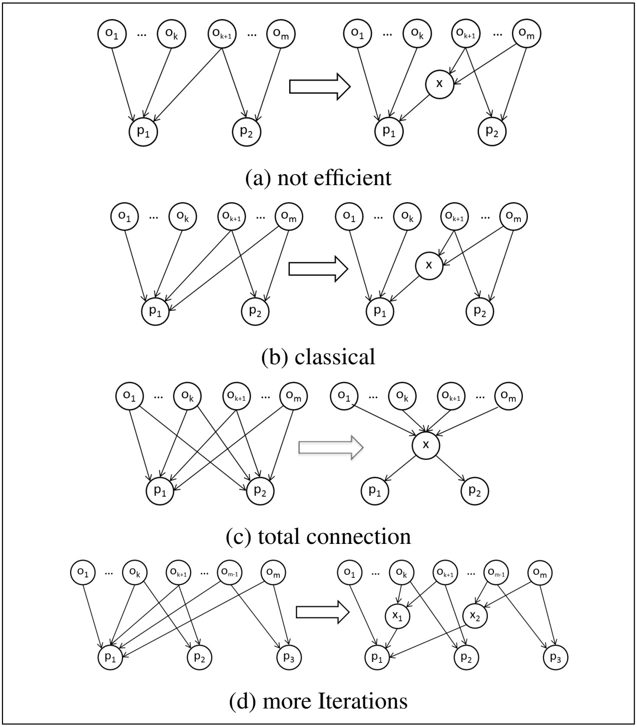


Figure 5: Different topologies, that can occur in a BN. The left side shows the intitial configuration, the right side shows the result after the *CfPD*

$o_{k+1},...o_m$. In this case the algorithm selects the node with the highest number of incoming edges, which is the node with the most parents. If the number is equal, it will choose randomly. The node x is inserted and divides the two sets of parent nodes: $o_1,...o_k$ and $o_{k+1},...o_m$.

c) Here we have a total connection. This means that every node $o_1,...o_m$ is connected to the nodes p_1 and p_2. The algorithm inserts a node, x, that divides each node o_i from another. In this case, the size of the CPT is not minimized, but the sum of entries is reduced. Initially the sum of all entries before is $m+2^m+2^m$ and afterwards we have a sum of $m+2^m+2$ afterwards.

d) This case shows a scenario where a node p_1 has a number of parent nodes in common with more than only one other node. In this case the algorithm can be processed two times and hence inserts more than one dividing node is inserted.

Value computation as an optimization problem

In this section, the computation of the probabilities of the changing and the newly-inserted nodes are formulated as an optimization problem. The aim is to minimize the error between the absolute probability of the occurrence of $P(p_1 = T|o_1,\dots,o_m)$ of the initial graph (P_{before}) and the absolute probability of the occurrence of $P(p_1 = T|x,o_1,\dots o_k)$ in the net resulting from the CfPD (P_{after}).

Initial values Let us define P_{before} first: Assume in the initial BN we have chosen p_1 in step 2 of the algorithm. Then m is the number of parents of p_1 and $Par(p_1) = \{o_1,\dots o_m\}$. P_{before} represents the absolute probability of the event, that $P(p_1 = T)$:

$$P_{before} = \sum_{o_1} ... \sum_{o_m} P(o_1)\cdot \ldots \cdot P(o_m)\cdot P(p_1 = T|o_1,...,o_m) \quad (6)$$

Now let us define P_{after}: Assume we introduced a new node x in the resulting BN. This node splits the original set of parents in $\{o_1,\dots o_k\}$ and $\{o_{k+1},\dots o_m\}$. P_{after} can then be calculated as follows:

$$\begin{aligned} P_{after} = \sum_{o_1}\dots\sum_{o_k}\sum_{x} P(o_1)\cdot\dots \\ \cdot P(o_k)\cdot P(x)\cdot P(p_1 = T|o_1,\dots o_k,x) \end{aligned} \quad (7)$$

The optimization problem Now we can formulate the optimization problem as a cost-minimizing function, where the cost is the difference between P_{before} and P_{after}. There are two sets of decision variables: $\sum_{o_1}\dots\sum_{o_k}\sum_{x} P(p1 = T|o_1,\dots,o_k,x)$ and $\sum_{o_{k+1}}\dots\sum_{o_m} P(x = T|o_{k+1},\dots o_m)$. The optimization problem for the first case, where no correlation between p_1 and x is assumed (see equation (2) above) is as follows:

$$\begin{aligned}
&1:\ \text{minimize } P_{before} - P_{after} \\
&\quad\ \text{subject to} \\
&2:\ \sum_{o_{k+1}}\dots\sum_{o_m}[P(x = F|o_{k+1},\dots,o_m) = \\
&\qquad\qquad = 1 - P(x = T|o_{k+1},\dots,o_m)] \\
&3:\ P_{before} - P_{after} \geq 0 \\
&4:\ \sum_{o_1}\dots\sum_{o_m}[P(p_1 = T|o_1,\dots,o_m) \leq \\
&\qquad \leq \sum_{x} P(p_1 = T|o_1,\dots,o_k,x)\cdot P(x|o_{k+1},\dots,o_m)] \\
&5:\ 0 \leq \sum_{o_{k+1}}\dots\sum_{o_m} P(x = T|o_{k+1},\dots,o_m) \leq 1 \\
&6:\ 0 \leq \sum_{o_1}\dots\sum_{o_k}\sum_{x} P(p1 = T|o_1,\dots,o_k,x) \leq 1
\end{aligned} \quad (8)$$

Taking into account that there is a correlation between p_1 and x we have to replace line 6 of the optimization problem above according to (3) by:

$$\begin{aligned}
&7:\ 0 \leq \sum_{o_1}\dots\sum_{o_k} P(p1 = T|o_1,\dots,o_k,x = F) \leq 1 \\
&8:\ \sum_{o_1}\dots\sum_{o_k} P(p1 = T|o_1,\dots,o_k,x = T) = 1
\end{aligned} \quad (9)$$

Line 1 represents the objective function. The aim is to minimize the difference between P_{before} and P_{after}, thus we have a minimization problem. Lines 2-8 represent the constraints that must be fulfilled. Line 2 assigns the values for the probability $P(x = F|o_{k+1},\dots,o_m)$ for all combinations of $\{o_{k+1},\dots,o_m\}$. In line 3 we assure that the new probability P_{before} is not larger than the probability P_{after}. In case of an error, i.e. $P_{before} \neq P_{after}$, the absolute probability of the occurrence of p_1 after the CfPD is less than before or in other words the occurrence is not overestimated. This constraint has been chosen because intutitively it seems more reliable to underestimate the occurrence of a variable than predicting an underestimation of a non-occurrence. Although, sometimes it seems more reliable to overestimate the occurrence of a variable at the time when e.g. a variable describes the failure of part of a system for example. In that case, the user has to differentiate and to determine whether to use P_{before} or rather P_{after} as minuend or subtrahend in line 3 and accordingly to use the $\leq$ for underestimation or the $\geq$ otherwise (see line 4). The fourth line assures that the probabilities, precisely the JPD of $P(p_1|o_1,\dots,o_k,x)$ is not larger than before. The last two lines only ensure the non-negativity of the decision variables, and that their values are not larger than 1. Assuming the correlation between x and p_1, the constraint for $P(p_1|o_1,\dots,o_k,x)$ is fixed by $P(p_1 = T|o_1,\dots,o_k,x = T) = 1$ according to (3). Hence line 6 of (8) is replaced by line 7 and 8 of (9). We took the BN $\mathcal{B} = (\mathcal{G},\mathcal{P})$ from figure 4a with its JPD $\mathcal{P}$ and $P_{before} = P(p_1|o_1,o_2) = 0.6202$ and performed the CfPD. This yields to the network $\mathcal{B}' = (\mathcal{G}',\mathcal{P}')$ in 4e. We calculated the JPT's of p_1 and x using the optimization approach on the one side and the EM-algorithm with 10 samples on the other. Now we can calculate the values for $P_{opt}(p_1|o_1,o_2)$ and $P^{10}_{EM}(p_1|o_1,o_2)$ as well as the JPD $\mathcal{P}'_{opt}$ of net $\mathcal{B}'$ for the optimization approach and the JPD $\mathcal{P}'^{10}_{EM}$ for the EM-algorithm respectively. Using the *Bayes Net Toolbox for Matlab* we receive the following results: $P_{opt}(p_1|o_1,o_2) = 0.6202$, $P^{10}_{EM}(p_1|o_1,o_2) = 0.7554$, $\mathcal{P}'_{opt} = 0.9915$ and $\mathcal{P}'^{10}_{EM} = 0.8620$. As we can see the optimization approach yields to a result, that do not change the probability of occurrence of p_1. Also the JPD only has a very little deviation. Whereas the result of the EM-algorithm with sample size 10 has a deviation of more then 13 % regarding the absolut probability of p_1. Also the JPD of the complete net varied of almost 14%. But we can obtain also good results for the EM-algorithm by incrementing the sample size. E.g. we receive $P^{50}_{EM}(p_1|o_1,o_2) = 0.6342$ and $\mathcal{P}'^{50}_{EM} = 0.9534$ for a database with 50 samples. Although the sample size could be increased almost further we will not get any better results for the JPD, then using the optimization approach.

Training Data

We created five different, randomly connected BNs with 30 nodes and 250 edges. Afterwards the CfPD algorithm was executed 10 times on these networks repeatedly. Table 1 shows the total number of CPT entries for the 10 iterations. Figure 6 represents the decreasing trend of the number of entries for each iteration. Already in the first iteration we can observe

iteration	$BNet_1$	$BNet_2$	$BNet_3$	$BNet_4$	$BNet_5$
0 (start)	339243	1138913	311404	829771	285098
1	210347	93409	181612	313803	156202
2	79405	78081	116844	184907	125514
3	63597	62273	85164	120139	92780
4	47597	54625	69836	88011	76972
5	31597	46451	53486	55755	60654

Table 1: Number of entries of the five random BNs. The first row (start) represents the total number of nodes before executing the algorithm.

a significant reduction of size. This is evident because the node with the highest possible number of parents and hence with the largest CPT is selected and reduced. For example the number of parents of the chosen node in $BNet_5$ is 19, hence this node has $2^{19} = 524288$ CPT entries. After executing CfPD on this net, the chosen node has 8 parents afterwards and the newly-inserted node has 12 parents. Hence the total number is reduced by $2^{19} - (2^8 + 2^{12}) = 431996$ entries. Figure 6 represents the trend of 10 iterations. For example

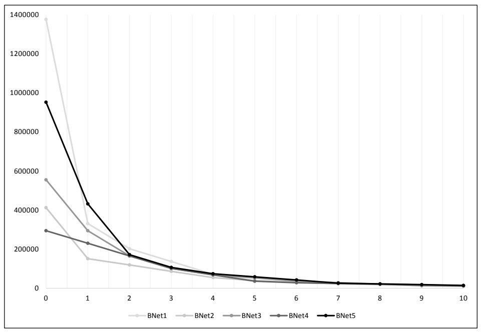


Figure 6: Processing the CfPD algorithm on 5 randomly-connected BNs with 30 nodes and 250 links.

$BNet_5$ has 40 nodes and only 14666 CPT entries in the resulting graph, which is 1.54% according to the initial graph. Taking $BNet_5$ as an example again, the algorithm can be processed 84 times on this network, which will lead to a network with 114 nodes and 760 (0.07%) CPT entries. Thus the size of reduction is decreasing by each iteration. The number of edges in a BN is restricted by $\frac{n\cdot(n-1)}{2}$. On the one hand, looking at a network with a high connectivity that has a higher number of edges provides the opportunity to execute CfPD several times and to lead to a high reduction of the number of CPT entries. On the other hand, if we consider a network that is less connected (hence it has only a few edges), we can only execute the algorithm a limited number of times and the reduction in the size of CPT entries will also be reduced. Figure 7 shows the difference of five iterations between five BNs with 30 nodes and 100 edges on the one side and 350 edges on the other. Therefore, the improvement regarding the total number of CPTs is very high at the beginning, but is flattened by raising the number of iterations. If we look at $Bnet_2$ from 6, the number of CPT entries after 14 iterations is 8255 compared to the start, when the network had 412167 entries, which is an improvement of over 98%. Therefore we have to weigh the improvement of reducing the number of entries in the CPTs on the one hand against adding a new node to the net on the other.

Cognition

We have shown that summing up nodes with a common feature by introducing a new node can improve the number of CPT entries significantly. We also pointed out that a high number of iterations will reduce the number of entries continuously, but the improvement decreases after each processing.

Now let us make a little extension to the learning scheme

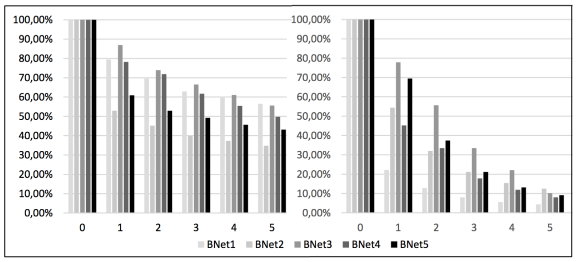
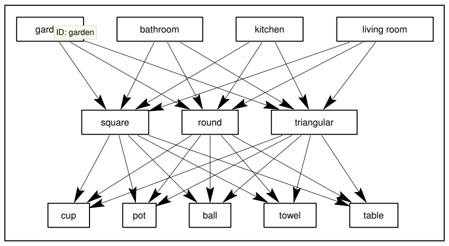
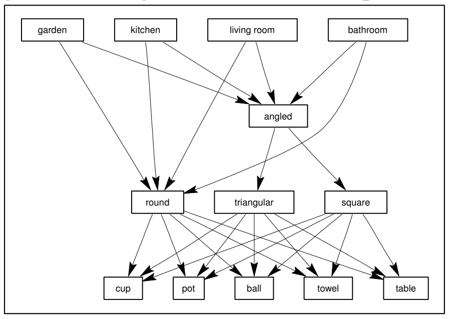


Figure 7: Decreasing trend of the number of CPTs after the first 5 iterations for two different random BNs. Left: BN with 30 nodes and 100 edges. Right: BN with 30 nodes and 350 edges

of CfPD by choosing an appropriate node or more precisely, a convenient level, manually. Assume we have a learning scheme that identifies *objects* by defining the actual *place* as well as the *shape* of the object. A simple example can be seen in figure 8. In this example, we have three *levels* of identifiers: place, shape and object see figure 8a. There are four example places in level one, three different shapes in level two and 5 example objects in level three. By obtaining knowledge about the place and the shape, the observer, also called agent, can identify a unique object with a suitable probability. CfPD may help us to improve these levels. Figure 8b is the resulting graph after executing CfPD on the second level, which means that the parents of the shape nodes, so level one, became divided. Assume the following scenario:



(a) A cognition scheme for identifying objects according their location and shape



(b) Graph after processing CfPD on the second level named *shapes*.

Figure 8: Example of a cognition scheme. 8a consists of a 3-level scheme with 12 nodes, 27 links and 92 CPT entries results in a cognition scheme with four levels, 13 nodes, 25 links and 80 CPT entries

The agent recognizes the situation, that he is in the kitchen and distinguishes a square-shaped object. Thus we have obtained evidence for the first and second level. Then the agent can assume that this object is a table with high probability and e.g. a cup with very low probability. CfPD can be used to improve a level of this network. If the level *shapes* includes forms such as *triangular, squared* and *round*, the algorithm can group together the first two by creating a new node and hence a new level called, for example *angled*.

Summary

By adding a new node to a BN and updating the links in the net, new probabilities must be computed and existing probabilities must be changed in order that the probability of occurrence of the observed node does not change. Therefore we formulated the problem as an optimization approach and minimized the costs between the probability of occurrence of the observed node before and afterwards. Other objectives, such as the Kullback-Leibler divergence might also be possible and can be used instead of the probability of occurrence. This will require some testing and might be a subject for future work. We tested the optimization problem on the classical approach applied in this paper, using *FICO XPress IVE 7.6* with random probabilities. The computed probabilities ensured the same probability of occurrence for the observed node at the initial network as well as for the node at the resulting net.

We showed that the size of the CPTs can be reduced in every step by repeating the algorithm to an existing BN. Furthermore the improvement during the first iterations is quite considerable and particularly in the case of highly-connected networks, a significant reduction in the total number of CPT entries can be achieved. By reducing the sizes of the CPTs the efficiency increases distinctly.

We provide an example how the CfPD algorithm can be used for cognition of objects. The algorithm is processed on a level of a BN instead of searching for a node with a large CPT. As a result the number of CPT entries is, of course, reduced and a new level is created that partitioned the chosen level more precisely.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Friedman, N. (n.d.). Learning belief networks in the presence of missing values and hidden variables..

Jensen, F. V. (2007). *Bayesian networks and decision graphs*. Springer.

Neapolitan, R. E. (2003). *Learning bayesian networks*. Prentice Hall.

Olesen, K. G., Kjaerluff, U., Jensen, F., & Jensen, F. V. (1989). A munin network for the median nerve - a case study on loops. *Applied Artificial Intelligence*.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*.

Röhrbein, F., Eggert, J., & Körner, E. (2009). Child-friendly divorcing: Incremental hierarchy learning in bayesian networks. In *IJCNN 2009*.