On Finding Relevant Variables in Discrete Bayesian Network Inference

Cory J. Butz  
butz@cs.uregina.ca  
University of Regina  
Canada

André E. dos Santos  
dossantos@cs.uregina.ca  
University of Regina  
Canada

Jhonatan S. Oliveira  
oliveira@cs.uregina.ca  
University of Regina  
Canada

Abstract
A central task in discrete Bayesian network (BN) inference is to determine those variables relevant to answer a given query. Two linear algorithms for this task explore the possibly relevant and active parts of a BN, respectively. We empirically compare these two methods along with a variation of each.

Introduction
Discrete Bayesian networks (BNs) (Pearl 1988) continue to be important today in a wide range of applications, including deep learning (Goodfellow, Bengio, and Courville 2016). Sum product networks (SPNs) (Poon and Domingos 2011) are a deep learning model for which impressive empirical results have been obtained in image completion, computer vision, classification, and speech recognition. Studies are investigating relationships between BNs and SPNs with interesting findings. Zhao, Melibari, and Poupart (2015) have shown that any complete and decomposable SPN is equivalent to a BN with size proportional to the size of the SPN. Moreover, Kazemi and Poole (2016) have investigated applying methods for detecting barren variables (Shachter 1986) in BNs to another deep learning model, called arithmetic circuits (Darwiche 2009). Thereby, BN advancements may aid both the BN and deep learning communities.

One fundamental task in BN inference is to determine the variables necessarily required to answer a query. These variables are called relevant. Consider a query \( P(X \mid Y) \), where \( X \) and \( Y \) are disjoint sets of variables in a BN \( B \), and let \( An(X \cup Y) \) be ancestors of \( X \cup Y \) in \( B \). (Geiger, Verma, and Pearl 1990) established that all variables not in \( X \cup Y \cup An(X \cup Y) \) are necessarily irrelevant to answer the query. In other words, the only variables that can possibly be relevant are restricted to those in \( X \), \( Y \), and \( An(X \cup Y) \). Consequently, we call the sub-DAG induced by these variables the possibly relevant part of the BN. Each variable within the possibly relevant part may or may not be relevant. (Geiger, Verma, and Pearl 1990) gave the first linear algorithm, which we will call GVP, to find relevant variables.

Bayes-Ball (Shachter 1998) is another linear algorithm often used for finding relevant variables. Bayes-Ball is a conceptually simple algorithm that works by considering how a hypothetical ball bounces in a BN according to specified rules. However, Bayes-Ball is a dual-purpose algorithm that can also concurrently test whether a given conditional independence (Pearl 1988) relation holds in a BN. More specifically, given disjoint sets \( X \) and \( Y \) in BN \( B \), Bayes-Ball can determine at the same time both the relevant variables for \( P(X \mid Y) \) and the variables conditionally independent of \( X \) given \( Y \). It does so by finding all variables that are reachable from \( X \) along active paths with respect to \( Y \). If, however, one is only interested in finding the relevant variables for \( P(X \mid Y) \), then Bayes-Ball can spend time exploring variables that are necessarily irrelevant to \( P(X \mid Y) \).

In this paper, we propose relevant active ancestors (RAA) as a novel method for determining the relevant variables for a query \( P(X \mid Y) \) posed to a discrete BN \( B \). All active paths are defined with respect to \( Y \). The active ancestors of a set \( W \) of variables, denoted \( A_W \), are the variables in \( W \) together with all variables in \( An(W) \) \( \setminus Y \) with an active, directed path to \( W \). The active ancestors of \( X \) are necessarily relevant. Unfortunately, the active ancestors of \( Y \) may contain irrelevant variables. Our solution is to first compute the active ancestors \( A_y \) for each \( y \in Y \) and then collect those \( A_y \) containing the missing relevant variables. We establish the soundness of RAA and show that it is a linear algorithm. As a variant of Bayes-Ball, we propose rp-relevant as another algorithm for finding relevant variables. In this case, rp-relevant is a slight modification of a faster method for testing independence in a BN (Butz, dos Santos, and Oliveira 2016). A thorough experimental analysis is performed between GVP, Bayes-Ball, RAA, and rp-relevant on 16 benchmark BNs using queries of 6 different sizes. RAA is almost always the fastest of the four.

Background
Let \( U = \{v_1, v_2, \ldots, v_n\} \) be a finite set of variables (nodes). Let \( B \) denote a directed acyclic graph (DAG) on \( U \). A directed path from \( v_i \) to \( v_k \) is a sequence \( v_1, v_2, \ldots, v_k \) with directed edges \( (v_i, v_{i+1}) \) in \( B \), \( i = 1, 2, \ldots, k - 1 \). For each \( v_i \in U \), the ancestors of \( v_i \), denoted \( An(v_i) \), are those variables having a directed path to \( v_i \), while the descendants of \( v_i \), denoted \( De(v_i) \), are those variables to which \( v_i \) has a directed path. For a set \( X \subseteq U \), we define \( An(X) \) in the obvious way. The children \( Ch(v_i) \) and parents \( Pa(v_i) \) of \( v_i \) are the sets \( v_j \) such that \( (v_i, v_j) \in B \) and \( (v_j, v_i) \in B \), respectively.
respectively. An undirected path in a DAG is a path ignoring directions. A singleton set \( \{v\} \) may be written as \( v \) and \( \{v_1, v_2, \ldots, v_n\} \) as \( v_1 v_2 \cdots v_n \). The cardinality of a set \( W \) is denoted \( |W| \).

A Bayesian network (BN) (Pearl 1988) is a DAG \( B \) on \( U \) together with conditional probability tables (CPTs) \( P(v_1|Pa(v_1)), P(v_2|Pa(v_2)), \ldots, P(v_n|Pa(v_n)) \). For example, Figure 1 (i) shows a BN, where CPTs \( P(a), P(b|a), \ldots, P(l|b) \) are not provided. We call a BN a, if no confusion arises. The product of the CPTs for \( B \) on \( U \) is a joint probability distribution \( P(U) \). Given pairwise disjoint \( X, Y, Z \subseteq U, \) the conditional independence (Pearl 1988) of \( X, Y, Z \) given \( Y \) holding in \( P(U) \) is denoted \( I(X,Y,Z) \). It is known that if \( I_{B}(X,Y,Z) \) holds in \( B \) by d-separation, then \( I(X,Y,Z) \) holds in \( P(U) \) (Pearl 1988).

\[
R = \{a, b, e, f, g, i\}, \tag{1}
\]

since only \( P(a), P(b|a), P(e|d), P(f|e), P(g|f), \) and \( P(i|a, e) \) are necessarily required to answer \( P(g|b, d, i) \).

(Geiger, Verma, and Pearl 1990) gave the first linear method, which we will call GVP, for finding relevant variables. There it is shown that all variables not in \( X \cup Y \cup \text{An}(X \cup Y) \) are necessarily irrelevant to answer \( P(X|Y) \). In other words, the only variables that can possibly be relevant to \( P(X|Y) \) are restricted to \( X \cup Y \cup \text{An}(X \cup Y) \). The call the sub-DAG of \( B \) restricted to these variables the possibly relevant part of \( B \). In our running example, the possibly relevant part of \( B \) is depicted in Figure 1 (ii). Given a query \( P(X|Y) \) posed to a BN, (Geiger, Verma, and Pearl 1990) finds relevant variables as follows. First, compute \( Y \cup \text{An}(Y) \). Next, find variables reachable from \( X \) along active paths with respect to \( Y \) in an iterative fashion by considering adjacent edges (to \( X \) initially) and adding those that are legal. (Geiger, Verma, and Pearl 1990) decide active paths in the BN by constructing a directed graph contain every edge of the BN in the forward and reverse direction, adding an auxiliary node \( s \) with an edge from \( s \) to every node in \( X \), and specifying a set \( F \) of illegal pairs of edges. The relevant variables are the reachable variables.

**Example 1.** Consider query \( P(g|b, d, i) \) posed to BN \( B \) in Figure 1 (i). The possibly relevant part is in Figure 1 (ii). A directed graph \( D' \) is then constructed with edges:

\[
\{(a, b), (b, c), (c, d), (d, e), (e, f), (f, g), (e, i), (a, i), \}

\{(b, a), (b, c), (d, e), (e, d), (f, e), (g, f), (i, e), (i, a)\}.
\]

Based upon the definition of active paths, the set \( F \) of illegal pairs of directed edges in \( D' \) includes

\[
[(a, b), (b, c)], [(c, d), (d, e)], \text{ and } [(a, i), (i, e)]. \tag{2}
\]

Mark nodes \( g \) and \( s \) as reachable and consider edge \( (s, g) \). There are 4 adjacent edges to evaluate, including \( (g, f) \). Since the pair of edges

\[
(s, g), (g, f) \tag{3}
\]

is legal (the pair is not in \( F \)), node \( f \) is marked as reachable. Now, in an iterative fashion, consider \( (g, f) \). The 3 adjacent edges to evaluate are \( (e, f), (f, e), \) and \( (f, g) \). As the pair

\[
(g, f), (f, e) \tag{4}
\]

is legal, node \( e \) is marked as reachable. The remainder of the example follows similarly, yielding reachable variables \( R = \{a, b, e, f, g, i\} \).
Finding Relevant using Active Ancestors

We begin by introducing the key notion of active ancestors. All active paths are defined with respect to $Y$ in $P(X|Y)$.

**Definition 1.** The active ancestors of a set $W$ of variables in BN $B$, denoted $A_W$, are the variables in $W$ together with all variables in $An(W) \setminus Y$ with an active, directed path to $W$.

The set $Y$ is not explicitly stated in the notation for $A_W$, since all active ancestor sets are defined with respect to $Y$.

**Example 3.** Given query $P(g|b,d,i)$ posed to $B$ in Figure 1 (i), $X = \{g\}$ and $Y = \{b,d,i\}$. Let us compute the active ancestors $A_X$. Variable $e$ is an active ancestor of $g$, since $e \in An(g) \setminus Y$ and there is an active, directed path $(a,b),(b,c),(c,d),(d,e),(e,f),(f,g)$ blocked by $b$ (or $d$). Moreover, the path $(a,i),(e,i),(e,f),(f,g)$ is active, but is not directed. Variable $g$ is in $A_X$, since $g \in X$. The set of all active ancestors of $X$ in $B$ is:

$$A_X = \{e,f,g\}.$$ (5)

The variables in $A_X$ are relevant to answer query $P(X|Y)$, that is, $A_X \subseteq R$. For example, $A_X$ in (5) is indeed a subset of $R$ in (1).

What remains is to find any missing relevant variables, namely, those in $R - A_X$. In our running example, $R - A_X = \{a,b,i\}$. To do this, we first compute the active ancestors of each variable $y \in Y$, denoted $A_y$. Second, we determine $R - A_X$ by finding all variables reachable along active paths from $X$, where both nodes of each edge in the path are common to at least one set of active ancestors $A_X$ for $X$ or $A_y$ for $y \in Y$.

**Example 4.** Let us compute the active ancestors for each variable of $Y = \{b,d,i\}$ in our running example. First, for $b \in Y$, consider computing $A_b$. Since $(a,b)$ is the only active, directed path to $b$, the active ancestors of $b \in Y$ are:

$$A_b = \{a,b\}.$$ (6)

Next, the active ancestors of $d \in Y$ are:

$$A_d = \{c,d\}.$$ (7)

Finally, the active ancestors of $i \in Y$ are:

$$A_i = \{a,e,i\}.$$ (8)

The active ancestors $A_b$, $A_d$, and $A_i$ in Example 4 and the active ancestors $A_X$ in Example 3 are depicted in Figure 2.

We now present the relevant active ancestors (RAA) algorithm. RAA computes the relevant variables for answering a query $P(X|Y)$ posed to a discrete BN $B$. RAA first computes the active ancestors $A_X$ for $X$. RAA then computes the active ancestors $A_y$ for each $y \in Y$. In order to avoid computing the active ancestors of a variable twice, a variable is marked as visited when it is first encountered. Moreover, when visiting $v$ for the first time, if a parent $v_i$ of $v$ will be scheduled to be visited, the edge $(v_i,v)$ in BN $B$ is marked. Marked edges will be used to compute $R$. The RAA algorithm is now given formally as Algorithm 2.
Algorithm 2 Relevant Active Ancestors (RAA) returns the relevant variables to answer query $P(X|Y)$ posed to BN $\mathcal{B}$.

1: procedure RAA($P(X|Y)$, $\mathcal{B}$) 
2: $A_{\mathcal{A}}(X, Y, \mathcal{B})$ \triangleright Compute $A_X$
3: for each $y \in Y$ do
4: $A_{\mathcal{A}}(y, Y, \mathcal{B})$ \triangleright Compute $A_y$
5: $R \leftarrow \{v \mid v$ is reachable along an active path from $X$ with respect to $Y$ in $\mathcal{B}$ using only marked edges}$
6: return $R$

Example 5. Given query $P(g|b, d, i)$ posed to the BN $\mathcal{B}$ in Figure 1 (i), let us run RAA to find the relevant variables. Here, $X = \{g\}$ and $Y = \{b, d, i\}$. Line 2 calls Algorithm 3 to compute $A_X$. In Algorithm 3, line 2 initializes $L = \{g\}$ as the variables to visit. Node $g$ is removed from $L$ in line 4, and since this is the first time $g$ is encountered, $g$ is marked as visited in line 6. Consider the parent $f$ of $g$ in line 7. Since $f \notin Y$ in line 8, the set of nodes to visit is updated as $L = \{f\}$ on line 9. The edge $(f, g)$ is marked in $\mathcal{B}$ on line 10. When considering node $f$ on the next iteration of the while loop in line 3, node $f$ is marked as visited, $L = \{e\}$, and edge $(e, f)$ is marked in $\mathcal{B}$. Consider node $e$ on the next iteration of the while loop. Node $e$ is marked as visited. When considering parent $d$ of $e$ in line 7, since $d \in Y$, it is not added as a node to visit, nor is edge $(d, e)$ marked in $\mathcal{B}$. As $L = \emptyset$, control returns to Algorithm 2. Thus, line 2 of RAA computed $A_X = \{e, f, g\}$, variables $e, f,$ and $g$ were marked as visited in $\mathcal{B}$, and edges $(e, f)$ and $(f, g)$ were marked in $\mathcal{B}$.

Next, consider the for loop in line 3 of RAA, which will compute $A_b = \{a, b\}$, $A_d = \{c, d\}$, and $A_i = \{a, e, i\}$. In the call to Algorithm 3 in line 4 when considering $b \in Y$, nodes $a$ and $b$ are marked in $\mathcal{B}$, and edge $(a, b)$ is marked in $\mathcal{B}$, too. Similarly, the call to Algorithm 3 for $d \in Y$ results in nodes $c$ and $d$, as well as edge $(c, d)$, being marked in $\mathcal{B}$. Finally, node $i$ and edges $(a, i)$ and $(e, i)$ are marked in $\mathcal{B}$ during the call to Algorithm 3 for $i \in Y$.

Now consider line 5 in RAA. The marked edges in $\mathcal{B}$ are $(a, b), (a, i), (c, d), (e, f), (e, i), (f, g)$.

Those variables reachable along active paths from $X = \{g\}$ with respect to $Y = \{b, d, i\}$ using only marked edges are $R = \{a, b, e, f, g, i\}$.

Algorithm 3 Active Ancestors (AA) computes the active ancestors of $W$ with respect to $Y$ in a BN $\mathcal{B}$.

1: procedure AA($W, Y, \mathcal{B}$) \triangleright Variables to visit
2: $L \leftarrow W$
3: while $L \neq \emptyset$ do
4: Remove $v$ from $L$
5: if $v$ is unmarked then \triangleright If not visited before
6: Mark $v$ as visited
7: for each $v_i \in Pa(v)$ do \triangleright For each parent
8: if $v_i \notin Y$ then \triangleright that is not in $Y$
9: $L \leftarrow L \cup \{v_i\}$ \triangleright visit $v_i$
10: Mark edge $(v_i, v)$ in $\mathcal{B}$

RAA is sound and has linear complexity.

Lemma 1. Algorithm 3 correctly computes the active ancestors $A_W$ of $W$ in $\mathcal{B}$ with respect to $Y$.

Theorem 1. The RAA algorithm correctly computes the relevant variables for query $P(X|Y)$ posed to a BN $\mathcal{B}$.

Given $P(g|b, d, i)$ posed to $\mathcal{B}$ in Figure 1 (i), RAA($P(g|b, d, i), \mathcal{B}$) returns $R = \{a, b, e, f, g, i\}$. These are precisely the relevant variables in (1).

The size of a DAG $\mathcal{B}$ is defined in terms of the number of nodes and the number of edges (Koller and Friedman 2009).

Theorem 2. RAA is linear in the size of DAG $\mathcal{B}$.

Experimental Results

We report on an empirical comparison of GVP, Bayes-Ball, RAA, and rp-relevant. All methods were implemented in Python using the NetworkX library (see networkx.github.io) and conducted on a 2.9 GHz Inter Core i7 with 8 GB RAM. The experiments reported in Figure 3 were carried out on the 16 BNs in Table 1. The second column of Table 1 reports the number of nodes of each BN, while the third column shows the percentage of leaves (nodes without children). For each BN, 1000 queries $P(X|Y)$ were randomly generated with each of $X$ and $Y$ being set to 1%, 5%, 10%, 15%, 20%, and 25% of $|U|$, yielding 6000 queries in total.

Figure 3 shows the average logarithmic time in seconds for each percentage group, where higher is faster. It can be seen that RAA was faster than Bayes-Ball and GVP in 91 out of 96 cases. RAA was faster than rp-relevant in all cases. RAA's performance is inversely related to query size.

Table 2 shows the average time savings as percentage of RAA over Bayes-Ball, GVP, and rp-relevant for each group.
The average time gains of RAA over Bayes-Ball ranged from 78% to 28%. Note that Bayes-Ball is almost always faster than RAA in the Andes BN. Andes is perhaps conducive for Bayes-Ball and hinders RAA because its topological structure has very few leaf variables relative to its size (11%). Fewer leaf variables may suggest both a meager number of barren variables and an abundant number of independent by evidence (Madsen and Jensen 1999) variables. This is favourable to Bayes-Ball, which explores barren but not independent by evidence, and unfavourable to RAA, which explores the converse.

The average time gains of RAA over GVP ranged from 73% to 46%. GVP will necessarily visit the nodes in the active part of $An(Y)$ twice, whereas RAA will visit these nodes only once. Moreover, GVP performs graphical manipulation, whereas RAA only performs graph traversal.
Table 1: Properties of the 16 BNs used in our experiments.

<table>
<thead>
<tr>
<th>BN</th>
<th>#Nodes</th>
<th>%Leaves</th>
</tr>
</thead>
<tbody>
<tr>
<td>child</td>
<td>20</td>
<td>35%</td>
</tr>
<tr>
<td>insurance</td>
<td>27</td>
<td>22%</td>
</tr>
<tr>
<td>water</td>
<td>32</td>
<td>25%</td>
</tr>
<tr>
<td>alarm</td>
<td>37</td>
<td>30%</td>
</tr>
<tr>
<td>hailfinder</td>
<td>56</td>
<td>23%</td>
</tr>
<tr>
<td>hepar2</td>
<td>70</td>
<td>59%</td>
</tr>
<tr>
<td>win95pts</td>
<td>76</td>
<td>21%</td>
</tr>
<tr>
<td>pathfinder</td>
<td>109</td>
<td>71%</td>
</tr>
<tr>
<td>munin1</td>
<td>186</td>
<td>17%</td>
</tr>
<tr>
<td>andes</td>
<td>223</td>
<td>11%</td>
</tr>
<tr>
<td>pigs</td>
<td>441</td>
<td>32%</td>
</tr>
<tr>
<td>link</td>
<td>724</td>
<td>18%</td>
</tr>
<tr>
<td>munin2</td>
<td>1003</td>
<td>18%</td>
</tr>
<tr>
<td>munin4</td>
<td>1038</td>
<td>17%</td>
</tr>
<tr>
<td>munin</td>
<td>1041</td>
<td>18%</td>
</tr>
<tr>
<td>munin3</td>
<td>1041</td>
<td>18%</td>
</tr>
</tbody>
</table>

Table 2: Average time savings as a percentage of RAA over Bayes-Ball, GVP, and rp-relevant for each group.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayes-Ball</td>
<td>78%</td>
<td>62%</td>
<td>48%</td>
<td>40%</td>
<td>34%</td>
<td>28%</td>
</tr>
<tr>
<td>GVP</td>
<td>73%</td>
<td>65%</td>
<td>57%</td>
<td>53%</td>
<td>50%</td>
<td>46%</td>
</tr>
<tr>
<td>rp-relevant</td>
<td>71%</td>
<td>71%</td>
<td>71%</td>
<td>73%</td>
<td>74%</td>
<td>75%</td>
</tr>
</tbody>
</table>

Lastly, we describe two possible applications of RAA in deep learning. Arithmetic circuits (ACs) (Darwiche 2003) are built from BNs and are used in deep learning (Poon and Domingos 2011). The computation process is linear and involves every node in the AC. (Kazemi and Poole 2016) attempt to exploit irrelevant variables in ACs when answering a query $P(Y | X)$ with $X \cup Y \subset U$. Irrelevant variables are either barren or independent by evidence (or both). Their approach, however, detects irrelevant variables in the given BN and then attempts to incorporate this information into the constructed AC. Let $W = X \cup Y \cup An(X \cup Y)$. All variables in $U - W$ are barren (Zhang and Poole 1994). Our key point is that the independent by evidence variables are $W - R$, where $R$ are the relevant variables returned by RAA.

Another application is (Zhao, Melibari, and Poupart 2015), where VE is applied on a BN to build a SPN. As with ACs, the SPN inference process involves all nodes. However, if SPNs are extended to answer queries $P(Y | X)$ with $X \cup Y \subset U$, then detecting relevant variables may be vital.

Conclusion

Determining relevant variables for a query posed to a BN is a central task in inference. In this paper, we have put forth Relevant Active Ancestors (RAA) as a new method for finding relevant variables. The correctness of RAA is stated in Theorem 1, while its linear complexity is in Theorem 2. RAA appears to be a very effective method in practice. The thorough empirical evaluation in Figure 3 shows that RAA tends to be faster than GVP (Geiger, Verma, and Pearl 1990) and Bayes-Ball (Shachter 1998), two linear approaches for finding relevant variables, by an average of 57% and 48%, respectively. Although linear, Bayes-Ball can consider variables that are necessarily irrelevant in its effort to be a dual-purpose algorithm.

References


