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 $P(X|Y)$, where X and Y are disjoint sets of variables in a BN $\mathcal{B}$, and let $An(X \cup Y)$ be ancestors of $X \cup Y$ in $\mathcal{B}$. (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 $\mathcal{B}$, Bayes-Ball can determine at the same time both the relevant variables for $P(X|Y)$ 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 $P(X|Y)$, then Bayes-Ball can spend time exploring variables that are necessarily irrelevant to $P(X|Y)$.

In this paper, we propose *relevant active ancestors* (RAA) as a novel method for determining the relevant variables for a query $P(X|Y)$ posed to a discrete BN $\mathcal{B}$. 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 $An(W) - Y$ with an active, directed path to W. A_X, 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 $\mathcal{B}$ denote a *directed acyclic graph* (DAG) on U. A *directed path* from v_1 to v_k is a sequence $v_1, v_2, \ldots, v_k$ with directed edges (v_i, v_{i+1}) in $\mathcal{B}$, $i = 1, 2, \ldots, k-1$. 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 those v_j such that $(v_i, v_j) \in \mathcal{B}$ and $(v_j, v_i) \in \mathcal{B}$,

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 $\mathcal{B}$ 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 $P(a), P(b|a)$, $\ldots, P(l|k)$ are not provided. We call $\mathcal{B}$ a BN, if no confusion arises. The product of the CPTs for $\mathcal{B}$ on U is a joint probability distribution $P(U)$. Given pairwise disjoint $X, Y, Z \subseteq U$, the *conditional independence* (Pearl 1988) of X and Z given Y holding in $P(U)$ is denoted $I(X, Y, Z)$. It is known that if $I_{\mathcal{B}}(X, Y, Z)$ holds in $\mathcal{B}$ by *d-separation*, then $I(X, Y, Z)$ holds in $P(U)$ (Pearl 1988).

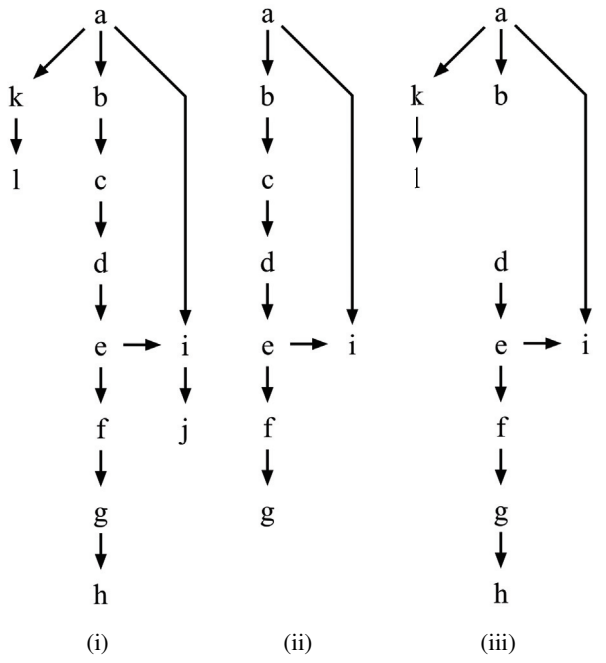


Figure 1: A BN $\mathcal{B}$ in (i). Given query $P(g|b, d, i)$, (ii) the *possibly relevant* part considered by GVP and RAA, and (iii) the *active* part considered by Bayes-Ball.

All methods discussed here for finding relevant variables are based upon the linear implementation of d-separation in (Geiger, Verma, and Pearl 1990). *d-Separation* (Pearl 1988) tests independencies in BNs and can be presented as follows (Darwiche 2009). Let X, Y, and Z be pairwise disjoint sets of variables in a DAG $\mathcal{B}$. We say X and Z are *d-separated* by Y, denoted $I_{\mathcal{B}}(X, Y, Z)$, if at least one node on every undirected path between X and Z is closed. There are three kinds of nodes v: (i) a *sequential* node means v is a parent of one of its neighbours and a child of the other; (ii) a *divergent* node is when v is a parent of both neighbours; and, (iii) a *convergent* node is when v is a child of both neighbours. A node v is either open or closed. A sequential or divergent node is *closed*, if $v \in Y$. A convergent node is *closed*, if $(v \cup De(v)) \cap Y = \emptyset$. A path with a closed node is *blocked*; otherwise, it is *active*.

Inference is a fundamental task in BNs. The task is to answer a given query $P(X|Y)$ posed to a BN $\mathcal{B}$, where X and Y are disjoint sets of variables in $\mathcal{B}$. More specifically, the problem is to identify those variables necessarily required to answer the query. These variables are called *relevant*, denoted R. Determining the relevant variables is useful as the query can be answered using only the CPTs of the relevant variables. As a running example, consider query $P(g|b, d, i)$ posed to the BN $\mathcal{B}$ in Figure 1 (i). The relevant variables are

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

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 $X \cup Y \cup An(X \cup Y)$ 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 $X \cup Y \cup An(X \cup Y)$. We call the sub-DAG of $\mathcal{B}$ restricted to these variables the *possibly relevant* part of $\mathcal{B}$. In our running example, the possibly relevant part of $\mathcal{B}$ 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 $Y \cup An(Y)$. 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 $\mathcal{B}$ in Figure 1 (i). The possibly relevant part is in Figure 1 (ii). A directed graph $\mathcal{D}'$ is then constructed with edges:*

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

Based upon the definition of active paths, the set F of illegal pairs of directed edges in $\mathcal{D}'$ includes

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

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 $R = \{a, b, e, f, g, i\}$.

Notice that variable i is necessarily a convergent variable in $\mathcal{B}$, but GVP treats it as sequential in (2). Similarly, variable a is necessarily divergent, by definition, but GVP treats it as sequential when considering $(b, a), (a, i)$.

Bayes-Ball (Shachter 1998) is another linear algorithm that concurrently solves two problems related to two disjoint sets X and Y in a BN $\mathcal{B}$. First, it can determine the relevant variables needed to answer $P(X|Y)$. Second, it can decide whether a set Z is conditionally independent of X given Y. Bayes-Ball solves these two problems using three main variables: V denotes the variables that have been visited; T denotes the variables marked on top; and B denotes the variables marked on bottom. The relevant variables are T. The independence $I(X, Y, Z)$ holds in $\mathcal{B}$, if $Z \subseteq (U - B)$. Bayes-Ball is given as Algorithm 1 (Shachter 1998).

Algorithm 1 Bayes-Ball gives the relevant variables T for query $P(X|Y)$ and B for the independence $I(X, Y, U-B)$.

```
1:  procedure BAYES-BALL(X, Y, B)
2:      Create a schedule of nodes to be visited, initialized
        with each node in X to be visited as if from a child.
3:      while there are still nodes scheduled to be visited do
4:          Remove any node v from the schedule.
5:          Mark v as visited.                    ▷ Update V
6:          if the visit to v is from a child and v ∉ Y then
7:              if v's top is not marked then
8:                  mark its top                  ▷ Update T
9:                  schedule its parents to be visited.
10:             if v's bottom is not marked then
11:                 mark its bottom               ▷ Update B
12:                 schedule its children to be visited.
13:         if the visit to v is from a parent then
14:             if v ∈ Y and v's top is not marked then
15:                 mark its top                  ▷ Update T
16:                 schedule its parents to be visited.
17:             if v ∉ Y and v's bottom is not marked then
18:                 mark its bottom               ▷ Update B
19:                 schedule its children to be visited.
20:     return T,B
```

Example 2. *Given $X = \{g\}$ and $Y = \{b, d, i\}$, let us run Bayes-Ball in the BN $\mathcal{B}$ of Figure 1 (i). The reader can verify that upon termination: the visited variables are $V = \{a, b, d, e, f, g, h, i, k, l\}$; the variables marked on top are $T = \{a, b, e, f, g, i\}$; and the variables marked on bottom are $B = \{a, e, f, g, h, k, l\}$. The variables marked on top T are precisely the relevant variables in (1). Moreover, the conditional independence*

$$I(X, Y, U - B) \equiv I(g, bdi, bcdij) \equiv I(g, bdi, cj)$$

holds in $\mathcal{B}$ meaning that the set of variables that are conditionally independent of g given $\{b, d, i\}$ is $\{c, j\}$.

Example 2 highlights the elegance of Bayes-Ball. In fact, Bayes-Ball and GVP can run on BNs with *functional nodes*, as well as on *influence diagrams* (Shachter 1998). These properties are outside of the scope of this study and are omitted.

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 $\mathcal{B}$, denoted A_W, are the variables in W together with all variables in $An(W) - Y$ 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 $\mathcal{B}$ 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 $e \in An(g) - Y$ and there is an active, directed path $(e, f), (f, g)$ from e to g. Node a is an ancestor of g, but not an active ancestor of g, since the directed path $(a, b), (b, c), (c, d), (d, e), (e, f), (f, g)$ is 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 $\mathcal{B}$ is:*

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

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*

$$A_b = \{a, b\}. \tag{6}$$

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

$$A_d = \{c, d\}. \tag{7}$$

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

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

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 $\mathcal{B}$. 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 $\mathcal{B}$ is marked. Marked edges will be used to compute R. The RAA algorithm is now given formally as Algorithm 2.

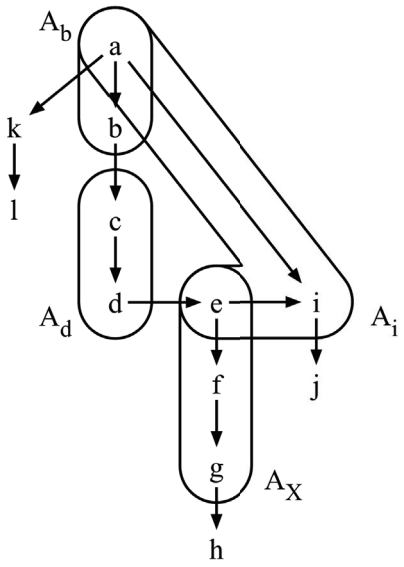


Figure 2: Active ancestors A_X, A_b, A_d, and A_i in Examples 3 and 4, given $P(g|b,d,i)$ posed to $\mathcal{B}$ in Figure 1 (i).

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: AA($X, Y, \mathcal{B}$) ▷ Compute A_X
3: **for** each $y \in Y$ **do**
4: AA($y, Y, \mathcal{B}$) ▷ 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

$$(a,b),(a,i),(c,d),(e,f),(e,i),(f,g). \tag{9}$$

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

$$R = \{a,b,e,f,g,i\}. \tag{10}$$

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

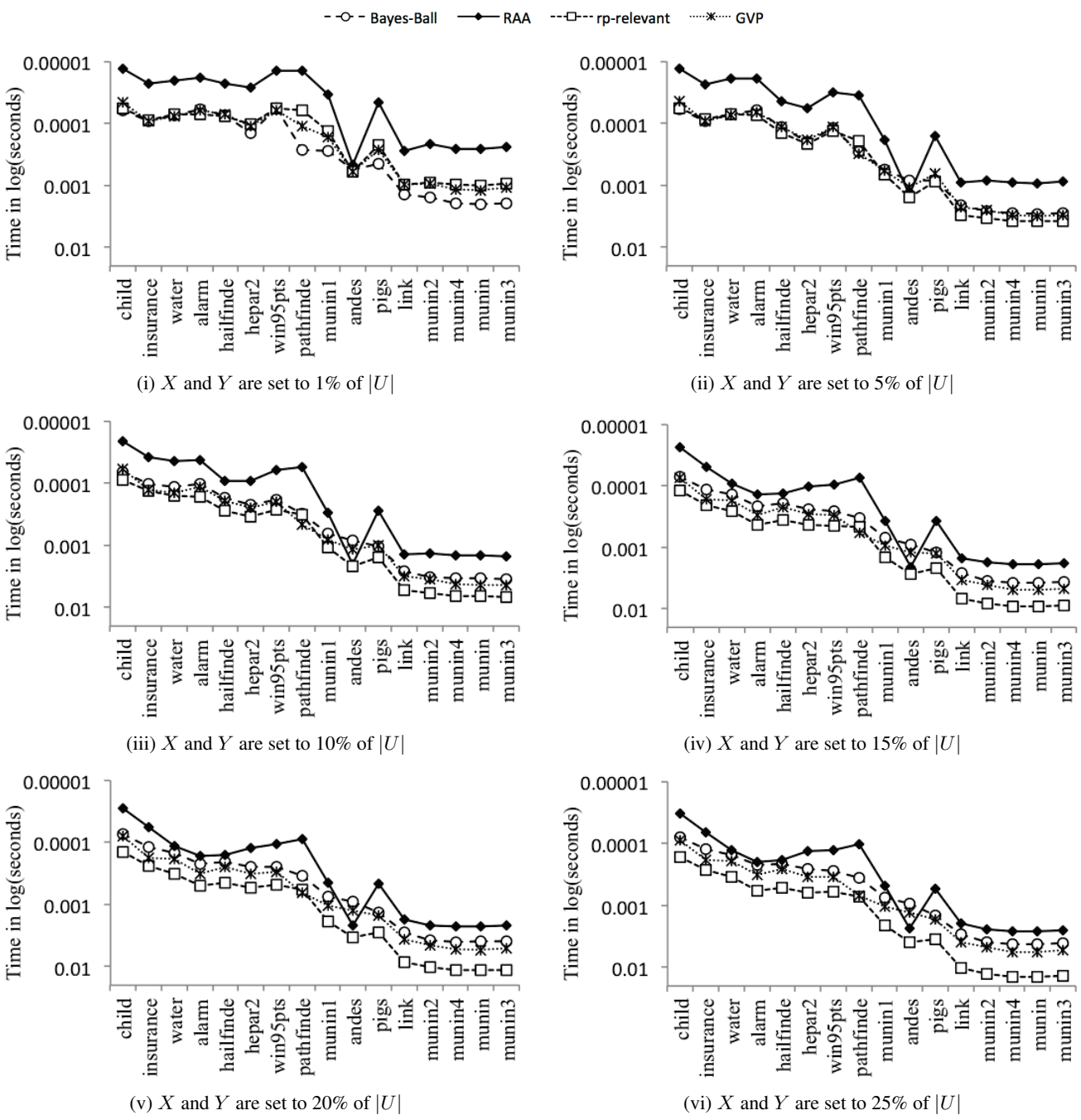
Theorem 2. *RAA is linear in the size of DAG $\mathcal{B}$.*

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.



(i) X and Y are set to 1% of $|U|$

(ii) X and Y are set to 5% of $|U|$

(iii) X and Y are set to 10% of $|U|$

(iv) X and Y are set to 15% of $|U|$

(v) X and Y are set to 20% of $|U|$

(vi) X and Y are set to 25% of $|U|$

Figure 3: The average time to answer 1000 random queries $P(X|Y)$ in each BN.

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.

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

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

Algorithm	1%	5%	10%	15%	20%	25%
Bayes-Ball	78%	62%	48%	40%	34%	28%
GVP	73%	65%	57%	53%	50%	46%
rp-relevant	71%	71%	71%	73%	74%	75%

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 $P(X|Y)$ 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 $P(X|Y)$ 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

Butz, C. J.; dos Santos, A. E.; and Oliveira, J. S. 2016. Relevant path separation: a faster method for testing independencies in Bayesian networks. In *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, 74–85.

Darwiche, A. 2003. A differential approach to inference in Bayesian networks. *Journal of the ACM* 50(3):280–305.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Geiger, D.; Verma, T.; and Pearl, J. 1990. Identifying independence in bayesian networks. *Networks* 20(5):507–534.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press.

Kazemi, S. M., and Poole, D. 2016. Lazy arithmetic circuits. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Madsen, A. L., and Jensen, F. V. 1999. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence* 113(1-2):203–245.

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

Poon, H., and Domingos, P. 2011. Sum-Product Networks: A New Deep Architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 337–346.

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

Shachter, R. D. 1998. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 480–487.

Zhang, N. L., and Poole, D. 1994. A simple approach to Bayesian network computations. In *Proceedings of the Tenth Canadian Artificial Intelligence Conference*, 171–178.

Zhao, H.; Melibari, M.; and Poupart, P. 2015. On the relationship between sum-product networks and Bayesian networks. In *Proceedings of Thirty-Second International Conference on Machine Learning*.