JMLR: Workshop and Conference Proceedings vol 52, 74-85, 2016 PGM 2016

Relevant Path Separation: A Faster Method for Testing Independencies in Bayesian Networks

Cory J. Butz BUTZ@CS.UREGINA.CA
André E. dos Santos DOSSANTOS@CS.UREGINA.CA
Jhonatan S. Oliveira OLIVEIRA@CS.UREGINA.CA
Department of Computer Science
University of Regina
Regina

Abstract

Directed separation (d-separation) played a fundamental role in the founding of *Bayesian networks* (BNs) and continues to be useful today in a wide range of applications. Given an independence to be tested, current implementations of d-separation explore the *active* part of a BN. On the other hand, an overlooked property of d-separation implies that d-separation need only consider the *relevant* part of a BN. We propose a new method for testing independencies in BNs, called *relevant path separation* (rp-separation), which explores the intersection between the active and relevant parts of a BN. Favourable experimental results are reported.

Keywords: Bayesian networks; d-separation; m-separation; testing independencies.

1. Introduction

Directed separation (d-separation) (Pearl, 1988) continues to be useful in a wide range of areas, including causal inference in statistics (Pearl, 2016), cause and correlation in biology (Shipley, 2016), extrapolation across populations (Pearl and Bareinboim, 2014), handling missing data (Mohan and Pearl, 2014), bioinformatics (Neapolitan, 2009), and deep learning (Goodfellow et al., 2016). The d-separation algorithm is a graphical method for determining which *conditional independence* relations are implied by the *directed acyclic graph* (DAG) of a *Bayesian network* (BN) (Pearl, 1988). With respect to a given independence to be tested, current implementations, including Bayes-Ball (Shachter, 1998) and Reachable (Koller and Friedman, 2009), find all nodes reachable along active paths, called the *active* part of a BN. This approach overlooks a crucial property of d-separation.

Lauritzen et al. (1990) proposed *m-separation* as another method for testing independencies in BNs. In the proof of correctness, it is established that all active paths of interest can only appear in what we call the *relevant* part of a BN. This property warrants attention in itself, since it has both theoretical and practical ramifications.

In this paper, we propose *relevant path separation* (rp-separation) as a new method for testing independencies in BNs. The salient feature of rp-separation is that it explores the *intersection* between the active and relevant parts of a BN. We introduce the notion of a *relevant* path and establish that *irrelevant* paths are either active paths that are doomed to become blocked or active paths that terminate before reaching a variable of interest. Rather than exploring all active paths, rp-separation displays impressive performance in practice by only exploring active paths that are relevant. In real-world or benchmark BNs, rp-separation is faster in 17 of 19 cases with an average time savings of 53%, culminating with being nearly twice as fast in the largest BN.

To test whether two sets X and Z of variables are conditionally independent in BN $\mathcal{B}$ given a third set Y of variables, denoted $I(X, Y, Z)$, it is shown that all irrelevant paths include at least one variable not in $XYZ \cup An(XYZ)$, where $An(XYZ)$ are the ancestors of XYZ in $\mathcal{B}$. Thus, the variables considered by rp-separation exactly coincide with those considered by m-separation. However, m-separation builds a secondary structure (an undirected graph), a process that involves adding and deleting undirected edges. We explicitly demonstrate several ways in which this computation is redundant and present a refinement of m-separation. Our experimental results show that the refinement leads to a time savings of 27% on average, but even the refined m-separation is slower than d-separation, let alone rp-separation.

This paper is organized as follows. Section 2 gives background information. We propose rp-separation in Section 3. Section 4 includes an empirical evaluation and analysis. We refine m-separation in Section 5. Section 6 discusses related work. Conclusions are drawn in Section 7.

2. Background

Let $U = \{v_1, v_2, \ldots, v_n\}$ be a finite set of variables. 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. 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 directed edge $(v_i, v_j) \in \mathcal{B}$ may be written as (v_j, v_i) in an undirected path. A variable v_k is called a *v-structure* (Koller and Friedman, 2009) in a DAG $\mathcal{B}$, if $\mathcal{B}$ contains directed edges (v_i, v_k) and (v_j, v_k), but not a directed edge between variables v_i and v_j. A singleton set $\{v\}$ may be written as v, $\{v_1, v_2, \ldots, v_n\}$ as $v_1 v_2 \cdots v_n$, and $X \cup Y$ as XY.

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 $P(a), P(b), \ldots, P(j|i)$ 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* $P(U)$ (Pearl, 1988). The *conditional independence* (Pearl, 1988) of X and Z given Y holding in $P(U)$ is denoted $I_P(X, Y, Z)$, where X, Y, and Z are pairwise disjoint subset of U. It is known that if $I(X, Y, Z)$ holds in $\mathcal{B}$, then $I_P(X, Y, Z)$ holds in $P(U)$.

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 BN $\mathcal{B}$. We say X and Z are *d-separated* by Y, denoted $I(X, Y, Z)$, if at least one variable on every undirected path from (any variable in) X to (any variable in) Z is closed. On a path, there are three kinds of variable v: (i) a *sequential* variable means v is a parent of one of its neighbours and a child of the other; (ii) a *divergent* variable is when v is a parent of both neighbours; and (iii) a *convergent* variable is when v is a child of both neighbours. A variable v is either open or closed. A sequential or divergent variable is *closed*, if $v \in Y$. A convergent variable is *closed*, if $(v \cup De(v)) \cap Y = \emptyset$. A path with a closed variable is *blocked*; otherwise, it is *active*.

Example 1 *Let us test $I(a, e, g)$ in the BN $\mathcal{B}$ of Figure 1 (i) using d-separation. Here $X = \{a\}$, $Y = \{e\}$, and $Z = \{g\}$. The path (a, d), (d, b), (b, e), (e, g) from X to Z is blocked by closed convergent variable d, since $d \cup De(d) = \{d, f, g, h\}$ and $\{d, f, g, h\} \cap Y = \emptyset$. On the contrary,*

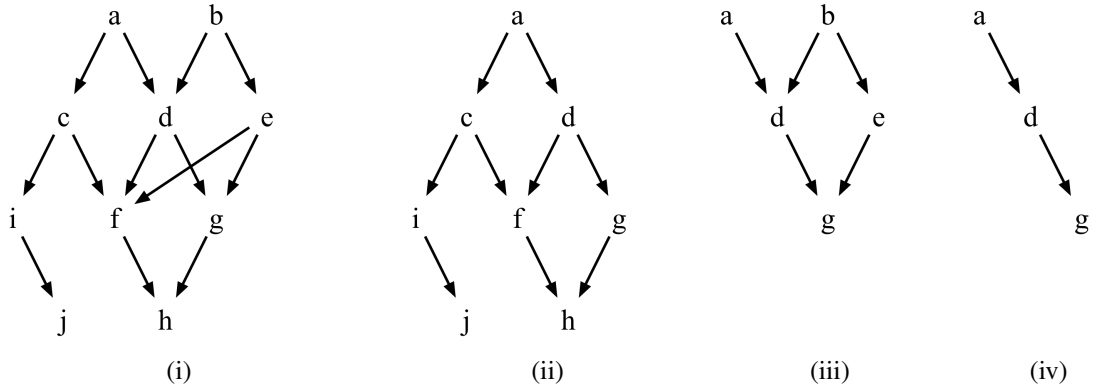

Figure 1: (i) testing independence $I(a, e, g)$ in a Bayesian network $\mathcal{B}$; (ii) the active part of $\mathcal{B}$; (iii) the relevant part of $\mathcal{B}$; (iv) the intersection of the active and relevant parts.

the path (a, d), (d, g) from X to Z is active, since d is an open sequential variable. As there exists an active path from a to g, $I(a, e, g)$ does not hold in $\mathcal{B}$ by d-separation.

Geiger et al. (1989) were the first to provide a linear time complexity algorithm for implementing d-separation. Their method, however, always explores the entire DAG. *Bayes-Ball* (BB) (Shachter, 1998) only explores the *active* part of the DAG. As will be discussed in Section 6, BB works using a hypothetical ball bouncing in a DAG and can be used for purposes outside the scope of our paper. Instead, we use the REACHABLE algorithm (Koller and Friedman, 2009), since it also explores the active part of a DAG, but with d-separation terminology. To test $I(X, Y, Z)$ in a BN $\mathcal{B}$, REACHABLE takes X, Y, and $\mathcal{B}$ as input and returns the set of all variables reachable from X along active paths. For pedagogical purposes, Example 2 will mention active paths explicitly.

Example 2 *Let us test $I(a, e, g)$ in the BN $\mathcal{B}$ of Figure 1 (i) using* REACHABLE*. Given $X = \{a\}$, $Y = \{e\}$, and $\mathcal{B}$, the set of variables reachable from X along active paths is denoted by R and initialized as $R = \{a\}$. As variable c is reachable from variable a along active path (a, c), the set of reachable variables is updated as $R = \{a, c\}$. Similarly, variable f can be reached along the active path $(a, c), (c, f)$. Subsequently, variable h can be reached along active path $(a, c), (c, f), (f, h)$ meaning R is updated as:*

$$R \quad = \quad \{a, c, f, h\}. \tag{1}$$

Variables i and j are reachable along active paths $(a, c), (c, i)$ and $(a, c), (c, i), (i, j)$, respectively. Hence, R is updated as:

$$R \quad = \quad \{a, c, f, h, i, j\}. \tag{2}$$

As depicted by the active part of $\mathcal{B}$ in Figure 1 (ii), it can be verified that the set of all variables reachable from X along active paths is $R = \{a, c, f, h, i, j, d, g\}$. Since at least one variable in Z was reachable from X along an active path, the independence $I(a, e, g)$ does not hold.

Example 2 and Figure 1 (ii) illustrate how REACHABLE finds all variables reachable along all active paths. Although linear, this approach can be wasteful.

3. Relevant Path Separation

Our purpose here is to give a faster method for testing independencies in BNs in practice. We begin by motivating our study.

m-Separation (Lauritzen et al., 1990) is another method for testing independencies in BNs. Let X, Y, and Z be pairwise disjoint sets of variables in a BN $\mathcal{B}$. m-Separation tests $I(X, Y, Z)$ with four steps: (i) prune $\mathcal{B}$ onto $XYZ \cup An(XYZ)$; (ii) construct the *moralization* (Lauritzen and Spiegelhalter, 1988) of the sub-DAG constructed in step (i) by adding an undirected edge between each pair of parents of a common child and then dropping directionality; (iii) delete Y and its incident edges from the undirected graph built in (ii); and (iv) if there exists a path from (any variable in) X to (any variable in) Z in (iii), then $I(X, Y, Z)$ does not hold; otherwise, $I(X, Y, Z)$ holds.

Theorem 1 shows m-separation is correct by establishing equivalence with d-separation.

Theorem 1 *(Lauritzen et al., 1990) Let X, Y, and Z be disjoint subsets of a DAG $\mathcal{B}$. Then, Y d-separates X from Z in $\mathcal{B}$ if and only if Y separates X from Z in the moralization of the sub-DAG of $\mathcal{B}$ onto $XYZ \cup An(XYZ)$.*

Proof ($\Leftarrow$) By contraposition, suppose Y does not d-separate X from Z. Then there exists an active path from X to Z. Every sequential variable on this path must be not in Y; otherwise, it would block the path. Similarly, every divergent variable on this path must be not in Y. On the contrary, every convergent variable on this path must either be in Y or have a descendent in Y; otherwise, the path is blocked. It follows that every variable on this active path is in $XYZ \cup An(XYZ)$. Hence, every variable in this path will appear in the moralized graph constructed onto $XYZ \cup An(XYZ)$. The moralization step will add an edge for each convergent variable on the active path, thus creating a path from X to Z not involving Y.

($\Rightarrow$) See (Lauritzen et al., 1990). ■

A key property of d-separation involving active paths is mentioned in the proof of Theorem 1.

Theorem 2 *Given $I(X, Y, Z)$ to be tested in a BN $\mathcal{B}$, all active paths between X and Z can only involve variables in $XYZ \cup An(XYZ)$.*

Proof Follows immediately from the proof of Theorem 1. ■

Let $I(X, Y, Z)$ be an independence to be tested in a BN $\mathcal{B}$. A path from (any variable in) X to (any variable in) Z is called *relevant*, if it only involves variables in $XYZ \cup An(XYZ)$; otherwise, it is called *irrelevant*. The *relevant part* of $\mathcal{B}$ is its sub-DAG onto $XYZ \cup An(XYZ)$.

Example 3 *Consider testing $I(a, e, g)$ in the BN $\mathcal{B}$ of Figure 1 (i). Then, $XYZ \cup An(XYZ) = \{a, e, g\} \cup \{a, b, d, e\} = \{a, b, d, e, g\}$. There are only two relevant paths. The path (a, d), (d, b), (b, e), (e, g) is relevant, since it only involves variables in $\{a, b, d, e, g\}$. Similarly, the path (a, d), (d, g) is relevant. On the contrary, the path (a, c), (c, f), (f, h), (h, g) is irrelevant, since it involves variables $c, f, h \notin \{a, b, d, e, g\}$. Moreover, the path (a, c), (c, i), (i, j) is irrelevant as $c, i, j \notin \{a, b, d, e, g\}$. The relevant part of $\mathcal{B}$ is shown in Figure 1 (iii).*

We now formally introduce a new method for testing independencies in BNs.

Definition 3 *Let X, Y, and Z be pairwise disjoint sets of variables in a BN $\mathcal{B}$. The independence test $I(X, Y, Z)$ holds if all relevant paths from (any variable in) X to (any variable in) Z are blocked; otherwise, $I(X, Y, Z)$ does not hold.*

We call the method *relevant path separation* (rp-separation) to emphasize that only the relevant part of a BN needs to be considered.

Example 4 *Let us test $I(a, e, g)$ in the BN $\mathcal{B}$ of Figure 1 (i) using rp-separation. Given $X = \{a\}$, $Y = \{e\}$, and $Z = \{g\}$, then*

$$XYZ \cup An(XYZ) \quad = \quad \{a, b, d, e, g\}. \tag{3}$$

There are only two relevant paths from X to Z to consider. The relevant path $(a, d), (d, g)$ is active, while the relevant path $(a, d), (d, b), (b, e), (e, g)$ is blocked as d is a closed convergent variable. Therefore, $I(a, e, g)$ does not hold in $\mathcal{B}$ by rp-separation.

We now show the correctness of rp-separation.

Theorem 4 *Independence $I(X, Y, Z)$ holds in a BN $\mathcal{B}$ by d-separation if and only if $I(X, Y, Z)$ holds in $\mathcal{B}$ by rp-separation.*

Proof Any path involving a variable not in $XYZ \cup An(XYZ)$ is ignored by rp-separation. Thereby, rp-separation can be seen as applying d-separation on $\mathcal{B}'$, where $\mathcal{B}'$ is the sub-DAG of $\mathcal{B}$ onto variables $XYZ \cup An(XYZ)$. However, testing $I(X, Y, Z)$ in $\mathcal{B}$ is equivalent to testing $I(X, Y, Z)$ in $\mathcal{B}'$ (Lauritzen et al., 1990). Therefore, the claim holds. ■

Although rp-separation only considers the relevant part of a BN, an efficient implementation, called RP-REACHABLE and given as Algorithm 1, traverses only the active paths within the relevant part. It is based upon the REACHABLE algorithm in (Koller and Friedman, 2009), which instead traverses the active part of a BN. More specifically, given $I(X, Y, Z)$ in a BN $\mathcal{B}$, Algorithm 1 takes X, Y, Z, and $\mathcal{B}$ as input and first marks those variables in $XYZ \cup An(XYZ)$. The algorithm keeps track of whether a variable v is visited from a child, denoted $(\uparrow, v)$, or visited from a parent, denoted $(\downarrow, v)$. In Algorithm 1, L is the set of variables to be visited, R is the set of reachable variables via active paths, and V is the set of variables that have been visited. Active paths from X are explored as long as they only involve variables in $XYZ \cup An(XYZ)$.

Example 5 *Let us test $I(a, e, g)$ in the BN $\mathcal{B}$ of Figure 1 (i) using* RP-REACHABLE*. The initialization sets $A = \{b, e\}$, $XYZ^{up} = \{a, b, d, e, g\}$, $L = \{(\uparrow, a)\}$, $V = \emptyset$, and $R = \emptyset$. The first iteration of the* while *loop in line 10 considers $(\uparrow, a)$. Next, $R = \{a\}$ and $V = \{(\uparrow, a)\}$. Line 21 considers children c and d of a. However, the check for relevant paths on line 22 results in $L = \{(\downarrow, d)\}$. That is, traversing from a to c is ignored because (a, c) is an irrelevant path. Similarly, when considering $(\downarrow, d)$ on the second iteration, line 21 considers children f and g of d. However, line 22 only adds $(\downarrow, g)$ and not $(\downarrow, f)$. The remainder follows similarly, yielding $R = \{a, d, g\}$.*

Example 5 shows that RP-REACHABLE only explores the intersection depicted in Figure 1 (iv).

Algorithm 1 Given an independence $I(X, Y, Z)$, RP-REACHABLE traverses all active paths from X within the relevant part of a BN $\mathcal{B}$.

```
1:  procedure RP-REACHABLE(X,Y,Z,B)
2:      ▷ Initialization
3:      A ← Y ∪ An(Y)
4:      XYZ^up ← XYZ ∪ An(XYZ)                          ▷ The relevant part of B
5:      for v ∈ X do                                    ▷ (Variable,direction) to be visited
6:          L ← L ∪ {(↑, v)}
7:      V ← ∅                                           ▷ (Variable,direction) marked as visited
8:      R ← ∅                                           ▷ Variables reachable via an active path
9:      ▷ Starting from X traverse relevant paths that are active
10:     while L ≠ ∅ do                                  ▷ While variables to be checked
11:         Select (d, v) in L
12:         L ← L − {(d, v)}
13:         if (d, v) ∉ V then                          ▷ If v has not been visited from direction d
14:             if v ∉ Y then
15:                 R ← R ∪ {v}                         ▷ v is reachable
16:             V ← V ∪ {(d, v)}                        ▷ Mark v as visited from direction d
17:             if d =↑ and v ∉ Y then
18:                 for v_i ∈ Pa(v) do                  ▷ v is open serial
19:                     if v_i ∈ XYZ^up then            ▷ Only explore relevant paths
20:                         L ← L ∪ {(↑, v_i)}
21:                 for v_i ∈ Ch(v) do                  ▷ v is open divergent
22:                     if v_i ∈ XYZ^up then            ▷ Only explore relevant paths
23:                         L ← L ∪ {(↓, v_i)}
24:             else if d =↓ then
25:                 if v ∉ Y then
26:                     for v_i ∈ Ch(v) do              ▷ v is open serial
27:                         if v_i ∈ XYZ^up then        ▷ Only explore relevant paths
28:                             L ← L ∪ {(↓, v_i)}
29:                 if v ∈ A then
30:                     for v_i ∈ Pa(v) do              ▷ v is open convergent
31:                         if v_i ∈ XYZ^up then        ▷ Only explore relevant paths
32:                             L ← L ∪ {(↑, v_i)}
33:     return R    ▷ All variables reachable from X via active paths within the relevant part of B
34: end procedure
```

Theorem 5 *Algorithm 1 is linear in the number of directed edges in a BN $\mathcal{B}$.*

Proof Consider $I(X, Y, Z)$ to be tested in a BN $\mathcal{B}$. Algorithm 1 extends REACHABLE by computing $XYZ \cup An(XYZ)$ in line 4. Algorithm 1 also differs from REACHABLE in lines 19, 22, 27, and 31 by checking whether a variable is in $XYZ \cup An(XYZ)$. As REACHABLE computes ancestral sets and tests set membership (Koller and Friedman, 2009), the claim follows. ■

4. Experimental Results and Analysis

We report on an empirical comparison of REACHABLE and RP-REACHABLE. Both methods were implemented in Python using the *NetworkX* library (see `networkx.github.io`). The experiments were conducted on a 2.3 GHz Inter Core i7 with 8 GB RAM. The evaluation was carried out on 19 real-world or benchmark BNs listed in the first column of Table 1. The second column of Table 1 reports the number of variables in each BN. For each BN, 1000 independencies $I(X, Y, Z)$ were randomly generated. Following REACHABLE in (Koller and Friedman, 2009), where X is a singleton set, in our experiments X, Y, and Z are kept to being singleton sets. Then $I(X, Y, Z)$ is tested by REACHABLE and RP-REACHABLE. The average time in seconds required by REACHABLE and RP-REACHABLE are reported in the third and fourth columns, respectively. Time savings by RP-REACHABLE over REACHABLE is shown in the fifth column. Table 1 shows that RP-REACHABLE was faster than REACHABLE on average by 53%.

Table 1: Comparison of REACHABLE and RP-REACHABLE with 1000 randomly generated independencies $I(X, Y, Z)$ in each BN. Average times are reported in seconds and winners in bold.

BN	Vars	REACHABLE	RP-REACHABLE	savings over REACHABLE
child	20	1.27E-04	**1.14E-04**	10%
insurance	27	1.93E-04	**1.77E-04**	8%
water	32	1.54E-04	**1.38E-04**	10%
mildew	35	**1.62E-04**	1.70E-04	-5%
alarm	37	**1.39E-04**	1.42E-04	-2%
barley	48	2.51E-04	**2.40E-04**	4%
hailfinder	56	1.70E-04	**1.52E-04**	11%
hepar2	70	3.14E-04	**1.95E-04**	38%
win95pts	76	1.21E-04	**9.90E-05**	18%
pathfinder	109	4.84E-04	**1.19E-04**	75%
munin1	186	5.70E-04	**2.60E-04**	54%
andes	223	8.21E-04	**6.81E-04**	17%
diabetes	413	2.38E-03	**2.23E-03**	6%
pigs	441	3.67E-04	**1.11E-04**	70%
link	724	1.20E-03	**3.70E-04**	69%
munin2	1003	7.66E-04	**1.95E-04**	75%
munin4	1038	1.57E-03	**2.47E-04**	84%
munin	1041	1.53E-03	**2.44E-04**	84%
munin3	1041	1.77E-03	**2.45E-04**	86%
Average Time		6.89E-04	**3.23E-04**	53%

Lemma 6 and Theorem 7 analyze why RP-REACHABLE tends to be faster in Table 1.

Lemma 6 *Given an independence $I(X, Y, Z)$ in a BN $\mathcal{B}$, the set of variables reachable from X along active paths in* RP-REACHABLE *is a subset of those determined in* REACHABLE.

In Example 5, RP-REACHABLE determined $R = \{a, d, g\}$, as depicted in Figure 1 (iv), whereas, in Example 2, REACHABLE determined $R = \{a, c, f, h, i, j, d, g\}$, as shown in Figure 1 (ii).

Theorem 7 *Consider testing $I(X, Y, Z)$ in a BN $\mathcal{B}$. If* REACHABLE *traverses an irrelevant path, then either the path will remain active and never reach a variable in $XYZ \cup An(XYZ)$ or it will necessarily become blocked by a closed convergent variable not in $XYZ \cup An(XYZ)$.*

The proofs of Lemma 6 and Theorem 7 will be provided in an extended paper.

Example 6 *Recall Example 2 where $I(a, e, g)$ is tested using* REACHABLE*. The active path (a, c), (c, f), (f, h) is explored, giving (1). However, spending time exploring this irrelevant path is wasteful, since (a, c), (c, f), (f, h), (h, g) is blocked by h, a closed convergent variable outside of $\{a, b, d, e, g\}$, as guaranteed by Theorem 7. Furthermore,* REACHABLE *explores the active path $(a, c), (c, i), (i, j)$ as indicated by (2). Again, time is wasted exploring this irrelevant path, since Theorem 7 ensures that such an active path must terminate before reaching g.*

The important point when testing $I(X, Y, Z)$ is that no active path from X to Z can involve a variable outside of $XYZ \cup An(XYZ)$. For instance, in Example 6, no active path from a to g involves variables c, f, h, i or j. Therefore, variables that are reachable via active paths, but that are not in $XYZ \cup An(XYZ)$, can be safely ignored.

Besides a practical advantage, Theorems 2 and 7 provide a clearer description of d-separation. When explaining the test of independence $I(X, Y, Z)$, the textbooks (Koller and Friedman, 2009; Darwiche, 2009; Pearl, 2016; Shipley, 2016; Kjærulff and Madsen, 2013; Jensen and Nielsen, 2007) all discuss traversing at least one irrelevant path from X to Z without mentioning that this path will necessarily become blocked. This is undesirable, since the d-separation test of $I(X, Y, Z)$ is to decide whether or not there exists an active path from X to Z. Testing irrelevant paths from X to Z does not aid in answering this question. For improved clarity, only relevant paths should be tested. For instance, in Example 4, the relevant path $(a, d), (d, g)$ is active, while the relevant path $(a, d), (d, b), (b, e), (e, g)$ is blocked.

4.1 Variations of rp-Separation

There are two main ways in which rp-separation can be implemented in practice. One approach is to mark and ignore those variables in the BN $\mathcal{B}$ not contained in $XYZ \cup An(XYZ)$. A second approach is to prune $\mathcal{B}$ onto $XYZ \cup An(XYZ)$, yielding a sub-BN $\mathcal{B}'$.

The third and fourth columns of Table 2 show the running times of both approaches to implementing rp-separation following the same setup as described for Table 1. The third column of Table 2 is for the implementation of rp-separation by marking and ignoring variables not in $XYZ \cup An(XYZ)$. The fourth column is for the implementation of rp-separation on a sub-BN $\mathcal{B}'$ built by pruning the BN $\mathcal{B}$ onto $XYZ \cup An(XYZ)$. Without exception, the marking approach was faster than the pruning approach, as highlighted by winning times in bold. The average time savings of marking over pruning is 20%.

5. Refining m-Separation

Here, we analyze and refine m-separation. We begin with an example of m-separation.

Table 2: In each BN, 1000 independencies are randomly generated. The third and forth columns indicate whether rp-separation is faster in practice using marking or pruning. The fifth and sixth columns compare m-separation and refined m-separation. Average times are reported in seconds and winners in bold.

BN	Vars	rp-separation by marking	rp-separation by pruning	m-separation	refined m-separation
child	20	**1.14E-04**	1.37E-04	2.33E-04	**2.11E-04**
insurance	27	**1.77E-04**	2.12E-04	4.65E-04	**3.61E-04**
water	32	**1.38E-04**	1.77E-04	5.29E-04	**3.56E-04**
mildew	35	**1.70E-04**	2.16E-04	5.57E-04	**3.98E-04**
alarm	37	**1.42E-04**	1.82E-04	4.25E-04	**3.33E-04**
barley	48	**2.40E-04**	2.94E-04	7.17E-04	**5.25E-04**
hailfinder	56	**1.52E-04**	1.98E-04	4.94E-04	**3.76E-04**
hepar2	70	**1.95E-04**	2.40E-04	5.53E-04	**4.19E-04**
win95pts	76	**9.90E-05**	1.35E-04	4.07E-04	**2.90E-04**
pathfinder	109	**1.19E-04**	1.43E-04	2.86E-04	**2.42E-04**
munin1	186	**2.60E-04**	3.29E-04	7.92E-04	**6.00E-04**
andes	223	**6.81E-04**	8.54E-04	2.49E-03	**1.59E-03**
diabetes	413	**2.23E-03**	2.71E-03	5.93E-03	**4.33E-03**
pigs	441	**1.11E-04**	1.51E-04	3.75E-04	**3.06E-04**
link	724	**3.70E-04**	4.92E-04	1.46E-03	**9.73E-04**
munin2	1003	**1.95E-04**	2.68E-04	7.19E-04	**5.57E-04**
munin4	1038	**2.47E-04**	3.26E-04	8.35E-04	**6.29E-04**
munin	1041	**2.44E-04**	3.22E-04	8.19E-04	**6.19E-04**
munin3	1041	**2.45E-04**	3.22E-04	7.84E-04	**6.04E-04**
Average Time		**3.23E-04**	4.06E-04	9.93E-04	**7.22E-04**

Example 7 *Consider testing $I(a, e, g)$ using m-separation in the BN $\mathcal{B}$ of Figure 1 (i). In step (i), the sub-DAG constructed onto $\{a, e, g\} \cup An(\{a, e, g\}) = \{a, b, d, e, g\}$ is depicted in Figure 2 (i). In step (ii), the moralization of the sub-DAG is depicted in Figure 2 (ii), where undirected edges (a, b) and (d, e) were added and then directionality was dropped. Edge (a, b) was added as a and b share common child d. Similarly, d and e share common child g. In step (iii), variable e and its incident edges (b, e), (d, e), and (e, g) are deleted, yielding the undirected graph in Figure 2 (iii). Since there exists a path from a to g in Figure 2 (iii), $I(a, e, g)$ does not hold in $\mathcal{B}$ by m-separation.*

The complexity of m-separation is $O(|E|^2)$, where E is the number of directed edges in the BN (Geiger et al., 1989). Although m-separation only considers relevant paths, it is lacking for two other reasons. First, it builds a sub-DAG in step (i) by pruning the given BN. The results in the third and fourth columns of Table 2 suggest that a pruning step is slower than a marking step. Second, steps (ii) and (iii) can be excessive.

Example 8 *Adding edge (a, b) is wasteful in Figure 2 (ii), since there already exists a path (a, d), (d, b) between a and b in the undirected graph in Figure 2 (iii).*

Example 8 shows that m-separation can add edges that do not change connectivity.

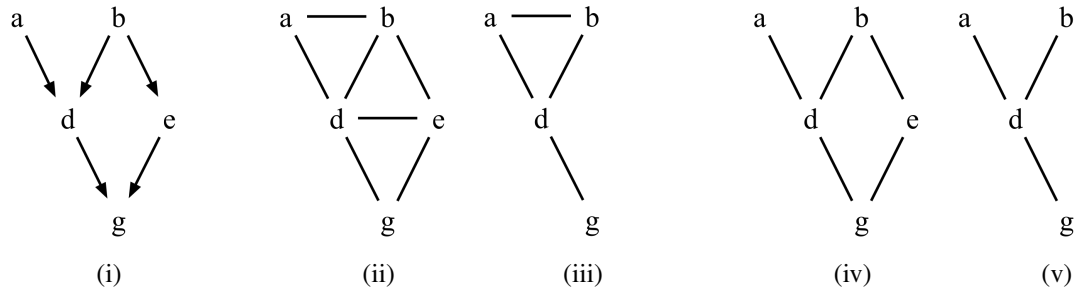

Figure 2: Testing $I(a, e, g)$ in the BN $\mathcal{B}$ in Figure 1 (i) with m-separation in (i)-(iii) and with refined m-separation in (i), (iv), and (v).

Example 9 *Adding edge (d, e) is wasteful in Figure 2 (ii), since (d, e) is deleted in Figure 2 (iii).*

Example 9 demonstrates that m-separation can add an edge in step (ii) only to delete it in step (iii). We now put forth a refinement of m-separation.

Let X, Y, and Z be pairwise disjoint sets of variables in a BN $\mathcal{B}$. Then, refined m-separation tests $I(X, Y, Z)$ with four steps: (i) mark and ignore variables of $\mathcal{B}$ not in $XYZ \cup An(XYZ)$; (ii) add undirected edge (v_i, v_j) between parents v_i and v_j of a common child v_k, only if $v_k \in Y$ and $v_i, v_j \notin Y$; (iii) delete Y and its incident edges from the undirected graph built in (ii); and (iv) if there exists a path from (any variable in) X to (any variable in) Z in (iii), then $I(X, Y, Z)$ does not hold; otherwise, $I(X, Y, Z)$ holds.

Example 10 *Consider testing $I(a, e, g)$ in the BN $\mathcal{B}$ of Figure 1 (i) using refined m-separation. In step (i), the variables under consideration are $\{a, e, g\} \cup \{a, b, d, e\} = \{a, b, d, e, g\}$ as illustrated in Figure 2 (i). In step (ii), variables a and b have variable d as common child, but $d \notin Y$. Similarly, variables d and e have common child $g \notin Y$. Thus, no undirected edges are added in step (ii), as depicted in Figure 2 (iv), where directionality has been dropped. Variable e and its incident edges (b, e) and (e, g) are deleted in step (iii), yielding Figure 2 (v). Since there exists a path from a to g in Figure 2 (v), $I(a, e, g)$ does not hold in $\mathcal{B}$ by refined m-separation.*

Example 10 underscores the advantage of the refined m-separation algorithm. It only adds required edges and it never adds an edge in step (ii) that will be subsequently deleted in step (iii).

The fifth and sixth columns of Table 2 provide the results of an empirical comparison between m-separation and refined m-separation. The experiments were conducted in the same manner as those in Table 1. The refined m-separation algorithm was faster than m-separation in all cases. The time savings were 27% on average. Furthermore, as the same randomly generated independencies were used in Tables 1 and 2, it can be seen that refined m-separation is slower than using REACHABLE, let alone RP-REACHABLE.

It is worth mentioning here that moralization is ideally suited when applied for building a *join tree* (Lauritzen and Spiegelhalter, 1988) for a given DAG. For this purpose, moralization ensures that a clique containing all variables in each CPT of the BN is formed in the undirected graph. On the contrary, moralization can be overkill when applied for testing independencies in BNs.

6. Related Work

Although the motivation and focus of this paper is on m-separation and d-separation, other methods for testing independencies also consider irrelevant active paths.

Bayes-Ball (BB) (Shachter, 1998) is a simple algorithm that can be applied for testing independencies and for determining requisite information in decision problems. Whereas (Geiger et al., 1989) runs in time linear in the size of the DAG, BB runs in time linear in the size of the *active* part of the DAG. Although the efficiency gain is modest (sub-linear versus linear), the computation of requisite information is performed at the same time as irrelevance is determined. However, when applied for testing independencies, we will show that BB can explore irrelevant active paths.

Due to space limitations, we overview BB and refer the reader to (Shachter, 1998) for details. To test $I(X, Y, Z)$ in a BN $\mathcal{B}$, BB takes X and Y as input. It sends a bouncing ball to visit variables in $\mathcal{B}$ starting from X. Depending on the type of variable with respect to Y and the direction from which the ball came (from parents or from children), the ball can *pass through* the variable, *bounce back*, or be *blocked*. The ball behaviour corresponds to reachability along active paths.

Example 11 *Let us test $I(a, e, g)$ in the BN $\mathcal{B}$ of Figure 1 (i) using BB. Starting at $X = \{a\}$, the ball can pass through c, pass through i, and bounce back at j. Moreover, the ball can pass through f, and then is blocked at h. The remainder of the example is immaterial to our discussion.*

Example 11 highlights the fact that BB explores irrelevant active paths. Although the path (a, c),(c, i),(i, j) is active, it is irrelevant as active paths from a to g can only involve variables in $\{a, e, g\} \cup An(\{a, e, g\})$. In addition, BB explores the irrelevant active path (a, c), (c, f), (f, h), (h, g), which is guaranteed by Theorem 7 to become blocked, by closed convergent variable h in this case.

The function DSEP in the *ggm* library (Marchetti, 2006), written in the R programming language, is based upon an alternative method of testing independencies given in (Wermuth and Cox, 2004). Although the test does not use REACHABLE per se, it can be seen as exploring irrelevant paths.

An empirical evaluation of RP-REACHABLE with BB and DSEP remains as future work.

7. Conclusion

When testing independence $I(X, Y, Z)$ in a BN $\mathcal{B}$, the definition of d-separation considers the entire BN, namely, checking whether every path from X to Z is active or blocked. The method in (Geiger et al., 1989) always checks every directed edge in $\mathcal{B}$, but still decides all active paths in linear time. The Bayes-Ball (Shachter, 1998) and REACHABLE (Koller and Friedman, 2009) algorithms do not necessarily explore the entire BN, i.e., they find all variables reachable from X along active paths. Here, we emphasize that $I(X, Y, Z)$ can be tested by considering only the *relevant* part of $\mathcal{B}$. The relevant part of $\mathcal{B}$ is its sub-DAG onto $XYZ \cup An(XYZ)$. Finally, we suggest RP-REACHABLE that only traverses the active paths within the relevant part. Figure 1 reflects these ideas using the test of independence $I(a, e, g)$. The experimental results in Table 1 show that RP-REACHABLE is 53% faster on average than REACHABLE in real-world or benchmark BNs. The work presented in this paper was inspired by Darwinian networks (Butz et al., 2016).

References

C. J. Butz, J. S. Oliveira, and A. E. dos Santos. On Darwinian networks. *Computational Intelligence*, in press, 2016.

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

D. Geiger, T. S. Verma, and J. Pearl. d-separation: From theorems to algorithms. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, pages 139–148, 1989.

I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016. URL `http://www.deeplearningbook.org`.

F. V. Jensen and T. D. Nielsen. *Bayesian networks and decision graphs*. Springer, 2007.

U. B. Kjærulff and A. L. Madsen. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer, 2nd edition, 2013.

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

S. L. Lauritzen and D. J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistical Society*, 50:157–244, 1988.

S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.

G. Marchetti. Independencies induced from a graphical markov model after marginalization and conditioning: The R package ggm. *Journal of Statistical Software*, 15(1), 2006.

K. Mohan and J. Pearl. On the testability of models with missing data. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33, 2014.

R. E. Neapolitan. *Probabilistic methods for bioinformatics: with an introduction to Bayesian networks*. Morgan Kaufmann, 2009.

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

J. Pearl. *Causal inference in statistics: a primer*. John Wiley & Sons Ltd, 2016.

J. Pearl and E. Bareinboim. External validity: From do-calculus to transportability across populations. *Statistical Science*, 29(4):579–595, 2014.

R. D. Shachter. Bayes-ball: 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*, pages 480–487, 1998.

B. Shipley. *Cause and correlation in biology*. Cambridge University Press, 2016.

N. Wermuth and D. R. Cox. Joint response graphs and separation induced by triangular systems. *Journal of the Royal Statistical Society*, 66(3):687–717, 2004.