Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Testing Independencies in Bayesian Networks with i-Separation

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

Christophe Gonzales
christophe.gonzales@lip6.fr
Université Pierre et Marie Curie
France

Abstract

Testing independencies in *Bayesian networks* (BNs) is a fundamental task in probabilistic reasoning. In this paper, we propose *inaugural-separation* (i-separation) as a new method for testing independencies in BNs. We establish the correctness of i-separation. Our method has several theoretical and practical advantages. There are at least five ways in which i-separation is simpler than *d-separation*, the classical method for testing independencies in BNs, of which the most important is that "blocking" works in an intuitive fashion. In practice, our empirical evaluation shows that i-separation tends to be faster than d-separation in large BNs.

Introduction

Pearl (1993) states that perhaps the founding of *Bayesian networks* (BNs) (Pearl 1988) made its greatest impact through the notion of d-separation. *Directed-separation* (d-separation) (Pearl 1986) is a graphical method for deciding which conditional independence relations are implied by the *directed acyclic graph* (DAG) of a BN. To test whether two sets X and Z of variables are conditionally independent given a third set Y of variables, denoted $I(X, Y, Z)$, d-separation checks whether every path from X to Z is "blocked" by Y. This involves classifying every variable between X and Z on each of these paths into one of three categories. This classification may involve consulting variables not appearing on the path itself. Unfortunately, many have had difficulties in understanding d-separation (Pearl 2009), perhaps due to the following two drawbacks. First, the same variable can assume different classifications depending on the path being considered. Second, sometimes a path is not "blocked" by Y even though it necessarily traverses Y.

This paper puts forth *inaugural-separation* (i-separation) as a novel method for testing independencies in BNs. We introduce the notion of an *inaugural* variable, the salient feature of which is that in testing $I(X, Y, Z)$, any path from X to Z involving an inaugural variable is "blocked." This means that paths involving inaugural variables can be ignored. On the paths not involving inaugural variables, only those variables belonging to Y need to be classified as to whether they belong to one category. Our method has several theoretical and practical advantages. On the theoretical side, i-separation is simpler than d-separation. Rather than testing all paths between X and Z, i-separation tests only those paths not involving inaugural variables. On these paths, instead of classifying all variables, i-separation classify only those variables that are members of Y. As compared to d-separation, which classifies variables into three categories, i-separation only requires binary classification. And classification in i-separation only involves consulting variables on the path itself. Thus, i-separation involves fewer paths, fewer variables, and fewer categories. In addition, "blocking" is intuitive in i-separation, namely, a path is "blocked" by Y if and only if it traverses through Y. From a practical perspective, our experimental results indicate that i-separation is especially effective in large BNs.

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 arcs (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 $An(X)$ and $De(X)$ 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 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) $P(v_1|Pa(v_1))$, $P(v_2|Pa(v_2))$, ..., $P(v_n|Pa(v_n))$. For example, Figure 1 shows a BN, where CPTs $P(a)$, $P(b)$, ..., $P(j|i)$ 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 $P(U)$

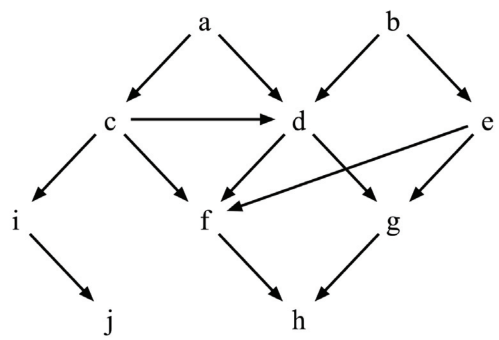

Figure 1: A DAG $\mathcal{B}$.

is denoted $I_P(X, Y, Z)$. 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 DAGs 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 $I(X, Y, Z)$, if at least one variable on every undirected path from X to 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 $I(a, de, g)$ in the DAG $\mathcal{B}$ of Figure 1 using d-separation. It can be verified that there are 17 undirected paths from a to g. Path (a, d), (d, g) is blocked, since sequential variable d is closed. Similarly, the path (a, c), (c, f), (d, f), (d, g) is blocked, since divergent variable d is closed. Moreover, the path (a, c), (c, f), (f, h), (g, h) is blocked, since convergent variable h is closed. It can be verified that the other 14 paths are blocked. Therefore, $I(a, de, g)$ holds. It can be verified that $I(a, d, g)$ does not hold in $\mathcal{B}$.*

i-Separation

Inaugural-separation (*i-separation*) is proposed as a novel method for testing independencies in BNs.

A variable v_k is called a *v-structure* (Pearl 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. For example, variable h is a v-structure in DAG $\mathcal{B}$ of Figure 1, since $\mathcal{B}$ contains directed edges (f, h) and (g, h), and does not contain a directed edge between variables f and g. Variable f is also a v-structure, since $\mathcal{B}$ contains directed edges (c, f) and (e, f), and does contain a directed edge between variables c and e. Similarly, d and g are also v-structures.

Given an independence $I(X, Y, Z)$ to be tested in a DAG $\mathcal{B}$, a variable v is *inaugural*, if either of the following two conditions are satisfied: (i) v is a v-structure and

$$(\{v\} \cup De(v)) \cap XYZ \quad = \quad \emptyset; \qquad (1)$$

or (ii) v is a descendant of a variable satisfying (i). We denote by V the set of all inaugural variables.

Example 2. *Consider testing $I(a, de, g)$ in the DAG $\mathcal{B}$ of Figure 1. Variable f is inaugural, since it is a v-structure and, by (1),*

$$(\{f\} \cup \{h\}) \cap \{a, d, e, g\} \quad = \quad \emptyset.$$

Consequently, by condition (ii), h is also inaugural, since h is a descendant of f. On the contrary, variable d is a v-structure, but is not inaugural, since

$$(\{d\} \cup \{f, g, h\}) \cap \{a, d, e, g\} \quad \neq \quad \emptyset.$$

The concept of a serial variable is needed in i-separation.

Definition 1. *Consider any undirected path $\ldots, (v_i, v_j), (v_j, v_k), \ldots$ passing through variable v_j in a DAG $\mathcal{B}$. We call v_j* serial, *if at most one of v_i and v_k is in $Pa(v_j)$.*

Example 3. *Referring to the DAG in Figure 1, consider the path (a, d), (d, g) passing through variable d. Since $a \in Pa(d)$ and $g \notin Pa(d)$, d is serial. On the other hand, variable d is not serial on the path (a, d), (d, b), since $a, b \in Pa(d)$.*

Note that sequential variables are serial, as are divergent variables. We now formally introduce i-separation.

Definition 2. *Let X, Y, and Z be pairwise disjoint sets of variables in a DAG $\mathcal{B}$. Then i-separation tests $I(X, Y, Z)$ by first pruning inaugural variables from $\mathcal{B}$. For every undirected path from X to Z in the resulting sub-DAG, if there exists a variable in Y that is serial, then $I(X, Y, Z)$ holds; otherwise, $I(X, Y, Z)$ does not hold.*

Example 4. *Let us test $I(a, de, g)$ in the DAG $\mathcal{B}$ of Figure 1 using i-separation. Inaugural variables f and h are pruned, yielding the sub-DAG in Figure 2 (i). Here, there are four undirected paths from a to g, as shown in (ii)-(v) of Figure 2. In (ii) and (iv), $e \in Y$ and e is serial. In (iii) and (v), $d \in Y$ and d is serial. Therefore, $I(a, de, g)$ holds. It can be verified that $I(a, d, g)$ does not hold in $\mathcal{B}$ by i-separation.*

We now present the main result of our paper.

Theorem 1. *Independence $I(X, Y, Z)$ holds in a DAG $\mathcal{B}$ by d-separation if and only if $I(X, Y, Z)$ holds in $\mathcal{B}$ by i-separation.*

Proof. ($\Rightarrow$) Suppose $I(X, Y, Z)$ holds in $\mathcal{B}$ by d-separation. By definition, all paths in $\mathcal{B}$ from X to Z are blocked. Thereby, all paths in $\mathcal{B}$ from X to Z not involving inaugural variables are blocked. By definition, $I(X, Y, Z)$ holds in $\mathcal{B}$ by i-separation.

($\Leftarrow$) Suppose $I(X, Y, Z)$ holds in $\mathcal{B}$ by i-separation. By definition, all paths in $\mathcal{B}$ from X to Z not involving inaugural variables are blocked. Let V be the set of all inaugural variables in $\mathcal{B}$. Consider any undirected path in $\mathcal{B}$ from X to

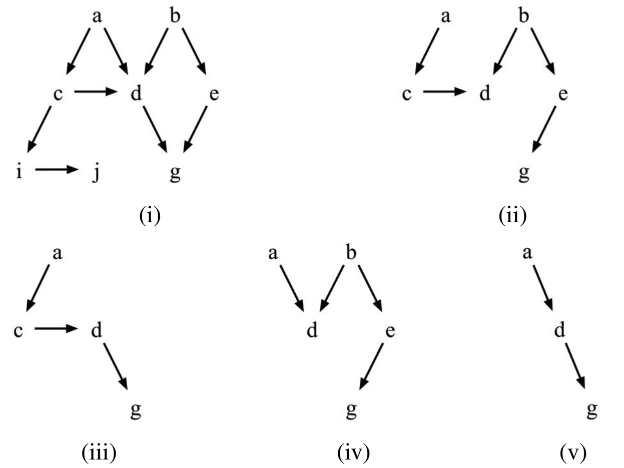

Figure 2: i-Separation prunes inaugural variables f and h from the DAG $\mathcal{B}$ of Figure 1 when testing $I(a, de, g)$, and then classifies only variables d and e in the four paths in (ii)-(v).

Z involving at least one inaugural variable $v \in V$. Without loss of generality, there are two cases to consider.

(i) (v_1, v),(v, v_2), where $v_1, v_2 \notin V$. This means that (v_1, v) and (v_2, v) are directed edges in $\mathcal{B}$; otherwise, v_1 and v_2 are members of $De(v)$, which, by (1), means that $v_1, v_2 \in V$. By (1), $(v \cup De(v)) \cap Y = \emptyset$. Thus, v is a closed convergent variable. Therefore, this path involving an inaugural variable v is blocked in d-separation.

(ii) (v_1, v),(v, v_2), where $v_1 \notin V$ and $v_2 \in V$. Here, $v_1 \notin V$ means that (v_1, v) is a directed edge in $\mathcal{B}$. First, suppose that (v_2, v) is a directed edge in $\mathcal{B}$. Therefore, v is a closed convergent variable. Thus, this path involving an inaugural variable v is blocked in d-separation. Second, suppose that (v, v_2) is a directed edge in $\mathcal{B}$. By (1), $De(v) \cap XYZ = \emptyset$. Thus, any undirected path from X using (v_1, v),(v, v_2) and continuing to Z necessarily traverses a convergent variable v'. Now, $v' \in De(v)$. Since v' is inaugural, by (1), $(v' \cup De(v')) \cap XYZ = \emptyset$. Thus, $(v' \cup De(v')) \cap Y = \emptyset$. By definition, v' is a closed convergent variable. Thus, this path involving an inaugural variable v is blocked in d-separation.

By (i) and (ii), $I(X, Y, Z)$ holds in $\mathcal{B}$ by d-separation. □

Example 5. *$I(a, de, g)$ holds in $\mathcal{B}$ of Figure 1 by d-separation in Example 1, and $I(a, de, g)$ holds in $\mathcal{B}$ by i-separation in Example 4. On the other hand, $I(a, d, g)$ neither holds by d-separation, nor by i-separation.*

Corollary 1. *When testing independence $I(X, Y, Z)$ in a DAG $\mathcal{B}$, if an undirected path $\ldots, (v_i, v_j), (v_j, v_k), \ldots$ passes through an inaugural variable v in $\mathcal{B}$, then this path is blocked by a closed convergent variable.*

Example 6. *When testing $I(a, de, g)$ in the DAG $\mathcal{B}$ of Figure 1 with d-separation, the path (a, c), (c, f), (f, h), (g, h) passes through inaugural variable f, for instance. By Corollary 1, this path is blocked in d-separation by a closed con-*

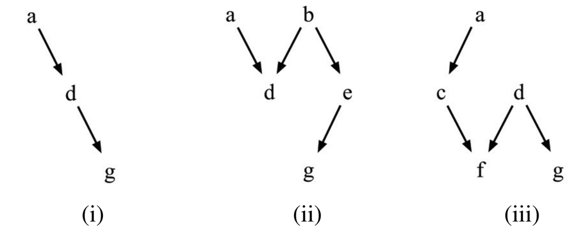

Figure 3: In d-separation, when testing $I(a, de, g)$ in $\mathcal{B}$ of Figure 1, variable d is closed sequential in (i), open convergent in (ii), and closed divergent in (iii).

vergent variable, which is variable h in this instance.

Advantages

Salient features of i-separation are described.

When testing $I(X, Y, Z)$ in a BN $\mathcal{B}$, by Theorem 1, any path involving an inaugural variable can be ignored.

Example 7. *When testing $I(a, de, g)$ in the BN in Figure 1, there are 17 undirected paths from a to g. By Theorem 1, 13 of these paths can be ignored, since they involve inaugural variables f or h. Only the four undirected paths of Figure 2 (ii)-(v) need to be considered when testing $I(a, de, g)$.*

In the paths not involving inaugural variables, i-separation classifies a variable only if it belongs to Y.

Example 8. *Recall the path in Figure 2 (ii) when testing $I(a, de, g)$. d-Separation will classify variables c, d, b, and e but i-separation will classify only d and e, since $d, e \in Y$.*

For the variables in Y on the paths not involving inaugural variables, i-separation classifies only for serial variables.

Example 9. *Recall testing $I(a, de, g)$ in $\mathcal{B}$ of Figure 1. i-Separation classifies only whether d and e are serial in Figure 2 (ii) and (iv), and whether d is serial in Figure 2 (iii) and (v).*

Recall that d-separation classifies a variable into one of three categories, namely, sequential, divergent, and convergent. It should be noted that in d-separation a variable can assume different classifications depending upon the path being considered. When testing $I(a, de, g)$ in $\mathcal{B}$ of Figure 1, variable d can be closed sequential, open convergent, and closed divergent, as illustrated in Figure 3, respectively.

Most importantly, the notion of "blocking" is sometimes counter-intuitive in d-separation.

Example 10. *Recall the three paths in Figure 3 considered by d-separation when testing $I(a, d, g)$ in $\mathcal{B}$ of Figure 1. Even though each of the three paths from variable a to variable g necessarily traverses through variable d, only the paths in (i) and (iii) are considered "blocked."*

Example 10 emphasizes that even though the path in Figure 3 (ii) necessarily traverses through variable d, the path is not considered as "blocked" by d. In i-separation, "blocking" works in the intuitive fashion.

Example 11. *In testing $I(a, de, g)$, i-separation checks for a serial variable in Y blocking each path of Figure 2 (ii)-(v). Variable e blocks the paths in (ii) and (iv), since $e \in Y$ and e is serial. Variable d blocks the paths in (iii) and (v), since $d \in Y$ and d is serial.*

One last advantage in testing whether a path is active or blocked is that i-separation only considers the variables on this path, whereas d-separation may necessarily consult descendants of some of these variables. For example, when testing $I(a, f, e)$ in $\mathcal{B}$ of Figure 1, consider the path (a, d), (b, d), (b, e). Here, i-separation will only examine variables b and d on the path, but d-separation will also consult variables f, g, and h that are not on the path, since checking whether convergent variable d is closed requires examining $De(d)$.

Experimental Results

Geiger at al. (1989) provide a linear-time complexity algorithm for implementing d-separation. Rather than checking whether every path between X and Z is blocked, the implementation determines all variables that are reachable from X on active paths. If a variable in Z is reached, then $I(X, Y, Z)$ does not hold.

The linear implementation of d-separation given in Algorithm 1 (Koller and Friedman 2009) has two phases. Phase I determines the ancestors $An(Y)$ of Y in the DAG $\mathcal{B}$ using the algorithm ANCESTORS (not shown). Phase II, uses the output of Phase I to determine all variables reachable from X via active paths. This is more involved, since the algorithm must keep 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.

Example 12. *Let us apply Algorithm 1 to test $I(nedbarea, markgrm, dgv5980)$ in the Barley BN (Kristensen and Rasmussen 2002) partially illustrated in Figure 4. Phase I determines $A = \{partigerm, jordinf, frspdag, saatid, markgrm\}$ in line 4. In Phase II, lines 6 and 7 set $L = \{(\uparrow, nedbarea)\}$. After initializing V and R to be empty, the main loop starts on line 10.*

Select $(\uparrow, nedbarea)$ on line 11. As $(\uparrow, nedbarea) \notin V$ on line 13 and $nedbarea \notin Y$ on line 14, variable $nedbarea$ is reachable, yielding $R = \{nedbarea\}$ on line 15. Next, set $V = \{(\uparrow, nedbarea)\}$ on line 16. Since $(\uparrow, nedbarea)$ satisfies line 17, on lines 18 and 19, $L = \{(\uparrow, komm)\}$. Then, lines 20 and 21 set $L = \{(\uparrow, komm), (\downarrow, nmin)\}$. This ends the iteration for $(\uparrow, nedbarea)$.

Starting the next iteration of the while loop, select $(\uparrow, komm)$. It can be verified at the end of this iteration, we have $L = \{(\downarrow, nmin), (\downarrow, nedbarea), (\downarrow, aar\_mod)\}$ and $R = \{nedbarea, komm\}$.

Select $(\downarrow, nmin)$ on line 11 for the next iteration. Again, it can be verified that at the end of the iteration, we will have obtained $L = \{(\downarrow, nedbarea), (\downarrow, aar\_mod), (\downarrow, jordn), (\downarrow, mod\_nmin)\}$ and

$$R = \{nedbarea, komm, nmin\}. \tag{2}$$

Algorithm 1 (Koller and Friedman 2009) Find nodes reachable from X given Y via active paths in DAG $\mathcal{B}$

1: **procedure** REACHABLE(X,Y,$\mathcal{B}$)
2: ▷ Phase I: insert Y and all ancestors of Y into A
3: $An(Y) \leftarrow$ ANCESTORS$(Y, \mathcal{B})$
4: $A \leftarrow An(Y) \cup Y$
5: ▷ Phase II: traverse active paths starting from X
6: **for** $v \in X$ **do** ▷ (Node,direction) to be visited
7: $L \leftarrow L \cup \{(\uparrow, v)\}$
8: $V \leftarrow \emptyset$ ▷ (Node,direction) marked as visited
9: $R \leftarrow \emptyset$ ▷ Nodes reachable via active path
10: **while** $L \neq \emptyset$ **do** ▷ While variables to be checked
11: Select (d, v) in L
12: $L \leftarrow L - \{(d, v)\}$
13: **if** $(d, v) \notin V$ **then**
14: **if** $v \notin Y$ **then**
15: $R \leftarrow R \cup \{v\}$ ▷ v is reachable
16: $V \leftarrow V \cup \{(d, v)\}$ ▷ Mark (d, v) as visited
17: **if** $d =\uparrow$ and $v \notin Y$ **then**
18: **for** $v_i \in Pa(v)$ **do**
19: $L \leftarrow L \cup \{(\uparrow, v_i)\}$
20: **for** $v_i \in Ch(v)$ **do**
21: $L \leftarrow L \cup \{(\downarrow, v_i)\}$
22: **else if** $d =\downarrow$ **then**
23: **if** $v \notin Y$ **then**
24: **for** $v_i \in Ch(v)$ **do**
25: $L \leftarrow L \cup \{(\downarrow, v_i)\}$
26: **if** $v \in A$ **then**
27: **for** $v_i \in Pa(v)$ **do**
28: $L \leftarrow L \cup \{(\uparrow, v_i)\}$
29: **return** R

The rest of the example follows similarly, yielding all reachable variables

$$R = \{nedbarea, komm, nmin, aar\_mod, jordn, mod\_nmin, ntilg, \ldots, aks\_vgt\}. \tag{3}$$

It can be verified that $dgv5980 \notin R$. Therefore, the independence $I(nedbarea, markgrm, dgv5980)$ holds.

The linear implementation of d-separation considers all active paths until they become blocked. Our key improvement is the identification of a class of active paths that are doomed to become blocked. By Corollary 1, any path from X to Z involving an inaugural variable is blocked.

Given an independence $I(X, Y, Z)$, Algorithm 2 determines the set of inaugural variables in $\mathcal{B}$.

Example 13. *Consider the Barley BN partially depicted in Figure 4. With respect to the independence $I(nedbarea, markgrm, dgv5980)$, algorithm 2 returns all inaugural variables, including $nmin$ and aar_mod.*

Algorithm 2 can be inefficient, since some inaugurals may not be reachable from X using active paths. For instance, in Figure 4, inaugural variable $ngtilg$ is not reachable from $nedbarea$ using active paths. Thereby, a more efficient approach is to mimic the linear implementation of

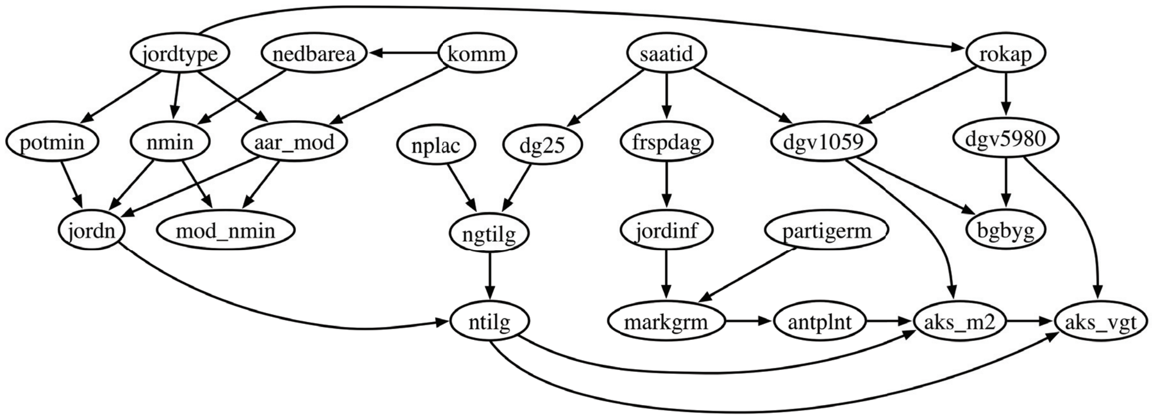

Figure 4: When testing $I(nedbarea, markgrm, dgv5980)$ in the Barley BN (only partially depicted), the traversal of paths from $nedbarea$ to $dgv5980$ can be stopped once they encounter either inaugural variables $aaar\_mode$ or $nmin$.

Algorithm 2 Find all inaugural variables in $\mathcal{B}$, given independence $I(X, Y, Z)$.

procedure ALL-INAUGURALS(X,Y,Z,$\mathcal{B}$)
  $\mathcal{I} \leftarrow \emptyset$ ▷ all inaugural
  $\mathcal{I}^* \leftarrow \emptyset$ ▷ temporary result
  $V' \leftarrow$ all v-structures in $\mathcal{B}$
  $An(XYZ) \leftarrow \text{ANCESTORS}(XYZ, \mathcal{B})$
  $V \leftarrow V' - (An(XYZ) \cup XYZ)$
  for $v \in V$ **do**
    $An(v) \leftarrow \text{ANCESTORS}(\{v\}, \mathcal{B})$
    if $An(v) \cap V = \emptyset$ **then**
      $\mathcal{I}^* \leftarrow \mathcal{I}^* \cup \{v\}$
  $\mathcal{I} \leftarrow \mathcal{I}^* \cup De(\mathcal{I}^*)$
  return $\mathcal{I}$

d-separation, except stopping the traversal of an active path if it encounters an inaugural variable or it becomes blocked.

In an active path, a variable is neither closed, nor inaugural. Therefore, a variable v to be tested can be considered inaugural, if it is a v-structure and $v \notin XYZ \cup An(XYZ)$. This test is given in Algorithm 3.

Algorithm 3 Test if a reachable variable v is inaugural.

procedure INAUGURAL(v,A,$\mathcal{B}$)
  if $v \notin A$ **then** ▷ If v not in $XYZ \cup An(XYZ)$
    if $Pa(v) > 1$ **then** ▷ If v is a v-structure
      return $true$ ▷ v is inaugural
  return $false$

The implementation of i-separation is presented in Algorithm 4.

Example 14. *Let us apply Algorithm 4 to test $I(nedbarea, markgrm, dgv5980)$ in the Barley BN partially depicted in Figure 4. Phase I is the same as in Example 12. In Phase II, lines 5 and 6 determine $A = \{komm,\ partigerm,\ jordinf,\ frspdag,\ saatid,\ rokap,\ jordtype,\ nedbarea, markgrm, dgv5980\}$. In Phase III, lines 8 and 9 set $L = \{(\uparrow, nedbarea)\}$. After setting $V = \emptyset$ and $R = \emptyset$, the main loop starts on line 12.*

Select $(\uparrow, nedbarea)$ in line 13. Now $R = \{nedbarea\}$. Here, $(\uparrow, komm)$ is added to L, but not $(\downarrow, nmin)$, since $nmin \in \mathcal{I}$. It can be verified that selecting $(\uparrow, komm)$ results in $R = \{nedbarea, komm\}$ and $L = \{(\downarrow, nedbarea)\}$. Hence, selecting $(\downarrow, nedbarea)$, results in $L = \emptyset$. Since $dgv5980 \notin R$, the independence $I(nedbarea, markgrm, dgv5980)$ holds.

Observe that, in Example 14, i-separation does not add variable $nmin$ to the set of nodes to be visited, since $nmin$ is inaugural. In contrast, d-separation adds $nmin$ to the set of nodes that are reachable as in (2), then subsequently adds $jordn$ and mod_nmin to the reachable set R in (3).

Table 1: Comparison of d-separation and i-separation with 1000 randomly generated independencies in each BN.

BN	$\|N\|$	Time d-Sep (s)	Time i-Sep (s)	Time Savings
Child	20	0.751	1.003	-34%
Insurance	27	1.544	1.876	-22%
Water	32	1.374	1.742	-27%
Mildew	35	1.272	1.287	-1%
Alarm	37	0.9698	1.077	-11%
Barley	48	2.838	3.259	-15%
Hailfinder	56	1.620	1.9876	-23%
Hepar2	70	3.9817	6.438	-62%
Win95pts	76	1.3366	1.4293	-7%
Pathfinder	135	7.964	14.2821	-79%
Munin1	186	12.9175	11.1387	14%
Andes	223	24.607	23.0223	6%
Diabetes	413	134.571	120.0226	11%
Pigs	441	16.739	10.7111	36%
Link	724	91.707	56.661	38%
Munin2	1003	57.536	38.396	33%
Munin4	1038	145.388	76.899	47%
Munin3	1041	140.15	63.163	55%

We now report an empirical comparison of d-separation

Algorithm 4 Find nodes reachable from X given Y via active paths in DAG $\mathcal{B}$.

procedure i-REACHABLE(X,Y,$\mathcal{B}$)
  $\triangleright$ Phase I: compute all ancestors of Y
  $An(Y) \leftarrow$ ANCESTORS$(Y, \mathcal{B})$
  $\triangleright$ Phase II: insert all ancestors of XYZ into A
  $An(XYZ) \leftarrow$ ANCESTORS$(XYZ, \mathcal{B})$
  $A \leftarrow XYZ \cup An(XYZ)$
  $\triangleright$ Phase III: traverse active paths starting from X
  for $v \in X$ **do**
    $L \leftarrow \{L \cup (\uparrow, v)\}$ $\triangleright$ visit v from child
  $V \leftarrow \emptyset$
  $R \leftarrow \emptyset$
  while $L \neq \emptyset$ **do**
    Select (d, v) from L
    $L \leftarrow L - \{(d, v)\}$
    if $(d, v) \notin V$ **then**
      $V \leftarrow V \cup \{(d, v)\}$
      $\triangleright$ Is v serial?
      if $v \notin Y$ **then**
        $R \leftarrow R \cup \{v\}$
        if $d = \uparrow$ **then** $\triangleright$ up from child
          for $v_i \in Pa(v)$ **do**
            if !(INAUGURAL$(v_i, A, \mathcal{B})$) **then**
              $L \leftarrow L \cup \{(\uparrow, v_i)\}$
          for $v_i \in Ch(v)$ **do**
            if !(INAUGURAL$(v_i, A, \mathcal{B})$) **then**
              $L \leftarrow L \cup \{(\downarrow, v_i)\}$
        else $\triangleright$ down from parent
          for $v_i \in Ch(v)$ **do**
            if !(INAUGURAL$(v_i, A, \mathcal{B})$) **then**
              $L \leftarrow L \cup \{(\downarrow, v_i)\}$
      $\triangleright$ Is v convergent?
      if $d = \downarrow$ **and** $v \in (Y \cup An(Y))$ **then**
        for $v_i \in Pa(v)$ **do**
          $L \leftarrow L \cup \{(\uparrow, v_i)\}$
  return R

and i-separation. Both methods were implemented in the Python programming language. The experiments were conducted on a 2.3 GHz Inter Core i7 with 8 GB RAM. The evaluation was carried out on 18 real-world or benchmark BNs listed in first column of Table 1. The second column of Table 1 reports characteristics of each BN. For each BN, 1000 independencies $I(X, Y, Z)$ were randomly generated, where X, Y, and Z are singleton sets, and tested by d-separation and by i-separation. The total time in seconds required by d-separation and i-separation are reported in the third and fourth columns, respectively. The percentage of time saved by i-separation is listed in the fifth column.

From Table 1, the implementation of i-separation is slower than that of d-separation on all BNs with 135 or fewer variables. The main reason is that Algorithm 1 only computes $An(Y)$, while Algorithm 4 computes $An(Y)$ as well as $An(XYZ)$. In small networks, the time required to compute $An(XYZ)$ is greater than the time saved by exploiting inaugural variables.

Table 1 also shows that i-separation is faster than d-separation on all BNs with 186 or more variables. Time savings appear to be proportional to network size, as larger networks can have more paths. Thus, the time taken by i-separation to computes $An(XYZ)$ is less than the time required to check paths unnecessarily. For example, consider the Barley network in Figure 4. One randomly generated independence was $I(nedbarrea, markrm, dgv5980)$. Here, $nmin$ and $aarmode$ are inaugural variables. Thus, i-separation only consider 4 tests, namely $(\uparrow, nedbarrea)$, $(\downarrow, nmin)$, $(\uparrow, komm)$, $(\downarrow, aarmod)$. No other nodes can be reached via active paths while ignoring inaugural variables. In sharp contrast, d-separation would consider these 4 tests as well as $(\downarrow, mod\_nmin)$, since both $nmin$ and $aarmod$ are open sequential variables. Thus, d-separation will continue exploring reachable variables along these active paths, until eventually determining each active path is blocked by a closed convergent variable.

Conclusion

We proposed *i-separation* as a new method for testing independencies in BNs. Any path from X to Z in $I(X, Y, Z)$ involving an inaugural variable is blocked. Therefore, these paths do not need to be checked and can be safely removed from the BN. In the remaining paths, only variables in Y of $I(X, Y, Z)$ need to be considered. Only one kind of variable, called *serial*, is utilized in i-separation. Finally, blocking works in the intuitive way. Our experimental results indicate that i-separation is especially effective in large BNs.

Acknowledgements

Research supported by NSERC Discovery Grant 238880.

References

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

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

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

Kristensen, K., and Rasmussen, I. A. 2002. The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture* 33(3):197 – 217.

Pearl, J. 1986. Fusion, propagation and structuring in belief networks. *Artificial Intelligence* 29:241–288.

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

Pearl, J. 1993. Belief networks revisited. *Artificial Intelligence* 59:49–56.

Pearl, J. 2009. *Causality*. Cambridge University Press.