Essential Hidden Variables
An Exploratory Algorithm

Brian Patterson and Dimitris Margaritis

Department of Computer Science
Iowa State University
Ames, IA 50010 USA

Abstract. In this paper we characterize an important class of hidden variables in Bayesian networks, *essential hidden variables*, and conduct a study on identifying a subclass thereof. The identification of hidden variables is of great importance because it holds the potential of helping in scientific discovery and advancing human knowledge. This study exhaustively identifies all possible independence-based essential sets of hidden variables in Bayesian networks of up to 8 attributes. Our study verifies the existence of families of distributions that cannot be perfectly represented by any Bayesian network that does not contain at least one (essential) hidden variable, and demonstrates that all such networks, up to the size examined, contain the same structural pattern around the hidden variables.

1 Introduction

Since the introduction of Bayesian networks by Pearl (1), the automated discovery and use of *hidden variables* (also called *latent variables*) to represent unmeasured or unmeasurable factors has been an open problem. A Bayesian network specifies a probability distribution over attributes using a graph in which each attribute of the distribution is represented by a vertex and types of path in the graph indicate influence (or the absence of influence) between attributes.

Informally, hidden variables are hypothesized attributes represented as nodes in the graph about which no experimental information is known. It has been shown that Bayesian networks with hidden variables represent a larger class of probabilistic distributions than ones represented by Bayesian networks without hidden variables (2).

The method described in this paper leads to a more concrete understanding of how Bayesian networks with hidden variables can represent a wider variety of distributions than the theoretical Stratified Exponential Families/Curved Exponential Families (SEF/CEF) distinction (2; 3) and a more complete result than the common cause characterization of causal networks (4; 5) for the network sizes we examine. A method of discovering only sets of hidden variables that enable this increased expressiveness would impact a variety of fields. To understand why this is the case, it is first important to understand how any hidden variable affects a Bayesian network.

1.1 Motivating Example

Consider the case of modeling four attributes of individuals who are high school students: extra-curricular involvement (EC), class attendance (CA), interest in school (IS), and average teacher rating for the teachers of the student (TR). For simplicity, assume that each of these attributes can only be high or low.

Let us say that a set of data collected indicates the following about the distribution of individuals in this domain: on average, for a randomly selected student irrespective of their class attendance (CA), their level of involvement with extra-curricular activities (EC) does not influence (has no statistical dependence with) their interest in school (IS). However, for a randomly chosen student with a high CA, he or she is more likely to be interested in school (high IS) if involved in many extra-curricular activities (high EC). For a randomly chosen student with low CA, he or she tends to not be interested in school if involved in few extra-curricular activities. Similarly, the data show no unconditional dependence between TR and CA but the same type of accumulating dependence when IS is in evidence.

This situation may occur if level of involvement in extra-curricular activities normally has no correlation with the student's interest in school and teacher rating alone does not influence class attendance. The data may indicate that, among those who attend class more, it tends to be the case that the student is involved in extra-curricular activities if and only if they already have a high interest in being in school. However, among students who rarely attend class, those who are bored by school (low IS) also tend to not be motivated to take part in extra-curricular activities (low EC). Similarly, the population of those with high IS might exhibit a correlation between how good their teacher is (TR) and whether they go to class (CA).

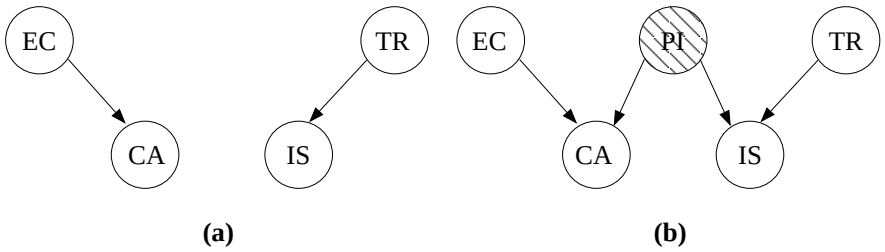

Fig. 1. Two Bayesian networks representing the discussed data (a) without and (b) with hidden variable AA, shown shaded.

A possible Bayesian network that represents this situation is depicted in Figure 1(a). While this network correctly represents finding EC and TR (unconditionally) unrelated to IS and CA (respectively), it does not represent the relationship between EC and IS given CA. This relationship can be represented by adding a hypothetical hidden variable we have labeled PI ("parental involvement") as shown in Figure 1(b). Even though PI cannot be measured, the existence of an attribute between class attendance (CA) and interest in school (IS) allows representation of the relationship between EC and IS given a high

CA. Intuitively, a high (low) EC and high (low) CA indicates a high (low) PI which, in turn, indicates that the student will enjoy (dislike) school. Similarly for the relationship between TR and CA given IS. However, if we look at a selection of individuals for which we do not know the value of CA, the EC rating will not influence IS in that group (see Definition 2 for how we determine these independence relations from a Bayesian network structure).

We show in this paper that PI is actually an *independence-based essential hidden variable*. This means that no network over the 4 measured attributes $\{EC, CA, IS, TR\}$ can represent all the independence relationships described by our hypothetical data without the presence of a hidden variable. Specifically, methods we examine give a way to verify that the structure implied by our data cannot be represented by a Bayesian network of size 4. Probabilistic inference using the network in Figure 1(b) will be, on average, more true to the distribution than the network in Figure 1(a). We call this hidden variable *essential* to representing the distribution with a Bayesian network because its presence enables the representation of a property (in this case, a set of independences) in the underlying distribution strictly more accurately, and no network without it can accomplish this.

1.2 Overview

Hidden variables are used in Bayesian networks for semantic reasons (e.g., Simon (6)) or for the compactness of the resulting Bayesian network (e.g., (7)).[1] Artificial intelligence (AI) research has centered on the use of hidden variables to simplify Bayesian networks while not necessarily altering the distribution that the network represents. In this paper, we will focus on finding structurally meaningful hidden variables, an aim closer to the goal of semantic expressiveness.

We begin with notation and definitions (Section 2) and then present an exploratory algorithm that searches for independence-based essential hidden variables (Section 3.1). Since the search space is exponential in the size of the network, we also present a number of optimizations to this algorithm (Section 3.2). We conclude with experimental results from this algorithm (Section 4).

Through a systematic examination of all networks up to size 8, we discover that an independence-based essential hidden set must have an embedded "W network" edge set (Definition 8) around each hidden variable. Although we are not able to produce any independence characterization of the relationships that must hold for an independence-based essential hidden variable to be present, our results for small networks may be useful in future work for establishing necessary and sufficient general conditions for essential hidden sets.

[1] Many believe these aims are the same (8). The argument is that hidden variables that optimize the compactness of a network must take advantage of some characteristic of the underlying probability distribution. Therefore, there must be some semantic property in the underlying distribution that allows the hidden variable to have a compacting effect.

2 Notation and Definitions

We assume that the reader is familiar with basic graph terminology (directed and divergent simple paths, graph skeletons, cliques, parents ($Par(X)$) and descendants ($Desc(X)$)) as well as the meaning of probabilistic independence terminology (conditional independence and dependence). We will refer to attributes of the hypothetical data set as "attributes" rather than "variables" to emphasize the fact that we are discussing visible, measured attributes of a possible data set. On the other hand, to emphasize that unmeasured attributes are not present in the data set, all unmeasured entities will be referred to as "hidden variables" (see Section 2.2 for further clarification).

We will use capital letters to indicate single attributes, lower case letters to indicate values of attributes, and bold capital letters to indicate sets of attributes. $\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} \mid \boldsymbol{Z}$ will denote that a set $\boldsymbol{X}$ is independent $\boldsymbol{Y}$ given a value of $\boldsymbol{Z}$, while dependence amongst the same sets will be denoted as $\boldsymbol{X} \not\perp\!\!\!\perp \boldsymbol{Y} \mid \boldsymbol{Z}$.

Sometimes it is necessary to clarify the context in which an independence exists. For attributes X, Y, and Z, $(X \perp\!\!\!\perp Y \mid Z)_P$ denotes that the independence relation $X \perp\!\!\!\perp Y \mid Z$ is reflected in probability distribution P while $(X \perp\!\!\!\perp Y \mid Z)_G$ denotes that it is reflected in graphical model G.

2.1 Bayesian Network Formalism

Definition 1 (The Markov Assumption). *The* Markov Assumption *for a graph $G = (\boldsymbol{V}, \boldsymbol{E})$ states that*

$$\forall X \in \boldsymbol{V},\ \{X \perp\!\!\!\perp [\boldsymbol{V} - Desc(X)] \mid Par(X)\}.$$

The Markov Assumption is critical for many proofs involving Bayesian networks that the Markov Assumption holds.

The following formal definitions related to Bayesian networks are adopted from Pearl (8):

Definition 2 (Independence Map (I-Map)). *A graph G is an* independence map *of distribution P over attributes $\boldsymbol{V}$ if there is a one-to-one correspondence between the elements of $\boldsymbol{V}$ and the vertices $\boldsymbol{V}$ of G such that for all disjoint subsets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ of elements we have $(\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} \mid \boldsymbol{Z})_G \Rightarrow (\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} \mid \boldsymbol{Z})_P$.*

Intuitively, a graph is a I-Map if all independences represented in the graph are represented in the distribution.

Definition 3 (Minimal I-Map). *A graph G is a* minimal I-map *of distribution P if no edges can be deleted from G without altering the property that G is an I-map of P.*

Definition 4 (Bayesian Network). *Given a probability distribution P on a set of attributes $\boldsymbol{V}$, a directed acyclic graph (DAG) D is called a* Bayesian network *of P if and only if D is a minimal I-map of P.*

To derive the independences represented in Bayesian networks following the Markov Assumption (as all Bayesian Networks in this paper do), we use the following *d-separation rules*:

Definition 5 (D-separation Rules (8)). *Given a directed, acyclic graph* $G = (\boldsymbol{V}, \boldsymbol{E})$*, for all disjoint sets* $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \subseteq \boldsymbol{V}$, $(\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} \mid \boldsymbol{Z})_G$ *if along every path between a node in* $\boldsymbol{X}$ *and a node in* $\boldsymbol{Y}$ *there is a node* W *satisfying one of the following two conditions: (1)* W *has converging arrows (called a* v-structure *at* W *between* $\boldsymbol{X}$ *and* $\boldsymbol{Y}$*) and none of* W *or Desc(*W*) are in* $\boldsymbol{Z}$*, or (2)* W *does not have converging arrows and* W *is in* $\boldsymbol{Z}$*.*

One can verify, using the rules of d-separation, whether any given conditional independence relation that logically follow from a Bayesian network structure in time polynomial in the number of variables in the domain.

We will be primarily interested in distributions that are *faithful*, as defined by Spirtes et al. (2000, p. 13):

> If all and only the conditional independence relations true in [the probability distribution] P are entailed by the Markov [assumption] applied to [graph] G, we will say that P and G are *faithful to one another*. We will, moreover, say that a distribution P is *faithful* provided there is some directed acyclic graph to which it is faithful. In the terminology of Pearl (8), if P and G are faithful to one another then G is a *perfect map* of P and P is a *DAG-Isomorph* of G.

As noted by many sources (e.g., (8)), many distributions are not faithful. As such, one goal of a procedure that generates a Bayesian network could be to get as close as possible to a perfect Bayesian network for the input distribution. Often this is balanced against computational efficiency concerns.

2.2 Hidden Variables

Definition 6 (Hidden Variables). *An attribute is said to be a* hidden variable *if nothing is known about the actual distribution of the attribute.*

Intuitively, hidden variables are an extreme form of missing data—hidden variables have all of their data missing. As such, hidden variables can never appear in any statement about independence in the distribution.

Note that the definition of a hidden variable is more general than just missing data. It includes situations where we also have no knowledge of the parametric family of distributions that the hidden variable's distribution is a member of, as well as when we also lack knowledge about the values of the parameters of the family. The concept of a hidden variable therefore includes attributes for which we do not even know the number of states that the variable can take.

Figure 2 depicts an example where H is a hidden variable. Independences generated from the graph with d-separation would include $1 \perp\!\!\!\perp 4 \mid 2$ and $1 \perp\!\!\!\perp 3$.

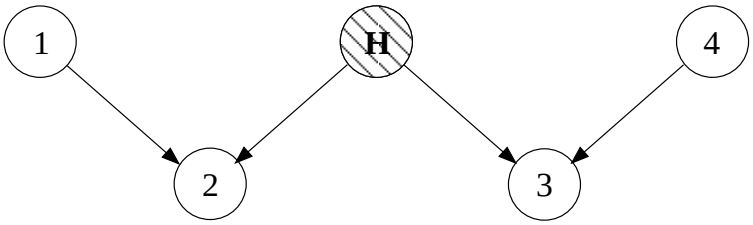

Fig. 2. H is labeled as an example hidden variable (attributes 1,2,3,4 are not hidden).

It also indicates $H \perp\!\!\!\perp 4 \mid 2$ and $2 \perp\!\!\!\perp 3 \mid H$ but we ignore them because they reference H directly or have H in evidence.

There are many perspectives on the use of sets of hidden variables in Bayesian networks. We will focus on hidden sets that enable representation of properties in the network not possible without a hidden set:

Definition 7 (Independence-Based Essential Hidden Set and Variable). *Consider any distribution P with n attributes $\boldsymbol{U}$ represented by a faithful Bayesian network B' ($\boldsymbol{U}'$,$\boldsymbol{E}$) where $\boldsymbol{U}' = \boldsymbol{U} \bigcup \boldsymbol{H}$ for some set of hidden variables $\boldsymbol{H}$ with cardinality k. If, for the set of independences $\boldsymbol{I}_{B'}$ implied by B', $\forall B \in \{n$ attribute legal Bayesian networks representing P with up to $k-1$ hidden variables$\}$ and independences $\boldsymbol{I}_B$ implied by network B, $\boldsymbol{I}_B \neq \boldsymbol{I}_{B'}$, then $\boldsymbol{H}$ is an* independence-based essential hidden set*. If $k = 1$, we call the hidden variable an* independence-based essential hidden variable*.*

Informally, an essential hidden set is a set of attributes that, when added to the visible attributes of a Bayesian network, induces a set of properties that cannot exist in any network with the same number of visible attributes and smaller hidden set. The independence-based essential hidden variable enables us to more closely approach a perfect Bayesian network through correctly representing a larger number of independences from the distribution correctly.

In Figure 2, H is an example of an independence-based essential hidden variable—no Bayesian network with 4 attributes can represent the independence and dependence relationships between attributes 1 through 4. Establishing this is one of the results of our algorithm (Section 3.1). We will call the structure of the network in Figure 2 the *W-network* and define its edge characteristics as follows:

Definition 8 (W-Network (Edge Characterization)). *A* W-network *contains 4 measured attributes labeled as 1,2,3,4 and a hidden variable H. It satisfies the following edge constraints:*

(1) $H \rightarrow 2$
(2) $H \rightarrow 3$
(3) $1 \rightarrow 2$
(4) $4 \rightarrow 3$
(5) No edge from H to 1 or from 1 to H
(6) No edge from H to 4 or from 4 to H

Note that a v-structure exists on either side of H (H to 2 to 1 and H to 3 to 4). This implies that, if we replace the hidden variable with a single edge between 2 and 3, the edge would have to point into both vertices.

3 An Algorithm for Identifying Essential Hidden Variables

3.1 Overview of the Algorithm

Our algorithm explores the space of graphs for essential hidden sets based on the idea that a Bayesian network B with $n-k$ attributes and k hidden variables is sometimes more powerful than any Bayesian network with only $n-k$ attributes and up to $k-1$ hidden variables. This happens when the essential hidden set will have an impact on the $n-k$ visible attributes of B that cannot be duplicated with fewer than k hidden variables. The specific set of properties we are examining for evaluating the increase in representational power is the set of independence relations representable by a Bayesian network.

Given as input a Bayesian network B with n vertices, the algorithm proceeds as follows: D-separation rules are applied to B to generate the set of independences I_B. We then choose a set of attributes $\boldsymbol{H}$ of size k to assume hidden by removing all independences in I_B referring directly to any $H \in \boldsymbol{H}$. While no independence is allowed to include H, the d-separation rules may allow H to influence the independences in B that refer to the remaining variables.

For the example network in Figure 2, suppose that we hide H $(k = 1)$. The list of independences not directly referring to $\boldsymbol{H} = \{H\}$ is $I_B = \{(1 \perp\!\!\!\perp 3), (1 \perp\!\!\!\perp 4), (1 \perp\!\!\!\perp 3 \mid 4), (2 \perp\!\!\!\perp 4), (1 \perp\!\!\!\perp 4 \mid 2), (1 \perp\!\!\!\perp 4 \mid 3), (2 \perp\!\!\!\perp 4 \mid 1)\}$[2].

In the general case, we then generate all Bayesian networks with $n-k$ vertices and up to $k-1$ hidden variables (referred to informally as "smaller networks") and compare the sets of independences represented by each smaller network to I_B. If the set of independences generated by some smaller network and I_B match exactly, we conclude that H is not an essential hidden variable and try hiding a different attribute as a possible hidden variable in B. If the two sets do not match exactly, we continue to the next smaller network. If finish examining all smaller networks without finding a network that generates exactly I_B, then we can conclude (by exhaustion) that H is an independence-based essential hidden variable.

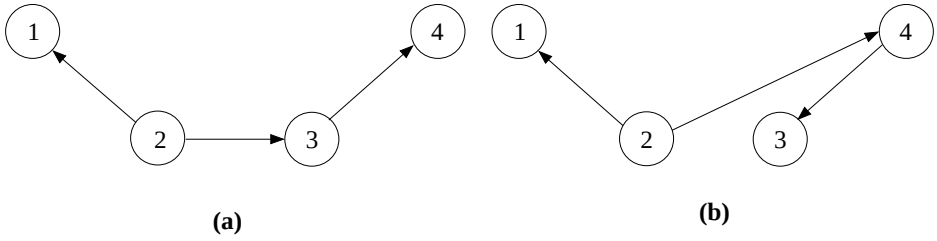

Fig. 3. Example networks with 4 vertices.

Continuing our example, consider the smaller networks over 4 attributes depicted in Figure 3. Amongst other differences between the independence sets of

[2] If H is allowed to be directly referenced as if it were visible, many more independences would result from application of the d-separation rules.

each of these two networks and the W-network (Figure 2), the network in Figure 3(a) does not have independence ($2 \perp\!\!\!\perp 4$) while the network of Figure 3(b) has the extra independence ($2 \perp\!\!\!\perp 3 \mid 4$). Thus these two networks are non-matches for the distribution the W-network represents. The algorithm then continues to examine the remaining networks over 4 attributes, trying to match the independences generated by each network with 4 attributes to I_B. In this example, I_B is not generated by any network with 4 vertices (and $k = 0$ hidden variables) so H is identified as an independence-based essential hidden variable.

The Basic Algorithm (Figure 4) presents the pseudocode that examines all networks with n vertices for essential hidden sets of size k, for a given k and n.

BASIC ALGORITHM—INDEPENDENCE-BASED ESSENTIAL HIDDEN SET IDENTIFICATION

Input: n (TOTAL NUMBER OF ATTRIBUTES POSSIBLE), k (SIZE OF THE HIDDEN SET TO TEST)
FOR (EACH BAYESIAN NETWORK B OVER n VERTICES V) {
  $L = \emptyset$
  $I =$ THE SET OF INDEPENDENCES IN B
  FOR (EACH ATTRIBUTE SET $\boldsymbol{H}$ OF SIZE k, IN THE NETWORK B) {
    $I_H = I -$ ANY INDEPENDENCES MENTIONING ANY $H \in \boldsymbol{H}$
    FOR($j = 1$ TO $k - 1$) {
      FOR (NETWORK B' OVER $n - k$ ATTRIBUTES AND j HIDDEN VARIABLES) {
        $I' =$THE INDEPENDENCES IN B'
        IF ($I_H == I'$)
          $\boldsymbol{H}$ NOT AN ESSENTIAL HIDDEN SET, BREAK AND TRY NEXT $\boldsymbol{H}$
      }
    }
    ADD $\boldsymbol{H}$ TO L
  }
  Output: (B, L)
}

Fig. 4. Basic Algorithm

3.2 Optimizations

The main problem with the Basic Algorithm is that it runs in exponential time—the number of possible directed, acyclic graphs over n variables $f(n)$ is characterized by the recursion $f(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i)$ (9) so $f(n) \in O(2^{n^2-2})$ (the log of this function is graphed in Figure 5). We therefore implemented a number of optimizations so that larger networks can be processed more efficiently. The major optimizations of graph isomorphism usage, the DETERMINEFAITHFUL algorithm, and other formal specifications are

discussed briefly in this section.[3] After all the optimizations are explained, a revised version of the Basic Algorithm including optimization detail is presented.

Graph Isomorphism Equivalence Classes A fundamental alteration to the Basic Algorithm is to test only Bayesian networks that are not graph isomorphic to any other Bayesian network that has already been examined. The Basic Algorithm tests several networks in the same graph isomorphism class. For example, if attribute 0 was identified to be an essential hidden variable in a 5 vertex graph (as in the W-network of Figure 2), attribute 1 would also be an essential hidden variable in the network where the identities of 0 and 1 are switched, attribute 2 when 0 and 2 are switched, etc. To illustrate the difference in the number of graphs that need to be tested, Figure 5 presents the logarithm of the total number of directed acyclic graphs for each vertex count (from (9)) and also the logarithm of the number of isomorphism classes (from (11)).

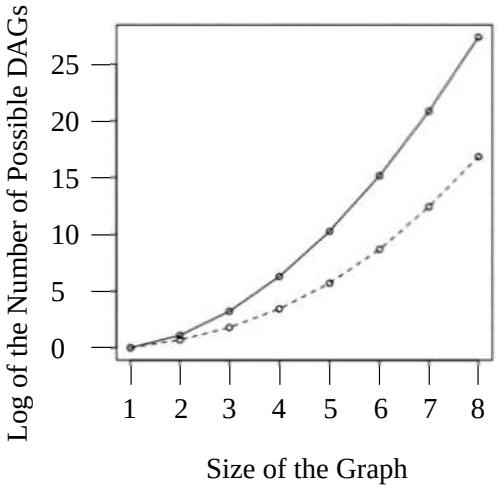

Fig. 5. Size of any directed, acyclic graph (DAG) against the log of the number of possible DAGs (solid line) and number of non-isomorphic possible DAGs (dotted line).

By only testing one graph from each isomorphism class, the search space of networks is greatly reduced. Unfortunately, this reduction in the search space is only helpful enough to enable search of graphs with up to 8 vertices. This is because, while the optimization results in an increasing reduction in the number of networks tested as number of attributes increases, isomorphism still only results in a small reduction in number of graphs—as we can see from Figure 5, the number of non-isomorphic graphs with n vertices is approximately the total

[3] One optimization, testing for connected components in the graph, was found to result in no efficiency gain (10) so it has been omitted from this discussion. An early optimization, memorization of the independences of the smaller networks, is not necessary if DETERMINEFAITHFUL is used.

number of graphs with $n-1$ vertices (for small n). Also, the problem of efficiently enumerating all graph isomorphism classes is expensive—it is suspected to not be in P (12).

Our modified algorithm generates graph isomorphism classes by reading graphs with 8 or fewer vertices from files generated by McKay's NAUTY toolkit (13). NAUTY (No AUTomorphisms, Yes?) is a program for computing automorphism groups of graphs and digraphs. One of its tools, geng (for "generate graphs") generates non-isomorphic graph class representatives very quickly by taking advantage of graph categorizations used by mathematicians and the feasible methods of generating non-isomorphic graphs within that class. NAUTY is acknowledged as the fastest overall graph automorphism and isomorphism detection toolkit with tight run time bounds of $O(n^2)$ and $\Omega(2^n)$ for the best and worst case (respectively) of processing a graph with n vertices (12).

With the testing of only non-graph-isomorphic Bayesian networks, it became necessary to be able to determine if two sets of independences are isomorphic. Two sets of independences are isomorphic if the labeling used in the larger network and the smaller network differ but the independences are the same when the labels of one set are sorted differently. Two sets of independences are tested for equivalence in the following way:

1. Verify that each set of independences has the same cardinality.
2. See if the independences are exactly the same without altering any labels.
3. Make sure the same number of attributes are in evidence for each corresponding independence. For example, if one independence set has 3 independences and each independence has 2 attributes in evidence, the other independence set must have 3 independences with 2 attributes in evidence also.
4. Recursively try every possible relabeling of vertices in one independence set.

Although the last step takes $O(n!)$ time to verify there is no such relabeling, the probability that two non-matching sets will make it to this step is low. In practice, this step rarely needs to be executed and usually results in finding a mismatch in far less than $O(n!)$ time.

An Algorithm to Test Faithfulness: DetermineFaithful Although little work has been done to identify essential hidden variables *per se* in causal network research, some of the work in discovering causality can be used in our independence-based essential hidden set verification algorithms.

The original inductive causation (IC) algorithm (14) assumes that the input attributes are causally sufficient. This means that there are no hidden variables mediating influence between any of the measured attributes. The algorithm works by looking at every pair of attributes A and B and determining if there is a subset of the other attributes that separates A and B (called a *separating set* of A and B). If not, by causal sufficiency we know that either A causes B or B causes A so we can add an undirected edge. V-structures are determined by looking at non-adjacent vertices A and B and an additional attribute C such that there is an edge from A to C and from B to C — if C does not render A

and B independent, the edges must be oriented so that there is a v-structure between A and B at C.

After these two initial steps are completed, there are a variety of methods to attempt to orient the remaining edges so as not to create a cycle or a new v-structure (4). The output of the algorithm is a *partially directed acyclic graph* (PDAG)[4] as, in some cases, it is not possible to use the restrictions of avoiding cycles and new v-structures to constrain the direction of the edges enough to orient all the edges. The *PC Algorithm* (5) is a specific approach to the IC algorithm, specifying the order in which we consider separating sets and starting with a complete graph (rather than one without edges). We can input the independences generated from the larger graph to the PC Algorithm to discover the PDAG that represents the distribution in graphical form. Each time the PC Algorithm queries for independence while discovering separating sets, the independence holds if and only if the input independences do.

However, sometimes the PDAG created by the PC Algorithm cannot be oriented into a Bayesian network without incorrectly adding or removing independences from the input set. Meek (16) provides a method to iteratively generate a member of the family of Bayesian networks representing the independences if this is possible. If it is not possible, the algorithm will generate a network with a cycle or that implies incorrect independences ((17)). For example, depending on which v-structure is discovered first, the independences represented by the hidden variable Bayesian network in Figure 2 will result in an edge between vertices 2 and 3 oriented in an arbitrary direction. However, this graph will not represent the same independences as the network in Figure 2.

For $k = 1$, this algorithm provides a sound and complete way to check whether a set of independences representing a distribution is faithful to a network given causal sufficiency and the independences of the distribution. For future reference, let us call this algorithm DETERMINEFAITHFUL.

When there is more than one hidden variable in the larger network ($k > 1$) and DETERMINEFAITHFUL cannot discover a B', we must resort to checking all graph isomorphism classes of smaller networks over $n - k$ attributes and $k - 1$ or fewer hidden variables. This is because DETERMINEFAITHFUL is not complete when there are hidden variables present — we cannot use it to look at smaller networks that include hidden variables as there is no guarantee that DETERMINEFAITHFUL will just fail to generate the hidden variables in the correct place(s). According to the theorems in Neapolitan (17) and our own testing, this procedure will result in the same set of conclusions as just checking all graph isomorphism classes over $n - k$ attributes and $k - 1 \cdots 0$ hidden variables but is faster overall as the $k = 0$ case (using DETERMINEFAITHFUL) is very fast.

Formal Simplifications Any Bayesian network with hidden variables can be *projected* into a network where all hidden variables have no parents and exactly two children (18). This new Bayesian network is called a *projection* and is defined as follows:

[4] Also called the *essential graph* in Gillispie (15).

Definition 9 (Projection (18)). *The* projection *of a Bayesian network B over observed attributes V_o is any Bayesian network B' where*

1. *B' represents the observed attributes V_o.*
2. *Every hidden vertex in B' has no parents.*
3. *Every hidden vertex in B' has exactly two children.*
4. *B and B' represent the same set of independences.*

Define a *hidden path* as one in which every internal node of the directed path is hidden. Using this idea, the Projection Algorithm given in Figure 6 takes as input any Bayesian network with hidden variables B and returns a projection B' of that network.

PROJECTION ALGORITHM (18)

Input: BAYESIAN NETWORK B
INITIALIZE NETWORK B' TO INCLUDE THE OBSERVED ATTRIBUTES AND NO EDGES.
FOR (EACH PAIR OF ATTRIBUTES X AND Y) {
  IF (THERE EXISTS A DIRECTED HIDDEN PATH FROM X TO Y IN B)
    ADD EDGE FROM X TO Y IN B'
  IF (THERE EXISTS A DIVERGENT HIDDEN PATH FROM X TO Y IN B) {
    ADD HIDDEN VARIABLE $v_{X,Y}$ TO B'
    ADD EDGES FROM $v_{X,Y}$ TO BOTH X AND Y IN B'
  }
}

Fig. 6. Projection Algorithm

Since the projection algorithm works for any network with hidden variables, we can apply it to the current algorithm to save time. We can ignore proposed hidden variables if they have any parents — according to projection (18), there will be another, equivalent network encountered in our search without these parents. Specifically, the network in which the edge to the parent of the hidden variable in the skipped case is deleted and replaced with a direct edge from that parent to each child of the hidden variable.

All of the relevant optimizations above were incorporated in the Optimized Algorithm (Figure 7). Note that we could not incorporate the limit of two children per hidden variable — our algorithm limits the number of hidden variables and the algorithm in Figure 6 relies on the creation of extra hidden variables.

4 Experimental Results

The goal of the experiments was to discover what conditions will always hold around essential hidden sets by analyzing the networks B and corresponding list L of essential hidden sets. If a set of edge or independence constraints holds in the neighborhood of every hidden variable in every essential hidden set, future work may be able to prove that these constraints imply essential hidden sets. Other

OPTIMIZED ALGORITHM—INDEPENDENCE-BASED ESSENTIAL HIDDEN SET IDENTIFICATION

Input: n (TOTAL NUMBER OF ATTRIBUTES POSSIBLE), k (SIZE OF THE HIDDEN SET TO TEST)
SEEN $= \emptyset$
FOR (EACH NETWORK B WITH n VERTICES NOT ISOMORPHIC TO A GRAPH IN SEEN)
{
  ADD B TO SEEN
  $I =$ THE SET OF INDEPENDENCES IN B
  $L = \emptyset$
  FOR (EACH SET $\boldsymbol{H}$ OF PARENTLESS ATTRIBUTES OF SIZE k, IN B) {
    $I_H = I-$ ANY INDEPENDENCES MENTIONING ANY $H \in \boldsymbol{H}$
    CALL DETERMINEFAITHFUL WITH INPUT I_H AND LIST OF ATTRIBUTES
    IF (DETERMINEFAITHFUL WAS SUCCESSFUL)
      $\boldsymbol{H}$ IS NOT AN ESSENTIAL HIDDEN SET, BREAK AND TRY ANOTHER $\boldsymbol{H}$
    }
    FOR($j = 1$ TO $k-1$) {
      SEEN' $= \emptyset$
      FOR (EACH NETWORK B' OVER $n-k$ ATTRIBUTES AND j HIDDEN VARIABLES NOT ISOMORPHIC TO A GRAPH IN SEEN') {
        $I' =$THE INDEPENDENCES IN B'
        IF ($I_H == I'$)
          $\boldsymbol{H}$ NOT AN ESSENTIAL HIDDEN SET, BREAK AND TRY NEXT $\boldsymbol{H}$
        ADD B' TO SEEN'
      }
    }
    ADD $\boldsymbol{H}$ TO L
  }
  OUTPUT (B, L)
}

Fig. 7. Optimized Algorithm

algorithms may then be able to exploit these experimentally verified constraints to find essential hidden sets through local tests.

The algorithms were implemented in Java 1.4.2 on a two-processor 2.8 Ghz Xeon computer with 2 Gb of RAM.

4.1 Edge Test Experiments

The edge tests were done to determine whether there exists a common set of edge constraints that the edges of the graph around the essential hidden set conform to. It was found that the subset of the edge constraints given in the definition of the W-network (Definition 8) held around all essential hidden variables—this means that the W-network is always found embedded in a network with an essential hidden variable and specifically that the hidden variable was always at the apex of the middle peak in the "W" of the W-network.

4.2 Independence Test Experiments

Similarly to the edge tests, various sets of independence constraints were tested against each network that contains an essential hidden set. An example of a set of independence constraints that we attempted to verify was $(1 \perp\!\!\!\perp 3 \mid 4)$, $(1 \not\perp\!\!\!\perp 3 \mid 2)$, $(1 \not\perp\!\!\!\perp \{2,4\})$, and $(2 \perp\!\!\!\perp 4 \mid 1)$. This example set of independences held in both networks over 5 vertices that had a single essential hidden variable (see Figure 9). However, neither that set nor any subset of those independences held in all networks with 6 or more vertices with an essential hidden variable. No set of independence constraints that we tried held in networks over 6 or more vertices. This indicates that it may not be possible to characterize essential hidden variables through independences among the visible attributes only.

4.3 Summary and Examples

Table 1. Quantitative Information about the Algorithm Runs

Size	Alg	# HVs	Num Nets	Found HVs	Total Minutes
4	B	1	543	0	0.0005
4	O	1	32	0	0.0013
4	O	2	32	0	0.0006
4	O	3	32	0	0.0003
5	B	1	29280	100	0.0576
5	O	1	303	2	0.0035
5	O	2	303	0	0.0012
5	O	3	303	0	0.0005
5	O	4	303	0	0.0004
6	B	1	> 54161	> 2185	> 2160
6	O	1	5985	107	0.2141
6	O	2	5985	2	0.0626
6	O	3	5985	0	0.0098
6	O	4	5985	0	0.0036
6	O	5	5985	0	0.0026
7	B	1	> 800	> 67	> 2160
7	O	1	243,669	8191	32.1151
7	O	2	243,669	184	49.3549
7	O	3	243,669	0	1.5390
7	O	4	243,669	0	0.2595
7	O	5	243,669	0	0.1409
7	O	6	243,669	0	0.1537
8	B	1	N/A	N/A	> 2160
8	O	1	$> 2,927,649$	$> 296,415$	> 2160
8	O	2	$> 11,510$	> 74	> 2160
8	O	3	$> 558,400$	> 21	> 2160
8	O	4	20,286,026	1	16.4701
8	O	5	20,286,026	0	23.3306
8	O	6	20,286,026	0	19.6434
8	O	7	20,286,026	0	15.6430

Table 1 displays the number of all Bayesian networks with independence-based essential hidden variables as well as average run times for Basic (B) and Optimized (O) Algorithms (see Figures 4 and 7). Tests for more than one hidden variable were only run with the Optimized Algorithm. The size of the hidden set tested k is indicated in the "# HVs" column.

The "Num Nets" column gives the number of Bayesian networks that each algorithm examined while "Found HVs" refers to the number of graphs that contain essential hidden sets. Thus, for networks with 6 vertices, the Optimized Algorithm found 107 of 5,985 isomorphism-equivalent classes of graphs contain at least one essential hidden set of size 1. Some algorithm runs could not be completed in 36 hours—this is indicated by the ">" notation that more networks exist. There are more essential hidden sets found using the Basic Algorithm than the Optimized Algorithm because the Basic Algorithm iterates over networks, while the Optimized Algorithm iterates over independence equivalence classes, so the same essential hidden variables are found multiple times using the Basic Algorithm under different node labelings.

Total time for each algorithm run is given in the last column of Table 1. Note that the results for incomplete analysis are biased towards smaller, simpler graphs on average (as graphs with fewer edges are tested first) so the time recorded are slightly lower than expected. Performance on graphs with 9 or more vertices is not reported as preparation to process these graphs took longer than the time allowed (36 hours). Test times for the Optimized Algorithm does not account for the time spent to generate the isomorphism classes of graphs—this was a total of 0.08 seconds for 6 vertices, 2.2 seconds for 7 vertices, and 260 seconds for 8 vertices (for smaller graphs, time was negligible and, for graphs larger than size 8, it took over a day).

One result of interest is the average number of Bayesian networks with essential hidden sets in each run of the optimized algorithm (that is, for each possible essential hidden set size). This gives us an estimate of the average probability of encountering unique networks that contain essential hidden sets — if this average is high, it is common for a network structure with a given number of attributes and hidden set size to have an essential hidden set.

A graph of the average number of Bayesian network with essential hidden variables per Bayesian network vs. k is given in Figure 8. As the number of attributes plus hidden variables (n) increases and the number of hidden variables in the set (k) decreases, the ratio of networks with essential hidden sets to total networks increases. This leads to the supposition that, for $n > 8$, the probability of a given network structure containing an essential hidden set of any size k will increase with n. This in turn hints that a fundamental understanding of the impact of essential hidden sets becomes more crucial as we increase the number of attributes.

Lastly, some examples of graphs over 5, 6, 7, and 8 vertices that were identified to contain essential hidden sets are given in Figure 9.

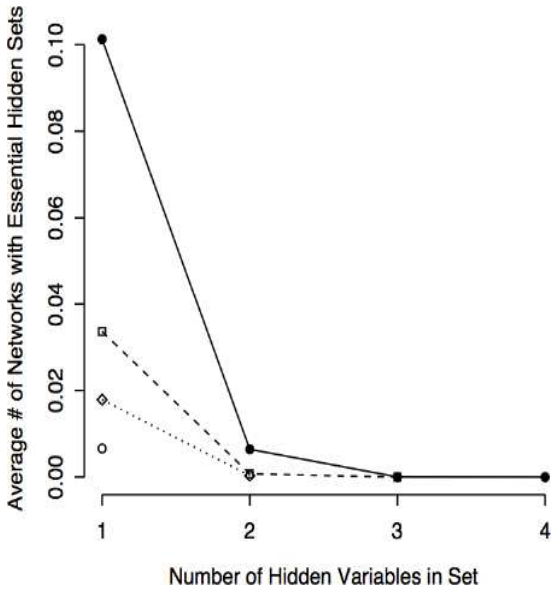

Fig. 8. Number of hidden variables tested for (k) vs. the average number of Bayesian network with essential hidden variables per Bayesian network examined of a given size (optimized algorithm only). Solid line and dots indicate size 8, dashed line and squares indicate size 7, dotted line and circles indicate size 6, and the single point indicates size 5 ratios.

5 Conclusions and Future Research

Our experiments show that networks of the sizes examined all contain the edges present in a W-network (Definition 8) embedded around the hidden variable. Precisely all the edges that are and are not allowed and what this implies for the independences between measured attributes has yet to be determined. However, if examining all the networks of a certain vertex count is required, significant optimizations will be needed to answer these questions within a reasonable time frame.

We had originally hoped to create an intelligent search algorithm of the space of networks but the d-separation rules used to generate independences when hidden variables are present do not allow this method to be more efficient than a complete search of the space. By examining a subset of possible Bayesian networks (that is, placing more restrictions on where hidden variables may appear or what overall distributions we are concerned about), more general conclusions about that subset can be drawn. If approximation is acceptable, we could draw a sample of networks and draw tentative conclusions from that sample.

An algorithm could be used to analyze the results of our algorithm and generate constraints similar to the independence or edge test constraints. This could illuminate whether an independence-based characterization of essential hidden variables exists and whether there is a "universal" edge-based characterization of essential hidden sets. These constraints could, in turn, be used to generate

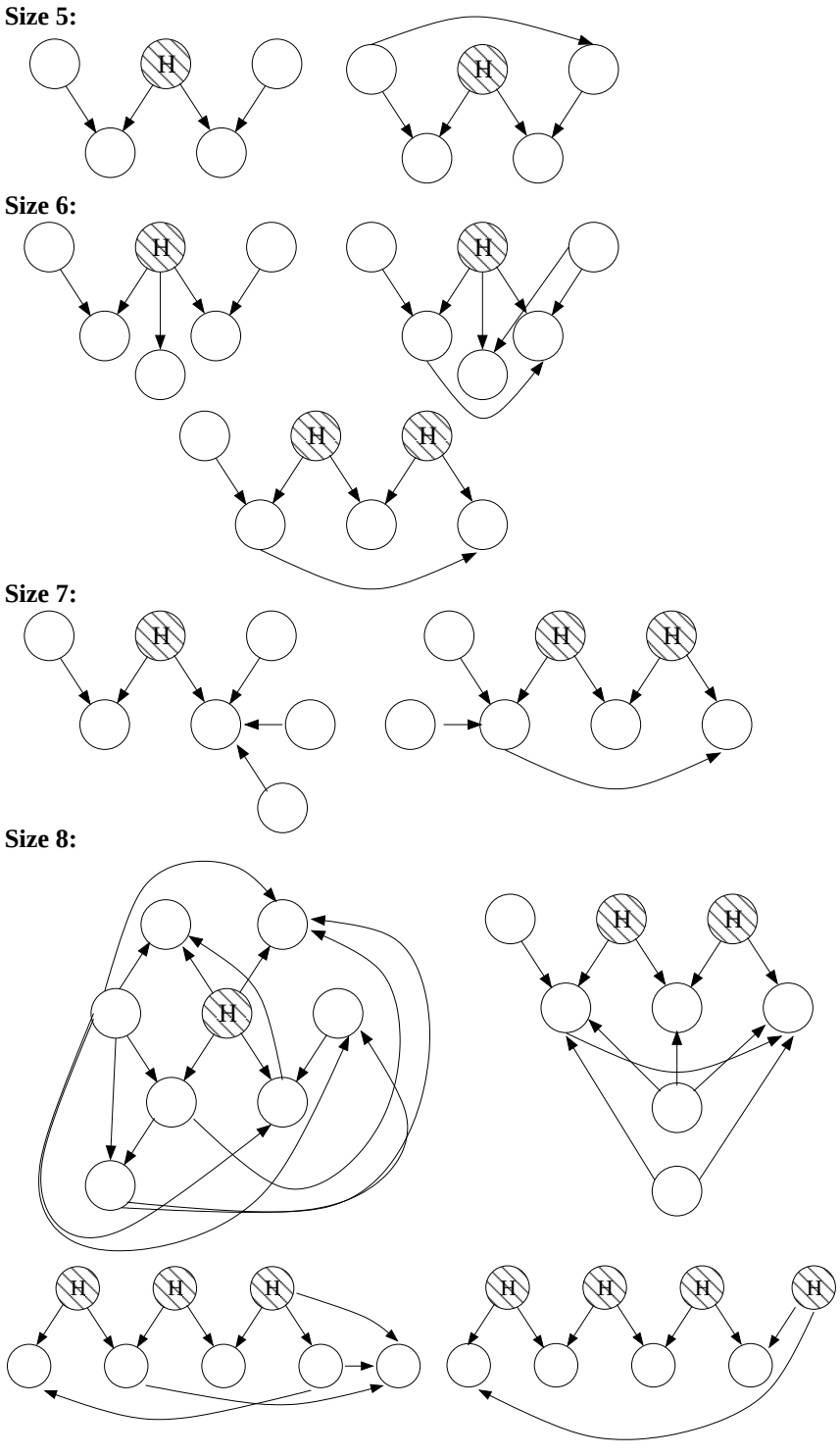

Fig. 9. Examples of Bayesian networks with essential hidden sets over 5 through 8 attributes.

tests for detecting whether an essential hidden set is present in attribute sets rather than deducing which variables are in a position to be essential amongst the attributes.

Finally, in our algorithms, we examined only the identification of a type of essential hidden set through independence constraints—it is possible that other essential hidden sets can be determined using other specification methods. Pearl and Tian (19) have made progress in this regard through the proposal of functional constraints existing around some essential hidden variables. It would also be interesting to see what algorithms are optimal for discovering these other types of hidden sets.

Acknowledgments

The first author would like to thank Jack Lutz and Daniel Ashlock for their suggestions on improvement when this paper was part of a Masters thesis. He would also like to thank the Iowa State Department of Computer Science and Graduate College for providing financial support to continue his research.

[1] Pearl, J.: A constraint-propagation approach to probabilistic reasoning. In: Proceedings of the 2nd Conference on Uncertainty in AI. (1986) 357–370
[2] Geiger, D., Heckerman, D., King, H., Meek, C.: Stratified exponential families: Graphical models and model selection. Technical Report MSR-TR-98-31, Microsoft Research (1998)
[3] Rusakov, D., Geiger, D.: Asymptotic model selection for naive bayesian networks. In: Proceedings of the 18th Conference on Uncertainty in AI. (2002) 438–445
[4] Pearl, J.: Causality. Cambridge University Press, Cambridge, U.K. (2000)
[5] Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search. 2 edn. The MIT Press, Cambridge, MA (2000)
[6] Simon, H.A.: Spurious correlation: A causal interpretation. Journal of the American Statistics Association **49** (1954) 467–479
[7] Binder, J., Koller, D., Russell, S., Kanazawa, K.: Adaptive probability networks with hidden variables. Machine Learning **29** (1997) 213–244
[8] Pearl, J.: Probabilistic Reasoning in Intelligent Systems. 2nd edn. Morgan Kaufmann Publishers, Inc., San Francisco, CA (1988)
[9] Robinson, R.W.: Counting labelled acyclic graphs. In Harary, F., ed.: New Directions in the Theory of Graphs. Academic Press, Inc., New York, New York (1971) 239–273
[10] Patterson, B.: Essential hidden variables: An introduction and novel algorithm for detection. Masters of science, Iowa State University (2004)
[11] McKay, B.: Combinatorial data. http://cs.anu.edu.au/people/bdm/data/digraphs.html (2004)
[12] Miyazaki, T.: The complexity of mckay's canonical labeling algorithm. In Einkelstien, Kantor, eds.: Conference on Discrete Mathematics and Computer Science (DIMACS). Volume 28. (1997)
[13] McKay, B.: The nauty page. http://cs.anu.edu.au/people/bdm/nauty/ (2004)
[14] Pearl, J., Verma, T.S.: A theory of inferred causation. In Allen, J., Fikes, R., Sandewall, E., eds.: Principles of Knowledge Representation and Reasoning, San Mateo, CA, Morgan Kaufmann (1991) 441–452
[15] Gillispie, S., Perlman, M.: The size distribution for markov equivalence classes of acyclic digraph models. Artificial Intelligence **141** (2002) 137–155
[16] Meek, C.: Strong completeness and faithfulness in bayesian networks. In Besnard, P., Hanks, S., eds.: UAI, Montreal, Canada, Morgan Kaufmann (1995) 411–418
[17] Neapolitan, R.: Learning Bayesian Networks. Prentice Hall (2003)
[18] Verma, T.S.: Graphical aspects of causal models. Technical Report R-191, UCLA (1993)
[19] Pearl, J., Tian, J.: On the testable implications of causal models with hidden variables. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), Edmonton, Canada (2002)