Journal of Machine Learning Research 5 (2004) 1287–1330 Submitted 3/04; Published 10/04

Large-Sample Learning of Bayesian Networks is NP-Hard

David Maxwell Chickering DMAX@MICROSOFT.COM
David Heckerman HECKERMA@MICROSOFT.COM
Christopher Meek MEEK@MICROSOFT.COM
Microsoft Research
Redmond, WA 98052, USA

Editor: David Madigan

Abstract

In this paper, we provide new complexity results for algorithms that learn discrete-variable Bayesian networks from data. Our results apply whenever the learning algorithm uses a scoring criterion that favors the simplest structure for which the model is able to represent the generative distribution exactly. Our results therefore hold whenever the learning algorithm uses a consistent scoring criterion and is applied to a sufficiently large dataset. We show that identifying high-scoring structures is NP-hard, even when any combination of one or more of the following hold: the generative distribution is perfect with respect to some DAG containing hidden variables; we are given an independence oracle; we are given an inference oracle; we are given an information oracle; we restrict potential solutions to structures in which each node has at most k parents, for all $k \geq 3$.

Our proof relies on a new technical result that we establish in the appendices. In particular, we provide a method for constructing the local distributions in a Bayesian network such that the resulting joint distribution is provably perfect with respect to the structure of the network.

Keywords: learning Bayesian networks, search complexity, large-sample data, NP-Hard

1. Introduction

Researchers in the machine-learning community have generally accepted that without restrictive assumptions, learning Bayesian networks from data is NP-hard, and consequently a large amount of work in this community has been dedicated to heuristic-search techniques to identify good models. A number of discouraging complexity results have emerged over the last few years that indicate that this belief is well founded. Chickering (1996) shows that for a general and widely used class of Bayesian scoring criteria, identifying the highest-scoring structure from small-sample data is hard, even when each node has at most two parents. Dasgupta (1999) shows that it is hard to find the polytree with highest maximum-likelihood score. Although we can identify the highest-scoring tree structure using a polynomial number of calls to the scoring criterion, Meek (2001) shows that identifying the best *path structure*—that is, a tree in which each node has degree at most two—is hard. Bouckaert (1995) shows that for domains containing only binary variables, finding the parameter-minimal structure that is consistent with an independence oracle is hard; we discuss this result in more detail below. Finally, Srebro (2001) shows that it is hard to find Markov networks with bounded tree width that maximize the maximum-likelihood score.

In this paper, we are interested in the large-sample version of the learning problem considered by Chickering (1996). The approach used by Chickering (1996) to reduce a known NP-complete problem to the problem of learning is to construct a complicated prior network that defines the

©2004 David Maxwell Chickering, David Heckerman and Christopher Meek.

Bayesian score, and then create a dataset consisting of a single record. Although the result is discouraging, the proof technique leaves open the hope that, in scenarios where the network scores are more "well behaved", learning is much easier.

As the number of records in the observed data grows large, most scoring criteria will agree on the same partial ranking of model structures; in particular, any *consistent* scoring criterion will—in the limit of large data—favor a structure that allows the model to represent the generative distribution over a structure that does not, and when comparing two structures that both allow the model to represent the generative distribution, will favor the structure that results in fewer model parameters. Almost all scoring criteria used in practice are consistent, including (1) any Bayesian criterion that does not rule out model structures apriori, (2) the minimum-description-length criterion, and (3) the Bayesian-information criterion.

In this paper, we consider the scenario when a learning algorithm is using a consistent scoring criterion with a large dataset. We assume that the learning algorithm has direct access to the generative distribution itself; the resulting learning problem is to identify the simplest DAG that allows the resulting Bayesian network to represent the generative distribution exactly. There are a number of algorithms that have been developed for this large-sample learning problem. The SGS algorithm (Spirtes, Glymour and Scheines, 2000), the GES algorithm (Meek, 1997; Chickering, 2002), and the KES algorithm (Nielsen, Kočka and Peña, 2003) all identify the optimal DAG if there exists a solution in which all independence and dependence relationships implied by that structure hold in the generative distribution (that is, the generative distribution is *DAG perfect* with respect to the observable variables). Unfortunately, none of these algorithms run in polynomial time in the worst case.

With some restrictive assumptions, however, we can accomplish large-sample learning efficiently. If (1) the generative distribution is DAG perfect with respect to the observable variables and (2) we know that there exists a solution in which each node has at most k parents (for some constant k), then we can apply the SGS algorithm to identify the best network structure in a polynomial number of independence tests. In particular, because we know the value k, we can limit the worst-case number of independence tests used by the algorithm. Alternatively, if (1) the generative distribution is DAG perfect with respect to *some* DAG that might contain vertices corresponding to hidden variables, and (2) we are given a total ordering over the variables that is consistent with the best structure, then we can find the best DAG using a polynomial number of calls to the scoring criterion by applying a version of the GES algorithm that greedily adds and then deletes the parents of each node.

Unfortunately, the assumptions needed for these special-case efficient solutions are not likely to hold in most real-world scenarios. In this paper, we show that in general—without the assumption that the generative distribution is DAG perfect with respect to the observables and without the assumption that we are given a total ordering—large-sample learning is NP-hard. We demonstrate that learning is NP-hard even when (1) the generative distribution is perfect with respect to a DAG (which contains hidden variables), (2) we are given an independence oracle, (3) we are given given an inference oracle, and/or (4) we are given an information oracle. We show that these results also apply to the problem of identifying high-scoring structures in which each node has at most k parents, for all $k \geq 3$.

A secondary contribution of this paper is a general result about Bayesian networks: in the appendices of this paper, we identify two properties of the local distributions in a Bayesian network

that are sufficient to guarantee that all independence and dependence facts implied by the structure also hold in the joint distribution. Our NP-hard proof relies on this result.

As an extension of our main result, we consider the case in which we are given an independence oracle, and we show in Theorem 15 that the resulting learning problem remains NP-hard. This theorem extends the independence-oracle result of Bouckaert (1995) in a number of ways. Perhaps most important, we place no restriction on the number of states for the (discrete) variables in the domain, which proves the conjecture in Bouckaert (1995) that learning with an independence oracle in non-binary domains is NP-hard. Another extension we make has to do with assumptions about the generative distribution. In the elegant reduction proof of Bouckaert (1995), the constructed independence oracle is consistent with a particular generative distribution that is not perfect with respect to any DAG. Although this distribution has properties that yield a much simpler reduction than our own, the results of this paper apply under the common assumption in the machine-learning literature that the generative distribution is, in fact, perfect with respect to some DAG. Furthermore, the DAG we use in the reduction, which contains hidden variables, has a sparse dependency structure: each node has at most two parents. Finally, our result extends the Bouckaert (1995) oracle-learning result to scenarios where we want to identify sparse (i.e., parent-count limited) model structures that are consistent with an oracle.

2. Background

In this section, we provide background material relevant to the rest of the paper. We denote a variable by an upper case token (e.g., A, B_i, Y) and a state or value of that variable by the same token in lower case (e.g., a, b_i, y). We denote sets with bold-face capitalized tokens (e.g., $\mathbf{A}, \mathbf{B}$) and corresponding sets of values by bold-face lower case tokens (e.g., $\mathbf{a}, \mathbf{b}$). Finally, we use calligraphic tokens (e.g., $\mathcal{B}, \mathcal{G}$) to denote Bayesian networks and graphs.

In this paper, we concentrate on Bayesian networks for a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, where each $X_i \in \mathbf{X}$ has a finite number of states. A *Bayesian network* for a set of variables $\mathbf{X}$ is a pair $(\mathcal{G}, \theta_{\mathcal{G}})$ that defines a joint probability distribution over $\mathbf{X}$. $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is an acyclic directed graph—or *DAG* for short—consisting of (1) nodes $\mathbf{V}$ in one-to-one correspondence with the variables $\mathbf{X}$, and (2) directed edges $\mathbf{E}$ that connect the nodes. $\theta_{\mathcal{G}}$ is a set of parameter values that specify the conditional probability distributions that collectively define the joint distribution.

We assume that each conditional probability distribution is a full table. That is, for each variable there is a separate (unconstrained) multinomial distribution given every distinct configuration of the parent values. For a variable X_i with r_i states, $r_i - 1$ parameters are both necessary and sufficient to specify an arbitrary multinomial distribution over X_i. Thus, assuming that there are q_i distinct parent configurations for X_i, the conditional distribution for X_i will contain $(r-1) \cdot q_i$ parameter values. We also assume that the number of states for each variable is some constant that does not depend on the number of variables in the domain.

Learning a Bayesian network from data requires both identifying the model structure $\mathcal{G}$ and identifying the corresponding set of model parameter values $\theta_{\mathcal{G}}$. Given a fixed structure, however, it is straightforward to estimate the parameter values. As a result, research on the problem of learning Bayesian networks from data is focused on methods for identifying one or more "good" DAG structures from data.

All independence constraints that necessarily hold in the joint distribution represented by any Bayesian network with structure $\mathcal{G}$ can be identified by the by the *d-separation* criterion of Pearl

(1988) applied to to $\mathcal{G}$. In particular, two nodes X and Y are said to be d-separated in a DAG $\mathcal{G}$ given a set of nodes $\mathbf{O}$ if and only if there is no $\mathbf{O}$-*active path* in $\mathcal{G}$ between X and Y; an $\mathbf{O}$-active path is a simple path for which each node Z along the path either (1) has converging arrows and Z or a descendant of Z is in $\mathbf{O}$ or (2) does not have converging arrows and Z is not in $\mathbf{O}$. By simple, we mean that the path never passes through the same node twice. If two nodes are not d-separated given some set, we say that they are *d-connected* given that set. We use $X \perp\!\!\!\perp_{\mathcal{G}} Y|\mathbf{Z}$ to denote the assertion that DAG $\mathcal{G}$ imposes the constraint—via d-separation—that for all values $\mathbf{z}$ of the set $\mathbf{Z}$, X is independent of Y given $\mathbf{Z} = \mathbf{z}$. For a probability distribution $p(\cdot)$, we use $X \perp\!\!\!\perp_p Y|\mathbf{Z}$ to denote the assertion that for all values $\mathbf{z}$ of the set $\mathbf{Z}$, X is independent of Y given $\mathbf{Z} = \mathbf{z}$ in p.

We say that a distribution $p(\mathbf{X})$ is *Markov* with respect to a DAG $\mathcal{G}$ if $X \perp\!\!\!\perp_{\mathcal{G}} Y|\mathbf{Z}$ implies $X \perp\!\!\!\perp_p Y|\mathbf{Z}$. Similarly, we say that $p(\mathbf{X})$ is *faithful* with respect to $\mathcal{G}$ if $X \perp\!\!\!\perp_p Y|\mathbf{Z}$ implies $X \perp\!\!\!\perp_{\mathcal{G}} Y|\mathbf{Z}$. If p is both Markov and faithful with respect to $\mathcal{G}$, we say that p is *perfect* with respect to $\mathcal{G}$. Note that if p is faithful with respect to $\mathcal{G}$, then $X \not\perp\!\!\!\perp_{\mathcal{G}} Y|\mathbf{Z}$ implies that *there exists* some x, y and $\mathbf{z}$ such that $p(x,y|\mathbf{z}) \neq p(x|\mathbf{z})p(y|\mathbf{z})$; there may be other values for which equality holds. We say that $p(\mathbf{X})$ is *DAG perfect* if there exists a DAG $\mathcal{G}$ such that $p(\mathbf{X})$ is perfect with respect to $\mathcal{G}$.

We say that a DAG $\mathcal{G}$ *includes* a distribution $p(\mathbf{X})$—and that $p(\mathbf{X})$ is *included by* $\mathcal{G}$—if the distribution can be represented by some Bayesian network with structure $\mathcal{G}$. Because we are only considering Bayesian networks that have complete tables as conditional distributions, $\mathcal{G}$ includes $p(\mathbf{X})$ if and only if $p(\mathbf{X})$ is Markov with respect to $\mathcal{G}$. We say that two DAGs $\mathcal{G}$ and $\mathcal{G}'$ are *equivalent* if the two sets of distributions included by $\mathcal{G}$ and $\mathcal{G}'$ are the same. Due to the complete-table assumption, an equivalent definition is that $\mathcal{G}$ and $\mathcal{G}'$ are equivalent if they impose the same independence constraints (via d-separation). For any DAG $\mathcal{G}$, we say an edge $X \rightarrow Y$ is *covered* in $\mathcal{G}$ if X and Y have identical parents, with the exception that X is not a parent of itself. The significance of covered edges is evident from the following result:

Lemma 1 (Chickering, 1995) *Let $\mathcal{G}$ be any DAG, and let $\mathcal{G}'$ be the result of reversing the edge $X \rightarrow Y$ in $\mathcal{G}$. Then $\mathcal{G}'$ is a DAG that is equivalent to $\mathcal{G}$ if and only if $X \rightarrow Y$ is covered in $\mathcal{G}$.*

As described above, when a Bayesian network has complete tables, the number of parameters is completely determined by its DAG and the number of states for each variable in the domain. To simplify presentation, we assume that the number of states for the variable corresponding to each vertex in a DAG is available implicitly, and therefore we can define the number of parameters associated with a DAG without reference to the corresponding state counts. In particular, we say that a DAG *supports* a number of parameters k when all Bayesian networks with that structure (defined over a particular domain) contain k parameters. The following result follows immediately from Lemma 1 for Bayesian networks with complete tables:

Lemma 2 (Chickering, 1995) *If $\mathcal{G}$ and $\mathcal{G}'$ are equivalent, then they support the same number of parameters.*

We say that DAG $\mathcal{H}$ *includes* DAG $\mathcal{G}$ if every distribution included by $\mathcal{G}$ is also included by $\mathcal{H}$. As above, an alternative but equivalent definition—due to the assumption of complete-table Bayesian networks—is that $\mathcal{H}$ includes $\mathcal{G}$ if every independence constraint implied by $\mathcal{H}$ is also implied by $\mathcal{G}$. Note that we are using "includes" to describe the relationship between a DAG and a particular distribution, as well as a relationship between two DAGs.

Theorem 3 (Chickering, 2002) *If $\mathcal{H}$ includes $\mathcal{G}$, then there exists a sequence of single edge additions and covered edge reversals in $\mathcal{G}$ such that (1) after each addition and reversal, $\mathcal{G}$ remains a DAG, (2) after each addition and reversal, $\mathcal{H}$ includes $\mathcal{G}$, and (3) after all additions and reversals, $\mathcal{G} = \mathcal{H}$.*

The "converse" of Theorem 3 will also prove useful.

Lemma 4 *If $\mathcal{F}$ can be transformed into $\mathcal{G}$ by a series of single edge additions and covered edge reversals, such that after each addition and reversal $\mathcal{F}$ remains a DAG, then $\mathcal{G}$ includes $\mathcal{F}$.*

Proof: Follows immediately from Lemma 1 and from the fact that the DAG that results from adding a single edge to $\mathcal{F}$ necessarily includes $\mathcal{F}$. □

3. Main Results

In this section, we provide the main results of this paper. We first define the decision problems that we use to prove that learning is NP-hard. As discussed in Section 1, in the limit of large data, all consistent scoring criteria rank network structures that include the generative distribution over those that do not, and among those structures that include the generative distribution, the criteria rank according to the number of parameters supported—with simpler structures receiving better scores. Thus, a natural decision problem corresponding to large-sample learning is the following:

LEARN
INSTANCE: Set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, probability distribution $p(\mathbf{X})$, and constant parameter bound d.
QUESTION: Does there exist a DAG that includes p and supports $\leq d$ parameters?

It is easy to see that if there exists an efficient algorithm for learning the optimal Bayesian-network structure from large-sample data, we can use that algorithm to solve LEARN: simply learn the best structure and evaluate the number of parameters it supports. By showing that LEARN is NP-hard, we therefore immediately conclude that the optimization problem of *identifying* the optimal DAG is hard as well. We show that LEARN is NP-hard using a reduction from a restricted version of the NP-complete problem FEEDBACK ARC SET. The general FEEDBACK ARC SET problem is stated by Garey and Johnson (1979) as follows:

FEEDBACK ARC SET
INSTANCE: Directed graph $\mathcal{G} = (\mathbf{V}, \mathbf{A})$, positive integer $k \leq |\mathbf{A}|$.
QUESTION: Is there a subset $\mathbf{A}' \subset \mathbf{A}$ with $|\mathbf{A}'| \leq k$ such that $\mathbf{A}'$ contains at least one arc from every directed cycle in $\mathcal{G}$?

Gavril (1977) shows that FEEDBACK ARC SET remains NP-complete for directed graphs in which no vertex has a total in-degree and out-degree more than three. We refer to this restricted version as DEGREE-BOUNDED FEEDBACK ARC SET, or *DBFAS* for short.

The remainder of this section is organized as follows. In Section 3.1, we describe a polynomial-time reduction from instances of DBFAS to instances of LEARN. In Section 3.2, we describe the main result of the appendices upon which Section 3.3 relies; in Section 3.3, we prove that there is a solution to an instance of DBFAS if and only if there is a solution to the instance of LEARN that results from the reduction, and therefore we establish that LEARN is NP-hard. In Section 3.4, we

extend our main result to the case when the learning algorithm has access to various oracles, and to the case when there is an upper bound on the number of parents for each node in the solution to LEARN.

For the remainder of this paper we assume—without loss of generality—that in any instance of DBFAS, no vertex has in-degree or out-degree of zero; if such a node exists, none of its incident edges can participate in a cycle, and we can remove that node from the graph without changing the solution.

3.1 A Reduction from DBFAS to LEARN

In this section, we show how to reduce an arbitrary instance of DBFAS into a corresponding instance of LEARN. To help distinguish between elements in the instance of DBFAS and elements in the instance of LEARN, we will subscript the corresponding symbols with D and L, respectively. In particular, we use $\mathcal{G}_D = (\mathbf{V}_D, \mathbf{A}_D)$ and k_D to denote the graph and arc-set bound, respectively, from the instance of DBFAS; from this instance, we create an instance of LEARN consisting of a set of variables $\mathbf{X}_L$, a probability distribution $p_L(\mathbf{X}_L)$, and a parameter bound d_L.

For each $V_i \in \mathbf{V}_D$ in the instance of DBFAS, we create a corresponding nine-state discrete variable V_i for $\mathbf{X}_L$. For each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ in the instance of DBFAS, we create seven discrete variables for $\mathbf{X}_L$: $A_{ij}, B_{ij}, C_{ij}, D_{ij}, E_{ij}, F_{ij}, G_{ij}$. Variables A_{ij}, D_{ij} and G_{ij} have nine states, variables B_{ij}, E_{ij} and F_{ij} have two states, and variable C_{ij} has three states. There are no other variables in $\mathbf{X}_L$ for the instance of LEARN. The probability distribution $p_L(\mathbf{X}_L)$ for the instance of LEARN is specified using a Bayesian network $(\mathcal{H}_L, \theta_{\mathcal{H}_L})$. The model is defined over the variables in $\mathbf{X}_L$, along with, for each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ from the instance of DBFAS, a single "hidden" binary variable H_{ij}. Let $\mathbf{H}_L$ denote the set of all such hidden variables. The distribution $p_L(\mathbf{X}_L)$ is defined by summing the distribution $p_L(\mathbf{H}_L, \mathbf{X}_L)$, defined by $(\mathcal{H}_L, \theta_{\mathcal{H}_L})$, over all of the variables in $\mathbf{H}_L$. The structure $\mathcal{H}_L$ is defined as follows. For each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ in the instance of DBFAS, the DAG contains the edges shown in Figure 1. The number of states for each node in the figure is specified in parentheses below the node.

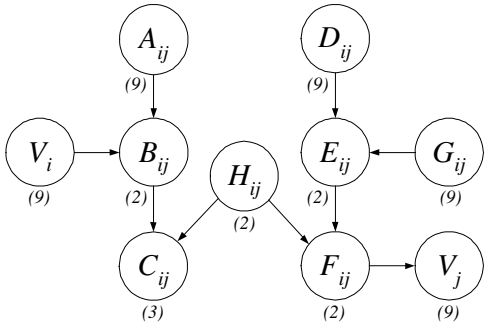
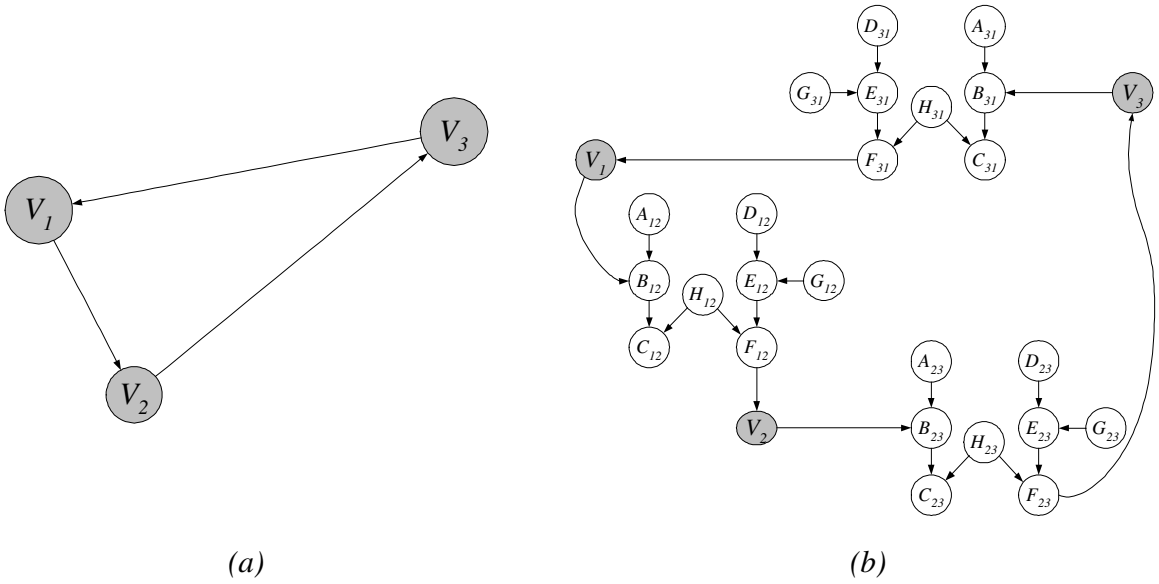


Figure 1: Edges in $\mathcal{H}_L$ corresponding to each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ from the instance of DBFAS. The number of states for each node is given in parentheses below the node.

For an example, Figure 2a shows an instance of DBFAS, and Figure 2b shows the resulting structure of $\mathcal{H}_L$.



(a) *(b)*

Figure 2: An example of the structure $\mathcal{H}_L$ that results from the reduction from a specific instance of DBFAS: (a) an instance of DBFAS consisting of three nodes V_1, V_2 and V_3 and (b) the corresponding structure $\mathcal{H}_L$.

We now specify the local probability distributions in $\theta_{\mathcal{H}_L}$. Let r_X denote the number of states of X, let $\mathbf{Pa}_X$ denote the set of parents of X in $\mathcal{H}_L$, and let $NNZ(\mathbf{pa}_X)$ denote the number of values in $\mathbf{pa}_X$ that are *not* equal to zero. Then for each node X in $\mathcal{H}_L$, the local probability distribution for X is defined as follows:

$$p(X = x|\mathbf{Pa}_X = \mathbf{pa}_X) = \begin{cases} \frac{1}{16} & \text{if } x = 0 \text{ and } NNZ(\mathbf{pa}_X) = 0 \\ \frac{1}{(r_X-1)}\frac{15}{16} & \text{if } x \neq 0 \text{ and } NNZ(\mathbf{pa}_X) = 0 \\ \frac{1}{64} & \text{if } x = 0 \text{ and } NNZ(\mathbf{pa}_X) = 1 \\ \frac{1}{(r_X-1)}\frac{63}{64} & \text{if } x \neq 0 \text{ and } NNZ(\mathbf{pa}_X) = 1 \\ \frac{1}{128} & \text{if } x = 0 \text{ and } NNZ(\mathbf{pa}_X) = 2 \\ \frac{1}{(r_X-1)}\frac{127}{128} & \text{if } x \neq 0 \text{ and } NNZ(\mathbf{pa}_X) = 2. \end{cases} \quad (1)$$

Because each node in $\mathcal{H}_L$ has at most two parents, the above conditions define every local distribution in $\theta_{\mathcal{H}_L}$.

Finally, we define the constant d_L in the instance of LEARN. Every node in $\mathcal{G}_D$ has either exactly one or exactly two parents because, in any instance of DBFAS, the total degree of each node is at most three and by assumption no node has an in-degree or an out-degree of zero. Let t_D denote the number of nodes in $\mathcal{G}_D$ from the instance of DBFAS that have exactly two in-coming edges; similarly, let $o_D = |\mathbf{V}_D| - t_D$ be the number of nodes that have exactly one in-coming edge. Then

we have

$$d_L = 186|\mathbf{A}_D| + 18k_D + 16(|\mathbf{A}_D| - k_D) + 16o_D + 32t_D. \quad (2)$$

We now argue that the reduction is polynomial. It is easy to see that we can specify the structure $\mathcal{H}_L$ and the bound d_L in polynomial time; we now argue that we can specify all of the parameter values $\theta_{\mathcal{H}_L}$ in polynomial time as well. Because each node in $\mathcal{H}_L$ has at most two parents, each corresponding conditional-probability table contains a constant number of parameters. Thus, as long as each parameter is represented using number of bits that is polynomial in the size of the instance of DBFAS, the parameters $\theta_{\mathcal{H}_L}$ can be written down in polynomial time. Each node has either two, three, or nine states, and thus it follows from the specification of $p(X = x|\mathbf{Pa}_X = \mathbf{pa}_X)$ in Equation 1 that each parameter is a fraction whose denominator is a power of two that can never exceed 1024 (i.e., $(9-1) \times 128$). Thus, when using a straight-forward binary representation for the parameter values, we can represent each such value exactly using at most ten (i.e., $\log_2 1024$) bits. Thus we conclude that the entire reduction is polynomial.

3.2 Specifying a Perfect Distribution

In our reduction from DBFAS to LEARN in the previous section, we specified the probability distribution $p_L(\mathbf{X}_L)$ using the Bayesian network $(\mathcal{H}_L, \theta_{\mathcal{H}_L})$. As we shall see in Section 3.3, our proof that LEARN is NP-hard requires that the distribution $p_L(\mathbf{H}_L, \mathbf{X}_L)$ is perfect with respect to the structure $\mathcal{H}_L$. In this section, we discuss the result from the appendices that guarantees that the local distributions defined by Equation 1 lead to an appropriate joint distribution.

Our results on perfectness are closely related to work on qualitative belief networks (QBNs), which are studied by (e.g.) Wellman (1990) and Druzdzel and Henrion (1993). In the appendices, we consider two properties of local probability distributions: one is related to the positive-influence property of QBNs, and the other is related to the positive-product-synergy property of QBNs. For a rigorous definition of these QBN concepts, see Druzdzel and Henrion (1993). Roughly speaking, a distribution has the positive-influence property if observing higher values of a parent node cannot decrease the probability of observing higher values of the target node when all other parent values are fixed. The positive-product-synergy property dictates how changes in the values for a *pair* of parents affects the probability of the target node, and is closely related to the function property *multivariate total positivity of order two* in the mathematics community (see Karlin and Rinott, 1980). The two QBN properties impose *non-strict* inequality constraints. For example, if the local distribution for a node Y has the positive-influence property, then increasing the value of one of its parents does not necessarily increase the probability of Y; it is instead constrained to not decrease. The positive-product-synergy property imposes an analogous non-strict inequality constraint.

In the appendices, we define strict versions of the QBN properties for a special class of distributions. The main result of the appendices (Lemma 17) is that for any Bayesian network in which each local distribution has both of our properties, the joint distribution is necessarily perfect with respect to the network structure. The following result provides a prescription for constructing distributions for which both our properties hold:

Lemma 5 *Let $(\mathcal{G}, \theta_{\mathcal{G}})$ be a Bayesian network, let r_X denote the number of states of node X, let $\mathbf{Pa}_X$ denote the set of parents of node X in $\mathcal{G}$, let NNZ$(\mathbf{pa}_X)$ denote the number of non-zero elements in the set $\mathbf{pa}_X$, and let α_X be a constant satisfying $0 < \alpha_X < 1$. If all of the local distributions in $\theta_{\mathcal{G}}$*

are defined as

$$p(X = x|\mathbf{Pa}_X = \mathbf{pa}_X) = \begin{cases} \alpha_X^{F(\mathbf{pa}_X)} & \text{if } x = 0 \\ \frac{1}{(r_X-1)}\left(1-\alpha_X^{F(\mathbf{pa}_X)}\right) & \text{otherwise,} \end{cases} \quad (3)$$

where

$$F(\mathbf{pa}_X) = 2 - \frac{1}{2}^{NNZ(\mathbf{pa}_X)},$$

then the distribution defined by $(\mathcal{G}, \theta_{\mathcal{G}})$ *is perfect with respect to* $\mathcal{G}$.

The local distributions defined by Equation 1 are simply specializations of Equation 3 where $\alpha_X = \frac{1}{16}$ for every X. Thus, the following corollary follows immediately from Lemma 5:

Corollary 6 *The distribution* $p_L(\mathbf{H}_L, \mathbf{X}_L)$ *resulting from the reduction is perfect with respect to* $\mathcal{H}_L$.

3.3 Reduction Proofs

In this section, we prove LEARN is NP-hard by demonstrating that there is a solution to the instance of DBFAS if and only if there is a solution to the instance of LEARN that results from the reduction. In the results that follow, we often consider sub-graphs of solutions to LEARN that correspond only to those nodes that are "relevant" to a particular arc in the instance of DBFAS. Therefore, to simplify the discussion, we use $\{V_i, V_j\}$ *edge component* to refer to a sub-graph defined by the nodes $\{V_i, A_{ij}, B_{ij}, C_{ij}, D_{ij}, E_{ij}, F_{ij}, G_{ij}, V_j\}$. We use *edge component* without reference to a particular V_i and V_j when an explicit reference is not necessary. Figure 3, which is key to the results that follow, shows two configurations of the edges in an edge component.

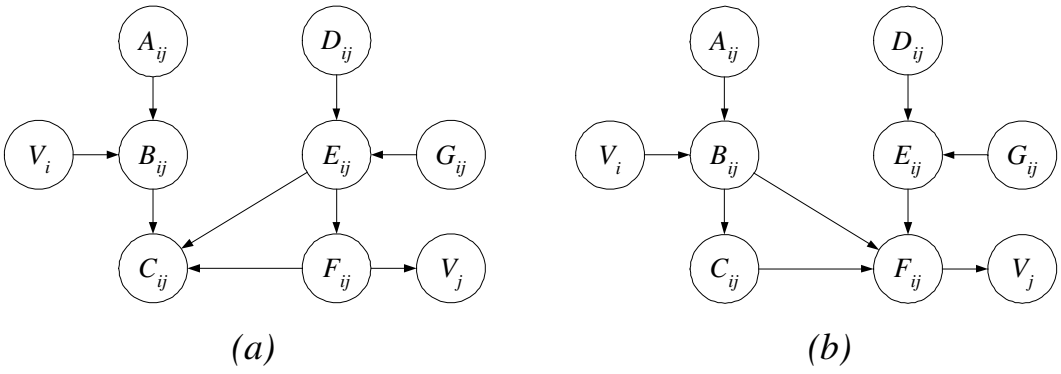


Figure 3: Two configurations of the edges in an edge component.

We first prove a preliminary result that is used in both of the main proofs of this section. Recall that $\mathcal{H}_L$ contains an additional "hidden" node H_{ij} within each edge component. We will be considering active paths in $\mathcal{H}_L$, but are only concerned about those in which the endpoints are in $\mathbf{X}_L$ and for which no H_{ij} is in the conditioning set; these active paths correspond to dependencies that exist within the (marginalized) distribution $p_L(\mathbf{X}_L)$. To simplify presentation, we define a $\mathbf{X}_L$-*restricted* active path to denote such an active path. In this and later results, we will demonstrate

that one DAG $\mathcal{F}_1$ includes another DAG $\mathcal{F}_2$ by showing that for any active path in $\mathcal{F}_2$, there exists a corresponding (i.e., same endpoints and same conditioning set) active path in $\mathcal{F}_1$.

Lemma 7 *Let $p_L(\mathbf{X}_L)$ be the distribution defined for the instance of LEARN in the reduction, and let $\mathcal{F}$ be any DAG defined over $\mathbf{X}_L$ such that each edge component in $\mathcal{F}$ contains the edges in either Figure 3a or in Figure 3b. Then $\mathcal{F}$ includes $p_L(\mathbf{X}_L)$.*

Proof: Let $\mathcal{H}_L$ be the DAG defining $p_L(\mathbf{X}_L)$ in the reduction. We prove that $\mathcal{F}$ includes $p_L(\mathbf{X}_L)$ by demonstrating that for every $\mathbf{X}_L$-restricted active path in $\mathcal{H}_L$, there exists a corresponding active path in $\mathcal{F}$. To do this, we construct an additional model $\mathcal{H}'$ that includes $\mathcal{H}_L$—and consequently $\mathcal{H}'$ can represent $p_L(\mathbf{X}_L)$ exactly—such that $\mathbf{X}_L$-restricted active paths in $\mathcal{H}'$ are easily mapped to their corresponding active paths in $\mathcal{F}$.

We create $\mathcal{H}'$ from $\mathcal{H}_L$ as follows. For each i and j, if the edge component in $\mathcal{F}$ is in the configuration shown in Figure 3a, we add the edge $E_{ij} \rightarrow H_{ij}$ to $\mathcal{H}$ and then reverse the (now covered) edge $H_{ij} \rightarrow F_{ij}$. Similarly, if the edge component in $\mathcal{F}$ is in the configuration shown in Figure 3b, we add the edge $B_{ij} \rightarrow H_{ij}$ to $\mathcal{H}$ and then reverse the edge $H_{ij} \rightarrow C_{ij}$. The resulting components in $\mathcal{H}'$ are shown in Figure 4a and Figure 4b, respectively. Because we created $\mathcal{H}'$ by

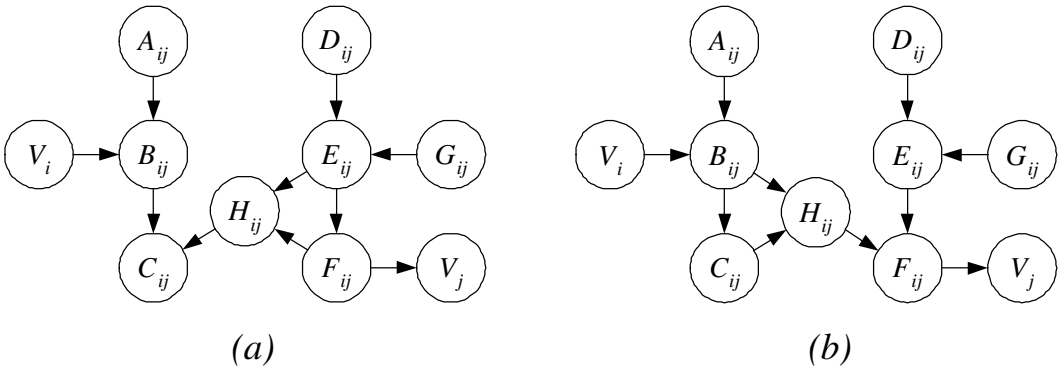


Figure 4: Edges in $\mathcal{H}'$ corresponding to the edge components in Figure 3

edge additions and covered edge reversals, we know by Lemma 4 that $\mathcal{H}'$ includes $\mathcal{H}_L$. It is now easy to see that any $\mathbf{X}_L$-restricted active path in $\mathcal{H}'$ has a corresponding active path in $\mathcal{F}$: simply replace any segment $X \rightarrow H_{ij} \rightarrow Y$ in the path by the corresponding edge $X \rightarrow Y$ from $\mathcal{F}$, and the resulting path will be active in $\mathcal{F}$. □

Theorem 8 *If there is a solution $\mathbf{A}'_D$ to the given instance of DBFAS with $|\mathbf{A}'_D| \leq k_D$, then there is a solution $\mathcal{F}_L$ to the instance of LEARN with $\leq d_L$ parameters.*

Proof: We create a solution DAG $\mathcal{F}_L$ as follows. For every arc $V_i \rightarrow V_j \in \mathbf{A}'_D$ in the DBFAS solution, $\mathcal{F}_L$ contains the edges shown in Figure 3a. For the remaining arcs $V_i \rightarrow V_j$ that are not in $\mathbf{A}'_D$, $\mathcal{F}_L$ contains the edges shown in Figure 3b. $\mathcal{F}_L$ contains no other edges. First we argue that $\mathcal{F}_L$ is acyclic. Each $\{V_i, V_j\}$ edge component in $\mathcal{F}_L$ is itself acyclic, and contains a directed path from V_i to V_j if and only if the corresponding arc $V_i \rightarrow V_j \in \mathbf{A}_D$ from the instance of DBFAS is not in $\mathbf{A}'_D$; if the corresponding arc from the instance of DBFAS is in $\mathbf{A}'_D$, $\mathcal{F}_L$ contains neither

a directed path from V_i to V_j, nor a directed path from V_j to V_i that is contained within the edge component. Therefore, for any hypothetical cycle in $\mathcal{F}_L$, there would be a corresponding cycle in $\mathcal{G}_D$ that passed entirely through arcs not in $\mathbf{A}'_D$, which is impossible assuming $\mathbf{A}'_D$ is a solution to DBFAS. From Lemma 7, we know that $\mathcal{F}_L$ includes $p_L(\mathbf{X}_L)$. Now we derive the number of parameters supported by $\mathcal{F}_L$. Within each edge component, the parents for A_{ij}, B_{ij}, D_{ij}, E_{ij} and G_{ij} are the same regardless of whether or not the arc is in $\mathbf{A}'_D$; it is easy to verify that for each edge component, the local distributions for these nodes contribute a total of 186 parameters. For each arc $V_i \rightarrow V_j \in \mathbf{A}'_D$, the corresponding nodes C_{ij} and F_{ij} contribute a total of $16+2=18$ parameters; for each arc $V_i \rightarrow V_j \notin \mathbf{A}'_D$, the nodes C_{ij} and F_{ij} contribute a total of $4+12=16$ parameters. For every node $V_i \in \mathbf{V}_D$ in the instance of DBFAS that has exactly two parents, the corresponding $V_i \in \mathbf{X}_L$ in the instance of LEARN will also have two parents. Similarly, for every node $V_i \in \mathbf{V}_D$ with exactly one parent, the corresponding $V_i \in \mathbf{X}_L$ has exactly one parent. By construction of $\mathcal{F}_L$, every parent node for any $V_i \in \mathbf{X}_L$ has two states (and is equal F_{ji} for some j), and therefore because each node $V_i \in \mathbf{X}_L$ has nine states, the total number of parameters used in the local distributions for these nodes is $16o_D + 32t_D$. Thus, we conclude that the number of parameters in $\mathcal{F}$ is exactly

$$186|\mathbf{A}_D| + 18|\mathbf{A}'_D| + 16(|\mathbf{A}_D| - |\mathbf{A}'_D|) + 16o_D + 32t_D.$$

Because $|\mathbf{A}'_D| \leq k_D$, we conclude from Equation 2 that the number of parameters in $\mathcal{F}_L$ is less than or equal to d_L, and thus $\mathcal{F}_L$ is a valid solution to the instance of LEARN. □

Theorem 9 *If there is a solution $\mathcal{F}_L$ to the instance of LEARN with $\leq d_L$ parameters, then there is a solution to the given instance of DBFAS with $|\mathbf{A}'_D| \leq k_D$.*

Proof: Given the solution $\mathcal{F}_L$, we create a new solution $\mathcal{F}_L{}'$ as follows. For every pair (V_i, V_j) corresponding to an edge $V_i \rightarrow V_j \in \mathbf{A}_D$ in the instance of DBFAS, if there is no directed path in $\mathcal{F}_L$ from V_i to V_j, then the corresponding edge component in $\mathcal{F}_L{}'$ contains the edges shown in Figure 3a. Otherwise, when there is at least one directed path in $\mathcal{F}_L$ from V_i to V_j, the corresponding edge component in $\mathcal{F}_L{}'$ contains the edges shown in Figure 3b. By construction, $\mathcal{F}_L{}'$ will contain a cycle only if $\mathcal{F}_L$ contains a cycle, and consequently we conclude that $\mathcal{F}_L{}'$ is a DAG. From Lemma 7, we know that $\mathcal{F}_L{}'$ includes $p_L(\mathbf{X}_L)$.

In the next two paragraphs, we argue that $\mathcal{F}_L{}'$ does not support more parameters than does $\mathcal{F}_L$. Consider the DAG $\mathcal{F}^0$ that is identical to $\mathcal{F}_L{}'$, except that for all i and j, the only parent of C_{ij} is B_{ij} and the only parent of F_{ij} is E_{ij} (see Figure 5). Because $\mathcal{F}^0$ is a subgraph of $\mathcal{F}_L{}'$, any active

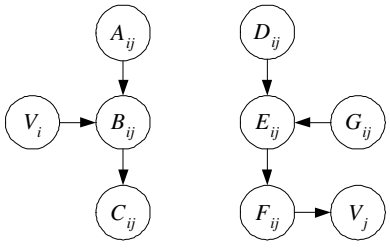


Figure 5: Edges within each edge component of $\mathcal{F}^0$

path in $\mathcal{F}^0$ must have a corresponding active path in $\mathcal{F}_L{}'$, and thus we conclude that $\mathcal{F}_L{}'$ includes

$\mathcal{F}^0$. The original solution $\mathcal{F}_L$ also includes $\mathcal{F}^0$ by the following argument: $\mathcal{F}^0$ is a strict sub-graph of $\mathcal{H}_L$ ($\mathcal{F}^0$ contains a subset of the edges and no H_{ij} nodes), and thus any active path in $\mathcal{F}^0$ has a corresponding $\mathbf{X}_L$-restricted active path in $\mathcal{H}_L$; because $\mathcal{H}_L$ is perfect with respect to the distribution $p_L(\mathbf{H}_L, \mathbf{X}_L)$ defined by $(\mathcal{H}_L, \theta_{\mathcal{H}_L})$ (Corollary 6), we know that any such $\mathbf{X}_L$-restricted active path in $\mathcal{H}_L$ corresponds to a dependence in $p_L(\mathbf{X}_L)$, and thus, because $\mathcal{F}_L$ includes $p_L(\mathbf{X}_L)$, there must be a corresponding active path in $\mathcal{F}_L$.

From Theorem 3, we know that there exists a sequence of edge additions and covered edge reversals that transforms $\mathcal{F}^0$ into $\mathcal{F}_L$, and another sequence of edge additions and covered edge reversals that transforms $\mathcal{F}^0$ into $\mathcal{F}_L{}'$. From Lemma 1 and Lemma 2, a covered edge reversal does not change the number of parameters supported by a DAG, and thus we can compare the number of parameters supported by the two DAGs by evaluating the increase in parameters that result from the additions within each of the two transformations. $\mathcal{F}^0$ can be transformed into $\mathcal{F}_L{}'$ by simply adding, for each edge component, the corresponding two extra edges in $\mathcal{F}_L{}'$. That is, we either (1) add the edges $E_{ij} \rightarrow C_{ij}$ and $F_{ij} \rightarrow C_{ij}$, resulting in an increase of 12 parameters, or (2) add the edges $B_{ij} \rightarrow F_{ij}$ and $C_{ij} \rightarrow F_{ij}$, resulting in an increase of 10 parameters. If $\mathcal{F}_L$ supports fewer parameters than $\mathcal{F}_L{}'$, there must be at least one $\{V_i, V_j\}$ edge component for which the total parameter increase from adding edges between nodes in that component is less than the corresponding increase in $\mathcal{F}_L{}'$. In order to reverse any edge in an edge component from $\mathcal{F}^0$, we need to first cover that edge by adding at least one other edge that is contained in that component; it is easy to verify that any such "covering addition" results in an increase of at least 16 parameters (adding $E_{ij} \rightarrow V_j$ results in this increase, and all other additions result in a larger increase). Thus we conclude that for the $\{V_i, V_j\}$ edge component, only edge additions are performed in the transformation from $\mathcal{F}^0$ to $\mathcal{F}_L$. H_{ij} does not exist in $\mathcal{F}_L$, and therefore because $p_L(\mathbf{H}_L, \mathbf{X}_L)$ is a DAG-perfect distribution (Corollary 6), C_{ij} and F_{ij} cannot be conditionally independent given any other nodes in $\mathbf{X}_L$; thus, in order for $\mathcal{F}_L$ to include $p_L(\mathbf{X}_L)$, there must be an edge between C_{ij} and F_{ij} in $\mathcal{F}_L$. We consider two cases, corresponding to the two possible directions of the edge between C_{ij} and F_{ij} in $\mathcal{F}_L$. If the edge is directed as $C_{ij} \rightarrow F_{ij}$, we know that there is a directed path between V_i and V_j in $\mathcal{F}_L$ because none of the edges from $\mathcal{F}^0$ can be reversed. By construction of $\mathcal{F}_L{}'$, this implies that the increase in parameters supported by $\mathcal{F}_L{}'$ attributed to this edge component is 10. In $\mathcal{H}_L$, F_{ij} and B_{ij} are d-connected given any conditioning set from $\mathbf{X}_L$ that contains C_{ij} (see Figure 1), and thus we know that $F_{ij} \not\perp\!\!\!\perp_{p_L} B_{ij} | \mathbf{S}$ for any $\mathbf{S} \subset \mathbf{X}_L$ that contains C_{ij}; this implies that the edge $B_{ij} \rightarrow F_{ij}$ must exist in $\mathcal{F}_L$, else we could find a conditioning set $\mathbf{S}$ that contains C_{ij} for which $F_{ij} \perp\!\!\!\perp_{\mathcal{F}_L} B_{ij} | \mathbf{S}$, which contradicts the fact that $\mathcal{F}_L$ includes $p_L(\mathbf{X}_L)$. But adding both $C_{ij} \rightarrow F_{ij}$ and $B_{ij} \rightarrow F_{ij}$ to $\mathcal{F}^0$ requires an addition of *at least* 10 parameters, contradicting the supposition that the parameter increase due to this edge component is smaller in $\mathcal{F}_L$ than in $\mathcal{F}_L{}'$. If the edge between C_{ij} and F_{ij} is directed as $F_{ij} \rightarrow C_{ij}$, we know that $\mathcal{F}_L$ must also contain the edge $E_{ij} \rightarrow C_{ij}$, lest (using the same logic as above) there would exist some conditioning set $\mathbf{S}$ containing F_{ij} such that $C_{ij} \perp\!\!\!\perp_{\mathcal{F}_L} E_{ij} | \mathbf{S}$ but $C_{ij} \not\perp\!\!\!\perp_{p_L} E_{ij} | \mathbf{S}$, contradicting the fact that $\mathcal{F}_L$ includes $p_L(\mathbf{X}_L)$. Adding both of these edges, however, requires an addition of *at least* 12 parameters; because the corresponding edge component in $\mathcal{F}_L{}'$ attributed *at most* 12 parameters in the transformation from $\mathcal{F}_L{}'$, this again contradicts the supposition that the parameter increase due to this edge component is smaller in $\mathcal{F}_L$ than in $\mathcal{F}_L{}'$.

Having established that $\mathcal{F}_L{}'$ is a solution to LEARN that supports fewer parameters than $\mathcal{F}_L$, we now use $\mathcal{F}_L{}'$ to construct a solution $\mathbf{A}'_D$ to the instance of DBFAS. For each $\{V_i, V_j\}$ edge

component in ${\mathcal{F}_L}'$, if that component contains the edges shown in Figure 3a, then we include in $\mathbf{A}'_D$ the arc $V_i \rightarrow V_j$. $\mathbf{A}'_D$ contains no other arcs.

We now argue that $\mathbf{A}'_D$ contains at least one arc from every cycle from the instance of DBFAS. Each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ that is *not* contained in $\mathbf{A}'_D$ has a corresponding edge component in ${\mathcal{F}_L}'$ for which there is a directed path from V_i to V_j. Thus, any hypothetical cycle in the instance of DBFAS that does not pass through an edge in $\mathbf{A}'_D$ has a corresponding directed cycle in ${\mathcal{F}_L}'$, which is impossible because ${\mathcal{F}_L}'$ is a DAG.

Finally, we argue that $\mathbf{A}'_D$ contains at most k_D arcs. Recall that o_D and t_D denote the number of nodes in $\mathcal{G}_D$ that have exactly one and two in-coming edges, respectively. As in the proof of Theorem 8, it is easy to verify that the number of parameters d'_L supported by ${\mathcal{F}_L}'$ is exactly

$$186|\mathbf{A}_D| + 18|\mathbf{A}'_D| + 16(|\mathbf{A}_D| - |\mathbf{A}'_D|) + 16o_D + 32t_D.$$

Given that $d'_L \leq d_L$, we conclude from Equation 2 that $|\mathbf{A}'_D| \leq k_D$. □

Given the previous results, the main result of this paper now follows easily.

Theorem 10 *LEARN is NP-hard.*

Proof: Follows immediately from Theorem 8 and Theorem 9. □

Also, due to the fact that the distribution in the reduction is obtained by marginalizing out the hidden variables in a DAG-perfect distribution, the following result is immediate.

Corollary 11 *LEARN remains NP-hard when we restrict the input probability distribution to be the marginalization of a DAG-perfect distribution.*

3.4 Extensions

Many approaches to learning Bayesian networks from data use independence tests or mutual-information calculations to help guide a search algorithm. In this section, we show that even if such tests and calculations could be obtained in constant time, the search problem remains hard. In particular, we show that Theorem 10 holds even when the learning algorithm has access to at least one of three oracles. Furthermore, we show that the problem remains hard when we restrict ourselves to considering only those solutions to LEARN for which each node has at most k parents, for all $k \geq 3$.

The first oracle we consider is an independence oracle. This oracle can evaluate independence queries in constant time.

Definition 12 (Independence Oracle)
An independence oracle *for a distribution $p(\mathbf{X})$ is an oracle that, in constant time, can determine whether or not $X \perp\!\!\!\perp_p Y|\mathbf{Z}$ for any X and Y in $\mathbf{X}$ and for any $\mathbf{Z} \subseteq \mathbf{X}$.*

The second oracle we consider can perform certain inference queries in constant time; namely, the inference oracle can return the joint probability of any constant-sized set of variables. This oracle can in turn be used to compute conditional probabilities in constant time using division.

Definition 13 (Constrained Inference Oracle)
A constrained inference oracle *for a distribution $p(\mathbf{X})$ is an oracle that, in constant time, can compute $p(\mathbf{Z} = \mathbf{z})$ for any $\mathbf{Z} \subseteq \mathbf{X}$ such that $|\mathbf{Z}| \leq k$ for some constant k.*

Some learning algorithms use mutual information—or an approximation of mutual information—from a distribution to help construct model structures. The *(conditional mutual) information* between variables X and Y given the set of variables $\mathbf{Z}$ is defined as

$$Inf(X;Y|\mathbf{Z}) = \sum_{x,y,\mathbf{z}} p(x,y,\mathbf{z}) \log \frac{p(x,y|\mathbf{z})}{p(x|\mathbf{z})p(y|\mathbf{z})}. \tag{4}$$

The third oracle we consider can compute the mutual information between two variable in constant time, given that there are only a constant number of variables in the conditioning set.

Definition 14 (Constrained Information Oracle)
A constrained information oracle *for a distribution $p(\mathbf{X})$ is an oracle that, in constant time, can compute $Inf(X;Y|\mathbf{Z})$ for any X and Y in $\mathbf{X}$ and for any $\mathbf{Z} \subseteq \mathbf{X}$ such that $|\mathbf{Z}| \leq k$ for some constant k.*

Theorem 15 *Theorem 10 holds even when the learning algorithm has access to (1) an independence oracle, (2) a constrained inference oracle, or (3) a constrained information oracle.*

Proof: We establish this result by demonstrating that we can implement all three of these oracles in polynomial time using the Bayesian network $(\mathcal{H}, \theta_{\mathcal{H}})$ from our reduction. Thus if LEARN can be solved in polynomial time when we have access to any of the constant-time oracles, it must also be solvable in polynomial time *without* any such oracle.

(1) holds immediately because we can test for d-separation in $\mathcal{H}$ in polynomial time. (3) follows from (2) because, given that each variable has some constant number of states, we can implement a constrained information oracle via Equation 4 by calling a constrained inference oracle a constant number of times.

Let $\mathbf{Z} \subseteq \mathbf{X}$ be any subset of the variables such that $|\mathbf{Z}| \leq k$ for some constant k. It remains to be shown how to compute $p(\mathbf{Z} = \mathbf{z})$ in polynomial time from $(\mathcal{H}, \theta_{\mathcal{H}})$. The trick is to see that there is always a cut-set of constant size that decomposes $\mathcal{H}$ into a set of polytrees, where each polytree has a constant number of nodes; within any polytree containing a constant number of nodes, we can perform inference in constant time. We define a cut-set $\mathbf{B}$ as follows: $\mathbf{B}$ contains every node B_{ij} for which (1) C_{ij} is in $\mathbf{Z}$ and (2) B_{ij} is not in $\mathbf{Z}$. Note that $\mathbf{B} \cap \mathbf{Z} = \emptyset$. Given conditioning set $\mathbf{B}$, no active path can contain a node C_{ij} as an interior (i.e., non-endpoint) node, even when any subset of $\mathbf{Z}$ is added to the conditioning set (see Figure 6): any such hypothetical active path must pass through at least one segment $B_{ij} \rightarrow C_{ij} \leftarrow H_{ij}$. But this is not possible, because every such segment is blocked: if C_{ij} is not in $\mathbf{Z}$, then the segment is blocked because C_{ij} has no descendants, and hence can have no descendants in the conditioning set; if C_{ij} is in $\mathbf{Z}$, then we know that $B_{ij} \in \mathbf{B}$ and thus the segment is blocked by B_{ij}.

Because no active path can pass through a node C_{ij}, it follows by construction of $\mathcal{H}$ that—given $\mathbf{B}$ and any subset of $\mathbf{Z}$—each node in $\mathbf{Z}$ is d-connected to only a constant number of other nodes in $\mathbf{Z}$. Furthermore, the structure of $\mathcal{H}$ that is bounded between the C_{ij} nodes forms a polytree. Thus, we can express $p(\mathbf{Z} = \mathbf{z})$ as

$$\begin{aligned} p(\mathbf{Z} = \mathbf{z}) &= \sum_{\mathbf{b}} p(\mathbf{Z} = \mathbf{z}, \mathbf{B} = \mathbf{b}) \\ &= \sum_{\mathbf{b}} \prod_{i} p(\mathbf{T}_i = \mathbf{t}_i(\mathbf{z}, \mathbf{b})), \end{aligned}$$

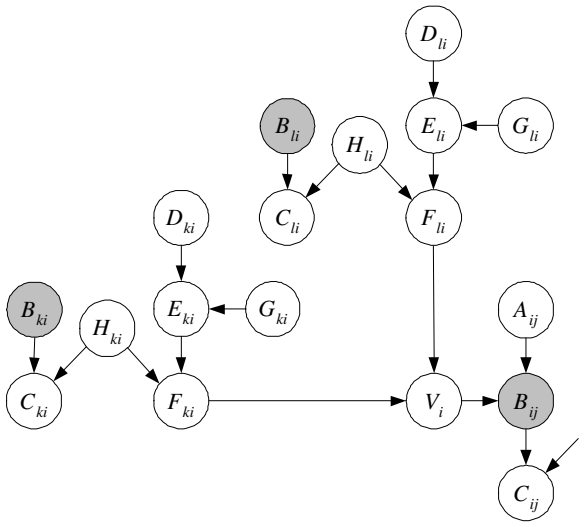


Figure 6: Portion of $\mathcal{H}$ showing that no active path can pass through any C_{ij} once B_{ij} is given.

where each $\mathbf{T}_i$ contains a constant number of variables—$\mathbf{t}_i(\mathbf{z},\mathbf{b})$ is the set of values for those variables as determined by $\mathbf{z}$ and $\mathbf{b}$—that constitute a polytree in $\mathcal{H}$. Thus, each term $p(\mathbf{T}_i = \mathbf{t}_i(\mathbf{z},\mathbf{b}))$ above can be computed in constant time using inference in a polytree. Because there are at most k nodes in $\mathbf{Z}$, the set $\mathbf{B}$ can contain at most k nodes. Therefore, given that each node in $\mathbf{B}$ has at most r states, there are at most r^k terms in the sum above—where both r and k are constants—and we conclude that $p(\mathbf{Z})$ can be computed in polynomial time. □

Finally, we prove that if we restrict LEARN to solutions in which each node has at most k parents, the problem remains NP-hard for all $k \geq 3$.

Theorem 16 *Theorem 15 holds even when solutions to LEARN are restricted to DAGs in which each node has at most k parents, for all* $k \geq 3$.

Proof: The case where $k = 3$ follows immediately from the proof of Theorem 8, where the constructed solution to LEARN is a DAG in which each node has at most three parents, and from the proof of Theorem 9, where the given solution to LEARN is converted into a (better) solution in which each node has at most three parents. It is easy to see that these proofs remain valid under a less restrictive ($k > 3$) bound on the number of parents, and thus the theorem follows. □

4. Conclusion

In this paper, we demonstrated that the problem of identifying high-scoring DAGs from large datasets when using a consistent scoring criterion is NP-hard. Together with the result of Chickering (1996) that the non-asymptotic learning problem is NP-hard, our result implies that learning is hard regardless of the size of the data. There is an interesting gap in the present results. In particular, Chickering (1996) proved that finite-sample learning is NP-hard when each node is restricted to have at most two parents, whereas in this paper we proved that large-sample learning is NP-hard with a three-parent restriction. This leads to the question of whether or not large-sample learning is

NP-hard when we restrict to two parents; we believe that this problem is probably NP-hard, and is worth further investigation.

In practice, the large-sample learning problem actually requires scanning a dataset with a large number of samples, as opposed to accessing a compact representation of the generative distribution. We could alternatively have defined a learning problem in which there is an actual data set supplied; the problem with this approach is that in order to guarantee that we get the large-sample ranking of model structures, we will need the number of data points to be so large that the size of the problem instance is exponential in the number of variables in the domain. Our results have practical importance when it is reasonable to assume that (1) there is enough data such that the relative ranking of those DAGs considered by the learning algorithm is the same as in the large-sample limit, and (2) the number of records in the data is small enough that we can compute the score for candidate structures in a reasonable amount of time.

As discussed in Section 1, there exist assumptions about the generative distribution that lead to efficient large-sample learning algorithms. These assumptions are not likely to hold in most real-world scenarios, but the corresponding "correct" algorithms can work well even if the assumptions do not hold. An interesting line of research is to investigate alternative, weaker assumptions about the generative distribution that lead to efficient learning algorithms and guarantee large-sample correctness.

Appendix A. Introduction to Perfectness Proofs

As described in Section 3.2, in these appendices we define two properties of local distributions, and prove that as long as these properties hold for every local distribution in the network, the corresponding joint distribution is perfect with respect to the network structure. Lemma 5 follows immediately from our main result once we demonstrate that the two properties hold for the family of distributions defined by Equation 3.

The two properties that we define are *binary-like lattice* (BLL) and *binary-like totally strictly positive* (BLTSP). The "binary like" aspect of both of these properties refers to the fact that the distributions are defined such that we can treat each variable *as if* it only has two states: a "distinguished" state and an "other" state. As first mentioned in Section 3.2, the BLL property of a local distribution is similar to the positive-influence property found in the QBN literature; it specifies that the probability of a node being in its "distinguished" state necessarily increases when we change a single parent from the "other" state to the "distinguished" state. The difference between BLL and the positive-influence property is that BLL requires that the probability strictly increase, whereas the positive-influence property requires that the probability does not decrease. The BLTSP property of a local distribution is similar to the positive-synergy property in the QBN literature. The intuition behind this property is that it requires that the (BLL) influence of a parent strictly increases with the number of other parents that are in the "distinguished" state. The difference between BLTSP and positive synergy is, as above, the requirement of a strict inequality.

Our main result demonstrates that if all local distributions in a Bayesian network are both BLL and BLTSP, then any active path corresponds to a dependence in the joint probability distribution defined by that network:

Lemma 17 *Let $(\mathcal{G}, \theta)$ be a Bayesian network in which all local distributions defined by θ are both BLL and BLTSP. Then the joint distribution represented by $(\mathcal{G}, \theta)$ is perfect with respect to $\mathcal{G}$.*

The proof of Lemma 17 is non-trivial, but the main technique can be understood as follows. We prove perfectness by demonstrating that for any active path between two nodes X and Y, there is a corresponding dependence in the joint distribution whenever the nodes in the conditioning set are all in their distinguished states. If X and Y are adjacent in $\mathcal{G}$ and if there are no nodes in the conditioning set, this dependence follows easily from the definition of BLL. We establish the general result by induction, using the simple case as the basis. In general, we show how to apply graph transformations that result in a simpler model for which our induction step applies. Each graph transformation is defined such that the original distribution over the non-observed nodes, when conditioned on the observed nodes, is represented exactly, and where every local distribution retains the BLL property; the BLTSP property is required in the original distributions to guarantee that the BLL property is retained as a result of each transformation.

The appendices are organized as follows. In Appendix B, we describe graph-transformation methods that can be applied to a Bayesian network. In Appendix C, we rigorously define BLL and BLTSP, and we demonstrate that the transformations described in Appendix B necessarily maintain the BLL property on every distribution. Finally, in Appendix D, we prove Lemma 17.

To simplify notation, we use $\mathcal{G}$ to denote a Bayesian network (as opposed to just the structure of that network) for the remainder of the paper, and we leave the parameter values θ implicit.

Appendix B. Graph Transformations

In this section, we describe a number of transformations that we apply to a Bayesian network in order to more easily prove our main result. For the remainder of the paper, we will assume that the domain of interest $\mathbf{V} = \{V_1, \ldots, V_n\}$ is decomposed into two sets of variables: $\mathbf{O}$ is the set of observed variables for which we are given a corresponding set of states $\mathbf{o}$, and $\mathbf{U}$ is the set of unobserved variables. In contrast to the "hidden" variables $\mathbf{H}_L$ described in Section 3.1, the unobserved variables $\mathbf{U}$ simply correspond to variables that are not in the particular conditioning set $\mathbf{O}$.

Given a Bayesian network $\mathcal{G}$ defined over the domain $\mathbf{O} \cup \mathbf{U}$, each transformation outputs a new Bayesian network $\mathcal{G}^T$. We will use $p(\cdot)$ and $p^T(\cdot)$ to denote the probability distributions defined by $\mathcal{G}$ and $\mathcal{G}^T$, respectively. As we see below, $\mathcal{G}^T$ may be defined over only a subset of the nodes in $\mathcal{G}$. Using $\mathbf{O}^T$ and $\mathbf{U}^T$ to denote the observed and unobserved nodes, respectively, that remain in $\mathcal{G}^T$, all of our transformations maintain the following invariant:

$$\forall \mathbf{u}^T \qquad p^T(\mathbf{u}^T | \mathbf{o}^T) = p(\mathbf{u}^T | \mathbf{o}). \tag{5}$$

Note that $\mathbf{o}$ and $\mathbf{o}^T$ are fixed (observed) values. In words, Equation 5 asserts that the distribution over the unobserved nodes that remain after the transformation is identical in the two models whenever we condition on the observed values.

B.1 Individual Transformations

There are five transformations that we use to prove our main result: edge deletion, edge reversal, node combination, barren node removal, and observed-child separation. For each transformation, there is both a structural change (e.g., an edge $X \rightarrow Y$ is added) and a corresponding change to the conditional distributions (e.g., the local distribution for Y is extended to include the new parent X). For the remainder of this paper, we assume that the local distributions in the model that result

from a transformation are obtained via inference from the original model. In the case where the parents of a node are identical in the two models, inference corresponds to copying the original local distribution.

Whenever the structure of the resulting model includes the original distribution, populating the local distributions via inference results in a new model that defines the original joint distribution. This follows because, by definition of inclusion, there exists a set of local distributions for the new model that yield the original joint distribution. Furthermore, these local distributions are unique (assuming that the original distribution is positive) and must match the corresponding conditional distributions from the original model; we use inference to ensure this match.

We say that a transformation is *valid* if (1) the preconditions of the transformation (e.g., there exists an edge $X \rightarrow Y$ in the model) are met and (2) the result of the transformation is an acyclic model. We now consider each transformation in turn.

B.1.1 EDGE DELETION

An *edge deletion* deletes an edge of the form $O \rightarrow Y$ from $\mathcal{G}$, where $O \in \mathbf{O}$ is an observed node, and replaces the local distribution in Y by the same local distribution except that the value o for O is fixed to its observed value (e.g.) o^0 (see Figure 7). Thus, if the parents of Y are $O \cup \mathbf{Z}$ in $\mathcal{G}$, then the new local distribution for Y in $\mathcal{G}^T$ is defined as

$$p^T(y|\mathbf{z}) = p(y|\mathbf{z}, o^0),$$

where the probability $p(y|\mathbf{z}, o^0)$ can be extracted directly from the local distribution of Y in $\mathcal{G}$. It is easy to see that for the resulting model $\mathcal{G}^T$ we have

$$\forall \mathbf{u} \qquad p^T(\mathbf{u}, \mathbf{o}) = p(\mathbf{u}, \mathbf{o})$$

(for fixed $\mathbf{o}$) and thus Equation 5 holds. Because deleting an edge can never create a cycle, an edge-deletion transformation (for an existing edge) is always valid.

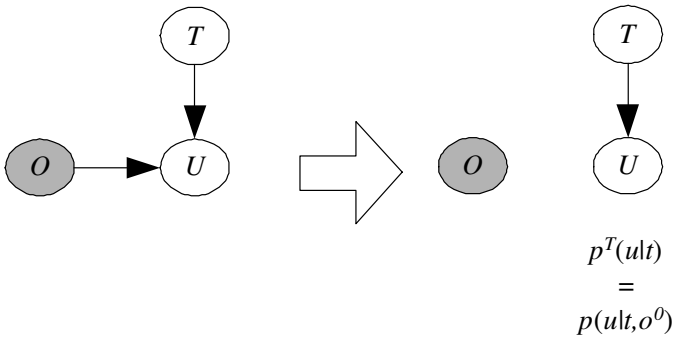


Figure 7: Example of an edge-deletion transformation. The observed value of node O is o^0. After deleting the edge $O \rightarrow U$, the local distribution for node U is identical to the original distribution when constrained to $o = o^0$.

B.1.2 EDGE REVERSAL

An *edge reversal*, originally defined by Howard and Matheson (1981), is a transformation that first "covers" an edge by adding new edges until the edge is covered, and then reverses the edge (see Figure 8). In particular, for an edge $H \rightarrow Y$, let $\mathbf{X}$ be the parents of H that are not parents of Y, let $\mathbf{Z}$ be the parents of both H and Y, and let $\mathbf{W}$ be the parents of Y that are not parents of H. The edge-reversal transformation adds the edge $X \rightarrow Y$ for every $X \in \mathbf{X}$, adds the edge $W \rightarrow H$ for every $W \in \mathbf{W}$, and reverses $H \rightarrow Y$.

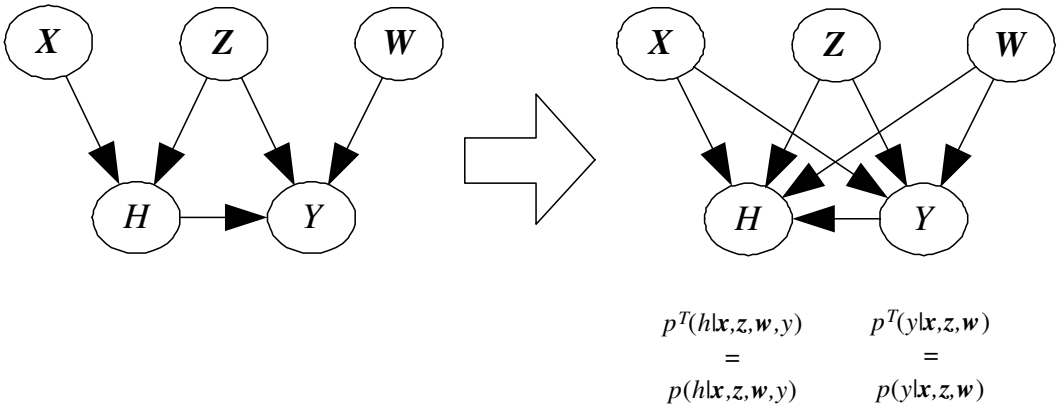


Figure 8: The relevant fragment of a graph structure for an edge-reversal transformation.

As shown in Figure 8, the local distributions for H and Y in $\mathcal{G}^T$ are defined by the joint distribution defined in $\mathcal{G}$. In contrast to the edge-deletion transformation, the local probability distributions for these nodes in $\mathcal{G}^T$ cannot simply be extracted from the corresponding distributions in $\mathcal{G}$; for example, we obtain the local distribution for H in $\mathcal{G}^T$ via inference as follows:

$$\begin{aligned} p^T(h|\mathbf{x},\mathbf{z},\mathbf{w},y) &= p(h|\mathbf{x},\mathbf{z},\mathbf{w},y) \\ &= \frac{p(h,y|\mathbf{x},\mathbf{z},\mathbf{w})}{p(y|h,\mathbf{x},\mathbf{z},\mathbf{w})} \\ &= \frac{p(h|\mathbf{x},\mathbf{z})p(y|h,\mathbf{z},\mathbf{w})}{\sum_i p(h^i|\mathbf{x},\mathbf{z})p(y|h^i,\mathbf{z},\mathbf{w})}, \end{aligned}$$

where $p(h|\mathbf{x},\mathbf{z})$ and $p(y|h,\mathbf{z},\mathbf{w})$ are the local distributions in $\mathcal{G}$.

Proposition 18 *If there is no directed path from H to Y other than the edge $H \rightarrow Y$, then the edge can be covered as described without creating a cycle.*

Proof: Suppose not. Using the notation from above, there must either be some $X \in \mathbf{X}$ for which adding $X \rightarrow Y$ creates a cycle, or there must be some $W \in \mathbf{W}$ for which adding $W \rightarrow H$ creates a cycle. Because there is already a directed path from X to Y, we immediately rule out the first case. If adding $W \rightarrow H$ creates a cycle, then there must already be a directed path from H to W. By appending $W \rightarrow Y$ to this directed path, we have a directed path from H to Y that is not the edge $H \rightarrow Y$, yielding a contradiction. □

Because no independence constraints are added as a result of adding an edge to a model, and because the edge is reversed only after being covered, the following result follows immediately from Proposition 18 and Lemma 1:

Proposition 19 *If in $\mathcal{G}$ there is no directed path from H to Y other than the edge $H \rightarrow Y$, then the edge-reversal transformation applied to $H \rightarrow Y$ is valid; and for the model $\mathcal{G}^T$ that results, the constraints of Equation 5 must hold.*

B.1.3 NODE COMBINATION

A *node combination* takes a set of nodes $\mathbf{Y}$, where each node in $\mathbf{Y}$ has no children, and replaces the set with the single *composite* node $Y = Comp(\mathbf{Y})$ whose states take on the cross product of the states of all nodes in $\mathbf{Y}$ (see Figure 9). The parents of Y are defined to be the union of all of the parents of the nodes in $\mathbf{Y}$. Because no node in $\mathbf{Y}$ has any children, it is easy to see that applying a node-combination transformation can never create a cycle, and thus the transformation is always valid.

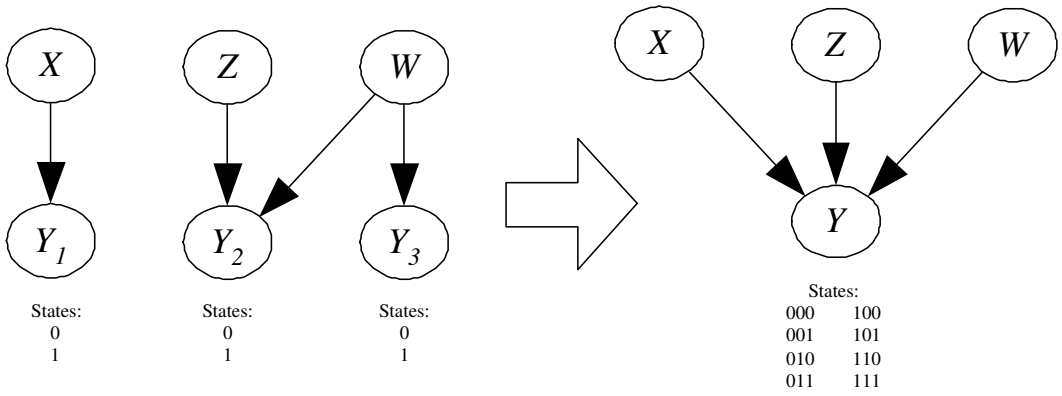


Figure 9: An example of a node-combination transformation. The state space of the combined node Y has a unique state for every possible combination of values for Y_1, Y_2, and Y_3.

The local distribution for the composite node Y is defined in the obvious way: the local probability in $\mathcal{G}^T$ of a composite state y given the parent values is simply the joint probability of the corresponding states of $\mathbf{Y}$ in the original model $\mathcal{G}$ given the same parent values.

Although the set of nodes in the Bayesian network $\mathcal{G}^T$ that results from a node combination is different than in the original network $\mathcal{G}$, it is important to understand that $\mathcal{G}^T$ represents a probability distribution over the original set of nodes. In particular, because the states of the composite node Y are defined to be the cross product of the states for all of the nodes in $\mathbf{Y}$, there is a one-to-one correspondence between states of Y and sets of all states of the nodes in $\mathbf{Y}$. Thus, given any Bayesian network containing composite nodes, we can always "unwind" those nodes into a clique of nodes, where each node in the clique—in addition to the adjacencies within the clique—has the same parents and children of the composite node. Because the nodes that made up the composite node form a clique, there are no independence constraints introduced by this unwinding process. For the remainder of this paper, when we discuss the joint distribution represented by a Bayesian

network, it is to be understood that we mean the distribution over the original domain; we leave implicit the unwinding process that can be performed so that the networks contain the same nodes.

B.1.4 BARREN NODE REMOVAL

An unobserved node $U \in \mathbf{U}$ is *barren* if U has no children. An observed node $O \in \mathbf{O}$ is barren if O has no parents and no children. The *barren-node-removal* transformation simply removes from $\mathcal{G}$ any barren nodes along with their incident edges (see Figure 10). Because a barren node has no children, no conditional distributions change (other than the deletion of the barren-node distribution). Because removing a barren node can never create a cycle, the transformation is always valid.

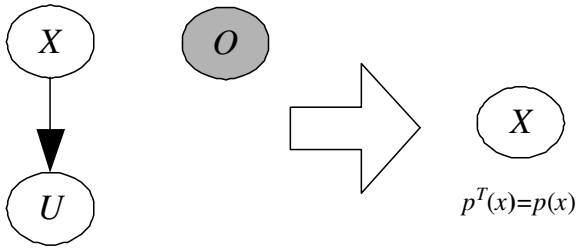


Figure 10: An example of a barren-node-removal transformation: both the unobserved node U and the observed node O are barren.

We now explain why Equation 5 must hold after removing an *unobserved* barren node. Letting $\mathbf{U}^T = \mathbf{U} \setminus U$ denote the unobserved nodes that remain after removing U, we can compute the joint probability in the original model over $\mathbf{U}^T$ and $\mathbf{O}$ as

$$p(\mathbf{u}^T, \mathbf{o}) = \sum_u p(u, \mathbf{u}, \mathbf{o}).$$

Because U has no children, we can "push the sum" all the way through to the last conditional distribution:

$$\begin{aligned} p(\mathbf{u}^T, \mathbf{o}) &= \prod_{x \in \mathbf{u}^T \cup \mathbf{o}} p(x|\mathbf{pa}^X) \left(\sum_u p(u|\cdot) \right) \\ &= \prod_{x \in \mathbf{u}^T \cup \mathbf{o}} p(x|\mathbf{pa}^X). \end{aligned}$$

Because $\mathcal{G}^T$ is identical to $\mathcal{G}$ except that it does not contain node U, it follows that the above product of conditional distributions is exactly the distribution represented by $\mathcal{G}^T$; thus $p^T(\mathbf{u}^T, \mathbf{o}) = p(\mathbf{u}^T, \mathbf{o})$ and Equation 5 must hold.

Equation 5 holds after removing an *observed* barren node O by a similar argument and because O is independent of every other node regardless of the conditioning set.

B.1.5 OBSERVED CHILD SEPARATION (OCS)

The observed-child-separation (OCS) transformation is a "macro" transformation that combines a node-combination transformation, an edge-reversal transformation, and an edge-deletion transfor-

mation. In particular, let H be any node in $\mathbf{U}$ that has at least one child in $\mathbf{O}$, and let $\mathbf{Y} = \{Y_1, \ldots, Y_m\}$ denote the set of all children of H that are in $\mathbf{O}$ that have no children themselves. For the OCS transformation (see the example in Figure 11), we first apply a node-combination transformation to the nodes in $\mathbf{Y}$, resulting in a model containing the composite node $Y = Comp(\mathbf{Y})$. Next, we apply an edge-reversal transformation on the edge $H \rightarrow Y$. Finally, we delete the resulting edge $Y \rightarrow H$ using an edge-deletion transformation. The OCS transformation is valid whenever the sub-transformations are valid.

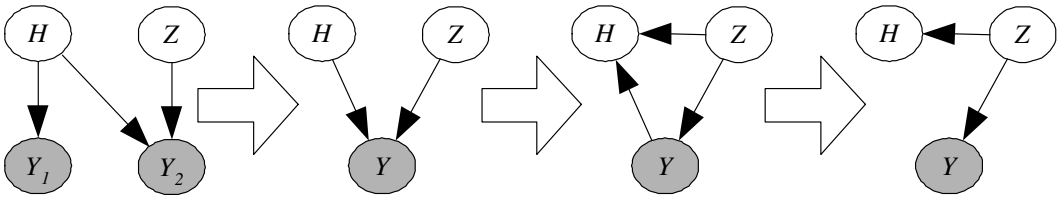


Figure 11: An example showing the sub-transformations that make up the OCS macro transformation.

Because we have already shown that the invariant of Equation 5 holds after each component transformation that makes up the OCS macro transformation, we conclude that Equation 5 holds as a result of this transformation as well.

In Figure 12, we show the relevant network fragment both before and after a general OCS transformation is applied, along with the local distributions in $\mathcal{G}^T$ that must be derived via inference in $\mathcal{G}$. The nodes Y_j—which are shaded in the figure—are the observed children of node H that will be separated from H using the transformation. The bold-font nodes $\mathbf{X}$, $\mathbf{Z}$, and $\mathbf{W}$ represent sets of nodes: each node in $\mathbf{X}$ is a parent of H but not a parent of any Y_j, each node in $\mathbf{Z}$ is a parent of H and a parent of at least one Y_j, and each node in $\mathbf{W}$ is not a parent of H but is a parent of at least one Y_j. In the figure, the term $\mathbf{y^0}$ in $p(h|\mathbf{x}, \mathbf{z}, \mathbf{w}, \mathbf{y^0})$ is shorthand for the set of all observed states $y_1^0, \ldots, y_n^0$ of the Y_j variables; we assume that y_j^0 is the observed state of Y_j for all j. Similarly, the term $\mathbf{y}$ in $p(\mathbf{y}|\mathbf{x}, \mathbf{z}, \mathbf{w})$ denotes an arbitrary set of states $y_1, \ldots, y_n$ for the observed Y_j variables.

B.2 Transformation Algorithms

In this section, we present two graph-transformation algorithms that, like the OCS "macro" transformation, apply a sequence of transformations to a model $\mathcal{G}$. We distinguish an "algorithm" from a "macro transformation" by the fact that in the former, the order in which we apply the individual transformations depends on the topology of the entire network structure. As in the case of the OCS macro transformation, we conclude that because the individual transformations that define the algorithm all maintain the invariant of Equation 5, the invariant holds for the algorithms as well.

We say that a node X in a graph is a *lowest* node with some property if no descendant of X in the graph also has that property. Thus when the graph is a DAG containing at least one node with a given property, there must always exist at least one lowest node with that property.

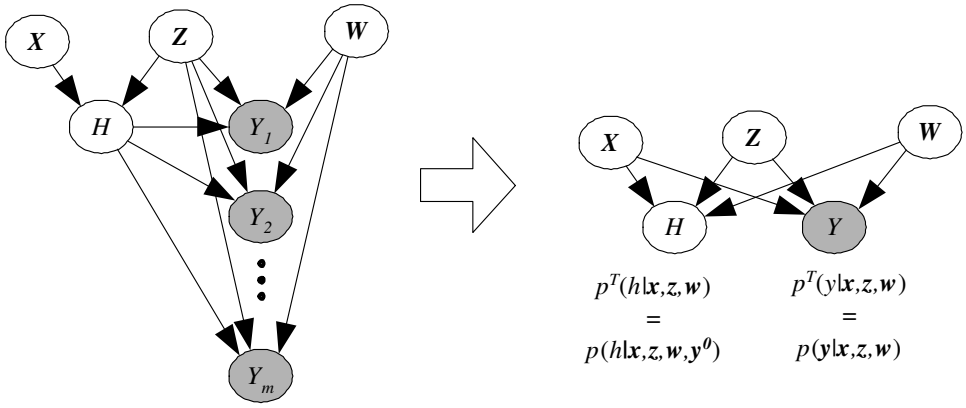


Figure 12: General OCS macro transformation.

B.2.1 THE UNOBSERVED PATH SHORTENING (UPS) ALGORITHM

The unobserved-path-shortening (UPS) algorithm is applied when $\mathcal{G}$ contains only unobserved nodes (all nodes from **O** have been removed before this algorithm is used). We say a node is a *root* if it has no parents. The algorithm takes as input any non-root node Y, and returns the model $\mathcal{G}^T$ in which all nodes have been deleted except for Y and its root-node ancestors **R**. For every $R \in \mathbf{R}$, the edge $R \rightarrow Y$ is in $\mathcal{G}^T$ (the edge need not be in $\mathcal{G}$); $\mathcal{G}^T$ contains no other edges (see Figure 13). In Figure 14, we show how the UPS algorithm is implemented by a sequence of the transformations presented in Section B.1.

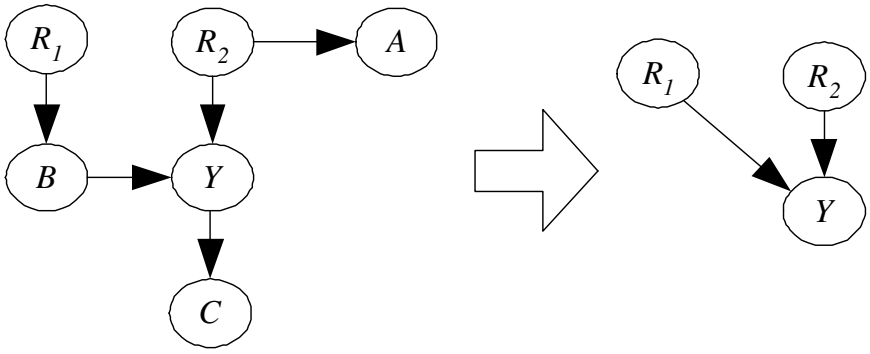


Figure 13: An example application of the UPS algorithm.

The following lemma demonstrates that the steps given in Figure 14 correctly implement the UPS algorithm using graph transformations.

Lemma 20 *Let $\mathcal{G}$ be a Bayesian network containing non-root node Y, and let* **R** *denote the set of root-node ancestors of Y in $\mathcal{G}$. Let $\mathcal{G}^T$ denote the Bayesian network that results from applying Algorithm UPS with inputs $\mathcal{G}$ and Y. Then after the algorithm completes, the nodes in $\mathcal{G}^T$ are precisely $\mathbf{R} \cup \{Y\}$, and the edges in $\mathcal{G}^T$ are precisely $R \rightarrow Y$ for every $R \in \mathbf{R}$.*

Algorithm UPS
Input: Bayesian network $\mathcal{G}$ and non-root node Y. (Let $\mathbf{R}$ denote the set of root-node ancestors of Y in $\mathcal{G}$, and assume that all nodes in $\mathcal{G}$ are unobserved.)
Output: Bayesian network $\mathcal{G}^T$ containing nodes $\mathbf{R} \cup \{Y\}$ and edges $R \rightarrow Y$ for every $R \in \mathbf{R}$

1. Set $\mathcal{G}^T = \mathcal{G}$
2. While $\mathcal{G}^T$ contains at least one barren node $B \neq Y$, delete B using the barren-node removal transformation.
3. While Y has at least one parent that is not a root node
4. Choose any lowest non-root H that is a parent of Y
5. Reverse the edge $H \rightarrow Y$ using the edge-reversal transformation
6. Delete the (now barren) node H using the barren-node-removal transformation.
7. Return $\mathcal{G}^T$

Figure 14: The unobserved path shortening (UPS) algorithm

Proof: First we note that after step 2 of the algorithm, every node in $\mathcal{G}^T$ other than Y is an ancestor of Y. This follows because, given that every node in $\mathcal{G}^T$ is unobserved, any non-ancestor of Y must either be barren or have some descendant (not equal to Y) that is barren.

At step 5, there cannot be any directed path from H to Y other than the edge $H \rightarrow Y$ because H is chosen to be a lowest parent of H. Thus, we know that the edge-reversal transformation at step 5 is always valid. Furthermore, because every node is an ancestor of Y, we know that Y must be the only child of H, and thus after the edge reversal, H must be barren (H cannot gain a child from the reversal), and we can always delete H in step 6.

By definition of an edge reversal, the only nodes that can gain parents in step 5 are Y and H. Because H is necessarily a non-root node, we conclude that every node in $\mathbf{R}$ will remain a root node after every edge reversal. The definition of an edge reversal also guarantees that any node other than H that is an ancestor of Y before the reversal will remain an ancestor after the reversal. Thus when the algorithm terminates, all nodes other than Y must be root-node parents of Y. □

B.2.2 THE OBSERVED NODE ELIMINATION (ONE) ALGORITHM

In this section, we describe the observed-node elimination (ONE) algorithm that deletes all of the observed variables from $\mathcal{G}$ such that, given certain preconditions, the invariant of Equation 5 holds on the resulting model. The details of the algorithm are shown in Figure 15.

In Figure 16, we show an example of the algorithm applied to a model. The original model is shown in Figure 16a, where the observed nodes E, F, and G are depicted by shading. In step 2 of the algorithm, the edges $F \rightarrow C$ and $F \rightarrow G$ are removed, resulting in the model from Figure 16b. For step 3, we see that all four unobserved nodes have at least one observed child. The only lowest nodes are C and D; (arbitrarily) choosing C first (shown with a thick border in the figure), we apply the OCS transformation (which "combines" the singleton node G, covers the edge $C \rightarrow G$ by

Algorithm ONE
Input: Bayesian network $\mathcal{G}$ consisting of observed nodes **O** and unobserved nodes **U**, set of observations **o** corresponding to nodes **O**
Output: Bayesian network $\mathcal{G}^T$ containing only the nodes in **U**

1. Set $\mathcal{G}^T = \mathcal{G}$.
2. For every edge $O \rightarrow X$ in $\mathcal{G}^T$ for which $O \in \mathbf{O}$ is observed, remove the edge using an edge-removal transformation.
3. While there exists an unobserved node in **U** that has at least one child from **O** in $\mathcal{G}^T$, apply the OCS transformation to $\mathcal{G}^T$ using any lowest such node.
4. Delete every node $O \in \mathbf{O}$ by applying the barren-node-elimination transformation.
5. Return $\mathcal{G}^T$.

Figure 15: The observed node elimination (ONE) algorithm

adding $B \rightarrow G$, then reverses and deletes the edge between C and G) resulting in the model shown in Figure 16c. Still in step 3, the lowest nodes are B and D; choosing B for the OCS transformation results in model shown in Figure 16d. In this model, EFG is the combined node of E, F, and G from the OCS transformation. Still in step 3, the lowest nodes are A and D; choosing D for the OCS transformation results in the model shown in Figure 16e. For the last iteration of step 3, A is the only node with an observed child, and applying the OCS transformation results in the model shown in Figure 16f. Finally, in step 4, the barren node EFG is deleted, resulting in the final model shown in Figure 16g that contains only observed nodes.

To help prove properties about Algorithm ONE, we use $\mathcal{G}^i$ to denote the Bayesian network that results after i iterations of the While loop at step 3. We define $\mathcal{G}^0$ to be the graph $\mathcal{G}^T$ that results after applying step 2 but before the first iteration of the While loop at step 3. We use H^i to denote the (lowest) node chosen in iteration i of the While loop, we use $\mathbf{Y}^i$ to denote the set of observed children of H^i on iteration i of the While loop, and we use $Y^i = Comp(\mathbf{Y}^i)$ to denote the composite node created by the OCS transformation in iteration i of the While loop.

As a result of applying the OCS transformation in step 3, we create the new composite node Y^i defined by the subset $\mathbf{Y} \subseteq \mathbf{O}$ of the observed variables. To simplify discussion, we find it convenient to treat **O** as a static set of nodes as opposed to a set that changes each time a new composite node is created. Thus, we will say that any composite node Y^i is contained in the set **O** when technically we should say that all nodes that have been combined to create Y are contained in **O**.

Lemma 21 *For all $\mathcal{G}^i$ (i.e., for all graphs considered in step 3 of Algorithm ONE), every node in* **O** *has zero children.*

Proof: The proposition clearly holds for $\mathcal{G}^0$ due to step 2. Toward a contradiction, let i be the first iteration of the While loop in which a node in **O** gains a child. By definition of the OCS transformation, the only nodes that can gain children are parents of H^i and parents of nodes in $\mathbf{Y}^i$. Because these nodes already have children, they cannot be in **O**. □

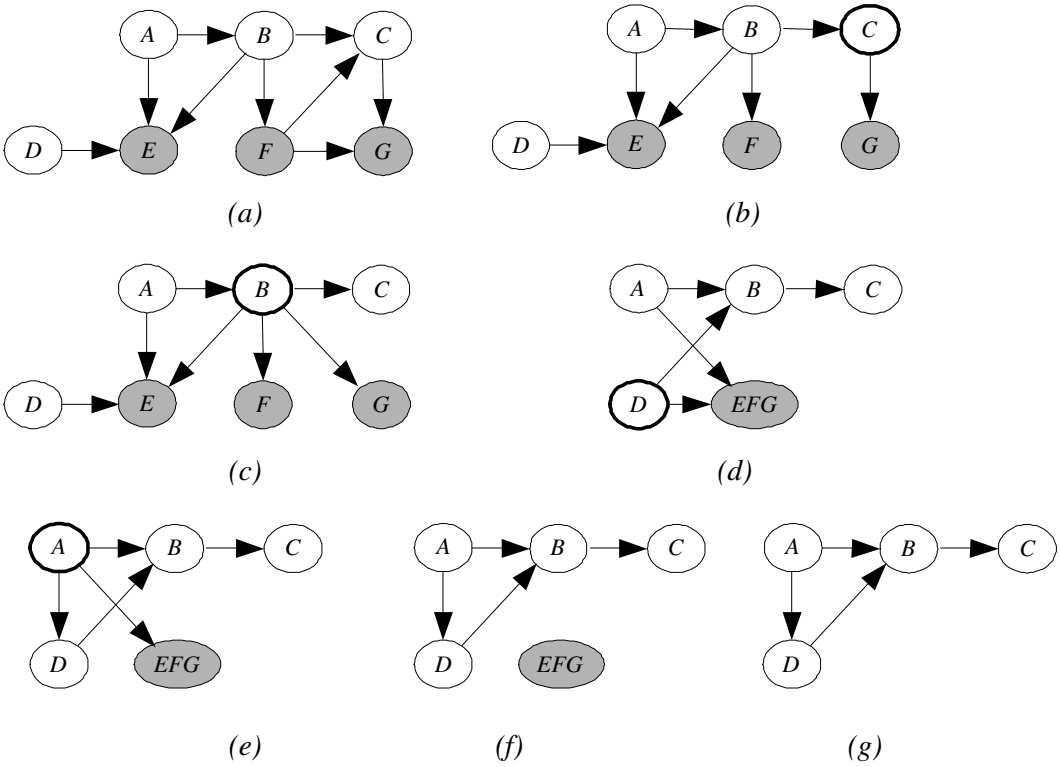


Figure 16: Example of the ONE algorithm applied to a model. Lowest nodes chosen in step 3 are shaded.

Recall from Section B.1.5 that the OCS transformation required that the observed children must have no children themselves. Lemma 21 guarantees that this property holds for every $\mathcal{G}^i$, and thus the OCS transformation can always be applied in step 3. We now demonstrate that the node H^i can be chosen by Algorithm ONE in step 3 at most once. An immediate consequence of this result is that step 3 terminates in at most $|\mathbf{U}|$ iterations.

Proposition 22 *Let $Anc^i(\mathbf{O})$ denote the set of nodes in $\mathbf{U}$ that are ancestors in $\mathcal{G}^i$ of at least one node in $\mathbf{O}$. Then $Anc^i(\mathbf{O}) = Anc^{i-1}(\mathbf{O}) \setminus H^i$.*

Proof: From the definition of the OCS transformation, at each iteration i this "macro" transformation first applies a node-combination transformation on the observed children $\mathbf{Y}^i$ of H^i, then applies an edge-reversal transformation on the edge $H^i \rightarrow Y^i$, and then applies the edge-removal transformation on $Y^i \rightarrow H^i$.

We first note that applying a node-combination transformation on nodes in $\mathbf{O}$ cannot change the set of ancestors of node in $\mathbf{O}$.

In order for a node to become a new ancestor of a node in $\mathbf{O}$, some edge $A \rightarrow B$ must be added to $\mathcal{G}^i$ such that before the addition, A is not an ancestor of a node in $\mathbf{O}$ and B is an ancestor of a node in $\mathbf{O}$. From the definition of the OCS transformation, the only edges that are added are either

(1) of the form $A \rightarrow Y^i$, where A is a parent of H^i and hence an ancestor of node $Y^i \in \mathbf{O}$, or (2) of the form $A \rightarrow H^i$, where A is the parent of $Y^i \in \mathbf{O}$. Thus we conclude that $Anc^i(\mathbf{O}) \subseteq Anc^{i-1}(\mathbf{O})$.

In order for a node to no longer be an ancestor of a node in $\mathbf{O}$, some edge $A \rightarrow B$ must be deleted such that after the deletion, A is no longer an ancestor of $\mathbf{O}$. The only edge that is deleted by the transformation is $H^i \rightarrow Y^i$. After the transformation, all parents of H^i are necessarily parents of Y^i, and thus the only node that can possibly no longer be an ancestor of a node in $\mathbf{O}$ is H^i. Because H^i was chosen as the lowest node with a child in $\mathbf{O}$, and because the only added edges were incident into H^i or a parent of H^i, we know that no descendant of H^i can be an ancestor of a node in $\mathbf{O}$, and the lemma follows. □

Corollary 23 *Algorithm ONE terminates after choosing each node from* $\mathbf{U}$ *in step 3 at most once.*

Proof: As in Lemma 22, we use $Anc^i(\mathbf{O})$ to denote the set of nodes in $\mathbf{U}$ that are ancestors in $\mathcal{G}^i$ of at least one node in $\mathbf{O}$. In order to be chosen in Step 3 during iteration i of the While loop, H^i must be in $Anc^i(\mathbf{O})$. From Lemma 22, if H^i is chosen during iteration i, it can never be an element of $Anc_j(\mathbf{O})$ for $j > i$. □

The next result demonstrates that by breaking ties as appropriate, we can guarantee that any particular unobserved node with no unobserved parents will be a root node after applying Algorithm ONE.

Lemma 24 *Let* $U \in \mathbf{U}$ *be any unobserved node with no unobserved parents in* $\mathcal{G}$*. If in Algorithm ONE we break ties in step 3 in favor of not selecting* U*, then* U *will be a root node in* $\mathcal{G}^T$*.*

Proof: Suppose not. Then at some iteration of the While loop in step 3, an unobserved parent W must be added to U by the algorithm. Let i be the first iteration in which this occurs. By definition of the OCS transformation, we conclude that $H^i = U$ and that W is a parent of Y^i that is not a parent of U. Because we break ties in favor of not choosing U, and because W has a child in $\mathbf{O}$, we conclude that there must be a directed path from W to U. But from Lemma 21 we conclude that the last edge in this directed path is from a node in $\mathbf{U}$ which means that U already has an unobserved parent. □

Appendix C. Properties of Local Distributions

In this appendix, we formally define the BLL and BLTSP properties that were described (informally) in Appendix A, and we show the conditions under which these properties are maintained in the conditional probability distributions as a result of the various graph transformations defined in Appendix B. We begin in Section C.1 by presenting some preliminary definitions and results. In Section C.2, we use this material to derive the main results of this section.

C.1 Preliminary Definitions and Results

In this section, we consider non-negative real-valued functions of $\mathbf{X}$, where $\mathbf{X} = (X_1, \ldots, X_n)$ is a set of variables such that the states of each variable X_i are totally ordered. By *totally ordered*, we mean totally ordered in the *non-strict* sense, where some states may have equal order. For convenience, we often write $f(x_1, \ldots, x_n)$ as $f(\ldots, x_i, \ldots)$ to emphasize the argument x_i.

Definition 25 *The function* $f(\mathbf{X})$ *is* lattice *if, for any* i *such that* $x_i > x_i'$*,*

$$f(\ldots, x_i, \ldots) > f(\ldots, x_i', \ldots),$$

where the other arguments are held fixed and are arbitrary.

For example, for a function with two binary arguments (X_1, X_2) with state orderings $x_1^0 > x_1^1$ and $x_2^0 > x_2^1$, we have $f(x_1^0, x_2^0) > f(x_1^0, x_2^1) > f(x_1^1, x_2^1)$ and $f(x_1^0, x_2^0) > f(x_1^1, x_2^0) > f(x_1^1, x_2^1)$.

Definition 26 *The function $f(\mathbf{X})$ is* totally non-strictly positive *if, for all $i < j$,*

$$f(\ldots, \max(x_i, x_i'), \ldots, \max(x_j, x_j'), \ldots) \cdot f(\ldots, \min(x_i, x_i'), \ldots, \min(x_j, x_j'), \ldots) \geq f(\ldots, x_i, \ldots, x_j, \ldots) \cdot f(\ldots, x_i', \ldots, x_j', \ldots). \tag{6}$$

The concept of total non-strict positivity is often referred to as *multivariate total positivity of order two* (see Karlin and Rinott, 1980). We now define a version of total positivity where the inequality in Equation 6 must be strict whenever equality does not hold trivially.

Definition 27 *The function $f(\mathbf{X})$ is* totally strictly positive *if, for all $i < j$,*

$$f(\ldots, \max(x_i, x_i'), \ldots, \max(x_j, x_j'), \ldots) \cdot f(\ldots, \min(x_i, x_i'), \ldots, \min(x_j, x_j'), \ldots) \geq f(\ldots, x_i, \ldots, x_j, \ldots) \cdot f(\ldots, x_i', \ldots, x_j', \ldots), \tag{7}$$

where the other arguments are held fixed and are arbitrary, and equality holds if and only if either

$$f(\ldots, \max(x_i, x_i'), \ldots, \max(x_j, x_j'), \ldots) = f(\ldots, x_i, \ldots, x_j, \ldots) \quad \text{and}$$
$$f(\ldots, \min(x_i, x_i'), \ldots, \min(x_j, x_j'), \ldots) = f(\ldots, x_i', \ldots, x_j', \ldots)$$

or

$$f(\ldots, \max(x_i, x_i'), \ldots, \max(x_j, x_j'), \ldots) = f(\ldots, x_i', \ldots, x_j', \ldots) \quad \text{and}$$
$$f(\ldots, \min(x_i, x_i'), \ldots, \min(x_j, x_j'), \ldots) = f(\ldots, x_i, \ldots, x_j, \ldots).$$

For example, a function f with two binary arguments (X_1, X_2) with state orderings $x_1^0 > x_1^1$ and $x_2^0 > x_2^1$ is totally strictly positive if $f(x_1^0, x_2^0)\ f(x_1^1, x_2^1) > f(x_1^0, x_2^1)\ f(x_1^1, x_2^0)$. Note that all other combinations of arguments yield trivial equalities. For example, applying the definition when $x_1 = x_1' = x_1^0$, $x_2 = x_2^0$, and $x_2' = x_2^1$, yields the constraint

$$f(\min(x_1^0, x_1^0), \min(x_2^0, x_2^1))\ f(\max(x_1^0, x_1^0), \max(x_2^0, x_2^1)) \geq f(x_1^0, x_2^1)\ f(x_1^0, x_2^0),$$

for which equality holds trivially by solving the left-hand side.

We now give properties of positive, lattice, and totally strictly positive functions whose arguments x are binary with values x^0 and x^1, and with state ordering $x^0 > x^1$. We call functions having only binary arguments *cube functions*. As a shorthand, we use (e.g.) f^{ij} to represent $f(x_1^i, x_2^j)$.

Proposition 28 *Given real numbers $\alpha_1 \geq \alpha_2$ in the interval $(0,1)$ and positive real numbers f^{00}, f^{01}, f^{10}, and f^{11} such that $f^{00} \geq f^{01} \geq f^{11}$ and $f^{00} \geq f^{10}$,*

$$\alpha_1 f^{00} + (1 - \alpha_1) f^{01} \geq \alpha_2 f^{10} + (1 - \alpha_2) f^{11}, \tag{8}$$

where equality holds if and only if $(f^{10} \geq f^{11}) \wedge (f^{00} = f^{10}) \wedge (f^{01} = f^{11}) \wedge ((f^{10} = f^{11}) \vee (\alpha_1 = \alpha_2))$. Equivalently,

$$f^{01} + \alpha_1 (f^{00} - f^{01}) \geq f^{10} + \alpha_2 (f^{10} - f^{11}). \tag{9}$$

Proof: There are two cases to consider.
Case 1: $f^{11} > f^{10}$. Here, the right-hand-side of 8 will be strictly less than $f^{11} \leq f^{01}$. Thus, because the left-hand-side of 8 will be at least f^{01}, 8 holds with strict inequality.
Case 2: $f^{10} \geq f^{11}$. Here, because $f^{00} \geq f^{10}$, $f^{01} \geq f^{11}$, and $0 < \alpha_1 < 1$,

$$\alpha_1 f^{00} + (1-\alpha_1) f^{01} \geq \alpha_1 f^{10} + (1-\alpha_1) f^{11}, \tag{10}$$

where equality holds if and only if $(f^{00} = f^{10}) \wedge (f^{01} = f^{11})$. Because $f^{10} \geq f^{11}$ and $\alpha_1 \geq \alpha_2$, we have

$$\alpha_1 f^{10} + (1-\alpha_1) f^{11} \geq \alpha_2 f^{10} + (1-\alpha_2) f^{11}, \tag{11}$$

where equality holds if and only if $(f^{10} = f^{11}) \vee (\alpha_1 = \alpha_2)$. Inequalities 10 and 11 imply 8, where equality holds if and only if $(f^{00} = f^{10}) \wedge (f^{01} = f^{11}) \wedge ((f^{10} = f^{11} \vee (\alpha_1 = \alpha_2))$. □

Proposition 29 *Given a positive, cube, and lattice function $f(X_1, X_2)$,*

$$\alpha_1 f^{00} + (1-\alpha_1) f^{01} > \alpha_2 f^{10} + (1-\alpha_2) f^{11}, \tag{12}$$

for any two real numbers $0 < \alpha_2 \leq \alpha_1 < 1$.

Proof: Because $f(X_1, X_2)$ is lattice, we have $f^{00} > f^{01} > f^{11}$ and $f^{00} > f^{10} > f^{11}$. The strict inequality 12 therefore follows by Proposition 28. □

Proposition 30 *Given a positive, cube, lattice, and totally strictly positive function $f(X_1, X_2)$,*

$$f^{00} + f^{11} > f^{01} + f^{10}. \tag{13}$$

Proof: We know

$$f^{00} f^{11} > f^{01} f^{10}.$$

Subtracting $f^{01} f^{11}$ from both sides, we obtain

$$(f^{00} - f^{01}) f^{11} > (f^{10} - f^{11}) f^{01}.$$

Because $f^{01} > f^{11}$, we have

$$f^{00} - f^{01} > f^{10} - f^{11}.$$

□

The remainder of this section contains propositions needed to prove the main results in Section C.2. We sometimes use functions having a range of $(0,1)$. We call such functions *unit functions*.

Proposition 31 *Given a real number f in $(0,1)$ and a positive, cube, and totally strictly positive function $g(Y_1, Y_2)$, the ratio*

$$\frac{f g^{00} + (1-f) g^{01}}{f g^{10} + (1-f) g^{11}}$$

is a strictly increasing function of f.

Proof: A straightforward algebraic arrangement yields

$$\frac{fg^{00}+(1-f)g^{01}}{fg^{10}+(1-f)g^{11}} = \frac{g^{01}}{g^{11}} + \frac{1}{1+(1-f)g^{11}/(fg^{10})}\left(\frac{g^{00}}{g^{10}} - \frac{g^{01}}{g^{11}}\right) \tag{14}$$

Because $g(Y_1,Y_2)$ is totally strictly positive, the difference of fractions at the end of Equation 14 is positive. The proposition then follows because g^{11}/g^{10} is positive. □

Proposition 32 *Given a unit, cube, lattice, and totally strictly positive function $f(X_1,X_2)$ and a positive, cube, and lattice function $g(Y)$,*

$$\frac{f^{00}g^0+(1-f^{00})g^1}{f^{01}g^0+(1-f^{01})g^1} > \frac{f^{10}g^0+(1-f^{10})g^1}{f^{11}g^0+(1-f^{11})g^1}. \tag{15}$$

Proof: By Proposition 30, we know that $f^{00}+f^{11} > f^{01}+f^{10}$. Therefore, we have

$$(g^0g^0+g^1g^1)f^{00}f^{11}+g^1(g^0-g^1)(f^{00}+f^{11}) > (g^0g^0+g^1g^1)f^{01}f^{10}+g^1(g^0-g^1)(f^{01}+f^{10}). \tag{16}$$

Adding g^1g^1 to both sides of inequality 16 and factoring, we obtain

$$(f^{00}g^0+(1-f^{00})g^1)(f^{11}g^0+(1-f^{11})g^1) > (f^{01}g^0+(1-f^{01})g^1)(f^{10}g^0+(1-f^{10})g^1).$$

Inequality 15 follows from the fact that terms in the denominator are positive. □

Proposition 33 *Given unit, cube, lattice, and totally strictly positive function $f(X_1,X_2)$ and a positive, cube, and totally strictly positive function $g(Y_1,Y_2)$, where $g(0,Y_2)$ is lattice,*

$$\frac{f^{00}g^{00}+(1-f^{00})g^{01}}{f^{01}g^{10}+(1-f^{01})g^{11}} > \frac{f^{10}g^{00}+(1-f^{10})g^{01}}{f^{11}g^{10}+(1-f^{11})g^{11}}. \tag{17}$$

Proof: The function $g(0,Y_2)$ is unit, cube, and lattice. Consequently, by Proposition 32, we have

$$\frac{f^{00}g^{00}+(1-f^{00})g^{01}}{f^{01}g^{00}+(1-f^{01})g^{01}} > \frac{f^{10}g^{00}+(1-f^{10})g^{01}}{f^{11}g^{00}+(1-f^{11})g^{01}}.$$

Interchanging the denominator on the left-hand-side with the numerator on the right-hand-side, we get

$$\frac{f^{00}g^{00}+(1-f^{00})g^{01}}{f^{10}g^{00}+(1-f^{10})g^{01}} > \frac{f^{01}g^{00}+(1-f^{01})g^{01}}{f^{11}g^{00}+(1-f^{11})g^{01}}. \tag{18}$$

Using Proposition 31 and that fact that $f^{01} > f^{11}$, we obtain

$$\frac{f^{01}g^{00}+(1-f^{01})g^{01}}{f^{01}g^{10}+(1-f^{01})g^{11}} > \frac{f^{11}g^{00}+(1-f^{11})g^{01}}{f^{11}g^{10}+(1-f^{11})g^{11}}$$

or, equivalently,

$$\frac{f^{01}g^{00}+(1-f^{01})g^{01}}{f^{11}g^{00}+(1-f^{11})g^{01}} > \frac{f^{01}g^{10}+(1-f^{01})g^{11}}{f^{11}g^{10}+(1-f^{11})g^{11}}. \tag{19}$$

Inequalities 18 and 19 imply

$$\frac{f^{00}g^{00}+(1-f^{00})g^{01}}{f^{10}g^{00}+(1-f^{10})g^{01}} > \frac{f^{01}g^{10}+(1-f^{01})g^{11}}{f^{11}g^{10}+(1-f^{11})g^{11}}.$$

which is equivalent to 17. □

Proposition 34 *Given a real number f in $(0,1)$ and a positive and cube function $g(Y_1,Y_2,Y_3)$, where $g(0,Y_2,Y_3)$, $g(1,Y_2,Y_3)$, and $g(Y_1,1,Y_3)$ are totally strictly positive and $g(Y_1,Y_2,0)$ and $g(Y_1,Y_2,1)$ are totally non-strictly positive,*

$$\frac{fg^{000}+(1-f)g^{001}}{fg^{010}+(1-f)g^{011}} > \frac{fg^{100}+(1-f)g^{101}}{fg^{110}+(1-f)g^{111}}. \tag{20}$$

Proof: Using Equation 14, we rewrite inequality 20 as follows:

$$\begin{aligned}\frac{g^{001}}{g^{011}} + \frac{1}{1+(1-f)g^{011}/(fg^{010})}\left(\frac{g^{000}}{g^{010}}-\frac{g^{001}}{g^{011}}\right) > \\ \frac{g^{101}}{g^{111}} + \frac{1}{1+(1-f)g^{111}/(fg^{110})}\left(\frac{g^{100}}{g^{110}}-\frac{g^{101}}{g^{111}}\right).\end{aligned} \tag{21}$$

Now, observe that inequality 21 has the same form as 9, with

$$\alpha_1 = \frac{1}{1+(1-f)g^{011}/(fg^{010})} \quad \text{and} \quad \alpha_2 = \frac{1}{1+(1-f)g^{111}/(fg^{110})}.$$

In addition, because $g(0,Y_2,Y_3)$ and $g(1,Y_2,Y_3)$ are totally strictly positive and $g(Y_1,Y_2,0)$ and $g(Y_1,Y_2,1)$ are totally (non-strictly) positive, we have

$$\frac{g^{000}}{g^{010}} > \frac{g^{001}}{g^{011}} \geq \frac{g^{101}}{g^{111}} \quad \text{and} \quad \frac{g^{000}}{g^{010}} \geq \frac{g^{100}}{g^{110}} > \frac{g^{101}}{g^{111}}.$$

Thus, the conditions of Proposition 28 apply, and 20 will hold if $\alpha_1 > \alpha_2$, or

$$\frac{1}{1+(1-f)g^{011}/(fg^{010})} > \frac{1}{1+(1-f)g^{111}/(fg^{110})}. \tag{22}$$

Rearranging inequality 22 and canceling the terms f and $(1-f)$, we obtain

$$\frac{g^{011}}{g^{010}} < \frac{g^{111}}{g^{110}}$$

which holds because $g(Y_1,1,Y_3)$ is totally strictly positive. □

Proposition 35 *Given a unit, cube, and lattice function $f(X)$ and positive, cube, and lattice function $g(Y_1,Y_2,Y_3)$, where $g(0,Y_2,Y_3)$, $g(1,Y_2,Y_3)$, and $g(Y_1,1,Y_3)$ are totally strictly positive and $g(Y_1,Y_2,0)$ and $g(Y_1,Y_2,1)$ are totally (non-strictly) positive,*

$$\frac{f^0g^{000}+(1-f^0)g^{001}}{f^0g^{010}+(1-f^0)g^{011}} > \frac{f^1g^{100}+(1-f^1)g^{101}}{f^1g^{110}+(1-f^1)g^{111}}. \tag{23}$$

Proof: By Proposition 34, we have

$$\frac{f^0g^{000}+(1-f^0)g^{001}}{f^0g^{010}+(1-f^0)g^{011}} > \frac{f^0g^{100}+(1-f^0)g^{101}}{f^0g^{110}+(1-f^0)g^{111}}. \tag{24}$$

Because $f^0 > f^1$ and $g(1,Y_2,Y_3)$ is totally strictly positive, Proposition 31 yields

$$\frac{f^0g^{100}+(1-f^0)g^{101}}{f^0g^{110}+(1-f^0)g^{111}} > \frac{f^1g^{100}+(1-f^1)g^{101}}{f^1g^{110}+(1-f^1)g^{111}}. \tag{25}$$

Inequalities 24 and 25 imply 23. □

Proposition 36 *Given a unit, cube, lattice and totally strictly positive function $f(X_1,X_2)$ and positive, cube, lattice function $g(Y_1,Y_2,Y_3)$, where $g(0,0,Y_3)$ is lattice, $g(0,Y_2,Y_3)$, $g(1,Y_2,Y_3)$, $g(Y_1,0,Y_3)$, and $g(Y_1,1,Y_3)$ are totally strictly positive and $g(Y_1,Y_2,0)$ and $g(Y_1,Y_2,1)$ are totally (non-strictly) positive,*

$$\frac{f^{00}g^{000}+(1-f^{00})g^{001}}{f^{01}g^{010}+(1-f^{01})g^{011}} > \frac{f^{10}g^{100}+(1-f^{10})g^{101}}{f^{11}g^{110}+(1-f^{11})g^{111}}. \tag{26}$$

Proof: Using Proposition 33 and the fact that $g(0,0,Y_3)$ is lattice and $g(Y_1,0,Y_3)$ is totally strictly positive, we get

$$\frac{f^{00}g^{000}+(1-f^{00})g^{001}}{f^{01}g^{000}+(1-f^{01})g^{001}} > \frac{f^{10}g^{100}+(1-f^{10})g^{101}}{f^{11}g^{100}+(1-f^{11})g^{101}}. \tag{27}$$

Because $f(X_1,1)$ is lattice, Proposition 35 yields

$$\frac{f^{01}g^{000}+(1-f^{01})g^{001}}{f^{01}g^{010}+(1-f^{01})g^{011}} > \frac{f^{11}g^{100}+(1-f^{11})g^{101}}{f^{11}g^{110}+(1-f^{11})g^{111}}. \tag{28}$$

Multiplying the left-hand-sizes of inequalities 27 and 28 and the right-hand-sides of the same inequalities, we obtain 26. □

C.2 The BLL and BLTSP Properties

In this section, we turn our attention from general non-negative real-valued functions to conditional probability distributions. In particular, we consider Bayesian networks for discrete, finite-valued variables in which the local distributions $p(y|x_1,\ldots,x_n)$ have the following properties.

Definition 37 *Given a set of variables Y, $X_1,\ldots,X_n$ such that each variable has a finite number of totally ordered states, the distribution $p(y|x_1,\ldots,x_n)$ is* lattice with respect to state y^0 *if the function $f(x_1,\ldots,x_n) = p(y^0|x_1,\ldots,x_n)$ is lattice.*

Definition 38 *Given a set of variables Y, $X_1,\ldots,X_n$ such that each variable has a finite number of totally ordered states, the distribution $p(y|x_1,\ldots,x_n)$ is* totally strictly positive with respect to state y^0 *if the function $f(x_1,\ldots,x_n) = p(y^0|x_1,\ldots,x_n)$ is totally strictly positive.*

We further concentrate on local distributions that are binary-like. In describing such distributions, we need the concept of a *distinguished* state of a variable. For the remainder of the paper, we use x^0 to denote the distinguished state of variable X.

Definition 39 *Given a set of variables Y, $X_1,\ldots,X_n$ such that each variable has a finite number of states and each variable X has a distinguished state x^0, the local distribution $p(y|x_1,\ldots,x_n)$ is* binary-like *if*

$$y \neq y^0 \text{ and } y' \neq y^0 \text{ implies } p(y|x_1,\ldots,x_n) = p(y'|x_1,\ldots,x_n) \tag{29}$$

$$x_i = x_i' \text{ or } (x_i \neq x_i^0 \text{ and } x_i' \neq x_i^0)\ i = 1,\ldots,n \text{ implies } p(y|x_1,\ldots,x_n) = p(y|x_1',\ldots,x_n'). \tag{30}$$

If condition 30 is satisfied for some particular state y, then $p(y|x_1,\ldots,x_n)$ is said to be binary-like with respect to y.

Thus, for a local distribution that is binary-like, if any non-distinguished state is replaced with another non-distinguished state on either side of the conditioning bar, the conditional probability remains the same. For a local distribution that is binary-like with respect to y, if any non-distinguished state on the right-hand side of the conditioning bar is replaced with another non-distinguished state, the conditional probability for state y remains the same. When appropriate, we use x^1 to denote an arbitrary non-distinguished state of X.

When working with distributions that are both binary-like and either lattice or totally strictly positive, we need to be careful how we assign the total ordering to the states for each variable X. In particular, in order for a distribution to be both binary-like and either lattice or totally strictly positive, all non-distinguished states must have equal order in the (non-strict) total ordering. We incorporate this condition in the following definitions. In addition, we use the ordering $x^0 > x^1$ for all variables X.

Definition 40 *Given a set of variables $Y, X_1, \ldots, X_n$ such that each variable has a finite number of totally ordered states, and each variable X has a distinguished state x^0, the distribution $p(y|x_1, \ldots, x_n)$ is* binary-like lattice (BLL) *if (1) the distribution is lattice with respect to y^0, (2) the distribution is binary-like and (3) if, for each variable X, x^0 is greatest in order and all non-distinguished states of X are equal in order. The distribution $p(y|x_1, \ldots, x_n)$ is* binary-like lattice (BLL) with respect to y^0 *if (1) the distribution is lattice with respect to y^0, (2) the distribution is binary-like with respect to y^0 and (3) if, for each variable X, x^0 is greatest in order and all non-distinguished states of X are equal in order.*

Definition 41 *Given a set of variables $Y, X_1, \ldots, X_n$ such that each variable has a finite number of totally ordered states, and each variable X has a distinguished state x^0, the distribution $p(y|x_1, \ldots, x_n)$ is* binary-like totally strictly positive (BLTSP) *if (1) the distribution is totally strictly positive with respect to y^0, (2) the distribution is binary-like and (3) if, for each variable X, x^0 is greatest in order and all non-distinguished states of X are equal in order. The distribution $p(y|x_1, \ldots, x_n)$ is* binary-like totally strictly positive (BLTSP) with respect to y^0 *if (1) the distribution is binary-like with respect to y^0, (2) the distribution is totally strictly positive with respect to y^0, and (3) if, for each variable X, x^0 is greatest in order and all non-distinguished states of X are equal in order.*

In the following sections, we consider the graph transformations of Section B.1, and investigate the conditions under which the BLL and the BLTSP properties are retained in the distributions that result from a transformation. We will say that a node is BLL (BLTSP) to mean that the local distribution for that node is BLL (BLTSP).

C.2.1 BLL AND BLTSP FOR THE EDGE-DELETION TRANSFORMATION

Lemma 42 (Edge Deletion, BLL and BLTSP) *Let $\mathcal{G}$ be a model containing the edge $O \rightarrow U$, where $O \in \mathbf{O}$ is observed, and let $\mathcal{G}^T$ denote the model that results from applying the edge-deletion transformation on $O \rightarrow U$ in $\mathcal{G}$. If U is BLL in $\mathcal{G}$ then U is BLL in $\mathcal{G}^T$, and if U is BLTSP in $\mathcal{G}$ than U is BLTSP in $\mathcal{G}^T$.*

Proof: Let $\mathbf{T}$ be the set of parents of U other than O. From the definition of an edge deletion, $\mathbf{T}$ will be the parents of U in $\mathcal{G}^T$ and, assuming (e.g.) o^* is the observed value of O, we have

$$p^T(u|\mathbf{t}) = p(u|\mathbf{t}, o^*)$$

for all u and $\mathbf{t}$. From the definition of BLL, if $p(u|\mathbf{t},o)$ is BLL, then it is also BLL when restricted to $o = o^*$. Similarly, from the definition of BLTSP, if $p(u|\mathbf{t},o)$ is BLTSP, then it is also BLTSP when restricted to $o = o^*$. □

C.2.2 BLL AND BLTSP FOR THE BARREN-NODE-REMOVAL TRANSFORMATION

Proposition 43 *Let $\mathcal{G}$ be a model containing barren node X, and let $\mathcal{G}^T$ denote the model that results from applying a barren-node-removal transformation to $\mathcal{G}$ on X. For any node $Y \neq X$ in $\mathcal{G}$, we have: (1) if Y is BLL in $\mathcal{G}$, then Y is BLL in $\mathcal{G}^T$, and (2) if Y is BLTSP in $\mathcal{G}$, then Y is BLTSP in $\mathcal{G}^T$.*

Proof: Follows immediately from the definition of a barren-node removal because the local distributions that remain in $\mathcal{G}^T$ are identical in $\mathcal{G}$. □

C.2.3 BLL AND BLTSP FOR THE OCS TRANSFORMATION

Lemma 44 (OCS, BLL in Y) *Consider the OCS transformation shown in Figure 12, where $p(h|\mathbf{x},\mathbf{z})$ is BLL. If $p(y_j|\mathbf{z},\mathbf{w},h)$ is BLL with respect to y_j^0, $j = 1,\ldots,m$, then $p^T(y|\mathbf{x},\mathbf{z},\mathbf{w}) = p(\mathbf{y}|\mathbf{x},\mathbf{z},\mathbf{w})$ is BLL with respect to $y^0 = \mathbf{y}^0$. If $m = 1$ and $p(\mathbf{y}|\mathbf{z},\mathbf{w},h) = p(y_1|\mathbf{z},\mathbf{w},h)$ is BLL, then $p^T(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is BLL.*

Proof: For notational simplicity, we use Y to denote the set of Y_j nodes in the original graph $\mathcal{G}$; that is, we use the node Y from $\mathcal{G}^T$ as shorthand for the set of nodes that were combined to create that variable.

First, we show that $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is either binary-like or binary-like with respect to y^0. From the sum rule of probability, we have

$$p(y|\mathbf{x},\mathbf{z},\mathbf{w}) = p(h^0|\mathbf{x},\mathbf{z})\ p(y|\mathbf{z},\mathbf{w},h^0) + \sum_{h \neq h^0} p(h|\mathbf{x},\mathbf{z})\ p(y|\mathbf{z},\mathbf{w},h). \tag{31}$$

When $m = 1$ and $p(y|\mathbf{z},\mathbf{w},h)$ is binary-like, because $p(h|\mathbf{x},\mathbf{z})$ is also binary like, we can rewrite Equation 31 as follows:

$$p(y|\mathbf{x},\mathbf{z},\mathbf{w}) = p(h^0|\mathbf{x},\mathbf{z})\ p(y|\mathbf{z},\mathbf{w},h^0) + (1 - p(h^0|\mathbf{x},\mathbf{z}))\ p(y|\mathbf{z},\mathbf{w},h^1). \tag{32}$$

Because $p(h|\mathbf{x},\mathbf{z})$ is binary-like with respect to h^0 and the remaining two terms in Equation 32 are binary-like, it follows that $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like. When $m \geq 1$ and $p(y_j|\mathbf{z},\mathbf{w},h)$ is binary-like with respect to y_j^0, $j = 1,\ldots,m$, Equation 32 with $Y = y^0$ still holds, because $p(y|\mathbf{z},\mathbf{w},h)$ is binary-like with respect to y^0. Consequently, $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like with respect to y^0. It remains to show that $p(y^0|\mathbf{x},\mathbf{z},\mathbf{w})$ is lattice. There are three cases to consider.

Case 1: $X \in \mathbf{X}$ changes. If $\mathbf{X}$ is empty, there is nothing to prove, so assume $\mathbf{X}$ is not empty. Here, we need to show that $p(y^0|x^0,\mathbf{x}',\mathbf{z},\mathbf{w}) > p(y^0|x^1,\mathbf{x}',\mathbf{z},\mathbf{w})$, where $\mathbf{X}' = \mathbf{X} \setminus \{X\}$. Using Equation 32 with $Y = y^0$ and omitting those variables that are held constant, we rewrite this condition as

$$\begin{aligned} p(h^0|x^0)\ p(y^0|h^0) + (1 - p(h^0|x^0))\ p(y^0|h^1) > \\ p(h^0|x^1)\ p(y^0|h^0) + (1 - p(h^0|x^1))\ p(y^0|h^1). \end{aligned} \tag{33}$$

Because $p(y_j^0|h)$ is lattice, $j = 1,\ldots,m$, we know that $p(y^0|h^i) = \prod_{j=1}^m p(y_j^0|h^i)$ is lattice—that is, $p(y^0|h^0) > p(y^0|h^1)$. Because $p(h^0|x)$, we have $p(h^0|x^0) > p(h^0|x^1)$. Consequently, inequality 33 holds.

Case 2: $W \in \mathbf{W}$ changes. If $\mathbf{W}$ is empty there is nothing to prove, so assume $\mathbf{W}$ is not empty. Here, we need to show that $p(y^0|\mathbf{x},\mathbf{z},w^0,\mathbf{w}') > p(y^0|\mathbf{x},\mathbf{z},w^1,\mathbf{w}')$, where $\mathbf{W}' = \mathbf{W} \setminus \{W\}$. Again using Equation 32 and omitting those variables that are held constant, this condition becomes

$$\begin{aligned} &p(h^0)\ p(y^0|w^0,h^0) + (1-p(h^0))\ p(y^0|w^0,h^1) > \\ &\quad p(h^0)\ p(y^0|w^1,h^0) + (1-p(h^0))\ p(y^0|w^1,h^1). \end{aligned} \tag{34}$$

First note that $p(y^0|w,h)$ is lattice. To see this fact, write

$$p(y^0|w,h) = \prod_a p(y_a^0|w,h) \prod_b p(y_b^0|h),$$

where each Y_a has parents W and H and each Y_b has parent H. Because there is at least one Y_a and $p(y_a^0|w,h)$ is lattice, it follows that $p(y^0|w,h)$ is lattice. Identifying $p(h^0)$ with $\alpha_1 = \alpha_2$ and $p(y^0|w^i,h^j)$ with f^{ij} in Proposition 29, and noting that f^{ij} is lattice because $p(y^0|w,h)$ is lattice, we find that inequality 34 holds.

Case 3: $Z \in \mathbf{Z}$ changes. If $\mathbf{Z}$ is empty there is nothing to prove, so assume $\mathbf{Z}$ is not empty. Here, we need to show that $p(y^0|\mathbf{x},z^0,\mathbf{z}',\mathbf{w}) > p(y^0|\mathbf{x},z^1,\mathbf{z}',\mathbf{w})$, where $\mathbf{Z}' = \mathbf{Z} \setminus \{Z\}$. Using Equation 32, this condition becomes

$$\begin{aligned} &p(h^0|z^0)\ p(y^0|z^0,h^0) + (1-p(h^0|z^0))\ p(y^0|z^0,h^1) > \\ &\quad p(h^0|z^1)\ p(y^0|z^1,h^0) + (1-p(h^0|z^1))\ p(y^0|z^1,h^1). \end{aligned} \tag{35}$$

By an argument analogous to that in Case 2 of this Lemma, it follows that $p(y^0|z,h)$ is lattice. Thus, identifying $p(h^0|z^i)$ with α_i and $p(y^0|z^i,h^j)$ with f^{ij} in Proposition 29 and using the fact that $p(h^0|z)$ and $p(y^0|z,h)$ are lattice, we establish inequality 35. □

Lemma 45 (OCS, BLL in H) *Consider the OCS transformation shown in Figure 12, where $p(h|\mathbf{x},\mathbf{z})$ is BLL. If $p(y_j|\mathbf{z},\mathbf{w},h)$ is BLTSP with respect to y_j^0, $j = 1,\ldots,m$, then $p^T(h|\mathbf{x},\mathbf{z},\mathbf{w}) = p(h|\mathbf{y}^0,\mathbf{x},\mathbf{z},\mathbf{w})$ is BLL.*

Proof: As in the proof of Lemma 44, we use Y to denote the set of Y_j nodes in the original graph $\mathcal{G}$.

From Bayes' rule and the definition of Y, we have

$$p(h|y^0,\mathbf{x},\mathbf{z},\mathbf{w}) = \frac{p(h|\mathbf{x},\mathbf{z})\ p(y^0|\mathbf{z},\mathbf{w},h)}{p(h^0|\mathbf{x},\mathbf{z})\ p(y^0|\mathbf{z},\mathbf{w},h^0) + (1-p(h^0|\mathbf{x},\mathbf{z}))\ p(y^0|\mathbf{z},\mathbf{w},h^1)}. \tag{36}$$

where $p(y^0|\mathbf{z},\mathbf{w},h) = \prod_{j=1}^m p(y_j^0|\mathbf{z},\mathbf{w},h)$. Because $p(h|\mathbf{x},\mathbf{z})$ is binary-like and $p(y_j|\mathbf{z},\mathbf{w},h)$ is binary-like with respect to y_j^0, $i = 1,\ldots,m$, it follows that $p(h|y^0,\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like. It remains to show that $p(h^0|y^0,\mathbf{x},\mathbf{z},\mathbf{w})$ is lattice.

Dividing the numerator and denominator of the right-hand-side of Equation 36 by the numerator and setting h to h^0, we obtain

$$p(h^0|y^0,\mathbf{x},\mathbf{z},\mathbf{w}) = \frac{1}{1 + \frac{(1-p(h^0|\mathbf{x},\mathbf{z}))}{p(h^0|\mathbf{x},\mathbf{z})} \frac{p(y^0|\mathbf{z},\mathbf{w},h^1)}{p(y^0|\mathbf{z},\mathbf{w},h^0)}}. \tag{37}$$

There are three cases to consider.

Case 1: $X \in \mathbf{X}$ changes. If $\mathbf{X}$ is empty, there is nothing to prove, so assume $\mathbf{X}$ is not empty. Here, we need to show that $p(h^0|y^0,x^0,\mathbf{x}',\mathbf{z},\mathbf{w}) > p(h^0|y^0,x^1,\mathbf{x}',\mathbf{z},\mathbf{w})$, where $\mathbf{X}' = \mathbf{X} \setminus \{X\}$. Using Equation 37 and the fact that $1/(1+a) > 1/(1+b)$ if and only if $1/a > 1/b$ for positive a and b, this condition becomes

$$\frac{p(h^0|x^0)}{(1-p(h^0|x^0))}\frac{p(y^0|h^0)}{p(y^0|h^1)} > \frac{p(h^0|x^1)}{(1-p(h^0|x^1))}\frac{p(y^0|h^0)}{p(y^0|h^1)},$$

which holds because $p(h^0|x)$ is lattice.

Case 2: $W \in \mathbf{W}$ changes. If $\mathbf{W}$ is empty, there is nothing to prove, so assume $\mathbf{W}$ is not empty. Here, we need to show that $p(h^0|y^0,\mathbf{x},\mathbf{z},w^0,\mathbf{w}') > p(h^0|y^0,\mathbf{x},\mathbf{z},w^1,\mathbf{w}')$, where $\mathbf{W}' = \mathbf{W} \setminus \{W\}$. By an argument similar to that in Case 1, this condition becomes

$$\frac{p(h^0)}{(1-p(h^0))}\frac{p(y^0|w^0,h^0)}{p(y^0|w^0,h^1)} > \frac{p(h^0)}{(1-p(h^0))}\frac{p(y^0|w^1,h^0)}{p(y^0|w^1,h^1)}.$$

Canceling the terms involving $p(h^0)$, we see that this inequality holds if $p(y^0|h,w)$ is totally strictly positive. To establish the latter fact, recall that

$$p(y^0|w,h) = \prod_a p(y_a^0|w,h)\prod_b p(y_b^0|h),$$

where each Y_a has parents W and H and each Y_b has parent H. Now note that the product of two functions that are positive and totally strictly positive is also positive and totally strictly positive, and that if $f(X_1,X_2)$ is positive and totally strictly positive and $g(X_1)$ is positive, then $f(X_1,X_2)\cdot g(X_1)$ is positive and totally strictly positive.

Case 3: $Z \in \mathbf{Z}$ changes. If $\mathbf{Z}$ is empty, there is nothing to prove, so assume $\mathbf{X}$ is not empty. Here, we need to show that $p(h^0|y^0,\mathbf{x},z^0,\mathbf{z},\mathbf{w}) > p(h^0|y^0,\mathbf{x},z^1,\mathbf{z},\mathbf{w})$, where $\mathbf{Z}' = \mathbf{Z} \setminus \{Z\}$. This condition becomes

$$\frac{p(h^0|z^0)}{(1-p(h^0|z^0))}\frac{p(y^0|z^0,h^0)}{p(y^0|z^0,h^1)} > \frac{p(h^0|z^1)}{(1-p(h^0|z^1))}\frac{p(y^0|z^1,h^0)}{p(y^0|z^1,h^1)}. \tag{38}$$

By an argument analogous to the last one in Case 2, $p(y^0|z,h)$ is totally strictly positive. Also, $p(h^0|x)$ is lattice. The inequality therefore holds. □

In the base case of the proof of our main result (Theorem 53), we require only that the distributions have the BLL property. The proof of the preceding lemma shows why BLTSP is a required property in the original model. Namely, in Case 2, $p(h^0|y^0,w)$ is lattice if and only if $p(y^0|w,h)$ is totally strictly positive.

Lemma 46 (OCS, BLTSP in Y) *Consider the OCS transformation shown in Figure 12, where $p(h|\mathbf{x},\mathbf{z})$ is BLL and BLTSP. If $p(y_j|\mathbf{z},\mathbf{w},h)$ is BLL with respect to y_j^0 and BLTSP with respect to y_j^0, $j = 1,\ldots,m$, then $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is BLTSP with respect to y^0. If $m = 1$ and $p(y|\mathbf{z},\mathbf{w},h) = p(y_1|\mathbf{z},\mathbf{w},h)$ is BLL and BLTSP, then $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is BLTSP.*

Proof: In the proof of Lemma 44, we showed that (1) if $m = 1$ and $p(y|\mathbf{z},\mathbf{w},h)$ is binary-like, then $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like; and (2) if $m \geq 1$ and $p(y_j|\mathbf{z},\mathbf{w},h)$ is binary-like with respect to y_j^0, $j = 1,\ldots,m$, then $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like with respect to y^0. It remains to show that $p(y^0|\mathbf{x},\mathbf{z},\mathbf{w})$ is totally strictly positive. There are six cases to consider.

Case 1: $X_1, X_2 \in \mathbf{X}$ changes. Assume $|\mathbf{X}| \geq 2$. Here, we need to show that

$$\frac{p(y^0|x_1^0, x_2^0, \mathbf{x}', \mathbf{z}, \mathbf{w})}{p(y^0|x_1^0, x_2^1, \mathbf{x}', \mathbf{z}, \mathbf{w})} > \frac{p(y^0|x_1^1, x_2^0, \mathbf{x}', \mathbf{z}, \mathbf{w})}{p(y^0|x_1^1, x_2^1, \mathbf{x}', \mathbf{z}, \mathbf{w})}. \tag{39}$$

where $\mathbf{X}' = \mathbf{X} \setminus \{X_1, X_2\}$. Using Equation 32 for $Y = y^0$ and omitting those variables that are held constant, we rewrite this condition as

$$\begin{aligned} &\frac{p(h^0|x_1^0, x_2^0)\ p(y^0|h^0) + (1 - p(h^0|x_1^0, x_2^0))\ p(y^0|h^1)}{p(h^0|x_1^0, x_2^1)\ p(y^0|h^0) + (1 - p(h^0|x_1^0, x_2^1))\ p(y^0|h^1)} > \\ &\quad \frac{p(h^0|x_1^1, x_2^0)\ p(y^0|h^0) + (1 - p(h^0|x_1^1, x_2^0))\ p(y^0|h^1)}{p(h^0|x_1^1, x_2^1)\ p(y^0|h^0) + (1 - p(h^0|x_1^1, x_2^1))\ p(y^0|h^1)}. \end{aligned} \tag{40}$$

Because $p(y_j^0|h)$ is lattice, $j = 1, \ldots, m$, we know that $p(y^0|h) = \prod_{j=1}^m p(y_j^0|h)$ is lattice. Thus, identifying $p(h^0|x_1^i, x_2^j)$ with f^{ij} and $p(y^0|h^i)$ with g^i in Proposition 32, we find that inequality 40 holds.

Case 2: $X \in \mathbf{X}$ and $W \in \mathbf{W}$ changes. Assume $|\mathbf{X}| \geq 1$ and $|\mathbf{W}| \geq 1$. Using Equation 32, we need to show the inequality

$$\begin{aligned} &\frac{p(h^0|x^0)\ p(y^0|w^0, h^0) + (1 - p(h^0|x^0))\ p(y^0|w^0, h^1)}{p(h^0|x^0)\ p(y^0|w^1, h^0) + (1 - p(h^0|x^0))\ p(y^0|w^1, h^1)} > \\ &\quad \frac{p(h^0|x^1)\ p(y^0|w^0, h^0) + (1 - p(h^0|x^1))\ p(y^0|w^0, h^1)}{p(h^0|x^1)\ p(y^0|w^1, h^0) + (1 - p(h^0|x^1))\ p(y^0|w^1, h^1)}. \end{aligned} \tag{41}$$

By an argument analogous to one in Case 2 of Lemma 45, we know that $p(y^0|w, h)$ is totally strictly positive. Therefore, identifying $p(y^0|w^i, h^j)$ with g^{ij} in Proposition 31 and noting that $p(h^0|x^0) > p(h^0|x^1)$, we establish inequality 41.

Case 3: $X \in \mathbf{X}$ and $Z \in \mathbf{Z}$ changes. Assume $|\mathbf{X}| \geq 1$ and $|\mathbf{Z}| \geq 1$. Using Equation 32, we need to show the inequality

$$\begin{aligned} &\frac{p(h^0|x^0, z^0)\ p(y^0|z^0, h^0) + (1 - p(h^0|x^0, z^0))\ p(y^0|z^0, h^1)}{p(h^0|x^0, z^1)\ p(y^0|z^1, h^0) + (1 - p(h^0|x^0, z^1))\ p(y^0|z^1, h^1)} > \\ &\quad \frac{p(h^0|x^1, z^0)\ p(y^0|z^0, h^0) + (1 - p(h^0|x^1, z^0))\ p(y^0|z^0, h^1)}{p(h^0|x^1, z^1)\ p(y^0|z^1, h^0) + (1 - p(h^0|x^1, z^1))\ p(y^0|z^1, h^1)}. \end{aligned}$$

Because $p(y^0|z^0, h) = \prod_{j=1}^m p(y_j^0|z^0, h)$, we know that $p(y^0|z^0, h)$ is lattice. By an argument analogous to one in Case 2 of Lemma 45, we know that $p(y^0|z, h)$ is totally strictly positive. Identifying $p(h^0|x^i, z^j)$ with f^{ij} and $p(y^0|z^i, h^j)$ with g^{ij}, Proposition 33 establishes this identity.

Case 4: $W_1, W_2 \in \mathbf{W}$ changes. Assume $|\mathbf{W}| \geq 2$. Using Equation 32, we need to show the inequality

$$\begin{aligned} &\frac{p(h^0)\ p(y^0|w_1^0, w_2^0, h^0) + (1 - p(h^0))\ p(y^0|w_1^0, w_2^0, h^1)}{p(h^0)\ p(y^0|w_1^0, w_2^1, h^0) + (1 - p(h^0))\ p(y^0|w_1^0, w_2^1, h^1)} > \\ &\quad \frac{p(h^0)\ p(y^0|w_1^1, w_2^0, h^0) + (1 - p(h^0))\ p(y^0|w_1^1, w_2^0, h^1)}{p(h^0)\ p(y^0|w_1^1, w_2^1, h^0) + (1 - p(h^0))\ p(y^0|w_1^1, w_2^1, h^1)}. \end{aligned}$$

Identifying $p(h^0)$ with f and $p(y^0|w_1^i, w_2^j, h^k)$ with g^{ijk}, Proposition 34 establishes this identity if we can show that g^{ijk} satisfies the strict and non-strict total positivity conditions of the proposition. To do so, write

$$p(y^0|w_1,w_2,h) = \prod_a p(y_a^0|w_1,w_2,h) \prod_b p(y_b^0|w_1,h) \prod_c p(y_c^0|w_2,h) \prod_d p(y_c^0|h),$$

where each Y_a has parents W_1, W_2, and H, each Y_b has parents W_1 and H, each Y_c has parents W_2 and H, and each Y_d has parent H. It is not difficult to show that, if there is at least one variable having both W_1 and W_2 as parents, then the product is totally strictly positive. If there is no such variable, however, the product is not totally strictly positive, because $g000/g010 = g100/g110$ and $g001/g011 = g101/g111$. Nonetheless, it is not difficult to show that the remaining four pairwise total strict positivity conditions hold. Consequently, the conditions of Proposition 34 hold.

Case 5: $Z \in \mathbf{Z}$ and $W \in \mathbf{W}$ changes. Assume $|\mathbf{Z}| \geq 1$ and $|\mathbf{W}| \geq 1$. Using Equation 32, we need to show the inequality

$$\frac{p(h^0|z^0)\ p(y^0|z^0,w^0,h^0) + (1-p(h^0|z^0))\ p(y^0|z^0,w^0,h^1)}{p(h^0|z^0)\ p(y^0|z^0,w^1,h^0) + (1-p(h^0|z^0))\ p(y^0|z^0,w^1,h^1)} > \frac{p(h^0|z^1)\ p(y^0|z^1,w^0,h^0) + (1-p(h^0|z^1))\ p(y^0|z^1,w^0,h^1)}{p(h^0|z^1)\ p(y^0|z^1,w^1,h^0) + (1-p(h^0|z^1))\ p(y^0|z^1,w^1,h^1)}.$$

Identifying $p(h^0|z^i)$ with f^i and $p(y^0|z^i, w^j, h^k)$ with g^{ijk}, Proposition 35 establishes this identity if g^{ijk} satisfies the strict and non-strict total positivity conditions of the Proposition. By an argument analogous to one in Case 4 of this Lemma, g^{ijk} satisfies these conditions.

Case 6: $Z_1, Z_2 \in \mathbf{Z}$ changes. Assume $|\mathbf{Z}| \geq 2$. Using Equation 32, we need to show the inequality

$$\frac{p(h^0|z_1^0,z_2^0)\ p(y^0|z_1^0,z_2^0,h^0) + (1-p(h^0|z_1^0,z_2^0))\ p(y^0|z_1^0,z_2^0,h^1)}{p(h^0|z_1^0,z_2^1)\ p(y^0|z_1^0,z_2^1,h^0) + (1-p(h^0|z_1^0,z_2^1))\ p(y^0|z_1^0,z_2^1,h^1)} > \frac{p(h^0|z_1^1,z_2^0)\ p(y^0|z_1^1,z_2^0,h^0) + (1-p(h^0|z_1^1,z_2^0))\ p(y^0|z_1^1,z_2^0,h^1)}{p(h^0|z_1^1,z_2^1)\ p(y^0|z_1^1,z_2^1,h^0) + (1-p(h^0|z_1^1,z_2^1))\ p(y^0|z_1^1,z_2^1,h^1)}.$$

Identifying $p(h^0|z_1^i, z_2^j)$ with f^{ij} and $p(y^0|z_1^i, z_2^j, h^k)$ with g^{ijk}, Proposition 36 establishes this identity if g^{ijk} satisfies the conditions of the Proposition. Because $p(y^0|z_1^0, x_2^0, h) = \prod_{j=1}^m p(y_j^0|z_1^0, z_2^0, h)$, we know that $p(y^0|z_1^0, z_2^0, h)$ is lattice. By an argument analogous to that of Case 4 of this Lemma, g^{ijk} satisfies the strict and non-strict total positivity conditions of Proposition 36. □

Putting the three previous lemmas together, we get the following general result for the OCS transformation.

Corollary 47 (OCS transformation) *Let $\mathcal{G}$ be a Bayesian network, and let $\mathcal{G}^T$ be the result of applying the observed-child-separation transformation on unobserved node H with observed children $\mathbf{Y}$. If the observed values for the children $\mathbf{Y}$ are the distinguished states $\mathbf{y}^0$ for those children, and if in $\mathcal{G}$, H is both BLL and BLTSP, and each observed child $Y_i \in \mathbf{Y}$ is both BLL with respect to y_i^0 and BLTSP with respect to y_i^0, then in $\mathcal{G}^T$, (1) H is BLL, (2) $Y = Comp(\mathbf{Y})$ is BLL with respect to y^0, and (3) $Y = Comp(\mathbf{Y})$ BLTSP with respect to y^0.*

Proof: (1), (2), and (3) follow immediately from Lemma 45, Lemma 44, and Lemma 46, respectively. □

C.2.4 BLL AND BLTSP FOR THE EDGE-REVERSAL TRANSFORMATION

Corollary 48 (Edge Reversal, BLL in Y) *Consider the edge-reversal transformation shown in Figure 8. If the local distributions for H and Y are BLL in $\mathcal{G}$, then the local distribution for Y in $\mathcal{G}^T$ is BLL.*

Proof: The local distribution for Y after reversing $H \rightarrow Y$ is exactly the same as the local distribution for Y after a one-child OCS transformation for $H \rightarrow Y$; thus, the corollary follows from the $m = 1$ case of Lemma 44. □

Corollary 49 (Edge Reversal, BLTSP in Y) *Consider the edge-reversal transformation shown in Figure 8. If the local distribution for H is BLL and BLTSP in $\mathcal{G}$, and if the local distribution for Y is BLL and BLTSP in $\mathcal{G}$, then the local distribution for Y in $\mathcal{G}^T$ is BLTSP.*

Proof: The local distribution for Y after reversing $H \rightarrow Y$ is exactly the same as the local distribution for Y after a one-child OCS transformation for $H \rightarrow Y$; thus, the corollary follows from the $m = 1$ case of Lemma 46. □

C.2.5 BLL FOR THE UPS ALGORITHM

We now show that if every node in $\mathcal{G}$ is BLL, then every node in the graph that results from applying the UPS algorithm is also BLL.

Lemma 50 *Let $\mathcal{G}$ be any Bayesian network, and let $\mathcal{G}^T$ be the result of applying the UPS algorithm with Bayesian network $\mathcal{G}$ and non-root node Y. If every node in $\mathcal{G}$ is BLL, then every node in $\mathcal{G}^T$ is BLL.*

Proof: The result follows because, from Corollary 48, after each edge $H \rightarrow Y$ is reversed, Y remains BLL; the property need not hold for H after the reversal because H is immediately removed. □

C.2.6 BLL FOR THE ONE ALGORITHM

Recall Algorithm ONE from Section B.2.2 which eliminates all observed nodes from a model. In this section, we show conditions under which the nodes that remain after the algorithm are BLL. As in Section B.2.2, we use $\mathcal{G}^i$ to denote the Bayesian network that results after i iterations of the While loop at step 3 of the algorithm, we define $\mathcal{G}^0$ to be the graph $\mathcal{G}^T$ that results after applying step 2 but before the first iteration of the While loop at step 3, we use H^i to denote the (lowest) node chosen in iteration i of the While loop, we use $\mathbf{Y}^i$ to denote the set of observed children of H^i on iteration i of the While loop, and we use $Y^i = Comp(\mathbf{Y}^i)$ to denote the composite node created by the OCS transformation in iteration i of the While loop.

Lemma 51 *Let $\mathcal{G}$ be a Bayesian network in which all nodes are both BLL and BLTSP, and let $\mathbf{o}$ be a set of observations for nodes $\mathbf{O}$. If for every $O \in \mathbf{O}$ that has at least one parent not in $\mathbf{O}$, the observation in $\mathbf{o}$ for O is the distinguished state o^0, then every node in the Bayesian network $\mathcal{G}^T$ that results from applying Algorithm ONE to $\mathcal{G}$ is BLL.*

Proof: From Lemma 42, we know that we retain the BLL and BLTSP properties of all nodes while removing edges in step 2, and thus all nodes in $\mathcal{G}^0$ are both BLL and BLTSP. Similarly,

from Proposition 43, step 4 does not affect the BLL or BLTSP properties of the nodes that remain. Thus, the lemma follows if we can show that for every OCS transformation applied in step 3 of the algorithm, every non-observed node retains the BLL property.

Consider the *ith* iteration of the While loop in step 3. From Corollary 47, if (1) the observed state for each child in $\mathbf{Y}^i$ is the distinguished state for that child, (2) H^i is both BLL and BLTSP, and (3) each child in $\mathbf{Y}^i$ is both BLL with respect to its distinguished state and BLTSP with respect to its distinguished state, then we are guaranteed that all observed variables retain the BLL property. We now demonstrate that these three preconditions of Corollary 47 hold for every iteration *i*.

After applying step 2 of the algorithm, any observed node without at least one observed parent will be completely disconnected from the graph, and thus precondition (1) is always satisfied. From Corollary 23, each unobserved node is chosen at most once in step 3. Because the parents (and hence the local distribution) for an unobserved node only change when it is chosen in step 3, we conclude that precondition (2) is always satisfied.

The only local distributions for nodes in $\mathbf{O}$ that change in iteration *i* are the nodes $\mathbf{Y}^i$, which are replaced by the single node Y^i. From Corollary 47, if preconditions (1) and (2) hold, and if every node $O \in \mathbf{O}$ is both BLL with respect to o^0 and BLTSP with respect to o^0 *before* the transformation, then every node $O \in \mathbf{O}$ is both BLL with respect to o^0 and BLTSP with respect to o^0 *after* the transformation. Because all nodes in $\mathbf{O}$ are initially both BLL with respect to their distinguished states and BLTSP with respect to their distinguished states, precondition (3) always holds and the lemma follows. □

Appendix D. Main Results

In this appendix, we prove Lemma 17 and Lemma 5 using results established in the previous appendices.

Lemma 52 *Let $\mathcal{G}$ be any Bayesian network in which all local distributions are BLL, and let X be any root node in $\mathcal{G}$. If X and Y are d-connected by an $\emptyset$-active path in $\mathcal{G}$, then $p(y^0|x^0) > p(y^0|x^1)$.*

Proof: From Lemma 50, we can apply Algorithm UPS to $\mathcal{G}$ and node Y, and every node in the resulting model $\mathcal{G}^T$ will be BLL. Furthermore, we know from Lemma 20 that X is a root-node parent of Y, and that all other nodes $\mathbf{Z}$ in $\mathcal{G}^T$ are also root-node parents of Y. Expressing the difference of interest using $\mathcal{G}^T$:

$$p(y^0|x^0) - p(y^0|x^1) = \left[\sum_{\mathbf{Z}} p^T(y^0|x^0,\mathbf{z})p^T(\mathbf{z}|x^0)\right] - \left[\sum_{\mathbf{Z}} p^T(y^0|x^1,\mathbf{z})p^T(\mathbf{z}|x^0)\right].$$

Because all nodes in $\mathbf{Z}$ are d-separated from X in $\mathcal{G}^T$ whenever Y is not in the conditioning set we have

$$p(y^0|x^0) - p(y^0|x^1) = \sum_{\mathbf{Z}} p(\mathbf{z}) \left[p^T(y^0|x^0,\mathbf{z}) - p^T(y^0|x^1,\mathbf{z})\right].$$

Every difference in the sum above is guaranteed to be greater than zero by definition of BLL. □

Theorem 53 *Let $\mathcal{G}$ be a Bayesian network in which all conditional distributions are BLL and BLTSP, and let $\mathbf{o}$ be a set of observations for nodes $\mathbf{O}$. If for every $O \in \mathbf{O}$ that has at least one parent not in $\mathbf{O}$, the observation in $\mathbf{o}$ for O is the distinguished state o^0, then if there is a $\mathbf{O}$-active path between X and Y in $\mathcal{G}$, then $p(y^0|x^0,\mathbf{o}^0) > p(y^0|x^1,\mathbf{o}^0)$.*

Proof: Without loss of generality, assume that X is not a descendant of Y in $\mathcal{G}$. Let $UAnc(X)$ denote the set of unobserved nodes in $\mathcal{G}$ for which there is a directed path to X though unobserved nodes. In other words, $UAnc(X)$ is the set of ancestors of X if we were to remove all of the nodes in $\mathbf{O}$ from $\mathcal{G}$. We prove the theorem by induction on the size of $UAnc(X)$.

For the basis, we consider the case when $|UAnc(X)| = 0$. From Lemma 51, we can use Algorithm ONE to convert $\mathcal{G}$ into a Bayesian network containing only unobserved nodes and for which every node is BLL. Furthermore, because X has no unobserved parents in $\mathcal{G}$, we can assume by Lemma 24 that X is a root node in the resulting model $\mathcal{G}^T$. Because there is a $\mathbf{O}$-active path between X and Y in $\mathcal{G}$, there must be a $\emptyset$-active path between X and Y in $\mathcal{G}^T$. Thus the base case follows from Lemma 52.

For the induction hypothesis, we assume the theorem is true whenever $|UAnc(X)|$ is less than k, and we consider the case when $|UAnc(X)| = k$. Let Z be any element of $UAnc(X)$ for which no parent of Z is also in $UAnc(X)$; that is, Z is a root-node ancestor of X in the graph that results from removing $\mathbf{O}$ from $\mathcal{G}$. Because $Z \notin \mathbf{O}$, we know that the theorem holds if

$$\begin{array}{rcl} p(z^0|x^0,\mathbf{o}^0)p(y^0|x^0,z^0,\mathbf{o}^0) & + & p(z^1|x^0,\mathbf{o}^0)p(y^0|x^0,z^1,\mathbf{o}^0) \\ & > & \\ p(z^0|x^1,\mathbf{o}^0)p(y^0|x^1,z^0,\mathbf{o}^0) & + & p(z^1|x^1,\mathbf{o}^0)p(y^0|x^1,z^1,\mathbf{o}^0). \end{array}$$

We conclude from Proposition 29—using $\alpha_1 = p(z^0|x^0,\mathbf{o}^0)$, $\alpha_2 = p(z^0|x^1,\mathbf{o}^0)$, and $f^{ij} = p(y^0|x^i,z^j,\mathbf{o}^0)$—that the following four conditions are sufficient to establish $p(y^0|x^0,\mathbf{o}^0) \geq p(y^0|x^1,\mathbf{o}^0)$:

1. $p(z^0|x^0,\mathbf{o}^0) \geq p(z^0|x^1,\mathbf{o}^0)$ (i.e., $\alpha_1 \geq \alpha_2$)
2. $p(y^0|x^0,z^0,\mathbf{o}^0) \geq p(y^0|x^0,z^1,\mathbf{o}^0)$ (i.e., $f^{00} \geq f^{01}$)
3. $p(y^0|x^0,z^1,\mathbf{o}^0) \geq p(y^0|x^1,z^1,\mathbf{o}^0)$ (i.e., $f^{01} \geq f^{11}$)
4. $p(y^0|x^0,z^0,\mathbf{o}^0) \geq p(y^0|x^1,z^0,\mathbf{o}^0)$ (i.e., $f^{00} \geq f^{10}$)

and that either of the following two conditions is sufficient to rule out equality, and thus establish the lemma:

5. $\big(p(y^0|x^0,z^0,\mathbf{o}^0) > p(y^0|x^0,z^1,\mathbf{o}^0)\big) \wedge$ $\big(p(z^0|x^0,\mathbf{o}^0) > p(z^0|x^1,\mathbf{o}^0)\big)$ (i.e., $(\alpha_1 > \alpha_2) \wedge (f^{00} > f^{01})$)
6. $p(y^0|x^0,z^0,\mathbf{o}^0) > p(y^0|x^1,z^0,\mathbf{o}^0)$ (i.e., $f^{00} \geq f^{10}$).

We consider two cases: in $\mathcal{G}$, either X and Y are d-separated by $\mathbf{O} \cup \{Z\}$ or they are not d-separated by $\mathbf{O} \cup \{Z\}$.

Suppose X and Y are d-separated by $\mathbf{O} \cup \{Z\}$. We can immediately conclude that equality holds for both (3) and (4). Because X and Y are not d-separated by $\mathbf{O}$, we conclude both that Z and Y are d-connected given $\mathbf{O}$ and that Y and Z are d-connected given $\mathbf{O} \cup X$. From this first d-connection fact and the fact that $|UAnc(Z)| = 0$, we conclude that (1) is a strict inequality by the base case of the induction. From the second d-connection fact, and because the preconditions of the theorem are not violated by adding $X = x^0$ to the observation set, we conclude that (2) is also a strict inequality by again deferring to the base case of the induction. Thus, all inequalities (1)-(4) hold, with (1)

and (2) holding as strict inequalities. Because condition (5) is simply the conjunction of the strict versions of (2) and (1), the theorem follows.

Suppose X and Y are not d-separated by $\mathbf{O} \cup \{Z\}$. In this case, the two d-connection facts from the previous case may or may not hold. If either or both of them hold, we can show that the corresponding inequality is strict using the same argument as above. If either or both of them do not hold, we conclude that equality holds for the corresponding inequality. Thus, we know that (1) and (2) both hold, although we have no guarantees on strictness. Because all of the parents of Z are necessarily in the conditioning set, the preconditions of the theorem are not violated by adding either $z = z^0$ or $z = z^1$ to the conditioning set. Because the result of either addition reduces $|UAnc(X)|$ by one, we conclude by induction that both (3) and (4) are strict inequalities. Thus, all inequalities (1)-(4) hold. Because condition (6) is simply the strict version of (4), the theorem follows. □

Theorem 53 is closely related to existing results in the QBN literature. In particular, Theorem 4 from Druzdzel and Henrion (1993) implies that in a graph satisfying the non-strict versions of BLL and BLTSP, our Theorem 53 holds except with the conclusion that $p(y^0|x^0, \mathbf{o}^0) \geq p(y^0|x^1, \mathbf{o}^0)$.

We now prove the main results of the appendices. We re-state the corollary here, adopting our convention of using $\mathcal{G}$ to denote both the structure and the parameters of a Bayesian network.

Lemma 17 *Let $\mathcal{G}$ be a Bayesian network in which all local distributions are both BLL and BLTSP. Then the joint distribution represented by $\mathcal{G}$ is perfect with respect to the structure of $\mathcal{G}$.*

Proof: Let $p(\cdot)$ denote the joint distribution defined by $\mathcal{G}$. Because $p(\cdot)$ is defined by a Bayesian network, we know it factors according to the structure of $\mathcal{G}$, and thus we need only show that $p(\cdot)$ is faithful with respect to the structure of $\mathcal{G}$. To demonstrate that $p(\cdot)$ is faithful, we consider an arbitrary d-connection fact in $\mathcal{G}$ and prove that there is a corresponding dependence in $p(\cdot)$. Let X and Y be any pair of nodes in $\mathcal{G}$ that are d-connected by some set $\mathbf{O}$ in $\mathcal{G}$. From Theorem 53, we know that for the observation $\mathbf{O} = \mathbf{o}^0$, we have $p(y^0|x^0, \mathbf{o}^0) > p(y^0|x^1, \mathbf{o}^0)$, and thus $p(\cdot)$ is faithful. □

We now prove Lemma 5; this lemma provides a method for constructing BLL and BLTSP distributions.

Lemma 5 *Let $\mathcal{G}$ be a Bayesian network, let r_Y denote the number of states of node Y, let $\mathbf{Pa}_Y$ denote the set of parents of node Y in $\mathcal{G}$, let $NNZ(\mathbf{pa}_Y)$ denote the number of non-zero elements in the set $\mathbf{pa}_Y$, and let α_X be a constant satisfying $0 < \alpha_X < 1$. If all of the local distributions are defined as*

$$p(y|\mathbf{pa}_Y) = \begin{cases} \alpha_X^{F(\mathbf{pa}_Y)} & \text{if } y = 0 \\ \frac{1}{(r_Y - 1)} \left(1 - \alpha_X^{F(\mathbf{pa}_Y)}\right) & \text{otherwise,} \end{cases} \tag{42}$$

where

$$F(\mathbf{pa}_Y) = 2 - 2^{-NNZ(\mathbf{pa}_Y)},$$

then the distribution defined by $\mathcal{G}$ is perfect with respect to the structure of $\mathcal{G}$.

Proof: Given Lemma 17, we need only show that $p(y|\mathbf{pa}_Y)$ is BLL and BLTSP. For every variable in $\mathcal{G}$, we define the distinguished state to be state zero, and we order the states such that state zero is greatest and all non-zero states are equal. Thus, according to the definition of BLL (Definition 40) and the definition of BLTSP (Definition 41), we need to show that $p(y|\mathbf{pa}_Y)$ is binary-like, lattice with respect to $y = 0$, and totally strictly positive with respect to $y = 0$.

Due to the definition of F in Equation 42, it follows immediately from Definition 39 that $p(y|\mathbf{pa}_Y)$ is binary-like. We now show that $p(y|\mathbf{pa}_Y)$ is lattice with respect to $y = 0$. From Equation

42, this follows as long as

$$\alpha^{2-\left(\frac{1}{2}\right)^{NNZ(\mathbf{pa}_Y^1)}} > \alpha^{2-\left(\frac{1}{2}\right)^{NNZ(\mathbf{pa}_Y^2)}}$$

when $\mathbf{pa}_Y^1$ and $\mathbf{pa}_Y^2$ are identical except that $\mathbf{pa}_Y^1$ contains one extra zero in some position. Due to the fact that $\alpha < 1$, the above condition is equivalent to $NNZ(\mathbf{pa}_Y^1) < NNZ(\mathbf{pa}_Y^2)$ (simplify by taking the logarithm base α, then subtracting constants, then multiplying by -1, and then taking the logarithm base $\frac{1}{2}$; the direction of the sign above thus changes three times). Because $\mathbf{pa}_Y^1$ contains exactly one more zero than does $\mathbf{pa}_Y^2$, $NNZ(\mathbf{pa}_Y^1) = NNZ(\mathbf{pa}_Y^2) + 1$ and we conclude that $p(y|\mathbf{pa}_Y)$ is lattice with respect to $y = 0$.

Finally, we show that $p(y|\mathbf{pa}_Y)$ is totally strictly positive with respect to $y = 0$. For an arbitrary pair of parents $\{X_i, X_j\} \subseteq \mathbf{Pa}_Y$, let $\mathbf{X}_{ij}$ denote the remaining parents. That is,

$$\mathbf{Pa}_Y = \{X_i, X_j\} \cup \mathbf{X}_{ij}.$$

From Definition 27 (and the example that follows it), it suffices to show that

$$p(y = 0|x_i^0, x_j^0, \mathbf{x}_{ij})\, p(y = 0|x_i^1, x_j^1, \mathbf{x}_{ij}) > p(y = 0|x_i^0, x_j^1, \mathbf{x}_{ij})\, p(y = 0|x_i^1, x_j^0, \mathbf{x}_{ij}).$$

Letting n_{ij} denote the number of non-zero elements in $\mathbf{x}_{ij}$ and plugging in Equation 42 yields

$$\alpha^{2-\left(\frac{1}{2}\right)^{n_{ij}+2}} \alpha^{2-\left(\frac{1}{2}\right)^{n_{ij}}} > \alpha^{2-\left(\frac{1}{2}\right)^{n_{ij}+1}} \alpha^{2-\left(\frac{1}{2}\right)^{n_{ij}+1}}.$$

Taking the logarithm (base α) of both sides (which reverses the sign of the inequality because $\alpha < 1$), subtracting 4 from both sides, and then dividing both sides by $-\frac{1}{2}^{n_{ij}}$ (which reverses the sign of the inequality once again) leaves

$$\left(\frac{1}{2}\right)^2 + 1 > \frac{1}{2} + \frac{1}{2},$$

which clearly holds. □

References

Bouckaert, R. R. (1995). *Bayesian Belief Networks: From Construction to Inference*. PhD thesis, Utrecht University, The Netherlands.

Chickering, D. M. (1995). A transformational characterization of Bayesian network structures. In Hanks, S. and Besnard, P., editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montreal, QU, pages 87–98. Morgan Kaufmann.

Chickering, D. M. (1996). Learning Bayesian networks is NP-Complete. In Fisher, D. and Lenz, H., editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

Dasgupta, S. (1999). Learning polytrees. In Laskey, K. and Prade, H., editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, pages 131–141. Morgan Kaufmann.

Druzdzel, M. J. and Henrion, M. (1993). Efficient reasoning in qualitative probabilistic networks. In *Proceedings of the Eleventh Annual Conference on Artificial Intelligence (AAAI-93)*, Washington, D.C., pages 548–553.

Garey, M. and Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman.

Gavril, F. (1977). Some NP-complete problems on graphs. In *Proc. 11th Conf. on Information Sciences and Systems*, Johns Hopkins University, pages 91–95. Baltimore, MD.

Howard, R. and Matheson, J. (1981). Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, volume II, pages 721–762. Strategic Decisions Group, Menlo Park, CA.

Karlin, S. and Rinott, Y. (1980). Classes of orderings of measures and related correlation inequalities. i. multivariate totally positive distributions. *Journal of Multivariate Analysis*, 10:467–498.

Meek, C. (1997). *Graphical Models: Selecting causal and statistical models*. PhD thesis, Carnegie Mellon University.

Meek, C. (2001). Finding a path is harder than finding a tree. *Journal of Artificial Intelligence Research*, 15:383–389.

Nielsen, J. D., Kočka, T., and Peña, J. M. (2003). On local optima in learning Bayesian networks. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, Acapulco, Mexico, pages 435–442. Morgan Kaufmann.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search (second edition)*. The MIT Press, Cambridge, Massachussets.

Srebro, N. (2001). Maximum likelihood bounded tree-width Markov networks. In Breese, J. and Koller, D., editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, Seattle, WA, pages 504–511. Morgan Kaufmann.

Wellman, M. P. (1990). Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303.