Learning Bayesian Networks: Search Methods and Experimental Results

David M. Chickering **Dan Geiger*** **David Heckerman**
Microsoft Research, Bldg 9S
Redmond WA, 98052-6399
dmax@cs.ucla.edu, dang@cs.technion.ac.il, heckerma@microsoft.com

Abstract

We discuss Bayesian approaches for learning Bayesian networks from data. First, we review a metric for computing the relative posterior probability of a network structure given data developed by Heckerman et al. (1994a,b,c). We see that the metric has a property useful for inferring causation from data. Next, we describe search methods for identifying network structures with high posterior probabilities. We describe polynomial algorithms for finding the highest-scoring network structures in the special case where every node has at most $k = 1$ parent. We show that the general case ($k > 1$) is NP-hard, and review heuristic search algorithms for this general case. Finally, we describe a methodology for evaluating learning algorithms, and use this methodology to evaluate various scoring metrics and search procedures.

1 Introduction

Recently, many researchers have begun to investigate methods for learning Bayesian networks, including Bayesian methods [Cooper and Herskovits, 1991, Buntine, 1991, York 1992, Spiegelhalter et al., 1993, Madigan and Raferty, 1994, Heckerman et al., 1994], quasi-Bayesian methods [Lam and Bacchus, 1993, Suzuki, 1993], and nonBayesian methods [Pearl and Verma, 1991, Spirtes et al., 1993]. Many of these approaches have the same basic components: a scoring metric and a search procedure. The scoring metric takes data and a network structure and returns a score reflecting the goodness-of-fit of the data to the structure. A search procedure generates networks for evaluation by the scoring metric. These approaches use the two components to identify a network structure or set of structures that can be used to predict future events or infer causal relationships.

The Bayesian approach can be understood as follows. Suppose we have a domain of variables $\{x_1, \ldots, x_n\} = U$, and a set of cases $\{C_1, \ldots, C_m\}$ where each case is an instance of some or of all the variables in U. We sometimes refer to D as a database. Further, suppose that we wish to determine the joint distribution $p(C|D,\xi)$—the probability distribution of a new case C, given the database and our current state of information ξ. Rather than reason about this distribution directly, we imagine that the data is a random sample from some Bayesian network structure B_S with unknown parameters. Using B_S^h to denote the hypothesis that the data is generated by network structure B_S, and assuming the hypotheses corresponding to all possible network structures form a mutually exclusive and collectively exhaustive set, we have

$$p(C|D,\xi) = \sum_{\text{all } B_S^h} p(C|B_S^h, D, \xi) \cdot p(B_S^h|D,\xi) \quad (1)$$

In practice, it is impossible to sum over all possible network structures. Consequently, we attempt to identify a small subset H of network-structure hypotheses that account for a large fraction of the posterior probability of the hypotheses. Rewriting Equation 1, we obtain

$$p(C|D,\xi) \approx c \sum_{B_S^h \in H} p(C|B_S^h, D, \xi) \cdot p(B_S^h|D,\xi) \quad (2)$$

where c is the normalization constant given by $1/[\sum_{B_S^h \in H} p(B_S^h|D,\xi)]$. From Equation 2, we see that only the relative posterior probability of hypotheses matter. Thus, rather than compute posterior probability, one typically computes $p(B_S^h, D|\xi) = p(B_S^h|\xi)\ p(D|B_S^h, \xi)$, or a *Bayes' factor*—$p(B_S^h|D,\xi)/p(B_{S0}^h|D,\xi)$—where B_{S0} is some reference structure such as the empty graph.

*Author's primary affiliation: Computer Science Department, Technion, Haifa 32000, Israel.

The approach is not only an approximation for $p(C|D,\xi)$ but a method for learning network structure, where relative posterior probability plays the role of scoring metric. For example, when $|H| = 1$, we learn a single network structure: the MAP (*maximum a posteriori*) structure of U. When $|H| > 1$, we learn a collection of network structures weighted by the relative posterior probability of their corresponding hypotheses. As we discuss in Section 3, learning about structure is useful, because we can sometimes infer causal relationships in a domain and consequently predict the effects of interventions.

In this paper, we review a scoring metric described by Heckerman et al. (1994a,b,c). The metric has the property of *likelihood equivalence*, which says that any two isomorphic network structures must have the same likelihood $p(D|B_S^h,\xi)$. In Section 3, we argue that this property is desirable for inferring causation from data. In addition, we examine search methods in detail. We describe polynomial algorithms for finding the highest-scoring networks in the special case where every node has at most $k = 1$ parent. Also, we show that the general case ($k > 1$) is NP-hard, even for a score-equivalent metric, and review heuristic search algorithms for this general case. Finally, we describe a methodology for evaluating learning algorithms, and use this methodology to evaluate various scoring metrics and search procedures.

2 Review of Bayesian Scoring Metrics

In this section, we review a Bayesian scoring metric for learning Bayesian networks containing only discrete variables described by Heckerman et al. (1994a,b,c)—herein referred to as HGC. The metric is closely related to the metrics described by Cooper and Herskovits (1991, 1992)—herein referred to as CH—and Buntine (1991). Although we limit our discussion to domains containing only discrete variables, many of the basic points translate to the more general case [Geiger and Heckerman, 1994].

CH derive a Bayesian scoring metric under the following assumptions:

1. The database D is a multinomial sample from some (possibly uncertain) structure B_S with (possibly uncertain) parameters Θ_{B_S}. We use B_S^h to denote the hypothesis that the database is a multinomial sample from structure B_S.[1] Given a belief-network structure B_S, we use Π_i to denote the parents of x_i. We use r_i to denote the number of states of variable x_i, and $q_i = \prod_{x_l \in \Pi_i} r_l$ to denote the number of instances of Π_i. We use the integer j to index these instances. That is, we write $\Pi_i = j$ to denote the observation of the jth instance of the parents of x_i. We use θ_{ijk} to denote the parameter associated with the kth state of variable x_i, given the jth state of Π_i. (We can think of θ_{ijk} as the long-run fraction of cases where $x_i = k$, in those cases where $\Pi_i = j$.) We use Θ_{ij} to denote the union of θ_{ijk} over k. The parameters of B_S (Θ_{B_S}) are the union of Θ_{ij} for all instances j of all variables x_i.

2. For all belief-network structures B_S, $\rho(\Theta_{B_S}|B_S^h) = \prod_i \prod_j \rho(\Theta_{ij}|B_S^h)$. (Throughout the paper, we use $\rho(\cdot|\cdot)$ to denote a conditional probability density.) The assumption corresponds to the *local* and *global independence* assumptions made by Spiegelhalter and Lauritzen (1990). We refer to this assumption simply as *parameter independence.*

3. If x_i has the same parents in any two belief-network structures B_{S1} and B_{S2}, then for $j = 1, \ldots, q_i$, $\rho(\Theta_{ij}|B_{S1}^h) = \rho(\Theta_{ij}|B_{S2}^h)$. We call this assumption *parameter modularity.* This assumption was made implicitly by CH and was made explicit by HGC.

4. For every belief-network structure B_S, and for all $\Theta_{ij} \subseteq \Theta_{B_S}$, $\rho(\Theta_{ij}|B_S^h)$ has the Dirichlet distribution $\rho(\Theta_{ij}|B_S^h) \propto \prod_k \theta_{ijk}^{N'_{ijk}-1}$.

Given these four assumptions, the prior densities of Θ_{B_S} for all structure B_S are determined by the Dirichlet exponents N'_{ijk}. To derive their metric, CH also make the following assumption:

5. All databases are *complete.* That is, every variable in every case in the database is observed.

Given this assumption, the parameters Θ_{B_S} for any network structure B_S remain independent and modular (in the sense of Assumptions 2 and 3) after the database has been observed. Consequently, CH obtain the following result:

$$p(D, B_S^h|\xi) = \tag{3}$$
$$p(B_S^h|\xi) \cdot \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}$$

where N_{ijk} is the number of cases in D where $x_i = k$ and $\Pi_i = j$, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, and $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$, and $\Gamma(\cdot)$ is the *Gamma* function. We call this expression or any expression proportional to it the BD (*B*ayesian *D*irichlet) metric. We discuss the specification of $p(B_S^h|\xi)$ later in this section.

HGC derive a special case of the BD metric. First, they argue for the property of *hypothesis equivalence*:

6. Hypotheses B_{S1}^h and B_{S2}^h are equivalent whenever B_{S1} and B_{S2} are isomorphic.

[1] There are some subtleties about the definition of B_S^h which are explored in HGC.

Two network structures are said to be *isomorphic* if they represent the same assertions of conditional independence. For example, in the three variable domain $\{x_1, x_2, x_3\}$, the network structures $x_1 \rightarrow x_2 \rightarrow x_3$ and $x_1 \leftarrow x_2 \rightarrow x_3$ represent the same independence assertion—x_1 and x_3 are independent given x_2—and are therefore isomorphic. We discuss the assumption of hypothesis equivalence further in the following section.

Next, HGC examine *complete belief-network structures*—those containing no missing arcs. Under the assumption of hypothesis equivalence, the hypotheses corresponding to any two such structures are the same, and are denoted $B^h_{S_C}$. They introduce the following assumptions:

7. $p(B^h_{S_C}|\xi) > 0$

8. $\rho(\Theta_{x_1,\ldots,x_n}|B^h_{S_C},\xi) = \prod_{x_1,\ldots,x_n} \theta_{x_1,\ldots,x_n}^{N'_{x_1,\ldots,x_n}-1}$

In words, Assumption 7 says that the data may be a random sample from a complete network structure. Assumption 8 says that the probability density of the parameters of the joint space of U conditioned on $B^h_{S_C}$ is Dirichlet. HGC (1994a,b) show that Assumptions 6 through 8 are consistent with CH's Assumptions 1 through 4. Namely, they show that the random-sample assumption (Assumption 1) and Assumptions 6 through 8 imply the assumption of parameter independence (Assumption 2) and the Dirichlet assumption (Assumption 4) for all complete network structures. As a consequence, they show that the Dirichlet exponents N'_{ijk} in the BD metric are constrained by the relation

$$N'_{ijk} = N' \cdot p(x_i = k, \Pi_i = j|B^h_{S_C}) \qquad (4)$$

where N' is the user's *equivalent sample size* for the domain.

HGC show that the BD metric so constrained has the property of *likelihood equivalence*, which says that the likelihoods $p(D|B^h_{S1},\xi)$ and $p(D|B^h_{S2},\xi)$ are equal whenever B_{S1} and B_{S2} are isomorphic. They call this special case the BDe metric ("e" for equivalence). In the next section, we argue that the property of hypothesis equivalence is useful for inferring causation from data. We note that Buntine (1991) proposed a special case of the BDe metric, which can be obtained by computing $p(x_i = k, \Pi_i = j|B^h_{S_C})$ in Equation 4 from a uniform joint distribution. In addition, Buntine stated that this metric satisfies the property which we call likelihood equivalence.

HGC show that the probabilities in Equation 4 may be computed from a *prior Bayesian network* provided by the expert: a Bayesian network for $p(U|B^h_{S_C},\xi)$. This prior network encodes the user's beliefs about what will happen in the next case, given $B^h_{S_C}$. As we shall see, a prior network also can be used to facilitate the assessment of the prior probabilities of network structure as well as initialize search.

HGC (1994c) demonstrate a much stronger result than that of HGC (1994a,b). Namely, under weak regularity conditions, they show that the random-sample assumption and the assumption of parameter independence, when combined with the assumption of hypothesis equivalence and the assumption that $p(B^h_{S_C}|\xi) > 0$ imply that the probability density of the joint-space parameters of U conditioned on $B^h_{S_C}$ must be Dirichlet (Assumption 8). In essence, they show that parameter independence in the context of learning Bayesian networks provides a new characterization of the Dirichlet distribution. As a result, they show that CH's basic assumptions for learning (Assumptions 1, 2, 3, and 5) when combined with hypothesis equivalence and the assumption $p(B^h_{S_C}|\xi) > 0$ imply the BDe metric.

To complete the specification of a Bayesian scoring metric, we need the prior probabilities of network structures $p(B^h_S|\xi)$.[2] HGC suggest a simple parametric formula for these probabilities. This formula punishes B^h_S for every arc that differs from those in the prior network used in the computations of Equation 4. Namely, let $\Pi_i(B^h_S)$ and $\Pi_i(P)$ denote the parents of x_i in B_S and the prior network, respectively. Let δ_i denote the number of nodes in the symmetric difference of $\Pi_i(B_S)$ and $\Pi_i(P)$. They set

$$p(B^h_S|\xi) = c \prod_{i=1}^{n} \kappa^{\delta_i} \qquad (5)$$

where $0 < \kappa \leq 1$ is the penalty factor, and c is a normalization constant. Buntine (1991) suggests a similar approach where penalty factors are different for different arcs.

We say that a metric is *prior equivalent* when the prior probabilities of any two isomorphic network structures are equal. We say a metric is *score equivalent* if it is both prior equivalent and likelihood equivalent. The parametric formula given in the previous paragraph is not prior equivalent. HGC (1994c) describe other parameterizations that are prior equivalent.

3 Inferring Causation

Bayesian networks were first described as a representation of conditional independence [Howard and Matheson, 1981, Pearl, 1988]. Recently, however, several researchers have begun to explore an additional causal

[2] For the sake of brevity, we refer to the probability of a network structure rather than the probability of a network-structure hypothesis.

semantics for these networks. They argue that the representation of causal knowledge is important, because such knowledge—unlike statistical knowledge—allows one to derive beliefs about a domain after intervention. For example, most of us believe that smoking causes lung cancer. From this knowledge, we infer that if we stop smoking, then we decrease our chances of getting lung cancer. In contrast, if we knew only that there was a statistical correlation between smoking and lung cancer, then we could not make this inference.

Causal networks, described by—for example—Pearl and Verma (1991), Spirtes et al. (1993), Druzdzel and Simon (1993), and Heckerman and Shachter (1994), represent such causal relationships among variables. In particular, a causal network for U is a Bayesian network for U, wherein it is asserted that each nonroot node x is caused by its parents. The precise meaning of cause and effect is not important for this discussion. The interested reader should consult the previous references.

Given the distinction between statistical and causal dependence, it would seem impossible to learn causal networks from data produced by observation alone and without intervention. For example, consider the simple three-variable domain $U = \{x_1, x_2, x_3\}$. If we find through the observation of data that the network structure $x_1 \rightarrow x_3 \leftarrow x_2$ is very likely, then we cannot conclude that x_1 and x_2 are causes for x_3. Rather, it may be the case that there is a hidden common cause of x_1 and x_3 as well as a hidden common cause of x_2 and x_3. If, however, we assume that every statistical association derives from causal interaction, and that there are no hidden common causes, then we can interpret learned networks as causal networks. In our example, under these assumptions, we can infer that x_1 and x_2 are causes for x_3. In our remaining discussion, we take these assumptions as given.

When learning Bayesian networks having only the traditional conditional-independence semantics, the assumption of hypothesis equivalence is reasonable, because two isomorphic network structures are identical by definition of isomorphism. If we want to interpret learned networks as being causal, however, the situation is not so straightforward. For purposes of this discussion, it is useful to decompose a scoring metric into two components: (1) determination of the prior probability of network structure $p(B_S^h|\xi)$ and (2) determination of the likelihood $p(D|B_S^h, \xi)$.

For most real-world problems that we have encountered, we have found that it is unreasonable to apply the assumption of hypothesis equivalence to the determination of the prior probabilities of causal-network structures. That is, we have found the assumption of prior equivalence to be unreasonable. For example, it is not unusual for one to believe that the proposition x_1 causes x_2 is more likely than the proposition x_2 causes x_1. Such a belief would be excluded by prior equivalence applied to the network structures $x_1 \leftarrow x_2$ and $x_1 \rightarrow x_2$.

In contrast, for most problems we have considered, we have found it reasonable to apply hypothesis equivalence to the determination of likelihood. That is, we have expected likelihood equivalence to hold. Of course, for any given problem, it is up to the decision maker to make this judgment. As a example of where the use of likelihood equivalence is unreasonable, imagine that a doctor may be uncertain as to which of two possible causal mechanisms are responsible for an observed correlation between two diseases, but certain that the existence of the first mechanism implies the hypothesis disease d_1 causes disease d_2 (B_{S1}^h), and the existence of the second mechanism implies the hypothesis disease d_2 causes d_1 (B_{S1}^h). In this scenario, the probability densities $\rho(\Theta_{d_1,d_2}|B_{S1}^h, \xi)$ and $\rho(\Theta_{d_1,d_2}|B_{S2}^h, \xi)$ may not be the same, in which case, likelihood equivalence will not hold. Without specific knowledge of competing causal mechanisms, however, we typically have no reason to believe that data will discriminate between two isomorphic network structures.

Under the assumption of likelihood equivalence, the ratio of posterior probabilities of two isomorphic network structures must be equal to the ratio of their prior probabilities. Consequently, if the priors on network structures are not too different, then typically, learning will produce many isomorphic network structures each having a large relative posterior probability. Furthermore, even for domains where the assumption of likelihood equivalence does not hold, there is a good chance that more than one hypothesis will have a large relative posterior probability. In such situations, we find it reasonable to average the causal assertions contained in individual learned networks. For example, in our three-variable domain, let us suppose that the data supports only the network structure $x_1 \rightarrow x_2 \rightarrow x_3$ and its isomorphic cousins $x_1 \leftarrow x_2 \leftarrow x_3$ and $x_1 \leftarrow x_2 \rightarrow x_3$. If each of the hypotheses corresponding to these structures have the same prior probability, then the posterior probability of each hypothesis will be $1/3$, and we infer that the proposition x_2 cases x_3 has probability $2/3$. Under these same conditions, the proposition that both x_1 and x_3 are causes of x_2 has probability 0.

4 Search Methods

In this section, we examine methods for finding network structures with high scores. Although our methods are presented in the context of Bayesian scoring metrics, they may be used in conjunction with other nonBayesian metrics as well.

Many search methods for learning network structure—including those that we describe—make use of a property of scoring metrics that we call decomposability. Given a network structure for domain U, we say that a measure on that structure is *decomposable* if it can be written as a product of measures, each of which is a function only of one node and its parents. From Equation 3, we see that the likelihood $p(D|B_S^h, \xi)$ given by the BD metric is decomposable. Consequently, if the prior probabilities of network structures are decomposable, then so is the BD metric. Thus, we can write

$$p(D, B_S^h|\xi) = \prod_{i=1}^{n} s(x_i|\Pi_i) \tag{6}$$

where $s(x_i|\Pi_i)$ is only a function of x_i and its parents. Most Bayesian and nonBayesian metrics to date are decomposable. Given a decomposable metric, we can compare the score for two network structures that differ by the addition or deletion of arcs pointing to x_i, by computing only the term $s(x_i|\Pi_i)$ for both structures.

4.1 Special-Case Polynomial Algorithms

We first consider the special case of finding the l network structures with the highest score among all structures in which every node has at most one parent.

For each arc $x_j \rightarrow x_i$ (including cases where x_j is null), we associate a weight $w(x_i, x_j) \equiv \log s(x_i|x_j) - \log s(x_i|\emptyset)$. From Equation 6, we have

$$\begin{aligned} \log p(D, B_S^h) &= \sum_{i=1}^{n} \log s(x_i|\pi_i) \qquad (7) \\ &= \sum_{i=1}^{n} w(x_i, \pi_i) + \sum_{i=1}^{n} \log s(x_i|\emptyset) \end{aligned}$$

where π_i is the (possibly) null parent of x_i. The last term in Equation 7 is the same for all network structures. Thus, among the network structures in which each node has at most one parent, ranking network structures by sum of weights $\sum_{i=1}^{n} w(x_i, \pi_i)$ or by score has the same result.

Finding the network structure with the highest weight ($l = 1$) is a special case of a well-known problem of finding *maximum branchings*, described as follows. A *tree-like network* is a connected directed acyclic graph in which no two edges are directed into the same node. The root of a tree-like network is a unique node that has no edges directed into it. A *branching* is a directed forest that consists of disjoint tree-like networks. A *spanning branching* is any branching that includes all nodes in the graph. A *maximum branching* is any spanning branching which maximizes the sum of arc weights (in our case, $\sum_{i=1}^{n} w(x_i, \pi_i)$). An efficient polynomial algorithm for finding a maximum branching was first described by Edmonds (1967). The general case ($l > 1$) was treated by Camerini et al. (1980).

These algorithms can be used to find the l branchings with the highest weights regardless of the metric we use, as long as one can associate a weight with every edge. Therefore, this algorithm is appropriate for any decomposable metric. When using metrics that are score equivalent, however, we have

$$s(x_i|x_j)s(x_j|\emptyset) = s(x_j|x_i)s(x_i|\emptyset)$$

Thus, for any two edges $x_i \rightarrow x_j$ and $x_i \leftarrow x_j$, the weights $w(x_i, x_j)$ and $w(x_j, x_i)$ are equal. Consequently, the directionality of the arcs plays no role for score-equivalent metrics, and the problem reduces to finding the l undirected forests for which $\sum w(x_i, x_j)$ is a maximum. For the case $l = 1$, we can apply a maximum spanning tree algorithm (with arc weights $w(x_i, x_j)$) to identify an undirected forest F having the highest score. The set of network structures that are formed from F by adding any directionality to the arcs of F such that the resulting network is a branching, yields a collection of isomorphic network structures each having the same maximal score. This algorithm is identical to the tree learning algorithm described by Chow and Liu (1968), except that we use a score-equivalent Bayesian metric rather than the mutual-information metric. For the general case ($l > 2$), we can use the algorithm of Gabow (1977) to identify the l undirected forests having the highest score, and then determine the l equivalence classes of network structures with the highest score.

4.2 Heuristic Search

A generalization of the problem described in the previous section is to find the l best networks from the set of all networks in which each node has no more than k parents. Unfortunately, even when $l = 1$, the problem for $k > 1$ is NP-hard. Furthermore, let us consider the following decision problem, which corresponds to our optimization problem with $l = 1$.

k-LEARN
INSTANCE: Set of variables U, database $D = \{C_1, \ldots, C_m\}$, where each C_i is an instance of all variables in U, scoring metric $M(D, B_S)$ and real value p,

QUESTION: Does there exist a network structure B_S defined over the variables in U, where each node in B_S has at most k parents, such that $M(D, B_S) \geq p$?

Höffgen (1993) shows that a similar problem for PAC learning is NP-complete. His results can be translated easily to show that k-LEARN is NP-complete for $k > 1$ when the BD metric is used. In the Appendix, we show that k-LEARN is NP-complete, even when we use the likelihood-equivalent BDe metric and the constraint of prior equivalence.

Therefore, unless $P = NP$, it is appropriate to use heuristic search algorithms for the general case $k > 1$. In this section, we review several such algorithms.

As is the case with essentially all search methods, the methods that we examine have two components: an initialization phase and a search phase. For example, let us consider the K2 search method described by CH. The initialization phase consists of choosing an ordering over the variables in U. In the search phase, for each node x_i in the ordering provided, the node from $\{x_i, \ldots, x_{i-1}\}$ that most increases the network score is added to the parent set of x_i, until no node increases the score or the size of Π_i exceeds a predetermined constant.

The search algorithms we consider make successive arc changes to the network, and employ the property of decomposability to evaluate the merit of each change. The possible changes that can be made are easy to identify. For any pair of variables, if there is an arc connecting them, then this arc can either be reversed or removed. If there is no arc connecting them, then an arc can be added in either direction. All changes are subject to the constraint that the resulting network contain no directed cycles. We use E to denote the set of eligible changes to a graph, and $\Delta(e)$ to denote the change in log score of the network resulting from the modification $e \in E$. Given a decomposable metric, if an arc to x_i is added or deleted, only $s(x_i|\Pi_i)$ need be evaluated to determine $\Delta(e)$. If an arc between x_i and x_j is reversed, then only $s(x_i|\Pi_i)$ and $s(x_j|\Pi_j)$ need be evaluated.

One simple heuristic search algorithm is *local search* [Johnson, 1985]. First, we choose a graph. Then, we evaluate $\Delta(e)$ for all $e \in E$, and make the change e for which $\Delta(e)$ is a maximum, provided it is positive. We terminate search when there is no e with a positive value for $\Delta(e)$. As we visit network structures, we retain l of them with the highest overall score. Using decomposable metrics, we can avoid recomputing all terms $\Delta(e)$ after every change. In particular, if neither x_i, x_j, nor their parents are changed, then $\Delta(e)$ remains unchanged for all changes e involving these nodes as long as the resulting network is acyclic. Candidates for the initial graph include the empty graph, a random graph, a graph determined by one of the polynomial algorithms described in the previous section, and the prior network.

A potential problem with local search is getting stuck at a local maximum. Methods for avoiding local maxima include iterated hill-climbing and simulated annealing. In *iterated hill-climbing*, we apply local search until we hit a local maximum. Then, we randomly perturb the current network structure, and repeat the process for some manageable number of iterations. At all stages we retain the top l networks structures.

In one variant of *simulated annealing* described by Metropolis et al. (1953), we initialize the system at some temperature T_0. Then, we pick some eligible change e at random, and evaluate the expression $p = \exp(\Delta(e)/T_0)$. If $p > 1$, then we make the change e; otherwise, we make the change with probability p. We repeat this selection and evaluation process α times or until we make β changes. If we make no changes in α repetitions, then we stop searching. Otherwise, we lower the temperature by multiplying the current temperature T_0 by a decay factor $0 < \gamma < 1$, and continue the search process. We stop searching if we have lowered the temperature more than δ times. Thus, this algorithm is controlled by five parameters: $T_0, \alpha, \beta, \gamma$ and δ. Throughout the process, we retain the top l structures. To initialize this algorithm, we can start with the empty graph, and make T_0 large enough so that almost every eligible change is made, thus creating a random graph. Alternatively, we may start with a lower temperature, and use one of the initialization methods described for local search.

5 Evaluation Methodology

Our methodology for measuring the learning accuracy of scoring metrics and search procedures is as follows. We start with a given network, which we call the *gold-standard network*. Next, we generate a database by repeated sampling from the given network. Then, we use a scoring metric and search procedure to identify one or more high-scoring network structures. Next, we use the database and prior knowledge to populate the probabilities in the new networks, called the *learned networks*. In particular, we set each probability $p(x_i = k|\Pi_i = j)$ to be the posterior mean of θ_{ijk} given the database, computed under the assumptions of the scoring metric. Finally, we quantitate learning accuracy by measuring the difference between the joint probability distributions of the gold-standard and learned networks. An advantage of our method is that there exists a clear correct answer: the gold-standard network. One argument against using our

method is that, by generating a database from a network, we guarantee that the assumption of exchangeability (time invariance) holds, and thereby bias results in favor of our scoring metrics. We can, however, simulate time varying databases in order to measure the sensitivity of our methods to the assumption of exchangeability (although we do not do so in this paper).

A principled candidate for a measure of learning accuracy is expected utility. Namely, given a utility function, a series of decisions to be made under uncertainty, and a model of that uncertainty (i.e., one ore more Bayesian networks for U), we evaluate the expected utility of these decisions using the gold-standard and learned networks, and note the difference. This utility function may include not only domain utility, but the costs of probabilistic inference as well [Horvitz, 1987]. Unfortunately, it is difficult if not impossible to construct utility functions and decision scenarios in practice. For example, a particular set of learned network structures may be used for a collection of decisions problems, some of which cannot be anticipated. Consequently, researchers have used surrogates for differences in utility, such as the mean square error, cross entropy, and differences in structure.

In this paper, we use two surrogate measures: cross-entropy and a structural difference. The cross-entropy measure [Kullback and Leibler, 1951] reflects how well the learned structures will predict the next case, whereas structural difference reflects the degree to which the learned structures have captured causal interactions.

Our cross-entropy measure is as follows. Let $q(U)$ and $p(U)$ denote the probability of an instance of U obtained from the gold-standard network and learned networks, respectively. We compute $p(U)$ using Equation 2. The cross entropy $H(q,p)$ is given by

$$H(q,p) = \sum_{x_1,\ldots,x_n} q(x_i,\ldots,x_n) \log \frac{q(x_i,\ldots,x_n)}{p(x_i,\ldots,x_n)} \quad (8)$$

Low values of cross entropy correspond to a learned distribution that is close to the gold standard. In HGC (1994a,c), we describe a relatively efficient method for computing the cross entropy of two networks that makes use of the network structures.

Our structural-difference measure is as follows. For a single learned network, the structural difference we use is $\sum_{i=1}^{n} \delta_i$, where δ_i is the symmetric difference of the parents of x_i in the gold-standard network and the parents of x_i in the learned network. For multiple networks, we take the average of the structural-difference scores, weighted by the relative posterior probabilities of the learned networks.

In several of our experiments described in the next section, we require a prior network. For these investigations, we construct prior networks by adding noise to the gold-standard network. We control the amount of noise with a parameter η. When $\eta = 0$, the prior network is identical to the gold-standard network, and as η increases, the prior network diverges from the gold-standard network. When η is large enough, the prior network and gold-standard networks are unrelated. Let (B_S, B_P) denote a Bayesian network with structure B_S and probabilities B_P. To generate the prior network, we first add 2η arcs to the gold-standard network, creating network structure B_{S1}. When we add an arc, we copy the probabilities in B_{P1} so as to maintain the same joint probability distribution for U. Next, we perturb each conditional probability in B_{P1} with noise. In particular, we convert each probability to log odds, add to it a sample from a normal distribution with mean zero and standard deviation η, convert the result back to a probability, and renormalize the probabilities. Then, we create another network structure B_{S2} by deleting 2η arcs and reversing 2η arcs that were present in the original gold-standard network. Next, we perform inference using the joint distribution determined by network (B_{S1}, B_{P1}) to populate the conditional probabilities for network (B_{S2}, B_{P2}). For example, if x has parents Y in B_{S1}, but x is a root node in B_{S2}, then we compute the marginal probability for x in B_{S1}, and store it with node x in B_{S2}. Finally, we return (B_{S2}, B_{P2}) as the prior network.

6 Experimental Results

We have implemented the metrics and search algorithms described in this paper. Our implementation is in the C++ programming language, and runs under Windows NT™ with a Pentium processor. We have tested our algorithms on small networks ($n \leq 5$) as well as the 35-node Alarm network for the domain of ICU ventilator management [Beinlich et al., 1989]. Here, we describe some of the more interesting results that we obtained using the Alarm network. In these preliminary experiments, we have learned only single network structures ($l = 1$).

To examine the effects of metric on learning accuracy, we measured the cross entropy and structural difference of learned networks with respect to the Alarm network for several variants of the BDe metric as well as a non-likelihood-equivalent variant of the BD metric. The results are shown in Figure 1. The metrics labeled BDe0, BDe2, and BDe4 correspond to the BDe metric with prior networks generated from the Alarm network with noise $\eta = 0, 2, 4$, respectively. The BDeu metric is the special case of the BDe metric described by Buntine (1991) where all instances of the joint space are equally likely. (The prior-network structure in this

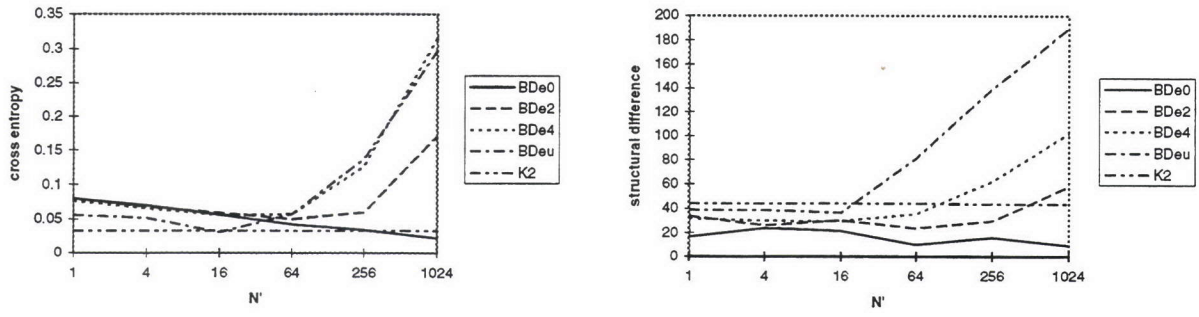


Figure 1: Metric comparisons.

case is the empty graph). The K2 metric (not to be confused with the K2 search algorithm) is the special case of the BD metric described by CH where all $N'_{ijk} = 1$. This metric is not likelihood equivalent.

In this comparison, we used local search initialized with a maximum branching and 10,000-case databases sampled from the Alarm network. For each value of equivalent sample size N' in the graph, the cross-entropy and structural-difference values shown in the figure represent an average across five learning instances, where in each instance we used a different database and (for the BDe2 and BDe4 metrics) a different prior network. We made the prior parameter κ a function of N'—namely, $\kappa = 1/(N' + 1)$—so that it would take on reasonable values at the extremes of N'. (When $N' = 0$, reflecting complete ignorance, all network structures receive the same prior probability. Whereas, when N' is large, reflecting a high degree of confidence, the prior network structure receives a high prior probability.) When computing the prior probabilities of network structure for the K2 metric, we used an empty prior network.

The qualitative behaviors of the BDe metrics were reasonable. When $\eta = 0$—that is, when the prior network was identical to the Alarm network—learning accuracy increased as the equivalent sample size N' increased. Also, learning accuracy decreased as the prior network deviated further from the gold-standard network, demonstrating the expected result that prior knowledge is useful. In addition, When $\eta \neq 0$, there was a value of N' associated with optimal accuracy, and this value decreased as η increased (this effect is more pronounced in the graph of structural differences). This result is not surprising. Namely, if N' is too large, then the deviation between the true values of the parameters and their priors degrade performance. On the other hand, if N' is too small, the metric is ignoring useful prior knowledge. Furthermore, as the deviation between the prior and gold-standard network structures increase, learning should be optimal for lower values of equivalent sample size. We speculate that results of this kind can be used to calibrate users in the assessment of N'.

Quantitative results show that, for low values of N', all metrics perform about equally well, with K2 producing slightly lower cross entropies and the BDe metrics producing slightly lower structural differences. One exception is that the BDe metric outperformed all other metrics for larger values of N', when the metric used the gold standard as a prior network. These results suggest that the expert should pay close attention to the assessment of equivalent sample size when using the BDe metric.

To investigate the effects of search initialization on learning accuracy, we initialized local search with random structures, prior networks for different values of η, a maximum branching, and the empty graph. The results are shown in Figure 2. In this comparison, we used the BDeu metric with $N' = 16$, $\kappa = 1/17$, and a 10,000-case database. We created 100 random structures by picking orderings at random, and then, for a given ordering, placing in the structure each possible arc with probability $\kappa/(1 + \kappa)$. (This approach produced a distribution of random network structures that was consistent with the prior probability of network structures as determined by Equation 5).

The curve in Figure 2 is a histogram of the local maxima achieved with random-structure initialization. Prior networks for both $\eta = 0$ and $\eta = 4$ produced local maxima that fell at the extreme low end of this curve. In addition, the maximum branching led to a local maximum with relatively low cross entropy. These results suggest that initialization with a prior network can be beneficial, even when the prior network differs substantially from the gold standard. The results also suggest that, if a prior network is not available, then a maximum branching can be a good starting point for search.

Finally, we investigated the effects of search algorithm

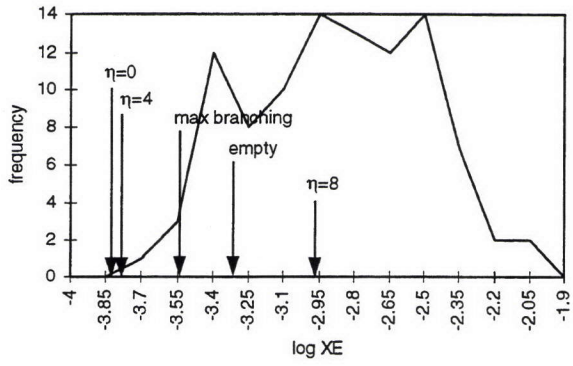


Figure 2: Comparisons of methods for initializaing search.

on learning accuracy. The results are shown in Table 1. In this comparison, we used the BDeu metric with $N' = 16$, $\kappa = 1/17$, and 30 databases of size 10,000. The algorithm K2opt is CH's K2 search algorithm (described in Section 4.2) initialized with an ordering that is consistent with the Alarm network. The algorithm K2rev is the same algorithm initialized with the reversed ordering. We included the latter algorithm to gauge the sensitivity of the K2 algorithm to variable order. Iterative local search used 30 restarts where, at each restart, the current network structure was modified with 100 random changes. (A single change was either an arc addition, deletion, or reversal.) The annealing algorithm used parameters $T_0 = 100, \alpha = 200, \beta = 100, \gamma = 0.9$, and $\delta = 60$. We found these parameters for iterative local search and annealing to yield reasonable learning accuracy after some experimentation. Local search, iterative local search, and annealing were initialized with a maximum branching.

K2opt obtained the lowest structural difference, whereas K2rev obtained the highest structural difference, illustrating the sensitivity of the K2 algorithm to variable ordering. All search algorithms—except K2rev—obtained low cross entropies. Overall, local search appeared to be the best algorithm, because it was both accurate and fast and did not require a variable ordering from the user. Neither iterative local search nor annealing offered much (if any) improvement over local search; and both algorithms were considerably slower than local search. Given the distribution of local maxima in Figure 2, however, we expect we will be able to increase learning accuracy of iterative local search and annealing after further tuning their parameters.

7 Summary

We have examined Bayesian approaches for learning Bayesian networks from data. First, we reviewed the BDe scoring metric described by HGC. We saw that this metric has a property useful for inferring causation called likelihood equivalence, which says that any two isomorphic network structures must have the same likelihood $p(D|B_S^h, \xi)$. Next, we described search methods for identifying network structures with high posterior probabilities. We presented polynomial algorithms for finding the highest-scoring network structures in the special case where every node has at most $k = 1$ parent. We showed that the general case ($k > 1$) is NP-hard, even when we use the restrictive BDe metric, and reviewed heuristic search algorithms for this general case. Finally, we described a methodology for evaluating learning algorithms. Using this methodology, we found that the learning accuracy of the BDe metric often exceeds that of CH's K2 metric, and that local search initialized with a prior-network structure or a maximum branching can be an effective search procedure.

Acknowledgments

We thank Koos Rommelse for his help with implementation and testing of the algorithms.

References

[Beinlich et al., 1989] Beinlich, I., Suermondt, H., Chavez, R., and Cooper, G. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine,* London. Springer Verlag, Berlin.

[Buntine, 1991] Buntine, W. (1991). Theory refinement on Bayesian networks. In *Proceedings of Seventh Conference on Uncertainty in Artificial Intelligence,* Los Angeles, CA, pages 52–60. Morgan Kaufmann.

[Camerini and Maffioli, 1980] Camerini, P. and Maffioli, L. F. F. (1980). The k best spanning arborescences of a network. *Networks*, 10:91–110.

[Chow and Liu, 1968] Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467.

[Cooper and Herskovits, 1991] Cooper, G. and Herskovits, E. (January, 1991). A Bayesian method for

Table 1: Cross entropy and learning times (mean ± s.d.) for various search algorithms across 30 databases.

	cross entropy	structural difference	learning time
K2opt	0.026 ± 0.002	3.9 ± 1.4	1.9 min
K2rev	0.218 ± 0.009	139.3 ± 1.7	2.9 min
local	0.029 ± 0.005	45.0 ± 7.8	2.1 min
iterative local	0.026±0.003	42.0 ± 9.9	251 min
annealing	0.029 ± 0.009	37.3 ± 11.5	25 min

the induction of probabilistic networks from data. Technical Report SMI-91-1, Section on Medical Informatics, Stanford University.

[Druzdzel and Simon, 1993] Druzdzel, M. and Simon, H. (1993). Causality in Bayesian belief networks. In *Proceedings of Ninth Conference on Uncertainty in Artificial Intelligence,* Washington, DC, pages 3–11. Morgan Kaufmann.

[Edmonds, 1967] Edmonds, J. (1967). Optimum brachching. *J. Res. NBS*, 71B:233–240.

[Gabow, 1977] Gabow, H. (1977). Siam journal of computing. *Networks*, 6:139–150.

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

[Garvil, 1977] Garvil, 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.

[Geiger and Heckerman, 1994] Geiger, D. and Heckerman, D. (1994). Learning Gaussian networks. In *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence,* Seattle, WA, pages 235–243. Morgan Kaufmann.

[Heckerman et al., 1994a] Heckerman, D., Geiger, D., and Chickering, D. (March, 1994). Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09a, Microsoft.

[Heckerman et al., 1994b] Heckerman, D., Geiger, D., and Chickering, D. (1994). Learning Bayesian networks: The combination of knowledge and statistical data. In *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence,* Seattle, WA, pages 293–301. Morgan Kaufmann.

[Heckerman et al., 1994c] Heckerman, D., Geiger, D., and Chickering, D. (Revised November, 1994). Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft.

[Heckerman and Shachter, 1994] Heckerman, D. and Shachter, R. (1994). A decision-based view of causality. In *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence,* Seattle, WA, pages 302–310. Morgan Kaufmann.

[Höffgen, 1993] Höffgen, K. (revised 1993). Learning and robust learning of product distributions. Technical Report 464, Fachbereich Informatik, Universität Dortmund.

[Horvitz, 1987] Horvitz, E. (1987). Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence,* Seattle, WA. Association for Uncertainty in Artificial Intelligence, Mountain View, CA.

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

[Johnson, 1985] Johnson (1985). How fast is local search? In *FOCS*, pages 39–42.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. (1951). Information and sufficiency. *Ann. Math. Statistics*, 22:79–86.

[Lam and Bacchus, 1993] Lam, W. and Bacchus, F. (1993). Using causal information and local measures to learn Bayesian networks. In *Proceedings of Ninth Conference on Uncertainty in Artificial Intelligence,* Washington, DC, pages 243–250. Morgan Kaufmann.

[Madigan and Rafferty, 1994] Madigan, D. and Rafferty, A. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, to appear.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). *Journal of Chemical Physics*, 21:1087–1092.

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

[Pearl and Verma, 1991] Pearl, J. and Verma, T. (1991). A theory of inferred causation. In Allen, J., Fikes, R., and Sandewall, E., editors, *Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452. Morgan Kaufmann, New York.

[Spiegelhalter et al., 1993] Spiegelhalter, D., Dawid, A., Lauritzen, S., and Cowell, R. (1993). Bayesian analysis in expert systems. *Statistical Science*, 8:219–282.

[Spiegelhalter and Lauritzen, 1990] Spiegelhalter, D. and Lauritzen, S. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605.

[Spirtes et al., 1993] Spirtes, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction, and Search*. Springer-Verlag, New York.

[Suzuki, 1993] Suzuki, J. (1993). A construction of Bayesian networks from databases based on an mdl scheme. In *Proceedings of Ninth Conference on Uncertainty in Artificial Intelligence*, Washington, DC, pages 266–273. Morgan Kaufmann.

[York, 1992] York, J. (1992). *Bayesian methods for the analysis of misclassified or incomplete multivariate discrete data*. PhD thesis, Department of Statistics, University of Washington, Seattle.

Appendix: k-LEARN is NP-Complete

As mentioned in Section 4.2, we easily can translate results in Höffgen (1993) for PAC learning to show that k-LEARN is NP-hard. In this section, we show that k-LEARN is NP-complete, even when we use the likelihood-equivalent BDe metric and the constraint of prior equivalence.

In proving our result, we must be careful in specifying the inputs to k-LEARN when the BDe metric is used. These inputs are (1) the relative prior probabilities of all network structures where each node has no more than k parents, (2) a database, and (3) parameters N'_{ijk} for some node–parent pairs and some values of i, j, and k. The input need only include enough parameters N'_{ijk} so that a score can be computed for all network structures where each node has no more than k parents. Consequently, we do not need parameters for nodes having more than k parents, nodes with parent configurations that always have zero prior probabilities, and values of i, j, and k for which there is no corresponding data in the database. Also, we emphasize that the parameters N'_{ijk} must be derivable from some joint probability distribution using Equation 4.

Given these inputs, we see from Equation 3, that the BDe metric for any given network structure and database can be computed in polynomial time. Consequently, k-LEARN is in NP.

In the following sections, we show that k-LEARN is NP-hard. In Section 7.1, we give a polynomial time reduction from a known NP-complete problem to 2-LEARN. In Section 7.2, we show that 2-LEARN is NP-hard using the reduction from Section 7.1, and then show that k-LEARN for $k > 2$ is NP-hard by reducing 2-LEARN to k-LEARN. In this discussion, we omit conditioning on background information ξ to simplify the notation.

7.1 Reduction from DBFAS to 2-LEARN

We show that 2-LEARN is NP-hard when $M(D, B_S)$ is the BDe metric by using a reduction from a restricted version of the feedback arc set problem. The general feedback arc set problem is stated in Garey and Johnson (1979) as follows:

FEEDBACK ARC SET
INSTANCE: Directed graph $G = (V, A)$, positive integer $K \leq |A|$.

QUESTION: Is there a subset $A' \subset A$ with $|A'| \leq K$ such that A' contains at least one arc from every directed cycle in G?

It is shown in Garvill (1977) 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.

Given an instance of DBFAS consisting of $G = (V, A)$ and K, our task is to specify, in polynomial time, the following components of the instance of 2-LEARN:

1. A variable set U
2. A database D
3. The relative prior probabilities of network structures

4. The necessary parameters N'_{ijk} (see comment at the beginning of this section)

5. A value p

Note that we need only specify relative prior probabilities because, as discussed in Section 1, the metric is arbitrary up to a proportionality constant. We then show that this instance of 2-LEARN has a solution if and only if the given instance of DBFAS has a solution. In the instance of DBFAS, we assume that no vertex has in-degree or out-degree of zero. If such a node exists, none of the incident edges can participate in a cycle, and we can remove them all from the graph.

To help distinguish between the instance of DBFAS and the instance of 2-LEARN, we adopt the following convention. We use the term *arc* to refer to a directed edge in the instance of DBFAS, and the term *edge* to refer to a directed edge in the instance of 2-LEARN.

We construct the variable set U as follows. For each node v_i in V, we include a corresponding binary variable v_i in U. We use $\mathcal{V}$ to denote the subset of U that corresponds to V. For each arc $a_i \in A$, we include five additional binary variables $a_{i1}, \ldots, a_{i5}$ in U. We use $\mathcal{A}_i$ to denote the subset of U containing these five variables, and define $\mathcal{A}$ to be $\mathcal{A}_1 \cup \ldots \cup \mathcal{A}_{|A|}$. We include no other variables in U.

The database D consists of a single case $C_1 = \{1, \ldots, 1\}$.

The relative prior probability of every network structure is one. This assignment satisfies our constraint of prior equivalence. From Equation 3 with database $D = C_1$ and relative prior probabilities equal to one, the BDe metric—denoted $M_{BDe}(D, B_S)$—becomes

$$M_{BDe}(C_1, B_S) = \prod_i \frac{N'_{ijk}}{N'_{ij}} \tag{9}$$

where k is the state of x_i equal to one, and j is the instance of Π_i such that the state of each variable in Π_i is equal to one. The reduction to this point is polynomial.

Max: Tell me again why you need the notation Π_i rather than Π_i.

To specify the necessary N'_{ijk} parameters, we specify a prior network and then compute the parameters using Equation 4. From Equation 9, we have

$$M_{BDe}(C_1, B_S) = \prod_i p(x_i = 1 | \Pi_i = 1, \ldots, 1, B^h_{S_C}) \tag{10}$$

To demonstrate that the reduction is polynomial, we show that the prior network can be constructed in polynomial time, and that Equation 10 can be computed from the prior network in polynomial time when Π_i contains no more than two variables. We establish the time complexity of prior-network construction in the following paragraphs. Although general probabilistic inference in a Bayesian network is NP-hard, in Section 7.3 (Theorem 12), we show that each probability in Equation 10 can be inferred from the prior network in constant time due to the special structure of the network.

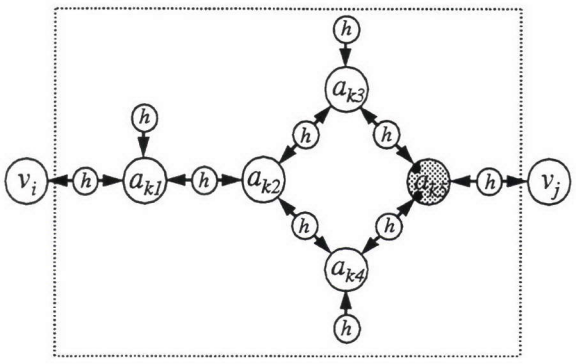


Figure 3: Subgraph of the prior net $\mathcal{B}$ corresponding to the *kth* arc in A from v_i to v_j.

We denote the prior Bayesian network $\mathcal{B} = (\mathcal{B}_S, \mathcal{B}_P)$. The prior network $\mathcal{B}$ contains both *hidden* nodes, which do not appear in U, and *visible* nodes which do appear in U. Every variable x_i in U has a corresponding visible node in $\mathcal{B}$ which is also denoted by x_i. There are no other visible nodes in $\mathcal{B}$. For every arc a_k from v_i to v_j in the given instance of DBFAS, $\mathcal{B}$ contains ten hidden binary nodes and the directed edges as shown in Figure 3.

In the given instance of DBFAS, we know that each node v_i in V is adjacent to either two or three nodes. For every node v_i in V which is adjacent to exactly two other nodes in G, there is a hidden node h_i in $\mathcal{B}$ and an edge from h_i to x_i. There are no other edges or hidden nodes in $\mathcal{B}$.

Every visible node in $\mathcal{B}$ is a sink and has exactly three hidden node parents, and every hidden node is a source with either one or two children. We use h_{ij} to denote the hidden node parent common to visible nodes x_i and x_j. We create the parameters $\mathcal{B}_P$ as follows. For every hidden node h_{ij} we set

$$p(h_{ij} = 0) = p(h_{ij} = 1) = \frac{1}{2}$$

Each visible node in $\mathcal{B}$ is one of two types. The type of a node is defined by its conditional probability distri-

bution. Every node a_{i5} in $\mathcal{B}$ (corresponding to the fifth variable created in U for the ith arc in the instance of DBFAS) is a *type II* node, and all other nodes are *type I* nodes. A type I node has the conditional probability distribution shown in Table 2.

Table 2: Conditional probability distribution for a type I node.

h_{ij}	h_{ik}	h_{il}	$p(x_i = 1 \mid h_{ij}, h_{ik}, h_{il})$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

We say that two variables in U are *prior siblings* if the corresponding nodes in the prior network $\mathcal{B}$ share a common hidden parent. We use S_{x_i} to denote the set of all variables in U which are prior siblings of x_i.

For each type II node a_{i5}, we define the *distinguished siblings* as the set $D_{a_{i5}} = \{a_{i3}, a_{i4}\} \subset S_{a_{i5}}$. Table 3 shows the conditional probability distribution of a type II node x_i with distinguished siblings $\{x_j, x_k\}$.[3]

Table 3: Conditional probability distribution for a type II node x_i with $D_{x_i} = \{x_j, x_k\}$.

h_{ij}	h_{ik}	h_{il}	$p(x_i = 1 \mid h_{ij}, h_{ik}, h_{il})$
0	0	0	$\frac{1}{3}$
0	0	1	1
0	1	0	$\frac{2}{3}$
0	1	1	0
1	0	0	$\frac{2}{3}$
1	0	1	0
1	1	0	$\frac{1}{3}$
1	1	1	0

There are $|V| + 5|A|$ visible nodes in $\mathcal{B}$, each visible node has at most three hidden node parents, and each probability table has constant size. Thus, the construction of $\mathcal{B}$ takes time polynomial in the size of the instance of DBFAS.

We now derive the value for p. From Equation 10, we obtain

$$
\begin{aligned}
&M_{BDe}(C_1, B_S) \\
&\quad = \prod_i p(x_i = 1 \mid \Pi_i = 1, \ldots, 1, B^h_{S_C}) \\
&\quad = \prod_i \delta^{3-|\Pi_i \cap S_{x_i}|} \cdot \frac{p(x_i = 1 \mid \Pi_i = 1, \ldots, 1, B^h_{S_C})}{\delta^{3-|\Pi_i \cap S_{x_i}|}} \\
&\quad = \delta^{\left(3n - \sum_i |\Pi_i \cap S_{x_i}|\right)} \prod_i \frac{p(x_i = 1 \mid \Pi_i = 1, \ldots, 1, B^h_{S_C})}{\delta^{3-|\Pi_i \cap S_{x_i}|}} \\
&\quad \equiv \delta^{\left(3n - \sum_i |\Pi_i \cap S_{x_i}|\right)} \prod_i s'(x_i \mid \Pi_i, S_i) \qquad (11)
\end{aligned}
$$

where $\delta < 1$ is a positive constant that we shall fix to be 15/16 for the remainder of the paper.

Let σ be the total number of prior sibling pairs as defined by $\mathcal{B}$, and let γ be the number of prior sibling pairs which are not adjacent in B_S. The sum $\sum_i |\Pi_i \cap S_{x_i}|$ is the number of edges in B_S which connect prior sibling pairs and is therefore equal to $\sigma - \gamma$. Rewriting Equation 11, we get

$$
\begin{aligned}
M_{BDe}(C_1, B_S) &= \delta^{(3n-(\sigma-\gamma))} \prod_i s'(x_i \mid \Pi_i, S_i) \\
&= \delta^{(3n-\sigma)} \cdot \delta^{\gamma} \prod_i s'(x_i \mid \Pi_i, S_i) \\
&= c' \cdot \delta^{\gamma} \prod_i s'(x_i \mid \Pi_i, S_i) \qquad (12)
\end{aligned}
$$

We now state 3 lemmas, postponing their proofs to Section 7.3. A network structure B_S is a *prior sibling graph* if all pairs of adjacent nodes are prior siblings. (Not all pairs of prior siblings in a prior sibling graph, however, need be adjacent.)

Lemma 1 *Let B_S be a network structure, and let $B_{S'}$ be the prior sibling graph created by removing every edge in B_S which does not connect a pair of prior siblings. Then it follows that $M_{BDe}(C_1, B_{S'}) \geq M_{BDe}(C_1, B_S)$*

Throughput the remainder of the paper, the symbol α stands for the constant 24/25.

Lemma 2 *If B_S is a prior sibling graph, then for every type I node x_i in B_S, if Π_i contains at least one element, then $s'(x_i \mid \Pi_i, S_i)$ is maximized and is equal to $m_1 = 64/135$. If $\Pi_i = \emptyset$, then $s'(x_i \mid \Pi_i, S_i) = \alpha \cdot m_1$.*

Lemma 3 *If B_S is a prior sibling graph, then for every type II node x_i in B_S, if $\Pi_i = D_{x_i}$, where D_{x_i} is the set of two distinguished siblings of x_i, then $s'(x_i \mid \Pi_i, S_i)$ is maximized and is equal to $m_2 = 40/81$. If $\Pi_i \neq D_{x_i}$ then $s'(x_i \mid \Pi_i, S_i) \leq \alpha \cdot m_2$.*

Finally, we define p in the instance of 2-LEARN as

$$p = c' m_1^{|V|} \left(m_1^4 m_2\right)^{|A|} \alpha^K \qquad (13)$$

[3] The type II node a_{i5} in Figure 1 is shaded and has small markers to indicate that a_{i3} and a_{i4} are its distinguished siblings.

where m_1 and m_2 are defined by Lemma 2 and 3 respectively, and c is the constant from Equation 12.

The value for p can be derived in polynomial time. Consequently, the entire reduction is polynomial.

7.2 Proof of NP-Hardness

In this section, we first prove that 2-LEARN is NP-hard using the reduction from the previous section. Then, we prove that K-LEARN is NP-hard for all $k > 1$, using a reduction from 2-LEARN.

The following discussion and lemma explain the selection of p made in Equation 13, which in turn facilitates the proof that 2-LEARN is NP-hard. Let γ_k be the number of prior sibling pairs $\{x_i, x_j\}$ which are not adjacent in B_S, where at least one of $\{x_i, x_j\}$ is in $\mathcal{A}_k$. We now argue that $\sum_k \gamma_k = \gamma$, where γ is the total number of prior sibling pairs not adjacent in B_S. First note that, as defined by $\mathcal{B}$, the prior siblings S_{v_i} for every node $v_i \in \mathcal{V}$ satisfy $S_{v_i} \subset \mathcal{A}$, and the prior siblings $S_{a_{kj}}$ for every node $a_{kj} \in \mathcal{A}_k$ satisfy $S_{a_{kj}} \subset \mathcal{V} \cup \mathcal{A}_k$ (see Figure 1). Thus, for any prior sibling pair $\{x_i, x_j\}$, there exists an $\mathcal{A}_k$ such that either both x_i and x_j are in $\mathcal{A}_k$, or exactly one of $\{x_i, x_j\}$ is in $\mathcal{A}_k$ and the other is in $\mathcal{V}$. Consequently, we have $\sum_k \gamma_k = \gamma$.

We can now express Equation 12 as

$$
\begin{aligned}
& M_{BDe}(C_1, B_S) \\
&= c' \left[\prod_{x_i \in \mathcal{V}} s'(x_i|\Pi_i, S_i) \right] \left[\prod_j \delta^{\gamma_j} \prod_{x_i \in A_j} s'(x_i|\Pi_i, S_i) \right] \\
&= c' \left[\prod_{x_i \in \mathcal{V}} s'(x_i|\Pi_i, S_i) \right] \left[\prod_j t(\mathcal{A}_j, \gamma_j) \right] \qquad (14)
\end{aligned}
$$

where $t(\mathcal{A}_j, \gamma_j) = \delta^{\gamma_j} \prod_{x_i \in A_j} s'(x_i|\Pi_i, S_i)$.

Lemma 4 *Let B_S be a prior sibling graph. If each node in $\mathcal{A}_k$ is adjacent to all of its prior siblings, and the orientation of the connecting edges are as shown in Figure 4, then $t(\mathcal{A}_k, \gamma_k)$ is maximized and is equal to $m_1^4 \cdot m_2$. Otherwise, $t(\mathcal{A}_k, \gamma_k) \leq \alpha \cdot m_1^4 \cdot m_2$.*

Proof: In Figure 4, every type I node in $\mathcal{A}_k$ has at least one prior sibling as a parent, and the single type II node has its distinguished siblings as parents. Thus, by Lemmas 2 and 3, the score $s'(x_i|\Pi_i, S_i)$ for each node $x_i \in \mathcal{A}_k$ is maximized. Furthermore, every pair of prior siblings are adjacent. Thus, we have

$$
\begin{aligned}
t(\mathcal{A}_j, \gamma_j) &= \delta^{\gamma_j} \prod_{x_i \in A_j} s'(x_i|\Pi_i, S_i) \\
&= \delta^0 \cdot m_1 \cdot m_1 \cdot m_1 \cdot m_1 \cdot m_2
\end{aligned}
$$

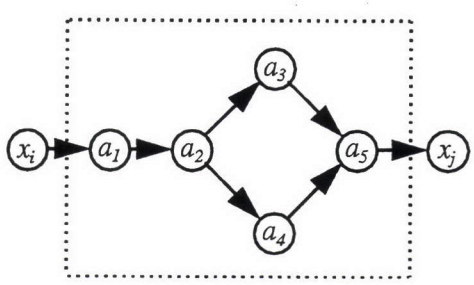


Figure 4: Optimal configuration of the edges incident to the nodes in $\mathcal{A}_k$ corresponding to the arc from v_i to v_j.

Suppose there exists another orientation of edges incident to the nodes in $\mathcal{A}_k$ such that that $t(\mathcal{A}_k, \gamma_k) > \alpha \cdot m_1^4 \cdot m_2$. Because $\delta < \alpha$ ($\frac{15}{16} < \frac{24}{25}$), every pair of prior siblings must be adjacent in this hypothetical configuration. Furthermore, every node in $\mathcal{A}_k$ must achieve its maximum score, else the total score will be bounded above by $\alpha \cdot m_1^4 \cdot m_2$. From Lemma 3, in order for a_{k5} to achieve its maximum score it must have its distinguished siblings a_{k3} and a_{k4} as parents. Because each node can have at most two parents, v_j must have a_{k5} as its parent. Both a_{k3} and a_{k4} must have a_{k2} as a parent, else they would be root nodes and by Lemma 2 would not attain a maximum score. Repeating this argument, a_{k2} must have a_{k1} as a parent and a_{k1} must have v_i as a parent. Because the resulting configuration is identical to Figure 4, the lemma follows. □

The next two theorems prove that 2-LEARN is NP-hard.

Theorem 5 *There exists a solution to the 2-LEARN instance constructed in Section 7.1 with $M_{BDe}(C_1, B_S) \geq p$ if there exists a solution to the given* DBFAS *problem with $A' \leq K$.*

Proof: Given a solution to DBFAS, create the solution to 2-LEARN as follows: For every arc $a_k = (v_i, v_j) \in A$ such that $a_k \notin A'$, insert the edges in B_S between the corresponding nodes in $\mathcal{A}_k \cup v_i \cup v_j$ as described in Lemma 4. For every arc $a_k = (v_i, v_j) \in A'$, insert the edges in B_S between the corresponding nodes in $\mathcal{A}_k \cup v_i \cup v_j$ as described in Lemma 4, except for the edge between a_{k1} and a_{k2} which is reversed and therefore oriented from a_{k2} to a_{k1}.

To complete the proof, we must first show that B_S is a solution to the 2-LEARN instance, and then show that $M_{BDe}(C_1, B_S)$ is greater than or equal to p.

Because each node in B_S has at most two parents, we know B_S is a solution as long as it is acyclic. Suppose B_S contains a cycle. Because there is no cycle con-

tained within any set $\mathcal{A}_k$, there must exist a sequence of nodes from $\mathcal{V}$ contained in this cycle such that for any pair of consecutive nodes (v_i, v_j) in the sequence, there is a directed path from v_i to v_j for which every intermediate node is in $\mathcal{A}$. However, we created the instance of 2-LEARN such that a directed path of this type exists if and only if $(v_i, v_j) \notin A'$. This implies there is a cycle in G for which none of the edges are contained in A', contradicting the fact that we have a solution to DBFAS.

We now derive $M_{BDe}(C_1, B_S)$. Let $\mathcal{A}^{opt}$ be the subset of $\mathcal{A}_k$ sets which correspond to the arcs in $A \setminus A'$. Rewriting Equation 14 we get

$$\begin{aligned} & M_{BDe}(C_1, B_S) \\ &= c' \left[\prod_{x_i \in \mathcal{V}} s'(x_i | \Pi_i, S_i) \right] \cdot \left[\prod_{\mathcal{A}_j \in \mathcal{A}^{opt}} t(\mathcal{A}_j, \gamma_j) \right] \\ &\quad \cdot \left[\prod_{\mathcal{A}_k \in \mathcal{A} \setminus \mathcal{A}^{opt}} t(\mathcal{A}_k, \gamma_k) \right] \end{aligned}$$

Every node $v_i \in \mathcal{V}$ has at least one prior sibling node as a parent because each node in the instance of DBFAS has an in-degree of at least one. Furthermore, Lemma 4 guarantees that for every $\mathcal{A}_k$ in $\mathcal{A}^{opt}$, $t(A_j, \gamma_j)$ equals $m_1^4 \cdot m_2$. Thus, we have

$$\begin{aligned} & M_{BDe}(C_1, B_S) \qquad (15) \\ &= c' m_1^{|\mathcal{V}|} \left[m_1^4 \cdot m_2 \right]^{|\mathcal{A}^{opt}|} \left[\prod_{\mathcal{A}_k \in \mathcal{A} \setminus \mathcal{A}^{opt}} t(\mathcal{A}_k, \gamma_k) \right] \\ &= c' m_1^{|V|} \left[m_1^4 \cdot m_2 \right]^{|\mathcal{A} \setminus \mathcal{A}'|} \left[\prod_{\mathcal{A}_k \in \mathcal{A} \setminus \mathcal{A}^{opt}} t(\mathcal{A}_k, \gamma_k) \right] \end{aligned}$$

Now consider any $\mathcal{A}_k$ in $\mathcal{A} \setminus \mathcal{A}^{opt}$. All prior sibling pairs for which at least one node is in this set are adjacent in B_S, so γ_k is zero. Furthermore, every node in this set attains a maximum score, except for the type I node a_{k2} which by Lemma 2 attains a score of $\alpha \cdot m_1$. Hence we conclude that $t(\mathcal{A}_k, \gamma_k)$ is equal to $m_1^4 \cdot m_2 \cdot \alpha$ and therefore

$$\begin{aligned} & M_{BDe}(C_1, B_S) \\ &= c' m_1^{|V|} \left[m_1^4 \cdot m_2 \right]^{|\mathcal{A} \setminus \mathcal{A}'|} \left[m_1^4 \cdot m_2 \cdot \alpha \right]^{|\mathcal{A} \setminus \mathcal{A}^{opt}|} \\ &= c' m_1^{|V|} \left[m_1^4 \cdot m_2 \right]^{|\mathcal{A} \setminus \mathcal{A}'|} \left[m_1^4 \cdot m_2 \right]^{|\mathcal{A}'|} \alpha^{|\mathcal{A}'|} \\ &= c' m_1^{|V|} \left[m_1^4 \cdot m_2 \right]^{|\mathcal{A}|} \alpha^{|\mathcal{A}'|} \end{aligned}$$

Because $\alpha < 1$ and $|A'| \leq K$ we conclude that $M_{BDe}(C_1, B_S) \geq p$. □

Theorem 6 *There exists a solution to the given* DBFAS *problem with $A' \leq K$ if there exists a solution to the 2-LEARN instance constructed in Section 7.1 with $M_{BDe}(C_1, B_S) \geq p$.*

Proof: Given the solution B_S to the instance of 2-LEARN, remove any edges in B_S which do not connect prior siblings. Lemma 1 guarantees that the BDe score does not decrease due to this transformation.

Now create the solution to DBFAS as follows. Recall that each set of nodes $\mathcal{A}_k$ corresponds to an arc $a_k = (v_i, v_j)$ in the instance of DBFAS. Define the solution arc set A' to be the set of arcs corresponding to those sets $\mathcal{A}_k$ for which the edges incident to the nodes in $\mathcal{A}_k$ are *not* configured as described in Lemma 4.

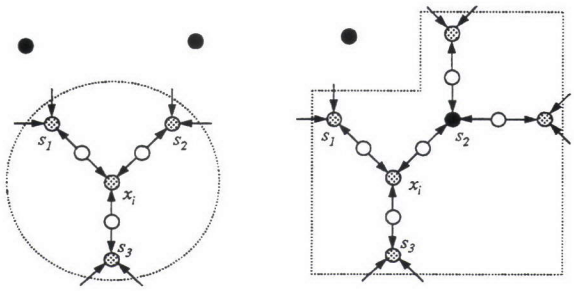
To complete the proof, we first show that A' is a solution to DBFAS, and then show that $|A'| \leq K$. Suppose that A' is not a solution to DBFAS. This means that there exists a cycle in G which does not pass through an arc in A'. For every arc (v_i, v_j) in this cycle, there is a corresponding directed path from v_i to v_j in B_S (see Figure 4). But this implies there is a cycle in B_S which contradicts the fact that we have a solution to 2-LEARN. From Lemma 4 we know that each set $\mathcal{A}_k$ which corresponds to an arc in A' has $t(\mathcal{A}_k, \gamma_k)$ bounded above by $\alpha \cdot m_1^4 \cdot m_2$. Because $M_{BDe}(C_1, B_S) \geq p$, we conclude from Equation 14 that there can be at most K such arcs. □

Theorem 7 *k-LEARN with $M_{BDe}(D, B_S)$ satisfying prior equivalence is NP-hard for every integer $k > 1$.*

Proof: Because 2-LEARN is NP-hard, we establish the theorem by showing that any 2-LEARN problem can be solved using an instance of k-LEARN.

Given an instance of 2-LEARN, an equivalent instance of k-LEARN is identical to the instance of 2-LEARN, except that the relative prior probability is zero for any structure that contains a node with more than two parents. Note that no new parameters N'_{ijk} need be specified. It remains to be shown that this assignment satisfies prior equivalence. We can establish this fact by showing that no structure containing a node with more than two parents is isomorphic to a structure for which no node contains more than two parents.

In HGC (1994c), it is shown that for any two isomorphic structures B_{S_1} and B_{S_2}, there exists a finite sequence of arc reversals in B_{S_1} such that (1) after each reversal B_{S_1} remains isomorphic to B_{S_2}, (2) after all reversals $B_{S_1} = B_{S_2}$, and (3) if the edge $v_i \rightarrow v_j$ is the next edge to be reversed, then v_i and v_j have the same parents with the exception that v_i is also a parent of v_j. It follows that after each reversal, v_i has the same number of parents as v_j did before the reversal, and v_j has the same number of parents as v_i did before the reversal. Thus, if there exists a node with l parents



(a) (b)

Figure 5: Two portions of $\mathcal{B}$ for which Theorem 8 applies. s_1, s_2 and s_3 are the elements of S_{x_i}. White nodes are hidden and black nodes are members of Π_i. In both (a) and (b), x_i is d-separated from any node outside the dashed line.

in some structure B_S, then there exists a node with l parents in any structure which is isomorphic to B_S. □

7.3 Proof of Lemmas

To prove Lemmas 1 through 3, we derive $s'(x_i|\Pi_i, S_{x_i})$ for every pair $\{x_i, \Pi_i\}$. Let x_i be any node. The set Π_i must satisfy one of the following mutually exclusive and collectively exhaustive assertions:

Assertion 1 For every node x_j which is both a parent of x_i and a prior sibling of x_i (i.e. $x_j \in \Pi_i \cap S_{x_i}$), there is no prior sibling of x_j which is also a parent of x_i.

Assertion 2 There exists a node x_j which is both a parent of x_i and a prior sibling of x_i, such that one of the prior siblings of x_j is also a parent of x_i.

The following theorem shows that to derive $s'(x_i|\Pi_i, S_{x_i})$ for any pair $\{x_i, \Pi_i\}$ for which Π_i satisfies Assertion 1, we need only compute the cases for which $\Pi_i \subseteq S_{x_i}$. Figure 5 shows two examples of the relevant portion of a prior network for which the theorem applies.

Theorem 8 *Let x_i be any node in B_S. If Π_i satisfies Assertion 1, then $s'(x_i|\Pi_i, S_{x_i}) = s'(x_i|\Pi_i \cap S_{x_i}, S_{x_i})$.*

Proof: From Equation 11, we have

$$s'(x_i|\Pi_i, S_{x_i}) = \frac{p(x_i|\Pi_i, B^{\epsilon}_{S_C})}{\delta^{3-|\Pi_i \cap S_{x_i}|}} \tag{16}$$

Thus, the theorem follows if we show that x_i is d-separated from all parents which are not prior siblings in $\mathcal{B}$ once the parents which are prior siblings are known.

Suppose there exists a parent x_k which is not a prior sibling of x_i and is not d-separated from x_i given $\Pi_i \cap S_{x_i}$. This implies that there is an active trail from x_i to x_k in $\mathcal{B}$ once we condition on these nodes. Any path—active or nonactive—from a nonsibling parent to x_i must pass through an element of S_{x_i}. Because each node in S_{x_i} is a sink, any active path from x_k to x_i must pass through some node x_j from $\Pi_i \cap S_{x_i}$. Because Π_i satisfies Assertion 1, however, we know that none of the siblings of x_j are in Π_i. Hence, no active path from x_k to x_i can pass through x_j. □

For the next two theorems, we use the following equalities.[4]

$$p(h_{ij}, h_{ik}, h_{il}) = p(h_{ij})p(h_{ik})p(h_{il}) \tag{17}$$

$$p(h_{ij}, h_{ik}, h_{il}|x_j) = p(h_{ij}|x_j)p(h_{ik})p(h_{il}) \tag{18}$$

$$p(h_{ij}, h_{ik}, h_{il}|x_j, x_k) = p(h_{ij}|x_j)p(h_{ik}|x_k)p(h_{il}) \tag{19}$$

$$p(h_{ij} = 0|x_i = 1) = \frac{2}{3} \tag{20}$$

Equation 17 follows because each hidden node is a root in $\mathcal{B}$. Equation 18 follows because any path from x_j to either h_{ik} or h_{il} must pass through some node $x \neq x_j$ which is a sink. Equation 19 follows from a similar argument, noting from the topology of $\mathcal{B}$ that $x \notin \{x_j, x_k\}$. Equation 20 follows from Tables 1 and 2, using the fact that $p(h_{ij} = 0)$ equals $1/2$.

Theorem 9 *Let x_i be any type I node in B_S for which Π_i satisfies Assertion 1.*

1. *If $|\Pi_i \cap S_{x_i}| = 0$ then $s'(x_i|\Pi_i, S_{x_i}) = \alpha \cdot m_1$*
2. *If $|\Pi_i \cap S_{x_i}| = 1$ then $s'(x_i|\Pi_i, S_{x_i}) = m_1$*
3. *If $|\Pi_i \cap S_{x_i}| = 2$ then $s'(x_i|\Pi_i, S_{x_i}) = m_1$*

Proof: Using Equations 17 through 20 and Table 2, we obtain the following relations:

1. If $|\Pi_i \cap S_{x_i}| = 0$ then $p(x_i = 1) = \delta^3 \cdot \alpha \cdot m_1$
2. If $|\Pi_i \cap S_{x_i}| = \{x_j\}$ then $p(x_i = 1|x_j = 1) = \delta^2 \cdot m_1$
3. If $|\Pi_i \cap S_{x_i}| = \{x_j, x_k\}$ then $p(x_i = 1|x_j = 1, x_k = 1) = \delta \cdot m_1$

[4] We drop the conditioning event $B^{\epsilon}_{S_C}$ and ξ to simplify notation.

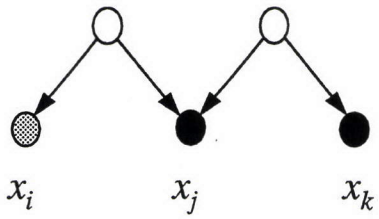


Figure 6: Portion of $\mathcal{B}$ for which Theorem 11 applies. White nodes are hidden and black nodes are members of Π_i. Node x_i has parents x_j and x_k, where $x_j \in S_{x_i}$ and $x_k \in S_{x_j}$. In this case, x_i is not d-separated from its non-sibling parent.

The theorem then follows from Equation 16. □

Theorem 10 *Let x_i be any type II node in B_S for which Π_i satisfies assertion 1.*

1. *If $|\Pi_i \cap S_{x_i}| = 0$ then $s'(x_i|\Pi_i) = \alpha^2 \cdot m_2$*
2. *If $|\Pi_i \cap S_{x_i}| = 1$ then $s'(x_i|\Pi_i) = \alpha \cdot m_2$*
3. *If $|\Pi_i \cap S_{x_i}| = 2$ and $\Pi_i \neq D_{x_i}$ then $s'(x_i|\Pi_i) = \alpha \cdot m_2$*
4. *If $\Pi_i = D_{x_i}$ then $s'(x_i|\Pi_i) = m_2$*

Proof: Using Equations 17 through 20 and Table 3, we obtain the following relations:

1. If $|\Pi_i \cap S_{x_i}| = 0$ then $p(x_i = 1) = \delta^3 \cdot \alpha^2 \cdot m_2$
2. If $|\Pi_i \cap S_{x_i}| = \{x_j\}$ then $p(x_i = 1|x_j = 1) = \delta^2 \cdot \alpha \cdot m_2$
3. If $|\Pi_i \cap S_{x_i}| = \{x_j, x_k\}$ and $\{x_j, x_k\} \neq D_{x_i}$ then $p(x_i = 1|x_j = 1, x_k = 1) = \delta \cdot \alpha \cdot m_2$
4. If $\Pi_i = \{x_j, x_k\}$ and $\{x_j, x_k\} = D_{x_i}$ then $p(x_i = 1|x_j = 1, x_k = 1) = \delta \cdot m_2$

The theorem then follows from Equation 16. □

Now we show that if Assertion 2 holds for the parents of some node, then we can remove the edge from the parent which is not a sibling without decreasing the score. Once this theorem is established, the lemmas follow.

Figure 4 shows the prior network $\mathcal{B}$ when Assertion 2 holds for a node x_i.

Theorem 11 *Let x_i be any node. If $\Pi_i = \{x_j, x_k\}$, where $x_j \in S_{x_i}$ and $x_k \in S_{x_j}$, then $s'(x_i|x_j) \geq s'(x_i|x_j, x_k)$.*

Proof: For any node we have

$$p(x_i = 1|x_j = 1, x_k = 1) = \frac{p(x_i = 1)p(x_k = 1|x_i = 1)p(x_j = 1|x_i = 1, x_k = 1)}{p(x_k = 1)p(x_j = 1|x_k = 1)}$$

Because x_i and x_k are not prior siblings, it follows that $p(x_k|x_i) = p(x_k)$. Therefore, we have

$$p(x_i = 1|x_j = 1, x_k = 1) = p(x_i = 1) \cdot \frac{p(x_j = 1|x_i = 1, x_k = 1)}{p(x_j = 1|x_k = 1)}$$

Expressing this equality in terms of $s'(x_i|\Pi_i, S_{x_i})$, noting that x_i has only one prior sibling as a parent, and canceling terms of δ, we obtain

$$s'(x_i|\{x_j, x_k\}, S_{x_i}) = s'(x_i|\emptyset, S_{x_i}) \frac{s'(x_j|\{x_i, x_k\}, S_{x_j})}{s'(x_j|\{x_k\}, S_{x_j})} \quad (21)$$

If x_j is a type I node, or if x_j is a type II node and x_i and x_k are *not* its distinguished siblings, then $s'(x_j|\{x_i, x_k\}, S_{x_j})$ equals $s'(x_j|\{x_k\}, S_{x_j})$. Thus, from Equation 21, we have

$$s'(x_i|\{x_j, x_k\}, S_{x_i}) = s'(x_i|\emptyset, S_{x_i}) < s'(x_i|\{x_j\}, S_{x_i}) \quad (22)$$

which implies that we can improve the local score of x_i by removing the edge from x_k.

If x_j is a type II node, and $D_{x_j} = \{x_i, x_k\}$, then $s'(x_j|\{x_i, x_k\}, S_{x_j})$ equals $(1/\alpha) \cdot s'(x_j|\{x_k\}, S_{x_j})$. Also, from Equation 21, we have

$$s'(x_i|\{x_j, x_k\}, S_{x_i}) = s'(x_i|\emptyset, S_{x_i}) \frac{1}{\alpha} = s'(x_i|\{x_j\}, S_{x_i}) \quad (23)$$

Therefore, we can remove the edge from x_k without affecting the score of x_i. □

The preceding arguments also demonstrate the following theorem.

Theorem 12 *For any pair $\{x_i, \Pi_i\}$, where $|\Pi_i| \leq 2$, the value $p(x_i = 1|\Pi_i)$ can be computed from $\mathcal{B}$ in constant time when the state of each of the variable in Π_i is equal to one.*