Noname manuscript No.
(will be inserted by the editor)

Scoring Bayesian Networks of Mixed Variables

Bryan Andrews · Joseph Ramsey · Greg Cooper

Received: date / Accepted: date

Abstract In this paper we outline two novel scoring methods for learning Bayesian networks in the presence of both continuous and discrete variables, that is, mixed variables. While much work has been done in the domain of automated Bayesian network learning, few studies have investigated this task in the presence of both continuous and discrete variables while focusing on scalability. Our goal is to provide two novel and scalable scoring function capable of handling mixed variables. The first method, the Conditional Gaussian (CG) score, provides a highly efficient option. The second method, the Mixed Variable Polynomial (MVP) score, allows for a wider range of relationships, including non-linearity, but is noticeably slower than CG. Both scores calculate log likelihood and degrees of freedom terms, which are incorporated into a Bayesian Information Criterion (BIC) score. Additionally, we introduce a structure prior for efficient learning of large networks and a simplification in scoring the discrete case which performs well empirically. While the core of this work focuses on applications in the search and score paradigm, we also show how the introduced scoring functions may be readily adapted as conditional independence tests for constraint-based Bayesian network learning methods. Lastly, we describe ways to simulate mixed variable networks and apply the simulations to derive performance results for the proposed methods.

Keywords Bayesian network learning · mixed variables · hybrid Bayesian network learning · causal search

Research reported in this publication was supported by grant U54HG008540 awarded by the National Human Genome Research Institute through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. This research was also supported by grant #4100070287 from the Pennsylvania Department of Health. The Department specifically disclaims responsibility for any analyses, interpretations, or conclusions. This research was also supported by the National Library of Medicine, Award Number R01LM012087. The content is solely the responsibility of the author and does not necessarily represent the official views of the granting agencies. This work has been completed without any conflicts of interest.

B. Andrews
University of Pittsburgh, Pittsburgh, PA 15260, USA
E-mail: bja43@pitt.edu

J. Ramsey
Carnegie Mellon University, Pittsburgh, PA 15213, USA
E-mail: jdramsey@andrew.cmu.edu

G. Cooper
University of Pittsburgh, Pittsburgh, PA 15260, USA
E-mail: gfc@pitt.edu

1 Introduction

Bayesian networks are a widely used graphical framework for representing probabilistic relationships among a set of variables. Under assumptions, we can interpret such a network as a causal model [19]. In general, a Bayesian network consists of two components, a structure component and a distribution component. The structure component encodes conditional independence relationships between variables allowing for efficient factorization, while the distribution component parameterizes the probabilistic relationships among the variables. In this paper, our interests lie in learning the structure component of Bayesian networks, represented by a Directed Acyclic Graph (DAG). Automated Bayesian network learning from data is an important and active area of research. However, few researchers have investigated this task in the presence of both continuous and discrete variables [2,7–9,13,16,18]. In the limited work that has been done, researchers either ignore the case where continuous variables are parents of a discrete variable. or do not provide solutions that scale much beyond 100 variables. The goal of this paper is to provide solutions for researchers working with datasets of hundreds of variables.

Most methods for learning Bayesian networks fall into one of two categories: search and score or constraint-based. Search and score methods heuristically search the space of structures using an objective function to evaluate fitness. Constraint-based methods use conditional independence tests to systematically assess the fit of each edge in a DAG. While the core of this paper focuses on the search and score paradigm, we also show how our proposed scoring functions may be readily adapted as a conditional independence test for constraint-based methods. For additional background information on Bayesian networks and learning their structures, see [5]. The remainder of this paper is organized as follows. Section 2 discusses scoring functions and the Bayesian Information Criterion (BIC). Sections 3 and 4 introduce the Conditional Gaussian (CG) score and the Mixed Variable Polynomial (MVP) score respectively. Section 5 details several adaptations of the introduced methods. Section 6 reports empirical results of the CG and MVP methods on data generated using simulation. Section 7 provides discussion and conclusions.

2 Scoring Bayesian Networks

Search and score methods utilize an objective function to evaluate the fitness of DAGs on a given dataset $\mathcal{D}$. Let S be a score function and $\mathcal{G}$ be a DAG containing m variables. Let Y_i be the i^{th} variable with parents Pa_i for $i \in \{1, 2, \ldots, m\}$. When scoring $\mathcal{G}$, most search algorithms require that S decomposes into local components involving only Y_i and Pa_i. This property is known as decomposability. Given a score is decomposable, we need only compare the differing local components when deciding between two DAGs. To solidify this concept, we say a score S is decomposable if it can be represented as a sum of local components. We score DAG $\mathcal{G}$ on dataset $\mathcal{D}$ using score S as,

$$S(\mathcal{G}, \mathcal{D}) = \sum_{i=1}^{m} s(Y_i|Pa_i),$$

where $s(Y_i|Pa_i)$ is the score of the current local component.

Note that several DAGs can encode the same set of conditional independence relationships. A set of DAGs which encodes the same independencies is known as a Markov Equivalence Class (MEC). If a scoring function S scores all DAGs in the same MEC equally, then S is score equivalent. To clarify, let $\mathcal{G}$ and $\mathcal{G}'$ be DAGs over dataset $\mathcal{D}$. If DAGs $\mathcal{G}$ and $\mathcal{G}'$ encode the same conditional independence relationships and S is score equivalent, then $S(\mathcal{G}, \mathcal{D}) = S(\mathcal{G}', \mathcal{D})$. This is a desirable trait because it allows search algorithms, such as Greedy Equivalent Search (GES) [4], to search over MECs directly.

Another favorable trait for scores, which algorithms such as GES require for optimality in the sample limit, is consistency. Consistency, in terms of a search and score procedure, states that a DAG $\mathcal{G}$ containing all the conditional independence relationships in the data $\mathcal{D}$ will score higher than a DAG $\mathcal{G}'$ which does not contain all such relationships, $S(\mathcal{G}, \mathcal{D}) > S(\mathcal{G}', \mathcal{D})$. Furthermore, if two DAGs $\mathcal{G}$ and $\mathcal{G}'$ both contain all the conditional independence relationships in $\mathcal{D}$, but $\mathcal{G}$ contains fewer parameters, then $\mathcal{G}$ will score higher, $S(\mathcal{G}, \mathcal{D}) > S(\mathcal{G}', \mathcal{D})$.

2.1 The Bayesian Information Criterion

The Bayesian Information Criterion (BIC) is a widely used scoring measure for model selection. Let M be a model we wish to score given a dataset $\mathcal{D}$. We can write the probability of model M given $\mathcal{D}$ using Bayes' rule as,

$$p(M|\mathcal{D}) = \frac{p(\mathcal{D}|M)p(M)}{p(\mathcal{D})}.$$

However, since $p(\mathcal{D})$ does not depend on M and remains constant across different models, we only consider,

$$p(M|\mathcal{D}) \propto p(\mathcal{D}|M)p(M). \tag{1}$$

BIC aims to approximate $p(M|\mathcal{D})$ in (1). For now, we assume $p(M)$ is distributed uniformly and thus drop it. Later in section 5.1, we introduce an alternative distribution for $p(M)$, which we find performs well in practice. Reference [10] shows, when allowing for a prior over the parameters, the logarithm of $p(\mathcal{D}|M)$ can be approximated as:

$$\log p(\mathcal{D}|M) \approx -2\ell(\boldsymbol{\theta}) + df \log n, \tag{2}$$

where $\ell(\boldsymbol{\theta})$ is the maximum log likelihood of the data, df are the degrees of freedom, and n is the sample size. The approximation on the right hand side of (2) characterizes the BIC, introduced by [15]. BIC is decomposable and can be readily applied to score Bayesian networks. In sections 3 and 4, we detail how to calculate the log likelihood and degrees of freedom terms for

BIC using our proposed scoring methods. We score a DAG $\mathcal{G}$ by calculating and summing over BIC values for each variable Y_i and its parents Pa_i in $\mathcal{G}$.

3 The Conditional Gaussian Score

The Conditional Gaussian (CG) score calculates conditional Gaussian mixtures using ratios of joint distributions. We make the following assumptions for the CG sore.

Assumption 1 *The data were generated from a Gaussian mixture where each Gaussian is defined for a setting of the discrete variables.*

Assuming the data are generated from such a model inherently biases the score towards favoring discrete parents of continuous children since the discrete variables define how the Gaussians in the mixture are formed. However, this assumption allows for very efficient calculations of the models to be computed by the score. In section 6, we see that even with such an assumption, CG performs quite well.

Assumption 2 *The instances in the data are independent and identically distributed.*

The data are assumed to be i.i.d. so that we can calculate the log likelihood as a sum over the marginal log probabilities for each instance in the data.

Since CG uses BIC as a framework to evaluate its approximations, the score is decomposable into a sum of parent-child relationships. In order to outline such a relationship, we introduce continuous variables C_1, C_2 and discrete variables D_1, D_2. Below we detail how CG forms approximations using these four variables with both a continuous and discrete child, however this procedure can easily be completed with any number of variables. We approximate the conditional distribution where C_1 is a child with parents C_2, D_1, and D_2 as,

$$
\begin{aligned}
p(C_1|C_2, D_1, D_2) &= \frac{p(C_1, C_2, D_1, D_2)}{p(C_2, D_1, D_2)} \\
&= \frac{p(C_1, C_2|D_1, D_2)p(D_1, D_2)}{p(C_2|D_1, D_2)p(D_1, D_2)}
\end{aligned}
\tag{3}
$$

and the conditional distribution where D_1 is a child with parents C_1, C_2, and D_2 as,

$$
\begin{aligned}
p(D_1|C_1, C_2, D_2) &= \frac{p(C_1, C_2, D_1, D_2)}{p(C_1, C_2, D_2)} \\
&= \frac{p(C_1, C_2|D_1, D_2)p(D_1, D_2)}{p(C_1, C_2|D_2)p(D_2)}.
\end{aligned}
\tag{4}
$$

In (3) and (4), we can straightforwardly calculate $p(C_1, C_2|D_1, D_2)$ and $p(C_1, C_2|D_2)$ using Gaussian distributions partitioned on the discrete variables and $p(D_1, D_2)$ and $p(D_2)$ using multinomial distributions. It is important to note that if we treat $p(C_1, C_2, D_1, D_2)$ as Gaussian for each setting of D_1 and D_2, then to calculate $p(C_1, C_2, D_2)$ correctly, we must treat each setting of D_2 as a mixture of Gaussians. We approximate this mixtures with single Gaussian since, by the central limit theorem, we expect these distributions to converge to a Gaussian. In section 6.1 we validate this approximation experimentally.

CG performs the above approximations in log space as differences of log distributions. The score of a DAG $\mathcal{G}$ given a dataset $\mathcal{D}$ is calculated as the sum across all BIC measures of similar parent-child relationships. Additionally, because of how CG computes each parent-child relation, the score is score equivalent and, assuming the denominator is a single Gaussian, CG is consistent. See the supplementary materials for proof.

In the remainder of the section introduction we provide a high-level overview of the CG method; Sections 3.1 - 3.2 provide details. Let Y_i be the i^{th} variable in $\mathcal{G}$ with the set Pa_i containing the parents of Y_i. Furthermore, let Pa_i consist of two mutually exclusive subsets Pc_i and Pd_i such that Pc_i and Pd_i hold the continuous and discrete parents of Y_i respectively. To evaluate the parent-child relationships between a variable Y_i and its parents Pa_i, CG calculates the log likelihood and degrees of freedom for two sets of variables, $\{Y_i \cup Pa_i\}$ and Pa_i. The log likelihood of Y_i given its parents Pa_i is computed as the difference between the log likelihood terms for $\{Y_i \cup Pa_i\}$ and Pa_i. Similarly, the degrees of freedom are calculated as the count difference in parameters used to fit $\{Y_i \cup Pa_i\}$ and Pa_i. When evaluating one of the two sets of variables, the data $\mathcal{D}$ for the variables in that set are first partitioned according to the discrete values. That is, we divide the data using a partitioning set Π_i with a partition for each combination assignment of the discrete variables. Further, we form a design matrix $\boldsymbol{X}_p$ for each partition $p \in \Pi_i$. $\boldsymbol{X}_p$ holds the values of the continuous variables in the set for partition p and is used to fit a Gaussian. We additionally fit a multinomial according to counts. Lastly, BIC computes the score using the log likelihood and degrees of freedom.

3.1 Modeling a Set of Variables

When using CG, we have three different kinds of sets to model: $\{Y_i \cup Pa_i\}$ where Y_i is continuous, $\{Y_i \cup Pa_i\}$ where Y_i is discrete, and Pa_i. They all follow the same generic format so we will describe the process in general while pointing out the subtle differences where they apply.

First we partition the data with respect to a partitioning set Π_i generated according to the discrete variables Pd_i. Note that if our set includes a discrete child Y_i, then the discrete variables are comprised of $\{Y_i \cup Pd_i\}$ and we partition according to these variables. Π_i contains a partition for every combination values in the discrete variables Pd_i. We define the partitioning set Π_i using a Cartesian product of the discrete variables. Let $|Pd_i| = d$, then partitioning set $\Pi_i = (Y_i) \times Pd_i(1) \times Pd_i(2) \times \cdots \times Pd_i(d)$ where Y_i is the set of values for the child (included only if Y_i is discrete), $Pd_i(1)$ is the set of values for the first discrete parent, $Pd_i(2)$ is the set of values for the second discrete parent, and so forth.

Let $|Pc_i| = c$, then for each partition $p \in \Pi_i$ we define a design matrix $\boldsymbol{X}_p$ with n_p observations and c variables corresponding to the variables in Pc_i. Here, if our set includes a continuous child Y_i, then we instead define $\boldsymbol{X}_p$ with $c+1$ variables corresponding to the variables in $\{Y_i \cup Pc_i\}$. That is,

$$\boldsymbol{X}_p = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1c} & (y_1) \\ x_{21} & x_{22} & \dots & x_{2c} & (y_2) \\ \vdots & \vdots & \ddots & \vdots & (\vdots) \\ x_{n_p1} & x_{n_p2} & \dots & x_{n_pc} & (y_{n_p}) \end{bmatrix},$$

where x_{jk} is the j^{th} value of the k^{th} variable in Pc_i and y_j is the j^{th} value of the child Y_i (included only if Y_i is continuous) for $j \in \{1, 2, \dots, n_p\}$ and $k \in \{1, 2, \dots, c\}$. In the definition of $\boldsymbol{X}_p$ above, we drop the subscript p on the elements in the matrix for a less cramped formulation.

3.2 Calculating the Log Likelihood and Degrees of Freedom

The calculations for the three aforementioned sets are identical in formulation, so without loss of generality, we demonstrate the log likelihood and degrees of freedom calculations for the set $\{Y_i \cup Pa_i\}$. The log likelihood for a set is calculated component-wise over each partition and summed together as follows,

$$\ell_{\{Y_i \cup Pa_i\}}(\boldsymbol{\theta}|\boldsymbol{X}) = \sum_{p \in \Pi_i} \ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p). \tag{5}$$

The degrees of freedom are calculated in a similar manner,

$$df_{\{Y_i \cup Pa_i\}}(\boldsymbol{\theta}) = \sum_{p \in \Pi_i} df_p(\boldsymbol{\theta}_p) - 1, \tag{6}$$

where the minus 1 term accounts for the redundant mixing component. Let N be the number of observations in the unpartitioned dataset. For each partition $p \in \Pi_i$, let k be the number of variables in $\boldsymbol{X}_p$ and $\boldsymbol{x}_j$ be the j^{th} observation in $\boldsymbol{X}_p$. Additionally, assume the subscript p is implicit on $\boldsymbol{x}_j$ for the remainder of this section. From [1], we calculate the Gaussian log likelihood for partition $p \in \Pi_i$ as,

$$\begin{aligned}\ell(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p|\boldsymbol{X}_p) = &-\frac{n_p k}{2}\log 2\pi - \frac{n_p}{2}\log|\boldsymbol{\Sigma}_p| \\ &-\frac{1}{2}\sum_{j=1}^{n_p}(\boldsymbol{x}_j - \boldsymbol{\mu}_p)^T \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu}_p),\end{aligned} \tag{7}$$

where $\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p$ are the vector valued mean and variance of the Gaussian distribution respectively. The maximum likelihood estimate $\hat{\boldsymbol{\Sigma}}_p$ is computed as,

$$\hat{\boldsymbol{\Sigma}}_p = \frac{1}{n_p}\sum_{j=1}^{n_p}(\boldsymbol{x}_j - \bar{\boldsymbol{x}_p})(\boldsymbol{x}_j - \bar{\boldsymbol{x}_p})^T. \tag{8}$$

Let $\boldsymbol{\mu}_p = \bar{\boldsymbol{x}_p}$. We note that while $\bar{\boldsymbol{x}_p}$ converges quickly to $\boldsymbol{\mu}_p$. Using the estimate in (8), the log likelihood in (7) simplifies to,

$$\ell(\hat{\boldsymbol{\Sigma}}_p|\boldsymbol{X}_p) = -\frac{n_p}{2}(\log|\hat{\boldsymbol{\Sigma}}_p| + k\log 2\pi + 1). \tag{9}$$

We use (9) to compute the log likelihood of a Gaussian conditioned on discrete variables. However, we still must add the log probability of an instance being from partition $p \in \Pi_i$ to each instance in partition p to calculate the desired joint log likelihood. These probabilities are computed using the maximum likelihood estimate from variables distributed according to a multinomial distribution. This is the count of instances in p denoted n_p over the total count N of all instances: $\frac{n_p}{N}$. Thus, we calculate the log likelihood for partition $p \in \Pi_i$ as,

$$\ell_p(\hat{\boldsymbol{\theta}}_p|\boldsymbol{X}_p) = -\frac{n_p}{2}(\log|\hat{\boldsymbol{\Sigma}}_p| + k\log 2\pi + 1) + n_p \log\frac{n_p}{N}. \tag{10}$$

We use (10) to calculate $\ell_p(\boldsymbol{\theta}_p|\boldsymbol{X}_p)$ in (5). To find the number of parameters in partition $p \in \Pi_i$, we count the number of unique terms in $\hat{\boldsymbol{\Sigma}}_p$ plus one for the mixing component in (10). Therefore,

$$df_p(\hat{\boldsymbol{\theta}_p}) = \frac{k(k+1)}{2} + 1. \tag{11}$$

We use (11) to calculate $df_p(\boldsymbol{\theta}_p)$ in (6).

Using the form of (3) and (4), we calculate the log likelihood and degrees of freedom terms as,

$$\ell_i(\hat{\boldsymbol{\theta}}|\boldsymbol{X}) = \ell_{\{Y_i \cup Pa_i\}}(\hat{\boldsymbol{\theta}}|\boldsymbol{X}) - \ell_{Pa_i}(\hat{\boldsymbol{\theta}}|\boldsymbol{X}), \tag{12}$$

$$df_i(\hat{\boldsymbol{\theta}}) = df_{\{Y_i \cup Pa_i\}}(\hat{\boldsymbol{\theta}}) - df_{Pa_i}(\hat{\boldsymbol{\theta}}). \tag{13}$$

BIC uses (12) and (13) to compute the score for the parent-child relationship of Y_i given Pa_i.

4 The Mixed Variable Polynomial Score

The Mixed Variable Polynomial (MVP) score uses higher order polynomial functions to approximate relationships between any number of continuous and discrete variables. Since MVP uses BIC as a framework to evaluate its approximations, the score is decomposable into a sum of parent-child relationships. The MVP method scores the decomposed local components of a DAG $\mathcal{G}$ using the approximating polynomial functions. To motivate the ideas underlying this approach, we note the implications of Weierstrass's approximation theorem.

Weierstrass Approximation Theorem. *Suppose f is a continuous real-valued function defined on the real interval $[a, b]$. For every $\epsilon > 0$, there exists a polynomial p such that for all $x \in [a, b]$, we have $|f(x) - p(x)| < \epsilon$.*

In short, as long as a function f is continuous and the contributing variables exist within a bounded interval, then there exists a polynomial function which approximates f to an arbitrary degree of accuracy. This brings us to our first two assumptions.

Assumption 1 *The sample space of each variable is finite.*

To shed some light on this assumption, we note that MVP's approximations are functions of the continuous variables. Thus, the motivation for assumption 1 becomes apparent as a prerequisite of the previously stated theorem; finite sample spaces are bounded. Additionally, we note that MVP forms partitions according to values of discrete variables. Therefore, to enforce a finite number of partitions, we require that the discrete variables have finite support.

Assumption 2 *Each continuous variable is a continuous function of its continuous parents plus additive Gaussian noise. The probability mass functions of each discrete variable are described by continuous nonzero functions of the variable's continuous parents.*

The motivation for this assumption follows from Weierstrass's approximation theorem since f, the function to be approximated, must be continuous. However, along with assuming continuity, we restrict the model class in the continuous child case to those with additive Gaussian noise. This assumption allows us to use least squares regression in order to obtain maximum likelihood estimates efficiently for continuous variables. Additionally, we assume nonzero probability in the discrete case so that, in the sample limit, our approximation will be non-negative. It is worth noting that we do not assume linearity unlike other commonly used scores.

Assumption 3 *There are no interaction terms between continuous parents when defining a child variable in the network.*

We make this assumption for tractability. Modeling interactions among the continuous parents is a combinatorial problem. Thus, we forgo these interaction terms.

Assumption 4 *The instances in the data are independent and identically distributed.*

The data are assumed to be i.i.d. so that we can calculate the log likelihood as a sum over the marginal log probabilities for each instance in the data.

Under these assumptions, the MVP score is consistent in the large sample limit with the correct choice of maximum polynomial degree. See the supplementary materials for proof. However, due to the use of nonlinear functions, it is not score equivalent. In section 6, we see that even without this property, the MVP score still performs quite well. Additionally, there has been work suggesting that asymmetric scores can be beneficial in inferring causation, most notably [17].

4.1 Partitioned Regression

Let Y_i be the i^{th} variable in $\mathcal{G}$ and Pa_i be the set containing the parents of Y_i. Furthermore, let Pa_i consist of two mutually exclusive subsets Pc_i and Pd_i such that Pc_i and Pd_i hold the continuous and discrete parents of Y_i respectively. In general, to evaluate the local score component between Y_i and its parents Pa_i, MVP first partitions the data with respect to the discrete parents Pd_i and performs least squares regression using the continuous parents Pc_i. The log likelihood and degrees of freedom for the model are calculated depending on the variable type of Y_i. Lastly, BIC computes the local score component using the log likelihood and degrees of freedom.

The partitioning set Π_i with respect to the discrete parents Pd_i contains a partition for every combination of values in the discrete parents. We define the partitioning set Π_i using a Cartesian product of the discrete parents Pd_i. Let $|Pd_i| = d$, then partitioning set $\Pi_i = Pd_i(1) \times Pd_i(2) \times \cdots \times Pd_i(d)$ where $Pd_i(1)$ is the set of values for the first discrete parent, $Pd_i(2)$ is the set of values for the second discrete parent, and so forth.

Let $|Pc_i| = c$, then for each partition $p \in \Pi_i$ we define a design matrix $\boldsymbol{X}_p$ with n_p observations and c variables corresponding to the variables in Pc_i. Additionally, we add a bias term and higher order polynomial terms for each variable in Pc_i, stopping at a maximum order specified by the function $g(n_p)$,

$$\boldsymbol{X}_p = \begin{bmatrix} 1 & x_{11} & \dots & x_{1c} & x_{11}^2 & \dots & x_{1c}^2 & \dots & x_{11}^{g(n)} & \dots & x_{1c}^{g(n)} \\ 1 & x_{21} & \dots & x_{2c} & x_{21}^2 & \dots & x_{2c}^2 & \dots & x_{21}^{g(n)} & \dots & x_{2c}^{g(n)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{nc} & x_{n1}^2 & \dots & x_{nc}^2 & \dots & x_{n1}^{g(n)} & \dots & x_{nc}^{g(n)} \end{bmatrix}.$$

In this paper, we report two choices for $g(n_p)$: $g(n_p) = 1$, and $g(n_p) = \lfloor \log n_p \rfloor$. We have also tried other choices, such as $g(n_p) = 3$, but found the above options provide the best solutions. In the definition of $\boldsymbol{X}_p$ above and $\boldsymbol{x}_j$ and $\boldsymbol{y}_p$ below, we drop the subscript p on their elements and on n_p for a less cramped formulation. Define $\boldsymbol{x}_j$ as the j^{th} observation in $\boldsymbol{X}_p$ and $\boldsymbol{y}_p$, whose values come from the child Y_i, as the target vector for least squares regression,

$$\boldsymbol{x}_j = \begin{bmatrix} 1 & x_{j1} & \dots & x_{jc} & x_{j1}^2 & \dots & x_{jc}^2 & \dots & x_{j1}^{g(n)} & \dots & x_{jc}^{g(n)} \end{bmatrix},$$

$$\boldsymbol{y}_p = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

We calculate the log likelihood of a variable Y_i given a set of parents Pa_i as the sum over the log likelihoods from each partition $p \in \Pi_i$,

$$\ell_i(\boldsymbol{\theta} | \boldsymbol{X}, \boldsymbol{y}) = \sum_{p \in \Pi_i} \ell_p(\boldsymbol{\theta}_p | \boldsymbol{X}_p, \boldsymbol{y}_p), \tag{14}$$

where $\ell_p(\boldsymbol{\theta}_p | \boldsymbol{X}_p, \boldsymbol{y}_p)$ is defined depending on whether Y_i is discrete or continuous. Similarly, the degrees of freedom for Y_i are calculated as the sum over parameter counts in each partition $p \in \Pi_i$,

$$df_i(\boldsymbol{\theta}) = \sum_{p \in \Pi_i} df_p(\boldsymbol{\theta}_p). \tag{15}$$

The BIC computes the local score component using the log likelihood $\ell_i(\boldsymbol{\theta} | \boldsymbol{X}, \boldsymbol{y})$ and degrees of freedom $df_i(\boldsymbol{\theta})$. For the remainder of this section, we assume the subscript p is implicit on $\boldsymbol{x}_j$ and any on $\boldsymbol{\beta}$ terms where it is omitted.

4.2 Modeling a Continuous Child

In the case where Y_i is continuous, for each partition $p \in \Pi_i$ with design matrix $\boldsymbol{X}_p$ and target vector $\boldsymbol{y}_p$, we use the maximum likelihood estimate to determine the log likelihood $\ell_p(\boldsymbol{\theta}_p | \boldsymbol{X}_p, \boldsymbol{y}_p)$ and degrees of freedom $df_p(\boldsymbol{\theta}_p)$. By assumption 2, $y_j = f(\boldsymbol{x}_j) + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$ is additive Gaussian noise with variance σ^2 and f is an arbitrary continuous function. Given our assumptions, there exists a polynomial function $\hat{f}$ which approximates f such that $y_j \approx \hat{f}(\boldsymbol{x}_j) + \epsilon$. Estimating the parameters of $\hat{f}$ using least squares regression, we have $\hat{y}_j \sim N(\boldsymbol{X}_p \boldsymbol{\beta}_p, \sigma_p^2)$. Therefore the log likelihood for each partition $p \in \Pi_i$ becomes,

$$\begin{aligned} \ell_p(\boldsymbol{\beta}_p, \sigma_p^2 | \boldsymbol{X}_p, \boldsymbol{y}_p) = &- \frac{n_p}{2} \log 2\pi - \frac{n_p}{2} \log \sigma_p^2 \\ &- \frac{(\boldsymbol{y}_p - \boldsymbol{X}_p \boldsymbol{\beta}_p)^T (\boldsymbol{y}_p - \boldsymbol{X}_p \boldsymbol{\beta}_p)}{2\sigma_p^2}, \end{aligned} \tag{16}$$

where the maximum likelihood estimates are computed as,

$$\hat{\sigma_p}^2 = \frac{(\boldsymbol{y}_p - \boldsymbol{X}_p \hat{\boldsymbol{\beta}}_p)^T (\boldsymbol{y}_p - \boldsymbol{X}_p \hat{\boldsymbol{\beta}}_p)}{n_p}, \tag{17}$$

$$\hat{\boldsymbol{\beta}}_p = (\boldsymbol{X}_p^T \boldsymbol{X}_p)^{-1} \boldsymbol{X}_p^T \boldsymbol{y}_p. \tag{18}$$

Using the estimates in (17) and (18), the log likelihood in (16) simplifies to,

$$\ell_p(\hat{\boldsymbol{\beta}}_p, \hat{\sigma_p}^2 | \boldsymbol{X}_p, \boldsymbol{y}_p) = -\frac{n_p}{2} (\log 2\pi + \log \hat{\sigma_p}^2 + 1), \tag{19}$$

which we use to calculate $\ell_p(\boldsymbol{\theta}_p | \boldsymbol{X}_p, \boldsymbol{y}_p)$ in (14). To find the number of parameters in each partition $p \in \Pi_i$ for the calculation of $df_p(\boldsymbol{\theta}_p)$ in (15), we count the number of terms in $\hat{\boldsymbol{\beta}}_p$,

$$df_p(\boldsymbol{\theta}_p) = c \cdot g(n) + 1. \tag{20}$$

The BIC uses (14) and (15) to compute the parent-child relationship for Y_i given Pa_i.

4.3 Modeling a Discrete Child

In the case where Y_i is discrete, for each partition $p \in \Pi_i$ with design matrix $\boldsymbol{X}_p$ and target vector $\boldsymbol{y}_p$, we calculate the log likelihood $\ell_p(\boldsymbol{\theta}_p | \boldsymbol{X}_p, \boldsymbol{y}_p)$ and degrees of freedom $df_p(\boldsymbol{\theta}_p | \boldsymbol{X}_p, \boldsymbol{y}_p)$ using least squares regression. Suppose Y_i consists of k categories. Let f_h calculate the probability for the h^{th} category in Y_i given its continuous parents Pc_i where $h \in \{1, \dots, k\}$. By Assumption 2, there exists a polynomial function $\hat{f}_h$ which approximates f_h arbitrarily well. With this in mind, we aim to approximate each f_h using the results of least squares regression $\boldsymbol{X}_p \boldsymbol{\beta}_h$ where $\hat{\boldsymbol{\beta}}_h$ is a vector of coefficients. Our end goal is to used the approximations of each f_h as a conditional probability mass functions in the log likelihood calculation.

Define the categories of $\boldsymbol{y}_p$ such that each $y_j \in \{1, \dots, k\}$. We expand $\boldsymbol{y}_p$ into k binary vectors where the h^{th} vector represents the h^{th} category and the j^{th}

element of vector $\boldsymbol{h}$ asserts whether or not the j^{th} observation is from category h. To represent these binary vectors in matrix notation, we define $1_{\{condition\}}$ as an indicator variable which is 1 if *condition* is true and 0 otherwise. The h^{th} binary vector is then defined as,

$$\mathbf{1}_{\{\boldsymbol{y}=h\}} = \begin{bmatrix} 1_{\{y_1=h\}} \\ 1_{\{y_2=h\}} \\ \vdots \\ 1_{\{y_n=h\}} \end{bmatrix}.$$

We further define $\mathbf{1}$ to represent an $n_p \times 1$ vector of ones. Using the binary vectors as our targets, we calculate the maximum likelihood estimate of least squares regression which gives us,

$$\hat{\boldsymbol{\beta}}_h = (\boldsymbol{X}_p^T \boldsymbol{X}_p)^{-1} \boldsymbol{X}_p^T \, \mathbf{1}_{\{\boldsymbol{y}=h\}}.$$

If we want to interpret the results of least squares regression $\boldsymbol{X}_p \hat{\boldsymbol{\beta}}_h$ as probability values, we first must ensure that,

1. $\sum_{h=1}^{k} \boldsymbol{X}_p \hat{\boldsymbol{\beta}}_h = \mathbf{1}$
2. $\boldsymbol{x}_j \hat{\boldsymbol{\beta}}_h \geq 0, \ \forall \, j \in \{1, \ldots, n_p\}, h \in \{1, \ldots, k\}$.

We prove condition 1 is necessarily true and that condition 2 holds in the sample limit. See the supplementary materials for proofs.

Unfortunately, there is no such guarantee the values given by least squares regression will be strictly non-negative outside of the sample limit. Instead we define a procedure to map the sets of values calculated for each j to the domain of valid probability distributions. This is done by requiring the minimum tentative probability for each j be at least $\frac{1}{n_p}$ while maintaining the condition that the probabilities sum to one. We choose to avoid setting values directly to zero in order to prevent assigning zero probability to any observed instances. Therefore, we settle for a term which tends towards zero.

Our procedure as follows:

1. Shift the tentative probabilities such that they are center about a non-informative center, $\frac{1}{k}$, by subtracting $\frac{1}{k}$.
2. Shrink the tentative probabilities such that the smallest value will equal $\frac{1}{n_p}$ after shifting back by $\frac{1}{k}$.
3. Shift the scaled values back by $\frac{1}{k}$ to the original center.

Since we only want to perform this procedure if we have an invalid probability distribution, we define $m_j = \min\{\frac{1}{n_p}, \boldsymbol{x}_j \hat{\boldsymbol{\beta}}_h\}$ so that m_j is either the minimum tentative probability for j or $\frac{1}{n_p}$. We calculate the scaling factor in step 2 in our procedure, α_j as,

$$\alpha_j (m_j - \frac{1}{k}) + \frac{1}{k} = \frac{1}{n_p},$$

where $(m_j - \frac{1}{k})$ is the shifted values. Solving for α_j we find,

$$\alpha_j = \frac{\frac{1}{n_p} - \frac{1}{k}}{m_j - \frac{1}{k}}.$$

Note, that if $m_j = \frac{1}{n_p}$, then $\alpha_j = 1$ and we do not perform a shift. We compute the log likelihood in the discrete case as

$$\ell_p(\hat{\boldsymbol{\theta}}_p | \boldsymbol{X}_p, \boldsymbol{y}_p) = \sum_{j=1}^{n_p} \log \left(\alpha_j \boldsymbol{x}_j \hat{\boldsymbol{\beta}}_{y_j} + \frac{1}{k}(1 - \alpha_j) \right). \quad (21)$$

We use (21) to calculate $\ell(\boldsymbol{\theta}_p | \boldsymbol{X}_p, \boldsymbol{y}_p)$ in (14). To find the number of parameters in each partition $p \in \Pi_i$, we count the number of terms across all $\hat{\boldsymbol{\beta}}_j$. Each $\hat{\boldsymbol{\beta}}_j$ has $c \cdot g(n_p) + 1$ parameters and j ranges over k values. However, since Proposition 1 shows the estimated probabilities sum to one, the number of free parameters is

$$df_p(\hat{\boldsymbol{\theta}_p}) = (k-1)(c \cdot g(n_p) + 1). \quad (22)$$

As before, BIC uses (14) and (15) to compute the parent-child relationship for Y_i given Pa_i.

5 Implementation Details and Adaptations

In this section we consider various adaptations of the two proposed scores. In section 5.1, we discuss a binomial structure prior which allows for efficient learning of large networks. In section 5.2, we discuss a simplification for scoring discrete children which performs well empirically. In section 5.3, we discuss how to adapt our scores into conditional independence test for constraint-based methods.

5.1 Binomial Structure Prior

We introduce a structure prior inspired by the binomial distribution. The idea is to give a prior over the number of parents a node has, so that if the algorithm is producing graphs that are too sparse, one may, by adjusting the binomial probability, encourage nodes to have more parents. Let n be the number of variables in our dataset. We view the addition of each edge as an independent event which occurs with probability $\frac{r}{n-1}$ where r is the expected number of parents for any given variable, and $n-1$ is the total number of possible parents. Then we have,

$$\pi(k) = \Big(\frac{r}{n-1}\Big)^k \Big(1 - \frac{r}{n-1}\Big)^{n-k-1},$$

where $\pi(k)$ is the prior probability that any given variable Y_i in DAG $\mathcal{G}$ has k parents. Often it is more convenient to work in log space. Thus we calculate the log prior probability as,

$$\log \pi(k) = k \log\left(\frac{r}{n-1}\right) + (n-k-1)\log\left(1 - \frac{r}{n-1}\right).$$

Usually, BIC assumes the prior probability of models in equation (1) is distributed uniformly. Using the binomial structure prior instead, we are able to adapt BIC such that it performs better with large networks. We report these findings in section 6.2.

5.2 Multinomial Scoring with Continuous Parents

Both scores presented in this paper reduce to multinomial scoring in the case of a discrete child with exclusively discrete parents. In this section, we discuss extending the use of multinomial scoring for discrete children and discrete parents to include continuous parents. Before starting a search, we create discretized versions of each continuous variable using equal frequency binning with a predefined number of bins b. Whenever scoring a discrete child, we replace any continuous parents with the precomputed discretized versions of those variables. This allows us to quickly and efficiently perform multinomial scoring for all discrete children. We report our finding when choosing $b = 3$ as a modification to CG in section 6.4.

5.3 As a Conditional Independence Test

To adapt either score into a conditional independence test, we calculate the log likelihood and degrees of freedom as usual, but perform a likelihood ratio test instead of scoring with BIC. Suppose we wish to test $Y_0 \perp\!\!\!\perp Y_1 | Z$ where Y_0 and Y_1 are variables and Z is a conditioning set in a dataset $\mathcal{D}$. Define ℓ_0 and df_0 respectively as the log likelihood and degrees of freedom for Y_0 given Pa_0 where $Pa_0 = \{Y_1\} \cup Z$. Further, define ℓ_0' and df_0' respectively as the log likelihood and degrees of freedom for Y_0 given Pa_0' where $Pa_0' = Z$. Perform a likelihood ratio test with test statistic $2(\ell_0 - \ell_0')$ and $df_0 - df_0'$ degrees of freedom. This tests whether the model encoding $Y_0 \perp\!\!\!\perp Y_1 | Z$ or the model encoding $Y_0 \not\perp\!\!\!\perp Y_1 | Z$ fits the data in $\mathcal{D}$ better. If the scoring method used is not score equivalent, then we must also perform a likelihood ratio test with test statistic $2(\ell_1 - \ell_1')$ and $df_1 - df_1'$ degrees of freedom where Y_0 and Y_1 are swapped. In this case we decide the variables are dependent if there is enough evidence in either test to support that hypothesis.

6 Simulation Studies

To simulate mixed data, we first randomly generate a DAG $\mathcal{G}$ and designate each variable in $\mathcal{G}$ as either discrete or continuous. $\mathcal{G}$ is generated by randomly defining a causal order and adding edges between the variables. Edges are added between randomly chosen pairs of nodes such that the connections are true to the prespecified ordering; they are continually added until the average degree of the graph reaches a user specified amount. Variables in the network without parents are generated according to Gaussian and Multinomial distributions. We create temporary discretized versions of each continuous variable using equal frequency binning with 2 to 5 bins uniformly chosen. In causal order, we simulate the remaining variables as follows. Continuous variables are generated by partitioning on the discrete parents and randomly parameterizing the coefficients of a linear regression for each partition. Discrete variables are generated via randomly parameterized Multinomial distributions of the variable being simulated, the discrete parents, and the discretized versions of the continuous parents. All temporary variables are removed after the simulation is complete. For all simulations, each variable is assigned either continuous or discrete with equal probability. Additionally, discrete variables will have a uniformly chosen number of categories between and including 2 and 5.

In order to prevent the number of multinomial cells for discrete variables from getting too large, we bound the maximum degree of any node in the generated graph to 5. In our experiment, we tested on graphs of average degree 2 and 4. Figure 1 and 2 show the distribution of the node degrees for different settings of average degree.

We compare CG with and without the discretization heuristic and MVP with $g(n_p) = 1$, $g(n_p) = \lfloor \log(n_p) \rfloor$ using the following performance measures.

AP	- adjacency precision: correctly predicted adjacent / all predicted adjacent
AR	- adjacency recall: correctly predicted adjacent / all true adjacent
AHP	- arrowhead precision: correctly predicted arrowheads / all predicted arrowheads
AHR	- arrowhead recall: correctly predicted arrowheads / all true arrowheads (in found adjacencies)
T	- elapsed time (seconds)

All results are averaged over 10 randomly simulated graphs and were run on a laptop with an Intel(R) Core I7 @ 3.1 GHz with 16GB of memory. The results in Tables 1 - 5, used the same simulated dataset and can be directly compared to each other. The results in Tables 6 and 7 each required a different set of simulation

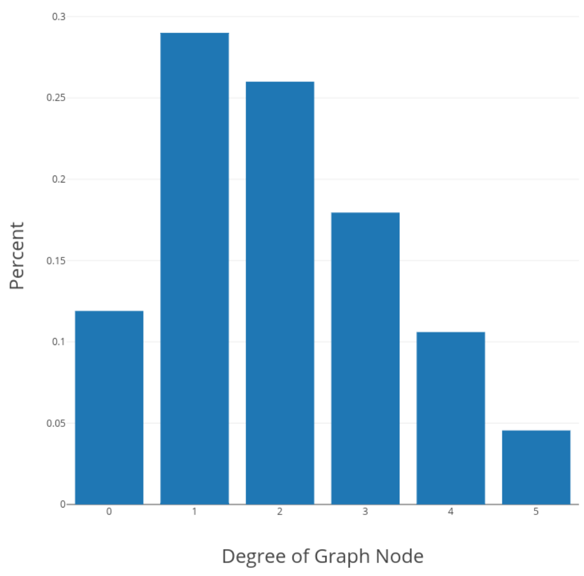

Fig. 1 Distribution of node degrees in graphs of average degree 2.

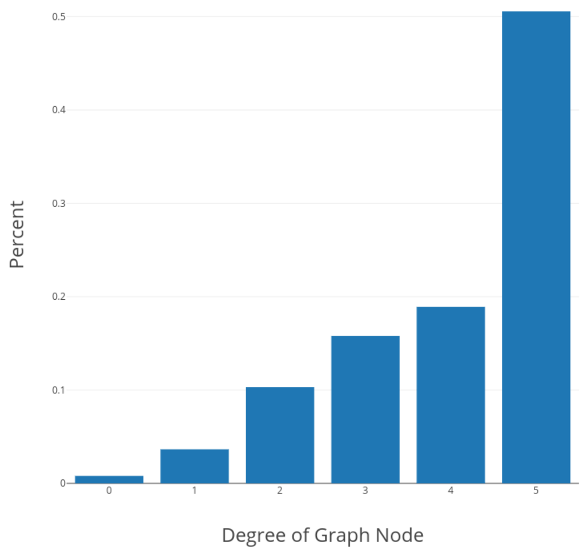

Fig. 2 Distribution of node degrees in graphs of average degree 4.

parameters and thus use different simulated datasets. Prior to running tests on any algorithm, the data were standardized.

6.1 The Conditional Gaussian Approximation

We empirically validated the choice of approximating a mixture of Gaussians with a single Gaussian in CG in Table 1. We denote the use of a single Gaussian as Approx and the use of the correct mixture calculation as Exact. Originally the results did not appear comparable as the approximate method output a much denser graph than the exact method. In the results shown, we tuned the density of the graphs using the binomial structure prior proposed in section 5.1. We can see that the approximation is as good (or better in some cases) than the exact method. In the comparisons, we simulate graphs of average degree 2 and 4 with 200 and 1,000 samples and 100 measured variables. Results are given with the binomial structure prior adjustment set to 1.

6.2 Binomial Structure Prior

We tested the usefulness of the binomial structure prior by simulating 200 and 1,000 samples from graphs of average degree 2 and 4 with 100 measured variables. The tests were performed using fGES [11], an optimized version of GES [4]. We compare our scoring functions with and without the binomial structure prior. Additionally we compare against extended BIC (EBIC) [3], an similar modification to BIC which aims to address the small-n-large-P situation. In these experiment the expected number of parents for the binomial structure prior is set to 1 and EBIC's gamma parameter is set to 0.5 upon suggestion of the authors. In Tables 2 and 3, we report our findings when the average degrees of the graphs are 2 and 4 respectively.

While we set the expected number of parents to 1 for the experiments presented in this paper, it is important to note that this parameter can be chosen to be any value greater than 0. By varying the expected number parents, we can define how sparse or dense we wish our output graph to be. The choice of a low value results in a relatively sparse graph and a high value in a denser one.

From Table 2 and 3, for both the binomial structure prior and EBIC, we see boosts in precision with a reduction in recall. Additionally, we see vast improvements to computation times. In general, EBIC seems to work better with small sample sizes. This makes sense, since EBIC is aimed at the small-n-large-P situation. However, for 1,000 samples, we find the binomial structure prior favorable over EBIC. In lieu of these results, we choose to use the binomial structure prior for the remainder of our score based experiments.

Table 1 Compares the approximate method to the exact method for CG on graphs of average degree 2 and 4 with 100 measured variables. Sample size is varied to be 200 or 1,000 and the binomial structure prior is used with the expected number of parents set to 1. The best results in each cell have been bolded and all reported results are averaged over 10 repetitions.

Sample Size		200					1,000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	Exact	0.56	**0.56**	0.37	**0.31**	1.03	0.79	0.79	0.65	**0.55**	2.94
	Approx	**0.82**	0.53	**0.75**	0.19	**0.31**	**0.91**	**0.81**	**0.85**	0.49	**0.59**
Avg Deg 4	Exact	0.59	0.39	0.44	0.26	0.73	0.84	**0.64**	0.73	**0.51**	3.73
	Approx	**0.82**	0.36	**0.69**	0.23	**0.17**	**0.92**	0.62	**0.84**	**0.51**	**0.99**

Table 2 Compares the use of different priors for CG, MVP 1, and MVP $\log n$ on graphs of average degree 2 with 100 measured variables. Sample size is varied to be 200 or 1,000. The best results in each cell have been bolded and all reported results are averaged over 10 repetitions.

Sample Size		200					1,000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
CG	Uniform	0.54	**0.59**	0.40	**0.38**	0.29	0.76	**0.83**	0.66	**0.59**	0.63
	EBIC	**0.85**	0.45	**0.80**	0.12	0.26	**0.93**	0.78	**0.88**	0.43	**0.53**
	Binomial	0.82	0.53	0.75	0.19	**0.18**	0.91	0.81	0.85	0.49	0.55
MVP 1	Uniform	0.36	**0.57**	0.24	**0.35**	9.32	0.70	**0.81**	0.56	**0.57**	6.43
	EBIC	**0.84**	0.39	**0.69**	0.17	**1.35**	**0.85**	0.71	**0.74**	0.46	**3.53**
	Binomial	0.53	0.53	0.35	0.31	1.53	0.83	0.77	0.70	0.52	3.93
MVP $\log n$	Uniform	0.37	**0.55**	0.23	**0.31**	7.51	0.77	**0.79**	0.60	**0.50**	14.47
	EBIC	**0.84**	0.31	**0.65**	0.09	**2.50**	**0.87**	0.65	**0.73**	0.37	**7.25**
	Binomial	0.52	0.51	0.33	0.28	2.51	0.84	0.76	0.68	0.47	8.54

Table 3 Compares the use of different priors for CG, MVP 1, and MVP $\log n$ on graphs of average degree 4 with 100 measured variables. Sample size is varied to be 200 or 1,000. The best results in each cell have been bolded and all reported results are averaged over 10 repetitions.

Sample Size		200					1,000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
CG	Uniform	0.66	**0.41**	0.52	**0.31**	0.23	0.87	**0.66**	0.79	**0.57**	1.09
	EBIC	**0.85**	0.30	**0.70**	0.16	**0.15**	**0.94**	0.57	**0.86**	0.43	0.92
	Binomial	0.82	0.36	0.69	0.23	0.16	0.92	0.62	0.84	0.51	**0.85**
MVP 1	Uniform	0.45	**0.42**	0.34	**0.30**	16.85	0.85	**0.66**	0.77	**0.56**	7.24
	EBIC	**0.84**	0.27	**0.69**	0.16	**0.98**	**0.92**	0.54	**0.84**	0.43	**4.29**
	Binomial	0.53	0.36	0.39	0.25	6.24	0.90	0.61	0.83	0.51	4.90
MVP $\log n$	Uniform	0.44	**0.37**	0.30	**0.23**	20.70	0.89	**0.62**	0.78	**0.49**	16.74
	EBIC	**0.85**	0.18	**0.64**	0.08	**2.06**	**0.94**	0.46	**0.84**	0.34	**9.25**
	Binomial	0.52	0.33	0.36	0.20	6.50	0.93	0.58	**0.84**	0.46	11.55

6.3 As a Conditional Independence Test

We tested the usefulness of the CG and MVP scores as conditional independence tests by simulating, 200 and 1,000 samples from graphs of average degree 2 and 4 with 100 measured variables. As a search algorithm, we used Conservative PC [12], which is a modified version of PC, with α set 0.001, and treat ambiguous triples as non-colliders [19]. Here we also use the discretization heuristic with $b = 3$ for CG, denoted CGd, however we do not use a structure prior since we are no longer scoring a full Bayesian network in this paradigm. We did not include results for a version of MVP which uses the discretization heuristic because it had little effect. The results are shown in Table 4.

In general, we find that our methods perform better as scores, but still perform reasonably well as conditional independence tests. This is promising for use in algorithms, such as FCI, that model the possibility of latent confounding [19].

6.4 Tests Against Baseline Scores

With little in the literature to compare against which scales to hundreds of variables, we define two simple baseline scores. The first of the two, which we denote MN, uses multinomial scoring for all cases. In order to do so, we essentially extend the discretization heuristic to the continuous child case so that we are always scoring with a multinomial. The second of the two, which we denote as LR, uses partitioned linear regression in the continuous child case and partitioned logistic regression in the discrete child case. Note that since logistic regression requires some sort of fixed point optimization, it will be slower than methods which have a closed form.

Table 4 Compares the use of CG, CGd, MVP 1, and MVP $\log n$ in the constraint-based paradigm with alpha set to 0.001 on graphs of average degree 2 and 4 respectively with 100 measured variables. Sample size is varied to be 200 or 1,000. The best results in each cell have been bolded and all reported results are averaged over 10 repetitions.

Sample Size		200					1,000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	CG	0.91	0.40	0.96	0.04	0.57	0.93	0.68	0.93	0.24	1.26
	CGd	0.92	**0.42**	**0.98**	**0.05**	**0.46**	**0.95**	**0.69**	**0.94**	**0.26**	**1.15**
	MVP 1	0.77	0.27	0.67	0.01	1.37	0.88	0.60	0.67	0.15	6.63
	MVP $\log n$	**0.97**	0.29	0.75	0.01	2.03	0.92	0.67	0.68	0.23	16.82
Avg Deg 4	CG	0.93	**0.26**	**0.93**	**0.05**	**0.44**	0.94	**0.50**	0.93	**0.24**	**4.47**
	CGd	0.93	**0.26**	0.89	**0.05**	**0.44**	**0.95**	**0.50**	**0.95**	**0.24**	8.55
	MVP 1	0.81	0.14	0.43	0.01	1.28	0.92	0.41	0.65	0.16	11.21
	MVP $\log n$	**0.98**	0.15	0.56	0.01	2.01	0.93	0.48	0.60	0.21	48.73

For this reason, we adapt a widely used toolkit, Lib Linear [6] to perform logistic regression in hopes that it will give readers more confidence in our timing results. As with MVP, the appended term on LR denotes the maximum degree of the polynomial basis used as the regressors.

We compared CG, CGd, MVP 1, LR 1, MVP $\log n$, LR $\log n$, and MN scores by simulating, 200 and 1,000 samples from graphs of average degree 2 and 4 with 100, measured variables. As a search algorithm, we used fGES. Here we also use the discretization heuristic with $b = 3$ for CGd and the binomial structure prior with the expected number of parents equal to 1 for all scores. Additionally, boldface text highlights the best performing score for each statistic in each cell of the table. The results are shown in Table 5 and 6. Furthermore, for the results in Table 6 we extend our method of simulating data. Since MVP is designed to handle non-linearity while CG is not, we modify the continuous child phase of data generation to allow for non-linearities. To do so, we additionally generate second, and third order polynomial terms. However, because of the nature of these non-linear functions, the values of the data often become unmanageably large. To correct for this, we resample a variable with square-root and cube-root relationships if the values are too large.

In Tables 5 and 6, we see that both CG and MVP easily scale to networks consisting of hundreds of variables while LR struggles to handle large networks. Additionally, there is almost not difference in performance between MVP and LR. This makes sense, since MVP is using an approximation of logistic regression in the discrete child case and performing all other cases identically. Complete discretization with MN is perhaps the most scalable solution, however due to the information loss from the discretization process, it takes a rather large hit in recall. We find the our scores are able to retain reasonable recall with high precision and low computation time. Additionally, we note that MVP performs similarly to CG with a slight advantage in the non-linear case.

In our final experiment, we aimed to access the scalability of our proposed methods. We simulate data according to the linear simulation described at the beginning of this section with 500 measured variables, 200 and 1,000 samples, and average degree 2 and 4. Note that for average degree 4, we have omitted results for MVP. This is because, while performing these experiment, we ran into issue with memory management. Further note that LR is not included at all. This is because LR (as implemented) cannot feasibly scale to such a large network due to time complexity.

In Table 7, we see that our methods are capable of scaling to, albeit sparse, graphs of 500 measured variables. Additionally, CG was able to scale to a slightly denser graph of 500 variables. In general, we see the same performance on these larger networks as before on the networks of 100 measured variables.

7 Conclusions

In this paper we outline two novel scoring methods for learning Bayesian networks in the presence of mixed data types. We provide solutions that are linear in the number of data instances for scoring and can be scaled to networks of 500 variables or more using a laptop. Further, we introduce a structure prior for learning large networks and a simplification for scoring discrete variables. The Conditional Gaussian (CG) score performs well, even on complex non-linear data, and was quite fast, returning a DAG on networks of 500 variables in less than 10 seconds (see Table 7). The Mixed Variable Polynomial (MVP) score generally had lower precision statistics than CG, but higher recall. Not surprisingly, since MVP can model non-linear relationships, it performed somewhat better overall on such data. However, MVP is considerably slower than CG. MVP performed similarly to the logistic regression based approach (LR), but MVP was 10 to 20 times faster. The fully discrete-

Table 5 Compares the use of CG, CGd, MVP 1, LR 1, MVP $\log n$, LR $\log n$, and MN using linear data from graphs of average degree 2 and 4 respectively with 100 measured variables. Sample size is varied to be 200 or 1,000 and the binomial structure prior is used with the expected number of parents set to 1. The best results in each cell have been bolded and all reported results are averaged over 10 repetitions.

Sample Size		200					1,000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	CG	0.82	**0.53**	0.75	0.19	0.29	0.91	**0.81**	0.85	0.49	0.56
	CGd	0.90	0.45	0.82	0.17	0.19	0.95	0.77	**0.93**	0.47	0.53
	MVP 1	0.53	**0.53**	0.35	0.31	1.92	0.83	0.77	0.70	**0.52**	4.01
	LR 1	0.53	**0.53**	0.36	**0.32**	18.55	0.84	0.78	0.71	0.51	52.34
	MVP $\log n$	0.52	0.51	0.33	0.28	2.63	0.84	0.76	0.68	0.47	8.55
	LR $\log n$	0.53	0.51	0.34	0.28	34.86	0.87	0.78	0.71	0.49	165.53
	MN	**0.93**	0.41	**0.85**	0.07	**0.09**	**0.97**	0.72	0.90	0.36	**0.48**
Avg Deg 4	CG	0.82	**0.36**	0.69	0.23	0.16	0.92	0.62	0.84	0.51	0.90
	CGd	0.92	0.32	0.80	0.18	0.15	0.96	0.58	**0.91**	0.48	0.73
	MVP 1	0.53	**0.36**	0.39	**0.25**	8.82	0.90	0.61	0.83	0.51	5.20
	LR 1	0.53	**0.36**	0.39	**0.25**	27.22	0.91	**0.63**	0.83	**0.52**	62.08
	MVP $\log n$	0.53	0.33	0.36	0.20	6.25	0.93	0.58	0.84	0.46	12.00
	LR $\log n$	0.53	0.33	0.36	0.20	45.97	0.93	0.59	0.84	0.47	215.93
	MN	**0.93**	0.26	**0.86**	0.07	**0.06**	**0.98**	0.51	0.84	0.36	**0.19**

Table 6 Compares the use of CG, CGd, MVP 1, LR 1, MVP $\log n$, LR $\log n$, and MN using non-linear data from graphs of average degree 2 and 4 respectively with 100 measured variables. Sample size is varied to be 200 or 1,000 and the binomial structure prior is used with the expected number of parents set to 1. The best results in each cell have been bolded and all reported results are averaged over 10 repetitions.

Sample Size		200					1,000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	CG	0.53	0.54	0.35	0.33	0.47	0.58	0.67	0.45	**0.45**	1.91
	CGd	0.82	0.52	0.61	0.23	0.23	0.83	0.68	0.67	0.39	1.03
	MVP 1	0.67	**0.55**	0.45	**0.32**	0.82	0.76	**0.69**	0.58	0.41	3.37
	LR 1	0.67	**0.55**	0.45	0.31	10.93	0.76	0.68	0.58	0.41	46.03
	MVP $\log n$	0.75	0.54	0.51	0.29	1.42	0.87	0.67	0.71	0.42	6.90
	LR $\log n$	0.75	0.54	0.51	0.29	22.07	0.87	0.67	0.71	0.42	158.09
	MN	**0.95**	0.49	**0.96**	0.05	**0.11**	**0.96**	0.65	**0.83**	0.31	**0.24**
Avg Deg 4	CG	0.48	0.34	0.34	**0.24**	0.57	0.70	0.51	0.60	0.41	1.38
	CGd	0.81	0.32	0.64	0.19	0.33	0.86	0.51	0.77	0.39	1.00
	MVP 1	0.71	**0.35**	0.53	0.23	1.03	0.83	0.52	0.73	0.41	3.82
	LR 1	0.72	**0.35**	0.54	0.23	12.50	0.82	0.52	0.72	0.40	48.62
	MVP $\log n$	0.81	0.33	0.63	0.23	1.88	0.93	0.52	**0.84**	0.41	8.29
	LR $\log n$	0.81	0.34	0.63	0.23	35.04	0.93	**0.53**	**0.84**	**0.42**	191.39
	MN	**0.95**	0.27	**0.85**	0.05	**0.06**	**0.95**	0.44	0.75	0.29	**0.20**

Table 7 Compares the use of CG, CGd, MVP 1, MVP $\log n$, and MN using linear data from graphs of average degree 2 and 4 respectively with 500 measured variables. Sample size is varied to be 200 or 1,000 and the binomial structure prior is used with the expected number of parents set to 1. Scores omitted with average graph degree 4 failed to return a result. The best results in each cell have been bolded and all reported results are averaged over 10 repetitions.

Sample Size		200					1,000				
Statistic		AP	AR	AHP	AHR	T (s)	AP	AR	AHP	AHR	T (s)
Avg Deg 2	CG	0.67	**0.51**	0.48	**0.28**	2.11	0.88	**0.77**	0.81	0.50	**7.28**
	CGd	0.86	0.44	0.75	0.19	**1.74**	0.94	0.71	**0.89**	0.46	8.42
	MVP 1	0.40	0.49	0.24	0.27	56.83	0.81	0.73	0.67	**0.51**	44.68
	MVP $\log n$	0.40	0.46	0.23	0.24	81.20	0.84	0.72	0.68	0.46	96.91
	MN	**0.93**	0.39	**0.84**	0.07	1.79	**0.97**	0.67	**0.89**	0.36	14.74
Avg Deg 4	CG	0.75	**0.35**	0.59	**0.25**	2.21	0.91	**0.61**	0.84	**0.51**	**9.70**
	CGd	0.88	0.31	**0.78**	0.19	2.22	0.95	0.58	**0.90**	0.48	16.59
	MN	**0.93**	0.26	0.77	0.07	**1.32**	**0.98**	0.51	0.86	0.37	12.43

variable approach (MN) performed surprisingly well in terms of precision and speed, although its recall was often lower that that of CG and MVP, and sometimes greatly lower. We also showed how the CG and MVP scoring methods can be readily adapted to provide conditional independence tests for constraint-based methods to support future use in algorithms such as FCI.

All experimental comparisons and simulation were completed within the Tetrad project [14] and the code is available in Tetrad's repository on GitHub[1].

There are several directions for future work. First, we would like to apply the methods to real datasets for which knowledge of the causal relationships is available. Second, we would like to expand the CG and MVP algorithms to model ordinal discrete variables. Third, we would like to investigate alternative basis functions for the MVP algorithm.

Acknowledgements We would like to thank Clark Glymour, Peter Spirtes, Takis Benos, Manatakis, Dimitrios, and Vineet Raghu for stimulating conversions on the topic of learning mixed variable Bayesian networks. Additionally, we would like to thank our reviewers for their helpful comments.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
2. Bøttcher, S.G.: Learning bayesian networks with mixed variables. Ph.D. thesis, Citeseer (2004)
3. Chen, J., Chen, Z.: Extended bic for small-n-large-p sparse glm. Statistica Sinica pp. 555–574 (2012)
4. Chickering, D.M.: Optimal structure identification with greedy search. Journal of Machine Learning Research **3**(Nov), 507–554 (2002)
5. Daly, R., Shen, Q., Aitken, S.: Review: learning bayesian networks: Approaches and issues. The Knowledge Engineering Review **26**(2), 99–157 (2011)
6. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research **9**, 1871–1874 (2008)
7. Heckerman, D., Geiger, D.: Learning bayesian networks: a unification for discrete and gaussian domains. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 274–284. Morgan Kaufmann Publishers Inc. (1995)
8. McGeachie, M.J., Chang, H.H., Weiss, S.T.: Cgbayesnets: conditional gaussian bayesian network learning and inference with mixed discrete and continuous data. PLoS Comput Biol **10**(6), e1003,676 (2014)
9. Monti, S., Cooper, G.F.: A multivariate discretization method for learning bayesian networks from mixed data. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 404–413. Morgan Kaufmann Publishers Inc. (1998)
10. Raftery, A.E.: Bayesian model selection in social research. Sociological Methodology pp. 111–163 (1995)
11. Ramsey, J., Glymour, M., Sanchez-Romero, R., Glymour, C.: A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. International Journal of Data Science and Analytics pp. 1–9 (2016)
12. Ramsey, J., Zhang, J., Spirtes, P.L.: Adjacency-faithfulness and conservative causal inference. arXiv preprint arXiv:1206.6843 (2012)
13. Romero, V., Rumí, R., Salmerón, A.: Learning hybrid bayesian networks using mixtures of truncated exponentials. International Journal of Approximate Reasoning **42**(1-2), 54–68 (2006)
14. Scheines, R., Spirtes, P., Glymour, C., Meek, C., Richardson, T.: The tetrad project: Constraint based aids to causal model specification. Multivariate Behavioral Research **33**(1), 65–117 (1998)
15. Schwarz, G., et al.: Estimating the dimension of a model. The Annals of Statistics **6**(2), 461–464 (1978)
16. Sedgewick, A.J., Ramsey, J.D., Spirtes, P., Glymour, C., Benos, P.V.: Mixed graphical models for causal analysis of multi-modal variables. arXiv preprint arXiv:1704.02621 (2017)
17. Shimizu, S., Hoyer, P.O., Hyvärinen, A., Kerminen, A.: A linear non-gaussian acyclic model for causal discovery. Journal of Machine Learning Research **7**(Oct), 2003–2030 (2006)
18. Sokolova, E., Groot, P., Claassen, T., Heskes, T.: Causal discovery from databases with discrete and continuous variables. In: European Workshop on Probabilistic Graphical Models, pp. 442–457. Springer (2014)
19. Spirtes, P., Glymour, C.N., Scheines, R.: Causation, Prediction, and Search. MIT Press (2000)

[1] https://github.com/cmu-phil/tetrad