Journal of Machine Learning Research 27 (2026) 1-41 Submitted 11/24; Published 1/26

Learning Bayesian Network Classifiers to Minimize Class Variable Parameters

Shouta Sugahara SUGAHARA@AI.LAB.UEC.AC.JP
The University of Electro-Communications
Tokyo, Japan

Koya Kato KATO@AI.LAB.UEC.AC.JP
The University of Electro-Communications
Tokyo, Japan

James Cussens JAMES.CUSSENS@BRISTOL.AC.UK
University of Bristol
Bristol, UK

Maomi Ueno UENO@AI.LAB.UEC.AC.JP
The University of Electro-Communications
Tokyo, Japan

Editor: Qiang Liu

Abstract

This study proposes and evaluates a novel Bayesian network classifier which can asymptotically estimate the true probability distribution of the class variable with the fewest class variable parameters among all structures for which the class variable has no parent. Moreover, to search for an optimal structure of the proposed classifier, we propose (1) a depth-first search based method and (2) an integer programming based method. The proposed methods are guaranteed to obtain the true probability distribution asymptotically while minimizing the number of class variable parameters. Comparative experiments using benchmark datasets demonstrate the effectiveness of the proposed method.

Keywords: probabilistic graphical models; Bayesian networks; classification; structure learning; generative models

1 Introduction

Deep learning can learn a huge number of variables and can provide more accurate classification than the Bayesian network approach. Nevertheless, the decisions made by a deep learning classifier are not explainable because the mathematical properties of deep learning are unclear and because the decisions of deep learning classifiers are deterministic. By contrast, the Bayesian network approach can estimate a *class-posterior distribution*, which consists of the conditional probabilities of the class variable given feature variables. In particular, probabilistic graphical models have high explainability: as opposed to a black box, they have a 'white-box' characteristic. This study proposes a novel Bayesian network classifier that provides high prediction accuracy and high explainability.

A popular Bayesian classifier is the naive Bayes classifier, in which the feature variables are conditionally independent given a class variable (Minsky, 1961). It is reasonable to

expect a naive Bayes classifier not to provide highly accurate classification because actual datasets were generated from more complex systems. Therefore, the general Bayesian network classifier (GBN), which has no structural constraints, would be expected to perform better than the naive Bayes classifier because GBNs are more expressive than naive Bayes classifiers.

However, Friedman et al. (1997) demonstrated that a GBN learned by maximizing an approximation to the marginal likelihood can be outperformed by naive Bayes. They argued that this is because marginal likelihood attempts to model conditional independences among variables that are not directly related to classification and therefore offers no guarantee of optimizing the conditional independences that matter for classification. For this reason, they proposed the conditional log likelihood (CLL) as a discriminative score. Since learning a structure that maximizes CLL is more computationally demanding than maximizing marginal likelihood, many approximate learning methods have been proposed (Greiner and Zhou, 2002; Grossman and Domingos, 2004; Carvalho et al., 2013; Mihaljević et al., 2018).

Nevertheless, these works did not explain why CLL would outperform marginal likelihood. For large datasets, because marginal likelihood has an asymptotic consistency, the classification accuracy achieved by maximizing marginal likelihood is expected to be comparable to that achieved by maximizing CLL. Differences observed between the two scores in prior studies may depend on the learning algorithms used there, which were approximate rather than exact.

Sugahara et al. (2018); Sugahara and Ueno (2021) proposed exact learning of Bayesian network classifiers by maximizing marginal likelihood subject to the constraint that the class variable is a parent of every feature variable. They empirically demonstrated that their method achieves higher classification accuracy than existing approximate methods that maximize CLL. However, because their method requires the class variable to be adjacent to all feature variables, the set of *class variable parameters*—parameters used to compute the class-posterior distribution—becomes unnecessarily large. Increasing *the number of class variable parameters (NCP)* slows the convergence of the estimated distributions.

To address the problem, this study proposes *Bayesian network optimal classifiers (BNOC)* which can asymptotically estimate the true probability distributions with the fewest NCP. To learn BNOC structures, we propose (1) a depth-first search based method and (2) an integer programming based method.

First, for the depth-first search based method, we formulate the search for a BNOC structure as a shortest-path-finding problem. Popular methods to solve shortest path problems for learning Bayesian networks are breadth-first search (Malone et al., 2011) and depth-first search (Malone and Yuan, 2014). Although their computational times increase exponentially as the numbers of variables increase, they can be decreased using branch-and-bound methods (Malone et al., 2011; Malone and Yuan, 2014). Nevertheless, the traditional heuristic functions used in the branch-and-bound method cannot be applied to learning a BNOC structure. In this study, we prove that the NCP of a naive Bayes classifier is the lower bound NCP of our model, and propose a depth-first branch-and-bound algorithm using the lower bound NCP as a heuristic function.

Another efficient approach for learning Bayesian networks is an integer programming (IP) based method (Cussens, 2012; Bartlett and Cussens, 2013; Liao et al., 2019), which formulates Bayesian network structure learning as an integer program and solves it using

off-the-shelf IP solvers, which are highly optimized and use decades of research in IP optimization to speed up the search for the optimal Bayesian network structure. We propose an IP-based method to search for a BNOC structure. The depth-first branch-and-bound method proposed in Section 4 consists of two steps: maximizing BDeu for best-parents search and minimizing NCP for structure search. On the other hand, the IP-based method concurrently conducts the two steps with a single objective function. In addition, it reduces the space complexity to $O(n\sum_{j=0}^{d}\binom{n}{j})$ where n is the number of feature variables when we give an upper bound d for the number of parents.

The proposed methods provide the four benefits presented below.

1. They guarantee asymptotic estimation of a structure which can asymptotically estimate the true probability distributions with the fewest NCP.
2. They yield an explainable graphical structure of the feature variables to predict the class variable.
3. The structure learned by them asymptotically does not include feature variables irrelevant to the class variable.
4. They are anytime algorithms.

Comparative experimentation has demonstrated the following: (1) For large datasets, the proposed methods achieved a structure which can asymptotically estimate the true probability distributions with the fewest NCP. (2) The proposed depth-first search method provided higher classification accuracy than almost all the compared methods did for small datasets. (3) The proposed IP-based method outperformed all the compared methods for large datasets. (4) With parameters estimated by maximizing CLL, the proposed depth-first search method provided higher classification accuracy than deep learning and random forests.

2 Background

This section provides a background on Bayesian networks and Bayesian network classifiers.

2.1 Bayesian networks

Let $\mathbf{V} = \{X_0, X_1, \ldots, X_n\}$ be a set of discrete variables, where each X_i $(i = 0, \ldots, n)$ takes values in $\{1, \ldots, r_i\}$. We write $X_i = k$ when X_i takes state k. For any subset $\mathbf{U} \subseteq \mathbf{V}$, let $q(\mathbf{U})$ denote the number of configurations of $\mathbf{U}$. When $\mathbf{U}$ takes the j-th configuration, we write $\mathbf{U} = j$.

A Bayesian network (BN) on $\mathbf{V}$ is an annotated directed acyclic graph (DAG) that encodes a joint probability distribution on $\mathbf{V}$. Formally, a BN is a pair $B = (G, \Theta_G)$, where G is a DAG whose nodes correspond to $X_0, \ldots, X_n$ and whose edges indicate direct dependencies. Let $\Pi_{X_i}^G$ be the parent set of X_i in G. The component Θ_G is the set of conditional probability parameters

$$\Theta_G = \bigcup_{i=0}^{n} \bigcup_{j=1}^{q(\Pi_{X_i}^G)} \bigcup_{k=1}^{r_i} \{\theta_{X_i=k|\Pi_{X_i}^G=j}\},$$

where each parameter $\theta_{X_i=k|\Pi^G_{X_i}=j}$ corresponds to a conditional probability of $X_i = k$ given $\Pi^G_{X_i} = j$. Letting P_B denote the probability distribution induced by B, the joint distribution $P_B(X_0, X_1, \ldots, X_n)$ can be represented as

$$P_B(X_0, X_1, \ldots, X_n) \;=\; \prod_{i=0}^{n} P_B\big(X_i \mid \Pi^G_{X_i}\big) = \prod_{i=0}^{n} \theta_{X_i|\Pi^G_{X_i}}.$$

The BN structure G represents conditional independence assertions in the probability distribution by *d-separation*. First, it is necessary to define *path* and *collider* to define d-separation.

Definition 1 Path
Let G be a DAG on $\mathbf{V}$. A finite sequence of pairwise distinct variables

$$\rho = \langle V_0, V_1, \ldots, V_l \rangle \quad (l \geq 1,\ V_i \in \mathbf{V})$$

is a path *between V_0 and V_l in G if and only if each consecutive pair is adjacent in G (i.e., there is an edge between them in either direction).*

Definition 2 Collider
Let G be a DAG on $\mathbf{V}$ and let $\rho = \langle V_0, \ldots, V_l \rangle$ be a path between V_0 and V_l in G. For an internal variable V_i of ρ $(1 \leq i \leq l-1)$, we say that V_i is a collider on ρ *if and only if there exist two incoming edges to V_i from V_{i-1} and V_{i+1} in G, that is,*

$$V_{i-1} \;\rightarrow\; V_i \;\leftarrow\; V_{i+1}.$$

Otherwise we say that V_i is a non-collider *on ρ.*

d-separation is defined as follows.

Definition 3 d-separation
Let G be a DAG on $\mathbf{V}$. Let $X, Y \in \mathbf{V}$ be distinct variables and let $\mathbf{U} \subseteq \mathbf{V} \setminus \{X, Y\}$. We say that X and Y are d-separated given $\mathbf{U}$ *in G if and only if every path ρ between X and Y satisfies at least one of the following conditions:*

1. *There exists a non-collider Z on ρ such that $\mathbf{U}$ includes Z.*
2. *There exists a collider Z on ρ such that $\mathbf{U}$ includes neither Z nor any of its descendants in G.*

We write $X \perp^G_d Y \mid \mathbf{U}$ to denote that X and Y are d-separated given $\mathbf{U}$ in G. If they are not d-separated, we write $X \not\perp^G_d Y \mid \mathbf{U}$.

Let P^{True} denote the true probability distribution over $\mathbf{V}$. Let $X \perp\!\!\!\perp Y \mid \mathbf{U}$ denote that X and Y are conditionally independent given $\mathbf{U}$ under P^{True}. A DAG G is an *independence map (I-map)* if all the d-separations in G entail conditional independence under P^{True}.

Definition 4 I-map
Let G be a DAG on $\mathbf{V}$. We say that G is an I-map *if and only if, for all pairwise disjoint subsets $\mathbf{X}, \mathbf{Y}, \mathbf{U} \subseteq \mathbf{V}$, the following implication holds:*

$$\mathbf{X} \perp^G_d \mathbf{Y} \mid \mathbf{U} \;\Rightarrow\; \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{U}.$$

If G is an I-map, then P^{True} factorizes according to G, so there exists Θ_G such that $P_{(G,\Theta_G)} = P^{\text{True}}$. Otherwise, there is no Θ_G such that $P_{(G,\Theta_G)} = P^{\text{True}}$.

Next, the following notation is introduced to support our discussion of learning BNs. Let $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d, \ldots, \boldsymbol{x}_N\}$ be a complete dataset consisting of N i.i.d. instances, where each instance $\boldsymbol{x}_d$ is a data vector $(x_{0,d}, x_{1,d}, \ldots, x_{n,d})$. The likelihood of a BN $B = (G, \Theta_G)$, given D, is

$$P_B(D) = \prod_{d=1}^{N} P_B(x_{0,d}, x_{1,d}, \ldots, x_{n,d}) = \prod_{i=0}^{n} \prod_{j=1}^{q(\Pi_{X_i}^G)} \prod_{k=1}^{r_i} \theta_{X_i=k|\Pi_{X_i}^G=j}^{N_{X_i=k,\Pi_{X_i}^G=j}},$$

where $N_{X_i=k,\Pi_{X_i}^G=j}$ represents the number of samples of $X_i = k$ when $\Pi_{X_i}^G = j$. In other words, $N = \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} N_{X_i=k,\Pi_{X_i}^G=j}$. The maximum likelihood estimators of $\theta_{X_i=k|\Pi_{X_i}^G=j}$ are

$$\hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\text{ML}(D)} = \frac{N_{X_i=k,\Pi_{X_i}^G=j}}{N_{\Pi_{X_i}^G=j}},$$

where $N_{\Pi_{X_i}^G=j} = \sum_{k=1}^{r_i} N_{X_i=k,\Pi_{X_i}^G=j}$.

We define $\Theta_{\Pi_{X_i}^G=j} = \bigcup_{k=1}^{r_i} \{\theta_{X_i=k|\Pi_{X_i}^G=j}\}$. The most popular parameter estimator of BNs is the *expected a posteriori* (EAP) of Equation (1), which is the expectation of $\theta_{X_i=k|\Pi_{X_i}^G=j}$ with respect to the density $p(\Theta_{\Pi_{X_i}^G=j} \mid D)$ of Equation (2), assuming the Dirichlet prior density $p(\Theta_{\Pi_{X_i}^G=j})$ of Equation (3).

$$\begin{aligned}
\hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\text{EAP}(D)} &= E(\theta_{X_i=k|\Pi_{X_i}^G=j} \mid D) \\
&= \int \theta_{X_i=k|\Pi_{X_i}^G=j} \cdot p(\Theta_{\Pi_{X_i}^G=j} \mid D) d\Theta_{\Pi_{X_i}^G=j} \\
&= \frac{\alpha_{X_i=k,\Pi_{X_i}^G=j} + N_{X_i=k,\Pi_{X_i}^G=j}}{\alpha_{\Pi_{X_i}^G=j} + N_{\Pi_{X_i}^G=j}}. \qquad (1)
\end{aligned}$$

$$p(\Theta_{\Pi_{X_i}^G=j} \mid D) = \frac{\Gamma(\alpha_{\Pi_{X_i}^G=j} + N_{\Pi_{X_i}^G=j})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{X_i=k,\Pi_{X_i}^G=j} + N_{X_i=k,\Pi_{X_i}^G=j})} \prod_{k=1}^{r_i} \theta_{X_i=k|\Pi_{X_i}^G=j}^{\left(\alpha_{X_i=k,\Pi_{X_i}^G=j} + N_{X_i=k,\Pi_{X_i}^G=j} - 1\right)}. \qquad (2)$$

$$p(\Theta_{\Pi_{X_i}^G=j}) = \frac{\Gamma(\alpha_{\Pi_{X_i}^G=j})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{X_i=k,\Pi_{X_i}^G=j})} \prod_{k=1}^{r_i} \theta_{X_i=k|\Pi_{X_i}^G=j}^{\alpha_{X_i=k,\Pi_{X_i}^G=j} - 1}. \qquad (3)$$

In Equations (1) through (3), $\alpha_{X_i=k,\Pi_{X_i}^G=j}$ denotes the hyperparameters of the Dirichlet prior distributions, with $\alpha_{\Pi_{X_i}^G=j} = \sum_{k=1}^{r_i} \alpha_{X_i=k,\Pi_{X_i}^G=j}$.

Because the BN structure is generally unknown, the structure must be learned from the observed data. Among structure-learning methods, the most common is the score-based approach. In this paradigm, one searches over DAGs on $\mathbf{V}$ for a DAG G that maximizes a chosen scoring criterion $Score(G, D)$. It is desirable to employ a scoring criterion that has an *asymptotic consistency*, defined below.

Definition 5 Asymptotic consistency of scoring criterion (Chickering, 2002) *Let G_1 be an arbitrary DAG on $\mathbf{V}$, and let G_2 be the DAG obtained by adding the edge $Y \to X$ to G_1. A scoring criterion Score has an* asymptotic consistency *if and only if the following two properties hold.*

- *If G_1 is an I-map and G_2 is not an I-map, then*

$$\lim_{N\to\infty} P(Score(G_1, D) > Score(G_2, D)) = 1.$$

- *If G_1 and G_2 both are I-maps, and if G_1 has fewer parameters than G_2, then*

$$\lim_{N\to\infty} P(Score(G_1, D) > Score(G_2, D)) = 1.$$

The marginal likelihood has an asymptotic consistency (Chickering, 2002). Moreover, the marginal likelihood has the following *asymptotic local consistency.*

Definition 6 Asymptotic local consistency of scoring criterion (Chickering, 2002) *Let G_1 be an arbitrary DAG on $\mathbf{V}$, and let G_2 be the DAG obtained by adding the edge $Y \to X$ to G_1. A scoring criterion Score has an asymptotic local consistency if and only if the following two properties hold.*

- $X \perp\!\!\!\perp Y \mid \Pi_X^{G_1} \Rightarrow \lim_{N\to\infty} P(Score(G_1, D) > Score(G_2, D)) = 1.$
- $X \not\perp\!\!\!\perp Y \mid \Pi_X^{G_1} \Rightarrow \lim_{N\to\infty} P(Score(G_1, D) < Score(G_2, D)) = 1.$

Assuming Dirichlet priors on $\Theta_{X_i|\Pi_{X_i}^G=j}$ for all $i \in \{0, 1, \ldots, n\}$ and $j \in \{1, \ldots, q(\Pi_{X_i}^G)\}$, the marginal likelihood admits a closed form. In particular, when the Dirichlet hyperparameters are set to

$$\alpha_{X_i=k,\Pi_{X_i}^G=j} = \frac{N'}{r_i q(\Pi_{X_i}^G)} \quad \text{for all } i \in \{0, \ldots, n\}, j \in \{1, \ldots, q(\Pi_{X_i}^G)\}, k \in \{1, \ldots, r_i\},$$

the marginal likelihood is called the *Bayesian Dirichlet equivalent uniform (BDeu)* (Buntine, 1991; Heckerman et al., 1995), and is given by

$$\begin{aligned}
\mathrm{BDeu}(G, D) &= \int P_{(G,\Theta_G)}(D)\; p(\Theta_G)\; d\Theta_G \\
&= \prod_{i=0}^{n} \prod_{j=1}^{q(\Pi_{X_i}^G)} \frac{\Gamma\left(\dfrac{N'}{q(\Pi_{X_i}^G)}\right)}{\Gamma\left(\dfrac{N'}{q(\Pi_{X_i}^G)} + N_{\Pi_{X_i}^G=j}\right)} \prod_{k=1}^{r_i} \frac{\Gamma\left(\dfrac{N'}{r_i q(\Pi_{X_i}^G)} + N_{X_i=k,\Pi_{X_i}^G=j}\right)}{\Gamma\left(\dfrac{N'}{r_i q(\Pi_{X_i}^G)}\right)},
\end{aligned}$$

where $N' > 0$ is the equivalent sample size (ESS) determined by users. This study uses the BDeu score, a widely used marginal likelihood that has asymptotic consistency; however, BIC or MDL (Schwarz, 1978; Rissanen, 1978), both of which are asymptotic approximations to the marginal likelihood, can also be used. The BIC(MDL) is represented as

$$\begin{aligned}\mathrm{BIC}(G, D) &= \sum_{d=1}^{N} \log P_{(G, \widehat{\Theta}_G^{\mathrm{ML}(D)})}(x_{0,d}, x_{1,d}, \ldots, x_{n,d}) - \frac{\log N}{2} \sum_{i=0}^{n} (r_i - 1) q(\Pi_{X_i}^G) \\ &= \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} N_{X_i=k, \Pi_{X_i}^G=j} \log \frac{N_{X_i=k, \Pi_{X_i}^G=j}}{N_{\Pi_{X_i}^G=j}} - \frac{\log N}{2} \sum_{i=0}^{n} (r_i - 1) q(\Pi_{X_i}^G), \quad (4)\end{aligned}$$

where $\widehat{\Theta}_G^{\mathrm{ML}(D)} = \bigcup_{i=0}^{n} \bigcup_{j=1}^{q(\Pi_{X_i}^G)} \bigcup_{k=1}^{r_i} \{\hat{\theta}^{\mathrm{ML}(D)}_{X_i=k|\Pi_{X_i}^G=j}\}$. The first term in (4), the log likelihood at the maximum likelihood estimates, is the fitting term, which reflects the degree of model fitting to the training data. The second term is the penalty term, which signifies the model complexity.

Both BDeu and BIC(MDL) are *decomposable*, i.e., their scores can be expressed as a sum of *local scores LS* depending only on one variable and its parents as

$$Score(G, D) = \sum_{i=0}^{n} LS(X_i, \Pi_{X_i}^G, D).$$

The decomposable score greatly facilitates the search for structures (Silander and Myllymäki, 2006; Bartlett and Cussens, 2013). Note, however, that even with a decomposable score, BN structure learning is known to be an NP-complete problem (Chickering, 1996).

2.2 Bayesian network classifiers

A Bayesian network classifier (BNC) can be interpreted as a BN for which X_0 is the class variable and for which $X_1, \ldots, X_n$ are feature variables. When $X_0, X_1, \ldots, X_n$ take values $x_0, x_1, \ldots, x_n$, the BNC $B = (G, \Theta_G)$ computes the class-posterior as follows.

$$\begin{aligned}P_B(x_0 \mid x_1, \ldots, x_n) &= \frac{P_B(x_0, x_1, \ldots, x_n)}{P_B(x_1, \ldots, x_n)} \\ &= \frac{\prod_{i=0}^{n} \theta_{X_i=x_i|\Pi_{X_i}^G=j_i(x_0)}}{\sum_{c=1}^{r_0} \theta_{X_0=c|\Pi_{X_0}^G=j_0(c)} \prod_{i=1}^{n} \theta_{X_i=x_i|\Pi_{X_i}^G=j_i(c)}} \\ &= \frac{\theta_{X_0=x_0|\Pi_{X_0}^G=j_0(x_0)} \prod_{i:X_0 \in \Pi_{X_i}^G} \theta_{X_i=x_i|\Pi_{X_i}^G=j_i(x_0)}}{\sum_{c=1}^{r_0} \theta_{X_0=c|\Pi_{X_0}^G=j_0(c)} \prod_{i:X_0 \in \Pi_{X_i}^G} \theta_{X_i=x_i|\Pi_{X_i}^G=j_i(c)}} \quad (5)\end{aligned}$$

In those equations, $j_i(c)$ denotes the index of the configuration taken by $\Pi_{X_i}^G$ when $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, and $X_0 = c$. From Equation (5), we can infer a class given only the values of the parents of X_0, the children of X_0, and the parents of the children of X_0, which comprise the *Markov blanket* of X_0.

However, Friedman et al. (1997) reported that a BNC minimizing MDL is unable to optimize the classification performance. Instead of the log likelihood for learning BNC structures, they proposed the sole use of CLL of the class variable given feature variables:

$$\begin{aligned} CLL(B, D) &= \sum_{d=1}^{N} \log P_B(x_{0,d} \mid x_{1,d}, \ldots, x_{n,d}) \\ &= \sum_{d=1}^{N} \log P_B(x_{0,d}, x_{1,d}, \cdots, x_{n,d}) - \sum_{d=1}^{N} \log \sum_{c=1}^{r_0} P_B(c, x_{1,d}, \cdots, x_{n,d}). \end{aligned}$$

Furthermore, they proposed conditional MDL (CMDL), which is a modified MDL replacing log likelihood with CLL. Consequently, they claimed that the BNC minimizing CMDL as a discriminative model showed better classification accuracy than that maximizing marginal likelihood as a generative model.

Unfortunately, CLL is not decomposable: we cannot describe the second term of CLL as a sum of the log parameters in Θ_G. This finding implies that no closed-form equation exists for the maximum CLL estimator for Θ_G. Therefore, learning the network structure which minimizes the CMDL requires a search method such as gradient descent over the space of parameters for each structure candidate. For that reason, exact learning of network structures by minimizing CMDL is computationally infeasible.

To resolve this difficulty, Friedman et al. (1997) proposed an *augmented naive Bayes classifier (ANB)* in which the class variable links directly to all feature variables. Links among feature variables are allowed. Formally, we define ANB as follows:

Definition 7 Augmented Naive Bayes Classifiers (Friedman et al., 1997)
Let G be a DAG on $\mathbf{V}$. A BNC with structure G is an augmented naive Bayes classifier (ANB) *if and only if the following condition is satisfied:*

$$\forall i \in \{1, \ldots, n\}, X_0 \in \Pi_{X_i}^G.$$

Actually, ANB ensures that all feature variables can contribute to classification. Later, restricted ANBs of various types were proposed, such as tree-augmented naive Bayes classifiers (TAN) (Friedman et al., 1997) and forest-augmented naive Bayes classifiers (FAN) (Lucas, 2004).

Various approximate methods have been proposed to maximize CLL. Carvalho et al. (2013) proposed an aCLL score, which is decomposable and computationally efficient. Moreover, Grossman and Domingos (2004) proposed a structure-learning method using a greedy hill-climbing algorithm to maximize the CLL while estimating the parameters by maximizing the log likelihood. Mihaljević et al. (2018) proved that all possible class-posteriors are covered in *class-rooted DAGs (CRDAGs)* defined as follows:

Definition 8 Class-Rooted DAGs
Let G be a DAG on $\mathbf{V}$. We say that G is a class-rooted DAG (CRDAG) *if and only if the class variable X_0 has no parents.*

Furthermore, they proposed the *minimal class-focused directed acyclic graph* (MC-DAGs), a reduced subclass of CRDAGs, and they proposed a greedy search algorithm in the space

of MC-DAGs using the CLL score. These reports described that the BNC maximizing the approximated CLL provides better performance than that provided by maximizing the approximated marginal likelihood.

However, because marginal likelihood has asymptotic consistency, the classification accuracies obtained by maximizing marginal likelihood are expected to be comparable to those obtained by maximizing CLL for large datasets. Differences found between the performances of the two scores in these earlier studies might depend on their learning algorithms used to maximize marginal likelihood. They were approximate learning algorithms, not exact learning algorithms.

Sugahara et al. (2018) compared the classification performances of the BNC with exact learning using marginal likelihood as a generative model and those with approximate learning using CLL as a discriminative model. The results indicated that, for large datasets, maximizing marginal likelihood provides better classification accuracy than maximizing CLL does. However, the findings also demonstrate that the classification accuracies obtained by exact learning of BNC using marginal likelihood are much lower than those obtained using other methods when the sample is small, and when the class variable has numerous parents in the exactly learned networks. When a class variable has numerous parents, estimation of the conditional probability parameters of the class variable becomes unstable because the parent configurations become numerous and the sample used for learning the parameters becomes sparse.

To resolve that difficulty, they proposed an exact learning ANB algorithm, which maximizes BDeu and which ensures that the class variable has no parents. As described in earlier reports, the ANB constraint was used to learn BNC as a discriminative model. In contrast, they used the ANB constraint to learn BNC as a generative model. Their proposed method is guaranteed to achieve an I-map with the fewest parameters among all possible ANB structures when the sample size is sufficiently large. They demonstrated that the proposed method improves the classification accuracy of the exact learning of GBNs for small datasets.

3 Bayesian Network Optimal Classifiers

Because ANB forces the class variable to be connected to every feature variable, the set of *class variable parameters*—those used to compute the class-posterior distribution—becomes unnecessarily large. A larger *number of class variable parameters (NCP)* in turn delays the convergence of the estimated distributions. The NCP in G is defined as

$$\mathrm{NCP}(G) = \sum_{i=0}^{n} \mathrm{NCP}_i\big(\Pi_{X_i}^{G}\big), \tag{6}$$

where, for a variable set $\mathbf{U} \subseteq \mathbf{V}$,

$$\mathrm{NCP}_i(\mathbf{U}) = \begin{cases} (r_i - 1)\, q(\mathbf{U}), & \text{if } i = 0 \text{ or } X_0 \in \mathbf{U}, \\ 0, & \text{otherwise.} \end{cases}$$

However, a GBN learned by maximizing BDeu does not guarantee minimization of NCP. Figure 1 provides a counterexample. Figure 1(a) illustrates the true model, which is a

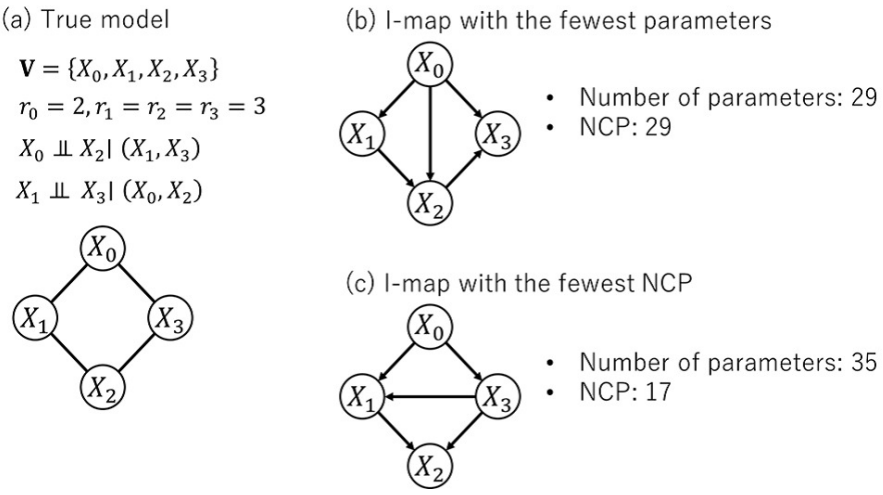

Figure 1: (a) True model, (b) I-map with the fewest parameters, and (c) I-map with the fewest NCP.

Markov network containing a cycle of four variables. For this model, there exists no BN that can perfectly represent the true conditional independencies and conditional dependencies. In this case, the I-map of the true model with the fewest parameters is shown in Figure 1(b), while the I-map of the true model with the fewest NCP is shown in Figure 1(c). Although Figure 1(b) has fewer overall parameters than Figure 1(c), Figure 1(c) has fewer NCP than Figure 1(b).

To address the problem, this study proposes *Bayesian network optimal classifiers (BNOC)* defined as follows:

Definition 9 Bayesian Network Optimal Classifiers
Let G be a CRDAG on $\mathbf{V}$. A BNC with structure G is a Bayesian network optimal classifier (BNOC) *if and only if the following conditions hold:*

1. *G is an I-map.*

2. *Among all I-map CRDAGs on $\mathbf{V}$, G has the fewest NCP.*

There are two reasons for restricting the BNOC structure to CRDAGs. First, this space is guaranteed to cover all possible posteriors of the class variable (Mihaljević et al., 2018). Second, Sugahara et al. (2018); Sugahara and Ueno (2021) experimentally demonstrated that BNCs with no parents of the class variable provide more accurate classification than GBNs do.

In the remainder of this section, we present a theorem and a corollary that are important to the search for a BNOC structure. Before that, we first introduce the following definition.

Definition 10 Variable-Order
Let $\mathbf{U} \subseteq \mathbf{V}$. A variable-order in $\mathbf{U}$ *is an ordered tuple listing each variable of $\mathbf{U}$ exactly*

once. Letting G be a DAG on $\mathbf{U}$ and σ be a variable-order in $\mathbf{U}$, we say that G is consistent with *σ if and only if*

$$\forall X \in \mathbf{U},\ \Pi_X^G \subseteq \mathbf{Pre}_X^\sigma,$$

where $\mathbf{Pre}_X^\sigma$ denotes the set of variables preceding X in σ.

We derive the following theorem.

Theorem 1 *Let σ be an arbitrary variable-order in $\mathbf{V}$. Let $\widehat{G}_\sigma$ be an arbitrary DAG on $\mathbf{V}$ that maximizes BDeu among all DAGs consistent with σ. Let $\mathcal{G}_\sigma^{\min}(\mathbf{V})$ be a set of all I-maps on $\mathbf{V}$ that minimizes NCP among all I-maps consistent with σ. Then*

$$\lim_{N\to\infty} P\left(\widehat{G}_\sigma \ \in\ \mathcal{G}_\sigma^{\min}(\mathbf{V})\right) = 1. \tag{7}$$

We provide a proof of Theorem 1 in the appendix. Note that this theorem holds for any structures, not only for CRDAGs. For any subset $\mathbf{U} \subseteq \mathbf{V}$ with $X_0 \in \mathbf{U}$, let $\boldsymbol{\sigma}_0(\mathbf{U})$ denote a set of variable-orders for $\mathbf{U}$ in which the first element is X_0. Then, Theorem 1 leads to the following corollary.

Corollary 1
Let $\mathcal{G}_{\mathrm{CR}}^{\min}(\mathbf{V})$ denote the set of all I-map CRDAGs on $\mathbf{V}$ with the fewest NCP. Then

$$\lim_{N\to\infty} P\left(\widehat{G}_{\sigma'} \in \mathcal{G}_{\mathrm{CR}}^{\min}(\mathbf{V})\right) = 1,$$

where

$$\sigma' = \arg\min_{\sigma\in\boldsymbol{\sigma}_0(\mathbf{V})} \mathrm{NCP}\big(\widehat{G}_\sigma\big).$$

4 Depth-First Branch-and-Bound Algorithm for Learning BNOC

Corollary 1 suggests the following search process for obtaining a BNOC structure.

1. For each variable-order, obtain the highest BDeu structure. (Each structure maximizes BDeu given each variable-order.)

2. Obtain a structure minimizing NCP among the structures obtained in Step 1.

Before presenting details of the procedure which must be used for our method, we introduce the following notation. Let $\mathbf{U}$ be an arbitrary subset of $\mathbf{V}$ with $X_0 \in \mathbf{U}$. We define the *best parents* of X_i in a candidate set $\mathbf{U}$ as the parent set which maximizes the local score in $\mathbf{U}$: $\widehat{\Pi}_{X_i}(\mathbf{U}) = \arg\max_{\mathbf{W}\subseteq\mathbf{U}} LS(X_i, \mathbf{W}, D)$. We let $\widetilde{G}(\mathbf{U})$ denote a CRDAG on $\mathbf{U}$ that minimizes NCP among all CRDAGs G for which there exists $\sigma \in \boldsymbol{\sigma}_0(\mathbf{U})$ such that G maximizes BDeu among CRDAGs consistent with σ. When a variable has no child in a structure, we say that it is a *sink* in the structure.

Learning a BNOC structure is equivalent to a shortest path problem for the CR reverse variable-order graph (CROG), which is a directed graph consisting of nodes corresponding to elements of $2^{\mathbf{V}} \setminus 2^{\mathbf{V}\setminus\{X_0\}}$. Intuitively, CROG can be understood as "picking nodes in

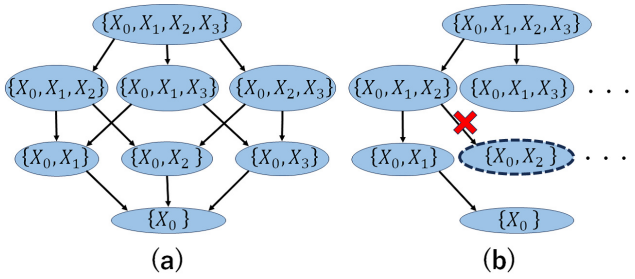

Figure 2: (a) CR reverse variable-order graph (CROG) of four variables. (b) Running example of the depth-first branch-and-bound algorithm.

reverse variable-order." For a variable $X_i \in \mathbf{V}$ and a variable set $\mathbf{U} \subseteq \mathbf{V}$, CROG has an edge from $\mathbf{U}$ to $\mathbf{U} \setminus \{X_i\}$. An example of CROG for $\mathbf{V} = \{X_0, X_1, X_2, X_3\}$ is presented in Figure 2(a). An edge from $\mathbf{U}$ to $\mathbf{U} \setminus \{X_i\}$ represents that a sink in $\widetilde{G}(\mathbf{U})$ is X_i and that the parents of X_i are $\widehat{\Pi}_{X_i}(\mathbf{U} \setminus \{X_i\})$. Moreover, the edge from $\mathbf{U}$ to $\mathbf{U} \setminus \{X_i\}$ has a cost $\mathrm{NCP}_i(\widehat{\Pi}_{X_i}(\mathbf{U} \setminus \{X_i\}))$. Each path from $\mathbf{V}$ to $\{X_0\}$ in CROG corresponds to each variable-order in $\boldsymbol{\sigma}_0(\mathbf{V})$. For each $\sigma \in \boldsymbol{\sigma}_0(\mathbf{V})$, following the path associated with σ yields the family of parent sets $\{\widehat{\Pi}_{X_i}(\mathbf{Pre}^{\sigma}_{X_i})\}_{i=1}^{n}$, which determine $\widehat{G}_\sigma$. The cost $c(p)$ of path p corresponding to σ is defined as

$$c(p) = \sum_{i=1}^{n} \mathrm{NCP}_i\big(\widehat{\Pi}_{X_i}(\mathbf{Pre}^{\sigma}_{X_i})\big) + r_0 - 1 = \mathrm{NCP}\big(\widehat{G}_\sigma\big).$$

By finding the shortest path p^* in which $c(p^*) \leq c(p)$ for any path p, one can obtain $\widetilde{G}(\mathbf{V})$.

Popular methods to solve the shortest path problems are breadth-first search and depth-first search. For both, their computational times increase exponentially along with an increasing number of variables. Their computational costs can be decreased using the branch-and-bound method. For a variable set $\mathbf{U} \subseteq \mathbf{V}$, we define $g(\mathbf{U})$ as the cost of a path from $\mathbf{V}$ to $\mathbf{U}$. Moreover, we define $h(\mathbf{U})$ as the lower bound of the cost of the path from $\mathbf{U}$ to $\{X_0\}$. Also, we use $f(\mathbf{U}) = g(\mathbf{U}) + h(\mathbf{U})$ for the cutting edges of CROG. When $f(\mathbf{U})$ is higher than a cost of the current best solution, we cut the node $\mathbf{U}$ because any path which passes $\mathbf{U}$ is not the best path.

Figure 2(b) depicts an example of depth-first search using the branch-and-bound method for $\mathbf{V} = \{X_0, X_1, X_2, X_3\}$. After expanding $\{X_0, X_1, X_2, X_3\}$ and after exploring $\{X_0, X_1, X_2\}$, we expand $\{X_0, X_1, X_2\}$ and explore $\{X_0, X_1\}$. Subsequently, we expand $\{X_0, X_1\}$ and explore $\{X_0\}$. The resulting structure is an I-map with the fewest NCP among all structures, which is consistent with the variable-order (X_0, X_1, X_2, X_3). The current best solution is updated to the NCP of this structure. Next, we expand $\{X_0, X_2\}$ if $f(\{X_0, X_2\})$ is lower than the current best solution, or cut $\{X_0, X_2\}$ otherwise. Subsequently, we expand $\{X_0, X_1, X_3\}$ if $f(\{X_0, X_1, X_3\})$ is lower than the current best solution, or otherwise cut $\{X_0, X_1, X_3\}$, and so on. Algorithm 1 presents the pseudocode of our proposed algorithm.

In prior work that addresses structure-learning as a shortest path problem, the goal has been to discover the structure with the minimum MDL (Malone et al., 2011; Yuan et al., 2011; Malone and Yuan, 2014). They employed the lower bound of MDL as a heuristic

Algorithm 1 Depth-first branch-and-bound algorithm

1: **function** MAIN(D)
D: input data
2: For all $i \in \{1, \ldots, n\}$ and for all $\mathbf{U} \in 2^{\mathbf{V}\setminus\{X_0, X_i\}}$, compute $\widehat{\Pi}_{X_i}(\mathbf{U} \cup \{X_0\})$ from the data D.
3: **for** $\mathbf{U} \in 2^{\mathbf{V}\setminus\{X_0\}}$ **do**
4: $h_{exact}(\mathbf{U} \cup \{X_0\}) \leftarrow \infty$
5: $g(\mathbf{U} \cup \{X_0\}) \leftarrow \infty$
6: **end for**
7: $optimal \leftarrow \infty$
8: $Repair \leftarrow \{(\mathbf{V}, 0)\}$
9: **while** $|Repair| > 0$ **do**
10: **for** $(\mathbf{S}, g) \in Repair$ **do**
11: **if** $g(\mathbf{S}) > g$ **then**
12: $g(\mathbf{S}) \leftarrow g$
13: $Repair' \leftarrow \text{EXPAND}(\mathbf{S}, Repair)$
14: **end if**
15: **end for**
16: $Repair \leftarrow Repair'$
17: **end while**
18: **end function**

19: **function** EXPAND($\mathbf{U}, Repair$)
20: **for** $X_i \in \mathbf{U}\setminus\{X_0\}$ **do**
21: $g \leftarrow g(\mathbf{U}) + \text{NCP}_i(\widehat{\Pi}_{X_i}(\mathbf{U}\setminus\{X_i\}))$
22: $duplicate \leftarrow exists(g(\mathbf{U}\setminus\{X_i\}))$
23: **if** $g < g(\mathbf{U}\setminus\{X_i\})$ **then**
24: $g(\mathbf{U}\setminus\{X_i\}) \leftarrow g$
25: **end if**
26: **if** *duplicate* and $g < g(\mathbf{U}\setminus\{X_i\})$ **then**
27: $Repair \leftarrow Repair \cup \{(\mathbf{U}, g)\}$
28: **end if**
29: $f \leftarrow h(\mathbf{U}\setminus\{X_i\}) + g(\mathbf{U}\setminus\{X_i\})$
30: **if** !*duplicate* and $f < optimal$ **then**
31: EXPAND($\mathbf{U}\setminus\{X_i\}, Repair$)
32: **end if**
33: **if** $h_{exact}(\mathbf{U}) > \text{NCP}_i(\widehat{\Pi}_{X_i}(\mathbf{U}\setminus\{X_i\})) + h_{exact}(\mathbf{U}\setminus\{X_i\})$ **then**
34: $h_{exact}(\mathbf{U}) \leftarrow \text{NCP}_i(\widehat{\Pi}_{X_i}(\mathbf{U}\setminus\{X_i\})) + h_{exact}(\mathbf{U}\setminus\{X_i\})$
35: **end if**
36: **end for**
37: **if** $\mathbf{U} == \{X_0\}$ **then**
38: $h_{exact}(\mathbf{U}) \leftarrow 0$
39: **end if**
40: **if** $optimal > h_{exact}(\mathbf{U}) + g(\mathbf{U})$ **then**
41: $optimal \leftarrow h_{exact}(\mathbf{U}) + g(\mathbf{U})$
42: **end if**
43: **return** $Repair$
44: **end function**

function used in the branch-and-bound method. However, because our method aims to minimize NCP, the existing heuristic functions are not applicable. We prove the following theorem and propose a new heuristic function for our learning algorithm.

Theorem 2 *Let G^* be a BNOC structure on $\mathbf{V}$. Also, let G^{NB} be the naive Bayes classifier consisting of a set of feature variables $\mathbf{V}_c$, which are children of the class variable in G^*. Then,* $\mathrm{NCP}(G^{\mathrm{NB}}) \leq \mathrm{NCP}(G^*)$ *holds.*

We provide a proof of Theorem 2 in the appendix. This theorem states that one can predict the lower bound of NCP of G^* given $\mathbf{V}_c$. We propose the heuristic function $h^*(\mathbf{U}) = \sum_{X_i \in (\mathbf{U} \cap \mathbf{V}_c)} \mathrm{NCP}_i(\{X_0\})$. The proposed heuristic function has consistency. A heuristic function with consistency guarantees that one can find the shortest path.

Definition 11 *For any node $\mathbf{U}$ or $\mathbf{R}$ in which there is an edge from $\mathbf{U}$ to $\mathbf{R}$ in a CROG, the heuristic function $h(\mathbf{U})$ has consistency if and only if $h(\mathbf{U}) \leq h(\mathbf{R}) + c(\mathbf{U}, \mathbf{R})$ holds, and where $c(\mathbf{U}, \mathbf{R})$ is the cost of the edge from $\mathbf{U}$ to $\mathbf{R}$.*

Theorem 3 *h^* has consistency.*

We provide a proof of Theorem 3 in the appendix.

Because we do not obtain $\mathbf{V}_c$ before learning structures, the proposed method predicts the $\mathbf{V}_c$ using feature selection with Bayes factor. Sugahara and Ueno (2021) proposed a method for learning the children of the class variable using the Bayes factor, which is a ratio between BDeu of independence model g_1 and that of dependence model g_2, i.e., $\mathrm{BDeu}(g_1, D)/\mathrm{BDeu}(g_2, D)$. Sugahara and Ueno (2021) determines the independence between the class variable and each feature variable when the Bayes factor is lower than a threshold. We employ this method to obtain $\mathbf{V}_c$ used in the proposed heuristic function. Note, however, that this feature selection method may select more variables than $\mathbf{V}_c$, which can potentially break consistency.

While Algorithm 1 shares the same $O(n2^n)$ time complexity as existing exact GBN learning methods, it offers superior practical performance. This efficiency stems from avoiding the parent set search for the class variable. As a result, the local-score computations are limited to $n2^n$, the best-parent computations are limited to $n2^{n-1}$, and the CROG size is restricted to 2^n nodes. This is a significant reduction compared to conventional approaches, which require $(n+1)2^n$ local-score computations, $(n+1)2^n$ best-parent computations, and 2^{n+1} nodes. Here, n denotes the count of feature variables. Moreover, we apply branch-and-bound pruning during the CROG search, further reducing the running time.

Additionally, it is notable that the structure learned using the proposed method differs from that learned by maximizing the marginal likelihood of GBN. Although learning GBN provides a graphical structure to approximate the probability distributions over all the variables, learning the proposed BNC yields the CI relation among the feature variables to optimize prediction of the class variable.

5 Integer Programming for Learning BNOC

In score-based approaches, integer programming (IP) provides a more efficient method than dynamic programming and breadth- or depth-first search (Cussens, 2012; Bartlett

and Cussens, 2013; Liao et al., 2019). In addition, IP solvers are anytime algorithms which can supply the best solution (here best BNOC structure) found at any point in time. Using IP, we propose a fast approximation method to obtain a BNOC structure. The depth-first branch-and-bound method proposed in Section 4 consists of two steps: maximizing BDeu for best-parents search and minimizing NCP for structure search. On the other hand, the IP-based method concurrently conducts these two steps with a single objective function. In addition, it reduces the space complexity to $O(n \sum_{j=0}^{d} \binom{n}{j})$ when we give an upper bound d for the number of parents. We first formulate IP for BN structure learning.

5.1 Encoding the BN Learning Problem as an Integer Program

In this section, we define the IP formulation to learn a BNOC structure, based on Cussens (2012); Bartlett and Cussens (2013); Liao et al. (2019). The variables in the IP encoding represent whether or not a node has a given parent set in the network. For every possible node, $X \in \mathbf{V}$ and parent set $\mathbf{W}$, a binary variable $I(\mathbf{W} \to X)$ is created which will be assigned the value 1 if $\mathbf{W}$ is the parent set of X in the BN or 0 otherwise. Using this encoding, one must write the objective function to maximise as a linear combination of these variables. One can then write the score to be optimised as the following linear expression of the IP variables.

$$\sum_{X \in \mathbf{V}} \sum_{\mathbf{W} \subseteq \mathbf{V} \setminus \{X\}} LS(X, \mathbf{W}, D) I(\mathbf{W} \to X). \tag{8}$$

Having defined the IP variables and objective function, it simply remains to define the constraints that must be obeyed by the $I(\mathbf{W} \to X)$ variables in order that they encode a valid network. There are two such constraints; each node must have exactly one parent set, and there cannot be any cycles in the network. The first of these constraints can be written very directly as follows.

$$\forall X \in \mathbf{V}, \quad \sum_{\mathbf{W} \subseteq \mathbf{V} \setminus \{X\}} I(\mathbf{W} \to X) = 1. \tag{9}$$

The acyclicity constraint is much less straightforward to encode. Observe that for any set of nodes (a cluster), if the graph is acyclic, there must be a node in the set which has no parents in that set, i.e. it either has no parents or all its parents are external to the set. This can be translated into the following linear set of linear inequalities, known as the *cluster constraints* introduced by Jaakkola et al. (2010).

$$\forall \mathbf{C} \subseteq \mathbf{V}, \quad \sum_{X \in \mathbf{C}} \sum_{\mathbf{W} \subseteq \mathbf{V} \setminus \{X\} \text{ s.t. } \mathbf{W} \cap \mathbf{C} = \emptyset} I(\mathbf{W} \to X) \geq 1. \tag{10}$$

Since there are exponentially many cluster constraints, only those that are necessary are added, and this is done by adding them as *cutting planes* during the course of IP solving (Cussens, 2012). Collecting these elements together, we can define the IP formulation to

learn a BNOC structure as follows.

$$\text{maximize} \quad \sum_{X \in \mathbf{V}} \sum_{\mathbf{W} \subseteq \mathbf{V} \setminus \{X\}} LS(X, \mathbf{W}, D)\, I(\mathbf{W} \to X) \tag{11}$$

$$\text{subject to} \quad \forall X \in \mathbf{V}, \quad \sum_{\mathbf{W} \subseteq \mathbf{V} \setminus \{X\}} I(\mathbf{W} \to X) = 1. \tag{12}$$

$$\forall \mathbf{C} \subseteq \mathbf{V}, \quad \sum_{X \in \mathbf{C}} \sum_{\mathbf{W} \subseteq \mathbf{V} \setminus \{X\} \text{ s.t. } \mathbf{W} \cap \mathbf{C} = \emptyset} I(\mathbf{W} \to X) \geq 1. \tag{13}$$

$$\forall X \in \mathbf{V} \setminus \{X_0\},\ \forall \mathbf{W} \subseteq \mathbf{V} \setminus \{X\}, \quad I(\mathbf{W} \to X) \in \{0, 1\}. \tag{14}$$

$$\forall \mathbf{W} \subseteq \mathbf{V} \setminus \{X_0\}, \quad I(\mathbf{W} \to X_0) = \begin{cases} 1 & \text{if } \mathbf{W} = \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

When the goal is classifier learning, the search space can be further reduced. In the IP above, for each feature variable we still explore parent sets that exclude the class variable. However, the parameters of feature variables that do not have the class variable as a parent are not used for classification (see (5)), so there is no need to consider such parent sets. Accordingly, we can reduce the search space by fixing decision variables in (14) as follows:

$$\forall X \in \mathbf{V} \setminus \{X_0\},\ \forall \mathbf{W} \subseteq \mathbf{V} \setminus \{X\} \text{ s.t. } X_0 \in \mathbf{W}, \quad I(\mathbf{W} \to X) \in \{0, 1\}. \tag{16}$$

$$\forall X \in \mathbf{V} \setminus \{X_0\},\ \forall \mathbf{W} \subseteq \mathbf{V} \setminus \{X\} \text{ s.t. } X_0 \notin \mathbf{W}, \quad \begin{cases} I(\mathbf{W} \to X) \in \{0, 1\} & \text{if } \mathbf{W} = \emptyset, \\ I(\mathbf{W} \to X) = 0 & \text{otherwise.} \end{cases} \tag{17}$$

This modification fixes about half of the decision variables in (14) and is therefore expected to significantly accelerate solve times.

To effect the reduction given by (15) and (17), GOBNILP was extended to allow the user to (1) specify the maximum number of parents for a particular child (a `childpalim` constraint) and (2) to specify that a particular variable must be present in all non-empty parent sets other than its own (a `required_parent_ifany` constraint). If `X0` is the class variable then putting the following two lines in a constraints file suffices to ensure that a BNOC structure is learned:

```
childpalim X0 0
required_parent_ifany X0
```

For efficiency reasons, binary variables $I(\mathbf{W} \to X)$ which violate these constraints are never created by GOBNILP, rather than being (pointlessly) created and then later removed.

5.2 Objective Function with NCP Penalty

This section proposes a *BDeu-NCP score* that integrates the two steps of the depth-first branch-and-bound method in section 4 into a single objective.

$$\begin{aligned}
&\text{BDeu-NCP}(G, D, \gamma) = \log BDeu(G, D) - \gamma \text{NCP}(G) \\
&= \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \left[\log \frac{\Gamma\left(\frac{N'}{q(\Pi_{X_i}^G)}\right)}{\Gamma\left(\frac{N'}{q(\Pi_{X_i}^G)} + N_{\Pi_{X_i}^G = j}\right)} + \sum_{k=1}^{r_i} \log \frac{\Gamma\left(\frac{N'}{r_i q(\Pi_{X_i}^G)} + N_{X_i=k, \Pi_{X_i}^G = j}\right)}{\Gamma\left(\frac{N'}{r_i q(\Pi_{X_i}^G)}\right)} \right] \\
&\quad - \gamma \sum_{i=0}^{n} \text{NCP}_i(\Pi_{X_i}^G), \qquad (18)
\end{aligned}$$

where γ is a hyperparameter representing the penalty weight for NCP. Since the BDeu-NCP score is decomposable, we use its local scores as the LS in the objective function (11), as follows:

$$\begin{aligned}
&\forall i \in \{0, 1, \ldots, n\}, \\
&LS(X_i, \mathbf{W}, D) = \sum_{j=1}^{q(\mathbf{W})} \left[\log \frac{\Gamma\left(\frac{N'}{q(\mathbf{W})}\right)}{\Gamma\left(\frac{N'}{q(\mathbf{W})} + N_{\mathbf{W}=j}\right)} + \sum_{k=1}^{r_i} \log \frac{\Gamma\left(\frac{N'}{r_i q(\mathbf{W})} + N_{X_i=k, \mathbf{W}=j}\right)}{\Gamma\left(\frac{N'}{r_i q(\mathbf{W})}\right)} \right] \\
&\qquad - \gamma \text{NCP}_i(\mathbf{W}).
\end{aligned}$$

The following theorem shows that there exists γ such that a structure with the highest BDeu-NCP score converges to a BNOC structure.

Theorem 4 *Let $\mathcal{G}_{\text{CR}}(\mathbf{V})$ denote the set of all CRDAGs on $\mathbf{V}$, and let $\mathcal{G}_{\text{CR}}^{\min}(\mathbf{V})$ denote the set of all I-map CRDAGs on $\mathbf{V}$ with the fewest NCP. Then the following holds:*

$$\begin{aligned}
&\forall G^* \in \mathcal{G}_{\text{CR}}^{\min}(\mathbf{V}),\ \exists c > 0,\ \forall G \in \mathcal{G}_{\text{CR}}(\mathbf{V}) \setminus \mathcal{G}_{\text{CR}}^{\min}(\mathbf{V}), \\
&\qquad \lim_{N \to \infty} P(\text{BDeu-NCP}(G^*, D, \gamma_N) > \text{BDeu-NCP}(G, D, \gamma_N)) = 1,
\end{aligned}$$

where $\gamma_N = c \log N$.

We provide a proof of Theorem 4 in the appendix.

6 Experiments

This section presents a description of experiments conducted to demonstrate the important benefits of the proposed method.

6.1 Comparing proposals with Bayesian network classifiers

6.1.1 Benchmark data

First, we compare the classification accuracies of the following methods using the benchmark datasets presented in Table 1.

1. *Naive Bayes*: Naive Bayes classifier
2. *GBN-CMDL* (Grossman and Domingos, 2004): Greedy learning GBN method using the hill-climbing search by minimizing CMDL while estimating parameters by maximizing log likelihood to calculate the CLL
3. *BNC2P* (Grossman and Domingos, 2004): Greedy learning method with at most two parents per variable using hill-climbing search by maximizing CLL while estimating parameters by maximizing log likelihood to calculate the CLL
4. *TAN-aCLL* (Carvalho et al., 2013): Exact learning TAN method by maximizing aCLL
5. *MC-DAGGES* (Mihaljević et al., 2018): Greedy learning method in the space of MC-DAGs using greedy equivalence search (Chickering, 2002) by maximizing CLL while estimating parameters by maximizing log likelihood to calculate the CLL
6. *GBN-BDeu*: Exact learning GBN method by maximizing BDeu
7. *ANB-BDeu*: Exact learning ANB method by maximizing BDeu
8. *fsANB-BDeu*: Exact learning ANB method by maximizing BDeu with feature selection by Bayes factor (Sugahara and Ueno, 2021)
9. *BNOC-BFS*: Learning BNOC using a breadth-first search to the shortest path problems of CROG
10. *BNOC-DFBnB*: Learning BNOC using the proposed depth-first branch-and-bound algorithm to the shortest path problems of CROG
11. *BNOC-IP*: Learning BNOC by solving the IP defined in expressions (11) – (13) and (15) – (17).

When evaluating classification accuracies, we use EAP estimators with $\alpha_{X_i=k,\Pi^G_{X_i}=j} = 1/(r_i q(\Pi^G_{X_i}))$ as conditional probability parameters of the respective classifiers (Ueno, 2010, 2011). We employ nested cross-validation for tuning the ESS $N' \in \{1, 10, 100, 1,000\}$ of each method (*GBN-BDeu*, *ANB-BDeu*, *fsANB-BDeu*, *BNOC-BFS*, and *BNOC-DFBnB*). Specifically, an outer ten-fold cross validation is used to evaluate the final classification accuracy, while an inner ten-fold cross validation is performed only on each training fold of the outer loop to select the ESS by grid search. The held-out test folds of the outer loop are never used in the tuning process. We tuned $\gamma \in \{1, 2, 4, 8, 16\}$ of *BNOC-IP* using the same inner-loop grid search on the training folds only. The ESS of *BNOC-IP* was fixed at $N' = 1$. *BNOC-BFS* and *BNOC-DFBnB* are implemented in C++. The source code is available online (`http://www.ai.lab.uec.ac.jp/software/`). We run the *BNOC-IP* with GOBNILP (Cussens, 2012), a system for BN structure learning that formulates the problem as an IP and solves it via branch-and-cut mixed-integer programming. The other methods are implemented in Java. As described throughout this paper, our experiments are conducted using a computer with a 3.2 GHz 16-core processor and 128 GB of memory.

No.	Dataset	$\lvert\mathbf{V}\rvert$	Sample size	SPP	cMB	Naive-Bayes	GBN-CMDL	BNC2P	TAN-aCLL	MC-DAG GES	GBN-BDeu	ANB-BDeu	fsANB-BDeu	BNOC-BFS	BNOC-DFBnB	BNOC-IP
1	Image Seg	19	2310	9.41×10^{-15}	1.5×10^{16}	0.9286	0.7994	0.9481	0.9290	0.9286	0.9390	0.9468	0.9468	0.9550	**0.9558**	0.9429
2	Pendigits	17	10992	1.13×10^{-13}	9.7×10^{15}	0.8805	0.8194	0.9541	0.9630	0.8898	**0.9641**	0.9636	0.9636	0.9601	0.9609	0.9553
3	Letter	17	20000	9.32×10^{-13}	8.3×10^{14}	0.7384	0.4729	0.8444	0.8142	0.7810	0.8350	0.8454	0.8435	0.8608	**0.8616**	0.7615
4	Lympho	19	148	1.63×10^{-7}	8.2×10^{7}	**0.8446**	0.7230	0.7973	0.8311	0.8176	0.7500	0.7770	0.7838	0.8041	0.7905	0.7581
5	EEG	15	14980	2.17×10^{-7}	360000	0.6874	0.7592	0.7304	0.7197	0.7166	0.7308	**0.7644**	**0.7644**	0.7304	0.7285	0.7427
6	BCW	10	683	3.42×10^{-7}	1.0×10^{8}	**0.9751**	0.8843	0.9590	0.9488	0.9722	**0.9751**	**0.9751**	**0.9751**	**0.9751**	**0.9751**	**0.9751**
7	Zoo	17	101	7.34×10^{-5}	12	0.9406	0.7921	0.9406	0.9307	0.9406	0.9307	0.9505	0.9604	0.9505	0.9307	**0.9609**
8	Hepatitis	20	80	7.63×10^{-5}	104869	0.8500	0.8000	**0.8875**	0.8750	0.8500	0.6125	0.5750	0.8375	0.7875	0.8000	0.8375
9	Wine	14	178	1.19×10^{-4}	18432	0.9831	0.9494	**0.9888**	0.9775	0.9831	0.9775	0.9663	0.9663	0.9775	0.9775	0.9827
10	Australian	15	690	2.23×10^{-4}	72	0.8464	0.8261	0.8348	0.8522	**0.8638**	0.8507	0.8420	0.8478	0.8551	0.8507	0.8609
11	Vehicle	19	846	8.07×10^{-4}	44339	0.4350	0.6123	0.5922	0.5816	0.5378	0.5768	**0.6253**	0.6135	0.6019	0.5827	0.5886
12	BC	10	277	8.33×10^{-4}	1	0.7401	0.6173	0.6895	0.7184	0.6282	0.7184	0.7040	**0.7473**	0.7401	0.7401	0.7075
13	Heart	14	270	1.22×10^{-3}	384	**0.8444**	0.8333	0.7963	0.8407	0.8111	0.8074	0.8407	0.8370	0.8074	0.8074	0.8259
14	HTRU2	9	17898	1.56×10^{-3}	40	0.9689	0.9668	**0.9796**	0.9764	0.9759	0.9787	0.9779	0.9779	0.9783	0.9784	0.9785
15	Voting	17	232	1.77×10^{-3}	64	0.9095	0.9698	**0.9741**	0.9181	0.9052	0.9655	0.9483	0.9307	0.9655	0.9698	0.9614
16	Solar Flare	11	1389	3.72×10^{-3}	1	0.7811	0.8243	0.8186	0.8229	0.7927	**0.8431**	0.8229	0.8373	**0.8431**	**0.8431**	**0.8431**
17	Glass	10	214	6.63×10^{-3}	96	0.5561	0.5607	0.6028	0.6308	0.5467	0.5701	**0.6449**	0.5911	0.6262	0.6075	0.5654
18	cmc	10	1473	2.66×10^{-2}	9	0.4671	0.4542	0.4630	0.4705	0.4650	0.4542	0.4481	0.4610	0.4623	0.4467	**0.4753**
19	H–Roth	5	132	2.29×10^{-1}	1	0.8182	0.6136	0.6515	0.6742	0.5909	0.6212	0.7879	0.7652	**0.8333**	**0.8333**	0.3786
20	Balance	5	625	3.33×10^{-1}	125	**0.9152**	0.3333	0.8672	0.8656	0.7072	**0.9152**	**0.9152**	**0.9152**	**0.9152**	**0.9152**	0.8223
21	Lenses	5	24	3.33×10^{-1}	2	0.7500	0.8333	0.6667	0.7083	0.7500	0.8333	0.7500	**0.8750**	**0.8750**	**0.8750**	0.6167
22	Iris	5	150	6.17×10^{-1}	9	0.9400	**0.9467**	0.9200	0.9400	0.9267	**0.9467**	0.9400	0.9400	**0.9467**	**0.9467**	**0.9467**
23	LED7	8	3200	2.50×10^{0}	64	0.7294	0.7372	**0.7384**	0.7350	0.7325	0.7294	0.7294	0.7294	0.7316	0.7325	0.7359
24	Banknote	5	1372	2.72×10^{0}	1	0.9249	**0.9413**	0.9388	0.9293	0.9388	0.9402	0.9410	0.9410	0.9410	0.9410	0.9351
average						0.8106	0.7529	0.8160	0.8189	0.7938	0.8111	0.8201	0.8354	0.8385	0.8354	0.7983
p-value (*BNOC-BFS vs. the others*)						0.0128	0.0024	0.0071	0.0466	0.0051	0.0012	0.0801	$p > 0.10$	-	-	-
p-value (*BNOC-DFBnB vs. the others*)						0.0847	0.0075	$p > 0.10$	$p > 0.10$	0.0278	0.0166	$p > 0.10$	$p > 0.10$	-	-	-
p-value (*BNOC-IP vs. the others*)						$p > 0.10$	$p > 0.10$	$p > 0.10$	$p > 0.10$	$p > 0.10$	$p > 0.10$	$p > 0.10$	$p > 0.10$	-	-	-

Table 1: Classification accuracies achieved using the respective methods.

This experiment uses 24 real datasets from the *UCI repository* (Lichman, 2013). Continuous variables are discretized using a standard discretization algorithm proposed by (Fayyad and Irani, 1993). In addition, data with missing values are removed from the datasets.

Table 1 presents the classification accuracies found for the respective classifiers. The datasets in Table 1 are listed in ascending order of samples per pattern (SPP). Here, SPP is defined as the sample size divided by the number of configurations of all the variables; this quantity measures the average number of samples per configuration. We also report "cMB", the number of configurations of the class variable's Markov blanket in the structure learned by *GBN-BDeu*. To assess the significance of the differences of *BNOC-BFS*, *BNOC-DFBnB*, and *BNOC-IP* from other methods, we apply Hommel's tests (Hommel, 1988), which are used as a standard in machine learning studies (Demšar, 2006). The p-values are presented at the bottom of Table 1. Table 2 presents the NCPs of structures learned using the respective methods. Table 3 presents the average runtimes of the respective methods. Moreover, "NTCE" in Table 3 presents the numbers of times *BNOC-DFBnB* cuts the edges of CROGs.

The results presented in Table 1 indicate that *BNOC-BFS* outperforms other methods at the $p < 0.1$ significance level, except for *fsANB-BDeu*. Moreover, the findings indicate that *BNOC-DFBnB* outperforms *Naive Bayes*, *GBN-CMDL*, *MC-DAGGES*, and *GBN-BDeu* at the $p < 0.1$ significance level.

For large SPP such as datasets Nos. 20 and 24, the classification accuracies of *Naive Bayes*, *BNC2P*, and *TAN-aCLL* tend to be worse than those of *BNOC-BFS* and *BNOC-DFBnB* because of the upper bound of the maximum number of parents. The small upper bound of the maximum number of parents tends to engender poor representational power of the structure (Ling and Zhang, 2003). For large SPPs such as Nos. 3 and 6, *GBN-CMDL* and *MC-DAGGES* yield much lower classification accuracy than *BNOC-BFS* does because the exact learning methods estimate the network structures more precisely than the greedy learning methods do.

The results indicate that *BNOC-BFS* and *BNOC-DFBnB* provide much higher classification accuracies than *GBN-BDeu* and *ANB-BDeu* do for small SPP, such as dataset No. 8. Table 2 shows that *BNOC-BFS* and *BNOC-DFBnB* provide smaller NCPs than *GBN-BDeu* and *ANB-BDeu* do in the dataset because *BNOC-BFS* and *BNOC-DFBnB* directly minimize NCP, but *GBN-BDeu* and *ANB-BDeu* minimize the numbers of all parameters. Consequently, *BNOC-BFS* and *BNOC-DFBnB* can avoid overfitting to the datasets. That feature improves the classification accuracies for small SPP.

The results show clearly that the classification accuracies of *BNOC-BFS* and *BNOC-DFBnB* are almost identical to those of *GBN-BDeu*, *ANB-BDeu*, and *fsANB-BDeu* for large SPP. Especially, our findings indicate that neither the difference between *BNOC-BFS* and *ANB-BDeu* nor the difference between *BNOC-BFS* and *fsANB-BDeu* is significant.

The classification accuracies of *BNOC-BFS* are approximately equal to those of *BNOC-DFBnB* for each dataset. However, *BNOC-BFS* provides higher average classification accuracy than that of *BNOC-DFBnB*. Because *BNOC-BFS* uses no branch-and-bound algorithm to cut edges of CROGs, *BNOC-BFS* estimates the BNOC structures more precisely than *BNOC-DFBnB* does.

No.	Naive-Bayes	GBN-CMDL	BNC2P	TAN-aCLL	MC-DAG GES	GBN-BDeu	ANB-BDeu	fsANB-BDeu	BNOC-BFS	BNOC-DFBnB	BNOC-IP
1	1091.0	34356.3	4374.7	2358.0	1091.0	2320.8	25640.8	25619.0	4323.6	5550.7	7264.2
2	1499.0	68425.8	10467.6	13435.0	930231.0	15252.9	15986.0	15986.0	9175	9887	12459.0
3	3665.0	109779123.0	32919.6	21761.0	113400.6	19121.1	31235.4	30179.8	12354.2	12354.2	6801.6
4	167.0	292.5	431.0	384.6	587.0	206510.5	727675.8	147.4	104.2	145.8	86.7
5	137.0	656785.6	494.2	1060.6	34781.4	2834.0	7515.0	7515.0	1849.2	1849.4	1849.0
6	163.0	1000.0	1126.9	1459.0	7469.2	153.1	163.0	163.0	161.2	161.2	154.0
7	146.0	70.4	126.6	295.8	146.0	415.7	1032.9	524.7	508	290	125.3
8	39.0	8.1	53.6	75.0	39.0	1977.3	6824.8	470.8	9.6	13.4	37.7
9	74.0	23.0	159.7	152.3	74.0	87.2	8183.6	8183.6	28.4	28.4	74.9
10	71.0	415.0	363.2	184.8	67.0	19.2	280.6	211.8	64	60	46.6
11	75.0	75.4	69.6	143.0	109.0	79.7	322.2	312.2	1377	1018.2	190.6
12	65.0	1122.0	352.9	144.8	14979.0	3.6	103.6	28.4	35.2	30.2	33.0
13	41.0	51.2	91.8	82.2	111.8	22.6	52.2	49.8	19.2	19.2	43.2
14	99.0	28672.0	221.3	645.0	3524.0	246.1	2119.0	2119.0	197.8	176.2	565.4
15	33.0	4.0	17.4	63.0	31.2	17.6	96.2	85.0	9.8	9	58.4
16	206.0	1381.4	624.6	839.6	8342.0	19.4	904.4	308.6	8	8	70.4
17	59.0	69.2	83.5	107.0	80.6	30.2	106.4	78.8	480.2	509.6	34.2
18	53.0	780.8	123.5	121.4	234.2	11.0	138.8	137.0	32.6	27.2	84.3
19	35.0	128.0	93.5	107.0	198.8	128.0	35.0	29.0	29	29	14.2
20	50.0	1250.0	179.2	194.0	458.0	48.4	50.0	50.0	50	50	46.0
21	17.0	8.3	26.7	29.3	17.0	8.2	17.3	8.3	8.3	8.3	7.0
22	26.0	12.6	55.6	62.0	102.8	18.0	36.8	36.8	18.8	18.8	14.8
23	79.0	124.0	136.5	139.0	119.0	78.1	79.0	79.0	94	98	79.0
24	29.0	126.0	101.9	71.0	355.0	247.6	315.4	315.4	15.4	15.4	72.0
average	330.0	4607262.7	2195.6	1829.8	46522.9	10402.1	34538.1	3859.9	1289.7	1348.2	1258.8

Table 2: Number of class variable parameters (NCP) of the learned structures obtained using the respective methods.

	Runtime											NTCE
No.	Naive-Bayes	GBN-CMDL	BNC2P	TAN-aCLL	MC-DAG GES	GBN-BDeu	ANB-BDeu	fsANB-BDeu	BNOC-BFS	BNOC-DFBnB	BNOC-IP	BNOC-DFBnB
1	0.0	2573.8	78.1	540.9	5.9	860.5	434.1	2713.0	5411.7	296.7	0.2	149209.5
2	0.0	8351.6	496.7	852.0	197.9	1227.2	605.7	6461.7	700.2	147.4	0.4	37821.8
3	0.0	60640.0	2420.7	2238.4	511.2	2231.4	1107.2	3671.8	504.2	103.2	0.8	78524.0
4	0.0	78.1	3.6	33.9	0.5	67.4	34.1	62.7	555.7	97.3	0.2	189638.4
5	0.0	1604.4	84.4	18.1	33.1	37.7	20.5	295.1	25.7	12.0	10.5	13069.7
6	0.0	5.0	1.1	2.2	0.5	0.3	0.2	2.0	0.3	0.1	0.1	98.0
7	0.0	40.7	1.4	8.0	0.3	5.7	2.8	6.5	377.2	18.2	0.1	6915.0
8	0.0	32.4	1.5	30.6	0.2	65.7	33.0	290.0	10044.8	250.8	0.6	386621.3
9	0.0	11.0	1.4	3.1	0.3	0.9	0.5	4.2	14.8	5.6	0.2	5945.4
10	0.0	54.9	3.6	4.9	0.7	7.2	3.7	10.3	21.3	7.7	0.2	8652.0
11	0.0	351.2	12.0	37.9	5.1	102.3	55.3	177.2	6527.6	206.5	341.2	159026.0
12	0.0	2.5	0.5	2.2	0.2	0.2	0.1	0.2	0.3	0.2	0.1	176.1
13	0.0	12.3	1.3	2.6	0.4	1.2	0.7	4.5	10.9	4.5	0.3	5104.1
14	0.0	98.2	12.7	2.2	6.1	0.3	0.2	3.2	0.2	0.3	0.1	74.5
15	0.0	29.1	1.4	5.6	0.3	8.3	4.1	23.9	427.1	34.8	0.8	22001.9
16	0.0	534.1	6.5	6.8	1.6	0.3	0.2	0.4	0.9	0.5	0.1	326.2
17	0.0	3.8	0.7	4.7	0.2	0.1	0.0	0.4	0.5	0.3	0.1	178.9
18	0.0	15.9	2.2	2.8	0.8	0.2	0.1	0.9	0.4	0.2	0.1	134.1
19	0.0	0.1	0.1	2.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.3
20	0.0	0.3	0.2	2.8	0.1	0.0	0.0	0.1	0.0	0.0	0.0	4.0
21	0.0	0.1	0.1	2.7	0.0	0.0	0.0	0.1	0.0	0.0	0.0	3.9
22	0.0	0.1	0.1	2.7	0.0	0.0	0.0	0.1	0.0	0.0	0.0	3.4
23	0.0	40.7	8.6	7.3	1.1	0.1	0.0	1.0	0.1	0.1	0.1	27.6
24	0.0	0.4	0.3	2.1	0.1	0.0	0.0	0.1	0.0	0.0	0.0	1.9
average	0.0	3103.4	130.8	159.1	32.0	192.4	96.0	572.0	1026.0	49.4	14.8	44315.0

Table 3: Average runtimes (s) of the respective methods and the numbers of times *BNOC-DFBnB* cuts edges.

As shown in Table 3, the average runtime of *BNOC-DFBnB* is shorter than those of any other method, except for *MCDAGGES*. Especially, *BNOC-DFBnB* provides much shorter runtime than *BNOC-BFS* does for large NTCE of *BNOC-DFBnB* such as datasets Nos. 1 and 8. The results demonstrate the efficiency of the proposed branch-and-bound algorithm of *BNOC-DFBnB*.

Overall, *BNOC-IP* achieves shorter runtimes than *BNOC-BFS* and *BNOC-DFBnB*, but its classification accuracy is substantially lower. This likely occurs because the structure learned by *BNOC-IP* is highly sensitive to the value of hyperparameter γ in the BDeu-NCP score. To improve accuracy, the tuning range for γ should be broadened or a finer grid of candidate values should be explored.

Finally, the classification accuracies of the ten methods for twelve large datasets from the UCI repository are compared. Table 4 presents the classification accuracies of the respective classifiers for the large datasets. For all methods other than *Naive-Bayes*, *BNC2P*, and *TAN-aCLL*, we imposed an upper bound on the number of parents of each variable. The column "Limitation on parent set size" in Table 4 indicates this upper bound. In Table 4, "time over (TO)" signifies that learning was not completed within a time limit of 6 hr. Also, "out of memory (MO)" signifies a failure to learn the structure because of memory insufficiency. The results presented in Table 4 demonstrate that *BNOC-IP* outperforms the other methods at the $p < 0.1$ significance level. It is noteworthy that *TAN-aCLL* fails to learn structures because of MO. Four exact learning methods fail to learn structures because of TO: *GBN-BDeu*, *ANB-BDeu*, *fsANB-BDeu*, and *BNOC-BFS*. Actually, the computational time and space of these exact learning methods increase exponentially with the number of variables. However, *BNOC-IP* and *BNOC-DFBnB* can be stopped at any time. The current best solution is obtainable because the depth-first search updates the current best solution sequentially. Although *BNOC-IP* showed no statistically significant difference from *BNOC-DFBnB*, it achieved higher mean classification accuracy. In particular, on the 242-variable dataset, the classification accuracy of *BNOC-IP* is much higher than that of *BNOC-DFBnB*, supporting its effectiveness on large-scale data.

6.1.2 Simulation data

To demonstrate important advantages of the proposed methods (*BNOC-BFS* and *BNOC-IP*) compared to *GBN-BDeu*, *ANB-BDeu*, and *fsANB-BDeu*, this section presents additional experiments using simulation data. Unlike the real-data comparisons in the previous subsection, simulations allow us to directly compare the difference between the true model and the structure estimated by each method, as well as the error between the true and estimated distributions. The experiments presented here compare NCPs of structures learned using the five methods and compare the Kullback–Leibler divergences (KLDs) between the referenced (set true) probability distribution of the class variable and that using the five methods from simulation datasets.

This experiment uses the following three networks: the Cancer network (Scutari, 2010) in the structure (1) of Figure 3, the Asia network (Scutari, 2010) in the structure (2) of Figure 3, and the Markov network in the structure (3) of Figure 3, which is identical to the structure in Figure 1(a).

No.	Dataset	Variables	Sample size	Limitation on parent set size	Naive-Bayes	GBN-CMDL	BNC2P	TAN-aCLL	MC-DAG GES	GBN-BDeu	ANB-BDeu	fsANB-BDeu	BNOC-BFS	BNOC-DFBnB	BNOC-IP
1	wdbc	31	569	5	0.9139	0.9209	0.9209	MO	0.9156	TO	TO	TO	TO	0.9350	0.9244
2	ionosphere	35	351	5	0.7550	0.8860	0.7493	MO	0.7493	TO	TO	TO	TO	0.8832	0.9288
3	kr-vs-kp	37	3196	5	0.6640	0.9252	0.8339	MO	0.7672	TO	TO	TO	TO	0.9252	0.9697
4	biodeg	42	1055	5	0.7877	0.8237	0.8284	MO	0.8066	TO	TO	TO	TO	0.7877	0.8407
5	Flowmeters_D	44	180	5	0.8333	0.8000	0.8833	MO	0.8500	TO	TO	TO	TO	0.8833	0.8333
6	Parkinson	48	240	5	0.7625	0.5000	0.4750	MO	0.7708	TO	TO	TO	TO	0.7708	0.7292
7	PAMAP2	53	174915	4	0.6864	0.6306	0.6502	MO	0.7052	TO	TO	TO	TO	0.8634	0.8782
8	spam	58	4601	5	0.8794	0.9270	0.9168	MO	0.8813	TO	TO	TO	TO	0.9331	0.9346
9	molecular	61	3190	4	0.9433	0.9241	0.9364	MO	0.9266	TO	TO	TO	TO	0.9464	0.9621
10	Nuclear	75	1047	4	0.9303	1.0000	1.0000	MO	0.9370	TO	TO	TO	TO	0.9914	0.9924
11	MI	116	544	3	0.9154	0.9081	0.9136	MO	0.9154	TO	TO	TO	TO	0.9375	0.9209
12	tuandromd	242	4464	2	0.9644	0.9635	0.9486	MO	0.9432	TO	TO	TO	TO	0.7986	0.9756
	average				0.8363	0.8508	0.8380	-	0.8474	-	-	-	-	0.8880	0.9075
	p-value(*BNOC-DFBnB* vs. the others)				0.0166	$p > 0.10$	$p > 0.10$	-	0.0207	-	-	-	-	-	-
	p-value(*BNOC-IP* vs. the others)				0.0064	0.0021	0.0139	-	0.0116	-	-	-	-	-	-

Table 4: Classification accuracies of the respective methods used for large datasets.

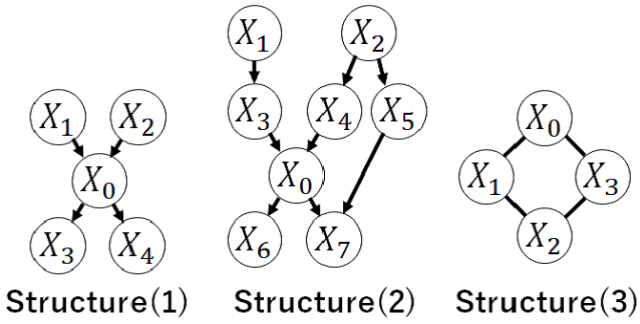

Figure 3: Structures: (1) the Cancer network, (2) the Asia network, and (3) a Markov network with a cycle.

From the three networks, we randomly generate datasets with sizes $N = 100, 1{,}000, 10{,}000,$ and 100,000. Based on the generated data, after learning BNC structures using the five methods, we evaluate the KLDs and the NCPs. Table 5 presents the NCPs and the average KLD over all feature variables' values. "Imin(GBN)" and "Imin(NP)" in Table 5 respectively signify whether the learned structure is an I-map with the lowest NCP in all GBN structures and in all CRDAGs.

The results demonstrate that, for the Cancer network, the average KLDs between the referenced (set true) probability distributions of the class variable and the ones learned by all five methods are identical when $N \geq 10,000$ because the five methods learn an I-map with the lowest NCP. Similarly, for the Asia network, because *GBN-BDeu*, *BNOC-BFS*, and *BNOC-IP* learn an I-map with the lowest NCP, the average KLDs between the referenced (set true) probability distributions of the class variable and those learned by *BNOC-BFS* and *BNOC-IP* are identical to those learned by exact learning *GBN-BDeu* when $N \geq$ 10,000.

Moreover, in the Markov network presented in Figure 3, the average KLDs between the referenced (set true) probability distributions of the class variable and those learned by *BNOC-BFS* and *BNOC-IP* are smaller than those learned by *GBN-BDeu* when $N \geq$ 10,000 because *BNOC-BFS* and *BNOC-IP* learn I-maps with the lowest NCP, but *GBN-BDeu* does not. The reason is that the *GBN-BDeu* does not asymptotically guarantee estimation of an I-map minimizing NCP, although it guarantees minimization of the number of all parameters.

Furthermore, we demonstrate the explainability of the proposed method. Figure 1(b) and 1(c) depict, respectively, the structures learned by *GBN-BDeu* and *BNOC-BFS* using simulation data with $N =$ 100,000 generated by Structure (3) depicted in Figure 3. The structure learned by *BNOC-BFS* (Figure 1(c)) has the true Markov blanket of the class variable, $\{X_1, X_3\}$. However, Figure 1(b) shows that the Markov blanket of the class variable learned by *GBN-BDeu* has an extra variable X_2 that does not affect the class variable. Because NCP of the structure learned by the proposed method is smaller than that of *GBN-BDeu*, the proposed method provides a simpler and understandable relation between the class variable and feature variables than *GBN-BDeu* does.

Network	Sample size	ANB-BDeu avgKLD	ANB-BDeu NCP	ANB-BDeu Imin (NP)	GBN-BDeu avgKLD	GBN-BDeu NCP	GBN-BDeu Imin (GBN)	fsANB-BDeu avgKLD	fsANB-BDeu NCP	fsANB-BDeu Imin (NP)	BNOC-BFS avgKLD	BNOC-BFS NCP	BNOC-BFS Imin (NP)	BNOC-IP avgKLD	BNOC-IP NCP	BNOC-IP Imin (NP)
Cancer (Structure (1))	100	5.11×10^{-2}	9	×	2.70×10^{-2}	5	×	5.87×10^{-2}	7	×	2.70×10^{-2}	5	×	6.73×10^{-2}	1	×
	1,000	4.11×10^{-2}	9	×	2.67×10^{-2}	5	×	3.97×10^{-2}	7	×	2.67×10^{-2}	5	×	4.19×10^{-2}	3	×
	10,000	1.27×10^{-3}	11	◯	1.27×10^{-3}	8	◯	1.27×10^{-3}	11	◯	1.27×10^{-3}	11	◯	1.27×10^{-3}	11	◯
	100,000	8.46×10^{-5}	11	◯	8.46×10^{-5}	8	◯	8.46×10^{-5}	11	◯	8.46×10^{-5}	11	◯	8.46×10^{-5}	11	◯
Asia (Structure (2))	100	8.81×10^{-2}	21	×	5.21×10^{-2}	5	×	1.04×10^{-1}	13	×	4.40×10^{-2}	3	×	4.40×10^{-2}	3	×
	1,000	3.70×10^{-2}	21	×	3.40×10^{-2}	9	×	4.89×10^{-2}	7	×	6.38×10^{-2}	7	×	6.38×10^{-2}	7	×
	10,000	2.58×10^{-2}	21	×	3.60×10^{-3}	10	×	2.88×10^{-2}	15	×	2.46×10^{-2}	11	×	1.75×10^{-2}	9	×
	100,000	1.94×10^{-3}	25	×	2.72×10^{-4}	10	◯	1.32×10^{-2}	17	×	2.72×10^{-4}	13	◯	2.72×10^{-4}	13	◯
Markov net (Structure (3))	100	2.20×10^{-1}	29	×	2.08×10^{-1}	3	×	2.20×10^{-1}	29	×	8.18×10^{-2}	5	×	2.86×10^{-1}	1	×
	1,000	6.47×10^{-2}	29	×	1.38×10^{-2}	17	×	6.47×10^{-2}	29	×	6.63×10^{-2}	5	×	6.63×10^{-2}	5	×
	10,000	2.96×10^{-3}	29	×	2.96×10^{-3}	27	×	2.96×10^{-3}	29	×	4.43×10^{-4}	17	◯	6.39×10^{-2}	5	×
	100,000	1.20×10^{-3}	29	×	1.20×10^{-3}	29	×	1.20×10^{-3}	29	×	7.94×10^{-5}	17	◯	7.94×10^{-5}	17	◯

Table 5: NCPs of structures learned by *ANB-BDeu*, *GBN-BDeu*, *fsANB-BDeu*, *BNOC-BFS*, and *BNOC-IP*, and the average KLDs between the referenced (set true) probability distributions of the class variable and those of the five methods.

6.2 Comparison of the Proposed method with Random Forest and Deep Learning

This subsection presents comparisons of the classification performance of the proposed method with those of *RF* (random forest) and those of *DL* (deep learning). We employ nested cross-validation: an outer ten-fold cross validation to evaluate the final classification accuracy, and an inner ten-fold cross validation performed only on each training fold of the outer loop to tune the hyper-parameters of RF and DL (as shown in Table 6). The test folds of the outer loop are never used in the tuning process. For *DL*, we employ the standard cross-entropy as the loss function.

6.2.1 Benchmark data

First, we compare the classification accuracies of the respective methods for benchmark datasets. In this experiment, we estimated conditional probability parameters of the structure learned by the proposed method using a maximum CLL estimator (MCE), which tends to provide higher classification accuracy than an EAP estimator does (Greiner and Zhou, 2002). Although learning structure to maximize CLL is computationally infeasible as mentioned in Section 2, only estimating parameters of a given structure to maximize CLL is relatively fast enough to be practically feasible. Calculation of the MCE requires a gradient descent algorithm because CLL has no closed-form equation for estimating the optimal parameters. However, the CLL surface of a BNC might have local (non-global) maxima, which leads to a poor classification accuracy. As a theorem to overcome this difficulty, it is known that there exists no local maximum in the CLL surface of a Markov network when the Markov network structure is *chordal*, i.e., all cycles of four or more variables have an edge that is not part of the cycle but which connects two variables of the cycle (Roos et al., 2005; Koller and Friedman, 2009). To obtain the global maxima of CLL, we transform the structure learned by *BNOC-BFS* into a chordal Markov network by adding edges and undirecting all the edges. Then we obtain a MCE of the chordal Markov network structure G^M using the steepest descent method using the following gradient of the CLL as

$$\frac{\partial}{\partial \phi_C(\mathbf{x})} CLL(G^M, \Phi, D) = N_{C=\mathbf{x}} - \sum_{d=1}^{N} P(\mathbf{x}(X_0) \mid x_{1,d}, \ldots, x_{n,d}), \tag{19}$$

where C denotes a clique in G^M, $\mathbf{x}$ stands for a configuration of C, $\phi_C(\mathbf{x})$ is a parameter of $C = \mathbf{x}$, Φ is a set of $\phi_C(\mathbf{x})$ for all cliques C and all configurations $\mathbf{x}$, $N_{C=\mathbf{x}}$ represents the number of samples of $C = \mathbf{x}$, and $\mathbf{x}(X_0)$ denotes the value of X_0 in $\mathbf{x}$. In the steepest descent method, we initialize all the parameters in $\mathbf{\Phi}$ to zero. We designate the resulting method as *proposal with chordalization and MCE (PCMCE)*. PCMCE performs classification by estimating the class-posterior distributions using the chordal Markov network.

Table 7 presents the classification accuracy of *RF*, *DL*, *BNOC-BFS*, and *PCMCE*. The results indicate that both *BNOC-BFS* and *PCMCE* provide lower classification accuracy than *RF* and *DL* do for datasets with small SPP, such as Nos. 1-5. When data become sparse, discriminative models *RF* and *DL* have better classification performance than our proposed BNCs do. On the other hand, *BNOC-BFS* and *PCMCE* provide slightly better average accuracy than *RF* and *DL* do for datasets with large SPP. Because the estimated

Method	Hyperparameter	Candidate
RF	Number of trees in the forest	10, 20, 30, 50, 100, 300
	Maximum depth of the tree	10, 20, 30, 50, 100, 300
DL	Hidden layer sizes	(50, 50, 50), (50, 100, 50), (100), (50, 50, 50, 50), (50, 50, 50, 50, 50)
	Activation function	tanh, relu
	Solver for weight optimization	sgd, adam
	Strength of the L2 regularization α	0.0001, 0.05
	Learning rate schedule for weight updates	constant, adaptive

Table 6: Candidate hyperparameters for *RF* and *DL*.

No.	Dataset	Variables	Sample size	SPP	*RF*	*DL*	*BNOC-BFS*	*PCMCE*
1	Image Segmentation	19	2310	9.41×10^{-15}	0.9706	0.9662	0.9550	0.9558
2	Pendigits	17	10992	1.13×10^{-13}	0.9914	0.9904	0.9601	0.9743
3	Letter	17	20000	9.32×10^{-13}	0.9649	0.9511	0.8608	0.8805
4	Lymphography	19	148	1.63×10^{-7}	0.8386	0.8248	0.8041	0.8176
5	EEG	15	14980	2.17×10^{-7}	0.8132	0.8062	0.7304	0.7408
6	Breast Cancer Wisconsin	10	683	3.42×10^{-7}	0.9693	0.9677	0.9751	0.9531
7	Zoo	17	101	7.34×10^{-5}	0.9500	0.9200	0.9505	0.9505
8	Hepatitis	20	80	7.63×10^{-5}	0.8625	0.8250	0.7875	0.7813
9	Wine	14	178	1.19×10^{-4}	0.9778	0.9719	0.9775	0.9551
10	Australian	15	690	2.23×10^{-4}	0.8449	0.8580	0.8551	0.8601
11	Vehicle	19	846	8.07×10^{-4}	0.6408	0.6277	0.6019	0.6135
12	Breast Cancer	10	277	8.33×10^{-4}	0.7111	0.7151	0.7401	0.7274
Average accuracy for datasets with small SPP (Nos. 1-12)					0.8779	0.8687	0.8498	0.8508
13	Heart	14	270	1.22×10^{-3}	0.8296	0.8370	0.8074	0.8037
14	HTRU2	9	17898	1.56×10^{-3}	0.9797	0.9797	0.9783	0.9788
15	Congressional Voting Records	17	232	1.77×10^{-3}	0.9567	0.9567	0.9655	0.9569
16	Solar Flare	11	1389	3.72×10^{-3}	0.8294	0.8409	0.8431	0.8431
17	Glass	10	214	6.63×10^{-3}	0.6310	0.6602	0.6262	0.6946
18	Contraceptive Method Choice	10	1473	2.66×10^{-2}	0.4772	0.4983	0.4623	0.4912
19	Hayes-Roth	5	132	2.29×10^{-1}	0.8088	0.7571	0.8333	0.8258
20	Balance Scale	5	625	3.33×10^{-1}	0.8287	0.9808	0.9152	0.9904
21	Lenses	5	24	3.33×10^{-1}	0.8167	0.7667	0.8750	0.8750
22	Iris	5	150	6.17×10^{-1}	0.9333	0.9467	0.9467	0.9333
23	LED7	8	3200	2.50×10^{0}	0.7269	0.7344	0.7316	0.7334
24	Banknote authentication	5	1372	2.72×10^{0}	0.9432	0.9432	0.9410	0.9406
Average accuracy for datasets with large SPP (Nos. 13-24)					0.8134	0.8251	0.8271	0.8389

Table 7: Classification accuracies of *RF*, *DL*, *BNOC-BFS*, and *PCMCE*.

class-posterior distribution converges to the true class-posterior distribution as $N \to \infty$, the classification accuracies of *BNOC-BFS* and *PCMCE* become high when the sample size becomes large.

Network	Sample size	*RF*	*DL*	*BNOC-BFS*	*PCMCE*
Cancer (Structure (1))	100	8.49×10^{-2}	5.29×10^{-2}	2.70×10^{-2}	2.88×10^{-2}
	1,000	5.33×10^{-2}	4.85×10^{-2}	2.67×10^{-2}	2.63×10^{-2}
	10,000	1.88×10^{-3}	8.38×10^{-3}	1.27×10^{-3}	1.45×10^{-3}
	100,000	2.73×10^{-4}	5.11×10^{-3}	8.46×10^{-5}	3.80×10^{-5}
Asia (Structure (2))	100	1.23×10^{-1}	1.51×10^{-1}	4.40×10^{-2}	4.37×10^{-2}
	1,000	1.82×10^{-1}	2.31×10^{-1}	6.38×10^{-2}	6.35×10^{-2}
	10,000	5.62×10^{-2}	3.11×10^{-2}	2.46×10^{-2}	2.40×10^{-2}
	100,000	3.26×10^{-2}	1.92×10^{-2}	2.72×10^{-4}	5.88×10^{-4}
Markov net (Structure (3))	100	2.73×10^{-1}	1.16×10^{-1}	8.18×10^{-2}	8.16×10^{-2}
	1,000	1.38×10^{-1}	7.86×10^{-2}	6.63×10^{-2}	6.65×10^{-2}
	10,000	1.30×10^{-1}	1.36×10^{-1}	4.43×10^{-4}	4.42×10^{-4}
	100,000	1.17×10^{-1}	1.44×10^{-1}	7.94×10^{-5}	7.94×10^{-5}

Table 8: Average KLDs between the true probability distribution of the class variable and those using *RF*, *DL*, *BNOC-BFS*, and *PCMCE*.

6.2.2 Simulation data

Finally, using simulated data, we compare the KLDs between the true class-posterior distribution and the class-posterior distributions estimated by *RF*, *DL*, *BNOC-BFS*, and *PCMCE*. We obtain the class-posterior distributions for *RF* and *DL* via scikit-learn's `RandomForestClassifier.predict_proba` and `MLPClassifier.predict_proba`, respectively. In scikit-learn, `RandomForestClassifier.predict_proba` returns the average, over trees, of the class frequencies in the terminal leaf reached by each tree (i.e., per-tree empirical class probabilities). `MLPClassifier.predict_proba` returns the network's output probabilities (softmax for multi-class, logistic for binary). In the same way as the preceding subsection, this experiment uses the three networks depicted in Figure 3 as referenced (set true) models. Datasets with sizes $N = 100, 1{,}000, 10{,}000, 100{,}000$ are generated randomly for each network. Table 8 presents the average KLD over all feature variables' values for each method and each sample size. The results indicate that the average KLDs of *BNOC-BFS* and *PCMCE* close to zero as sample size increases, but those of *RF* and *DL* do not. This difference of tendencies demonstrates that *BNOC-BFS* and *PCMCE* guarantee asymptotic estimation of the true probability distribution, although *RF* and *DL* do not. Therefore, the proposed method is superior to *RF* and *DL* when accurate estimation of probabilities is necessary, as it is for decision-making tasks. The differences between the average KLDs of *BNOC-BFS* and *PCMCE* for all datasets might be caused by errors of EAP estimator and MCE.

Minimizing NCP reduces the average number of children of the class variable (see the end of Section 6.1.2). This may improve explainability. The proposed method is expected to be more explainable than DL and RF in both probability estimation and variable selection.

7 Conclusions

We proposed Bayesian network optimal classifiers (BNOC) which can asymptotically estimate the true probability distribution with the fewest NCP. We proposed two algorithms: one based on depth-first branch-and-bound and one based on integer programming. The experimental results demonstrated the following: (1) BNOC outperforms almost all the other BNC learning methods at the $p < 0.1$ significance level, (2) that BNOC can estimate the true probability distribution while minimizing NCP, (3) that BNOC provides a shorter runtime than that of almost any other method, and (4) BNOC has higher classification accuracy than random forests and deep learning when the sample size becomes large.

Deep learning and random forests suffer from the "black-box" problem: it is often difficult to understand how they arrive at their decisions. In contrast, the BNOC has explainability while keeping good classification performance and high estimation accuracy of class-posterior distributions.

Exact algorithms for learning optimal Bayesian networks based on score-based approach are still limited to a relatively small number of variables. As an alternative approach to learn larger networks, constraint-based approaches have been proposed. Such approaches learn networks by conditional independence tests and the direction of edges using the orientation rules. Our future work is to extend the proposed method to learn larger networks using the constraint-based approaches with a modified conditional independence test to minimize NCP.

Acknowledgments and Disclosure of Funding

Parts of this research were reported in an earlier conference paper published by Sugahara et al. (2024). The authors are grateful to Dr. Brandon Malone for helpful comments on this work. We also thank him for providing us a code of the depth-first search algorithm for learning Bayesian networks. This research was funded by JSPS KAKENHI Grant Numbers 25K22839, 25K21270, 24H00739, and 22K19825.

Appendix A.

This appendix presents the proofs of Lemma 1 and Theorems 1–4. We begin with the following lemma.

Lemma 1 *For all pairwise disjoint subsets* $\mathbf{X}, \mathbf{Y}, \mathbf{A}, \mathbf{B} \subseteq \mathbf{V}$,

$$\mathbf{X} \perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, \mathbf{Y}) \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid (\mathbf{A}, \mathbf{B}) \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A}.$$

Proof From the decomposition property of conditional independence (Pearl, 1988), $\mathbf{X} \perp\!\!\!\perp (\mathbf{Y}, \mathbf{B}) \mid \mathbf{A} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{B} \mid \mathbf{A}$ holds. The contraposition of the implication above is $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A} \vee \mathbf{X} \not\perp\!\!\!\perp \mathbf{B} \mid \mathbf{A} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp (\mathbf{Y}, \mathbf{B}) \mid \mathbf{A}$. Clearly, one obtains

$$\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp (\mathbf{Y}, \mathbf{B}) \mid \mathbf{A}. \tag{20}$$

From the intersection property of conditional independence (Pearl, 1988), $\mathbf{X} \perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, \mathbf{Y}) \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid (\mathbf{A}, \mathbf{B}) \Rightarrow \mathbf{X} \perp\!\!\!\perp (\mathbf{Y}, \mathbf{B}) \mid \mathbf{A}$ holds. The contraposition of the implication presented above is

$$\mathbf{X} \not\perp\!\!\!\perp (\mathbf{Y}, \mathbf{B}) \mid \mathbf{A} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, \mathbf{Y}) \vee \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid (\mathbf{A} \cup \mathbf{B}). \tag{21}$$

From (20) and (21), we obtain $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, \mathbf{Y}) \vee \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid (\mathbf{A}, \mathbf{B})$. Taking the contrapositive, we also have $\mathbf{X} \perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, \mathbf{Y}) \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid (\mathbf{A}, \mathbf{B}) \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A}$. ■

Next, we introduce the following proposition.

Proposition 1 (Pearl, 2000)
Let G be a DAG on $\mathbf{V}$. For every $X \in \mathbf{V}$, let $\mathbf{ND}_X^G$ denote the set of non-descendants of X in G. Then

$$\forall X \in \mathbf{V},\ X \perp_d^G (\mathbf{ND}_X^G \setminus \Pi_X^G) \mid \Pi_X^G.$$

We now present the following proof of Theorem 1.

Theorem 1 *Let σ be an arbitrary variable-order in $\mathbf{V}$. Let $\widehat{G}_\sigma$ be an arbitrary DAG on $\mathbf{V}$ that maximizes BDeu among all DAGs consistent with σ. Let $\mathcal{G}_\sigma^{\min}(\mathbf{V})$ be a set of all I-maps on $\mathbf{V}$ that minimizes NCP among all I-maps consistent with σ. Then*

$$\lim_{N \to \infty} P\left(\widehat{G}_\sigma \ \in\ \mathcal{G}_\sigma^{\min}(\mathbf{V})\right) = 1. \tag{22}$$

Proof Let $\mathcal{H}_\sigma(\mathbf{V})$ be the set of I-maps on $\mathbf{V}$ that minimizes the number of all the parameters among all I-maps consistent with σ. From the asymptotic consistency of BDeu (Chickering, 2002) described in Definition 5,

$$\lim_{N \to \infty} P\left(\widehat{G}_\sigma \in \mathcal{H}_\sigma(\mathbf{V})\right) = 1. \tag{23}$$

Therefore, it is enough to show that

$$\mathcal{H}_\sigma(\mathbf{V}) \subseteq \mathcal{G}_\sigma^{\min}(\mathbf{V}). \tag{24}$$

Let $H = (\mathbf{V}, \mathbf{E}_H)$ be an arbitrary DAG in $\mathcal{H}_\sigma(\mathbf{V})$, where $\mathbf{E}_H$ is the set of edges in H. Then (24) is equivalent to

$$H \in \mathcal{G}_\sigma^{\min}(\mathbf{V}) \tag{25}$$

Let $\mathcal{G}_\sigma(\mathbf{V})$ denote a set of all the I-maps on $\mathbf{V}$ consistent with σ. Then, a sufficient condition for (25) to hold is

$$\forall G \in \mathcal{G}_\sigma(\mathbf{V}),\ \mathbf{E}_H \subseteq \mathbf{E}_G, \tag{26}$$

where $\mathbf{E}_G$ denotes the set of edges of G. We prove that (26) is true by contradiction. First, we assume that there exists $G' = (\mathbf{V}, \mathbf{E}_{G'}) \in \mathcal{G}_\sigma(\mathbf{V})$ such that $\mathbf{E}_H \nsubseteq \mathbf{E}_{G'}$. This assumption engenders that there exist distinct variables $X, Y \in \mathbf{V}$ such that

$$(Y \to X) \in \mathbf{E}_H \text{ and } (Y \to X) \notin \mathbf{E}_{G'}.$$

Letting $\mathbf{A} = \Pi_X^H \setminus \{Y\}$, then we obtain

$$X \not\perp\!\!\!\perp Y \mid \mathbf{A} \tag{27}$$

from $(Y \to X) \in \mathbf{E}_H$ and the asymptotic local consistency of BDeu (Chickering, 2002) described in Definition 6. Let $\mathbf{B}$ represent a set of variables $\mathbf{Pre}_X^\sigma \setminus \Pi_X^H$. From the contrapositive of Lemma 1, we obtain

$$X \not\perp\!\!\!\perp Y \mid \mathbf{A} \Rightarrow X \not\perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, Y) \vee X \not\perp\!\!\!\perp Y \mid (\mathbf{A}, \mathbf{B}). \tag{28}$$

From (27) and (28), $X \not\perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, Y) \vee X \not\perp\!\!\!\perp Y \mid (\mathbf{A}, \mathbf{B})$ holds, i.e., $X \perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, Y) \Rightarrow X \not\perp\!\!\!\perp Y \mid (\mathbf{A}, \mathbf{B})$ holds. Because $X \perp\!\!\!\perp \mathbf{B} \mid (\mathbf{A}, Y)$ holds from the local independences (described in Proposition 1) in H, we obtain

$$X \not\perp\!\!\!\perp Y \mid (\mathbf{A}, \mathbf{B}). \tag{29}$$

However, $(X \perp_d^{G'} Y \mid (\mathbf{A}, \mathbf{B}))$ holds because X and Y are not adjacent in G' and because no variable in $\mathbf{A} \cup \mathbf{B}$ is a descendant of both X and Y in G'. This result contradicts (29). Therefore no such G' exists, and (26) holds. Consequently (24) follows, and combining (23) and (24) proves (22). ■

In the remainder of this appendix, we prove Theorems 2–4.

Theorem 2 *Let G^* be a BNOC structure on $\mathbf{V}$. Also, let G^{NB} be the naive Bayes classifier consisting of a set of feature variables $\mathbf{V}_c$, which are children of the class variable in G^*. Then,* $\mathrm{NCP}(G^{\mathrm{NB}}) \leq \mathrm{NCP}(G^*)$ *holds.*

Proof Because the parent of feature variables in $G^{\mathrm{NB}}(\mathbf{V}_c)$ is only X_0, we obtain

$$\mathrm{NCP}(G^{\mathrm{NB}}(\mathbf{V}_c)) = \sum_{X_i \in \mathbf{V}_c} \mathrm{NCP}_i(\{X_0\}) + r_0 - 1,$$

where $\mathrm{NCP}_i(\{X_0\}) = (r_i - 1)r_0$. Because $X_0 \in \Pi_X^{G^*}$ for all $X \in \mathbf{V}_c$, we obtain

$$\forall X_i \in \mathbf{V}_c, \mathrm{NCP}_i(\{X_0\}) \leq \mathrm{NCP}_i(\Pi_{X_i}^{G^*}).$$

Consequently, we obtain

$$\begin{aligned}
\mathrm{NCP}(G^{\mathrm{NB}}(\mathbf{V}_c)) &= \sum_{X_i \in \mathbf{V}_c} \mathrm{NCP}_i(\{X_0\}) + r_0 - 1 \\
&\leq \sum_{X_i \in \mathbf{V}_c} \mathrm{NCP}_i(\Pi_{X_i}^{G^*}) + r_0 - 1 \\
&= \mathrm{NCP}(G^*).
\end{aligned}$$

■

Theorem 3 *h^* has consistency.*

Proof For any pair of nodes $(\mathbf{U}, \mathbf{R})$ in which $\mathbf{R}$ has an incoming edge from $\mathbf{U}$ in a CROG, let $c(\mathbf{U}, \mathbf{R})$ represent the cost of the edge from $\mathbf{U}$ to $\mathbf{R}$. Moreover, let X_j be a variable included in $\mathbf{U} \setminus \mathbf{R}$. When $X_j \notin \mathbf{V}_c$, we obtain

$$\begin{aligned}
h^*(\mathbf{U}) &= \sum_{X_i \in (\mathbf{U} \cap \mathbf{V}_c)} \mathrm{NCP}_i(\{X_0\}) \\
&= \sum_{X_i \in (\mathbf{R} \cap \mathbf{V}_c)} \mathrm{NCP}_i(\{X_0\}) \\
&\leq \sum_{X_i \in (\mathbf{R} \cap \mathbf{V}_c)} \mathrm{NCP}_i(\{X_0\}) + \mathrm{NCP}_j(\widehat{\Pi}_{X_j}(\mathbf{U} \setminus \{X_j\})) \\
&= h^*(\mathbf{R}) + c(\mathbf{U}, \mathbf{R}).
\end{aligned}$$

When $X_j \in \mathbf{V}_c$, the following equation holds using $X_0 \in \widehat{\Pi}_{X_j}(\mathbf{U} \setminus \{X_j\})$.

$$\begin{aligned}
h^*(\mathbf{U}) &= \sum_{X_i \in (\mathbf{U} \cap \mathbf{V}_c)} \mathrm{NCP}_i(\{X_0\}) \\
&= \sum_{X_i \in (\mathbf{U} \cap \mathbf{V}_c) \setminus \{X_j\}} \mathrm{NCP}_i(\{X_0\}) + \mathrm{NCP}_j(\{X_0\}) \\
&= \sum_{X_i \in (\mathbf{R} \cap \mathbf{V}_c)} \mathrm{NCP}_i(\{X_0\}) + \mathrm{NCP}_j(\{X_0\}) \\
&\leq \sum_{X_i \in (\mathbf{R} \cap \mathbf{V}_c)} \mathrm{NCP}_i(\{X_0\}) + \mathrm{NCP}_j(\widehat{\Pi}_{X_j}(\mathbf{U} \setminus \{X_j\})) \\
&= h^*(\mathbf{R}) + c(\mathbf{U}, \mathbf{R}).
\end{aligned}$$

Consequently, we obtain

$$h^*(\mathbf{U}) \leq h^*(\mathbf{R}) + c(\mathbf{U}, \mathbf{R}).$$

■

Theorem 4 *Let $\mathcal{G}_{\mathrm{CR}}(\mathbf{V})$ denote the set of all CRDAGs on $\mathbf{V}$, and let $\mathcal{G}^{\min}_{\mathrm{CR}}(\mathbf{V})$ denote the set of all I-map CRDAGs on $\mathbf{V}$ with the fewest NCP. Then the following holds:*

$$\forall G^* \in \mathcal{G}^{\min}_{\mathrm{CR}}(\mathbf{V}),\ \exists c > 0,\ \forall G \in \mathcal{G}_{\mathrm{CR}}(\mathbf{V}) \setminus \mathcal{G}^{\min}_{\mathrm{CR}}(\mathbf{V}),$$
$$\lim_{N\to\infty} P(\text{BDeu-NCP}(G^*, D, \gamma_N) > \text{BDeu-NCP}(G, D, \gamma_N)) = 1, \qquad (30)$$

where $\gamma_N = c \log N$.

Proof We will prove that (30) holds for the $\gamma_N = c \log N$ with the following positive constant c:

$$c \;=\; \frac{1}{2}\left(1 + \sum_{i=0}^{n}(r_i - 1)\left(q(\Pi^{G^*}_{X_i}) - q(\Pi^{G'}_{X_i})\right)\right), \qquad (31)$$

where G' is an arbitrary I-map CRDAG with the fewest number of all the parameters. We split the proof of (30) into two cases according to whether G is an I-map.

(1) G is not an I-map.

For a large dataset, from Theorem 1 of Ueno (2010), the BDeu-NCP score can be approximated by:

$$\begin{aligned}
\text{BDeu-NCP}(G, D, \gamma_N) = &\sum_{i=0}^{n}\sum_{j=1}^{q(\Pi^G_{X_i})}\sum_{k=1}^{r_i}\left(\frac{N'}{r_i\, q(\Pi^G_{X_i})} + N_{X_i=k,\Pi^G_{X_i}=j}\right)\\
&\times \log\frac{\left(\dfrac{N'}{r_i\, q(\Pi^G_{X_i})} + N_{X_i=k,\Pi^G_{X_i}=j}\right)}{\left(\dfrac{N'}{q(\Pi^G_{X_i})} + N_{\Pi^G_{X_i}=j}\right)}\\
&- \frac{1}{2}\sum_{i=0}^{n}\sum_{j=1}^{q(\Pi^G_{X_i})}\sum_{k=1}^{r_i}\frac{r_i - 1}{r_i}\log\left(1 + \frac{r_i\, q(\Pi^G_{X_i})\, N_{X_i=k,\Pi^G_{X_i}=j}}{N'}\right)\\
&- \gamma_N\, \mathrm{NCP}(G) + O(1), \qquad (32)
\end{aligned}$$

where N' is a positive constant. Let $f_1(G, D, \gamma_N)$ denote the first term of (32) as follows:

$$f_1(G, D, \gamma_N) = \sum_{i=0}^{n}\sum_{j=1}^{q(\Pi^G_{X_i})}\sum_{k=1}^{r_i}\left(\frac{N'}{r_i\, q(\Pi^G_{X_i})} + N_{X_i=k,\Pi^G_{X_i}=j}\right)\log\frac{\left(\dfrac{N'}{r_i\, q(\Pi^G_{X_i})} + N_{X_i=k,\Pi^G_{X_i}=j}\right)}{\left(\dfrac{N'}{q(\Pi^G_{X_i})} + N_{\Pi^G_{X_i}=j}\right)}$$

The logarithm's argument in $f_1(G, D, \gamma_N)$ equals the EAP estimator (see (1)) with $\alpha_{X_i=k|\Pi^G_{X_i}} = N'/(r_i q(\Pi^G_{X_i}))$ as follows:

$$\frac{\left(\dfrac{N'}{r_i\, q(\Pi^G_{X_i})} + N_{X_i=k,\Pi^G_{X_i}=j}\right)}{\left(\dfrac{N'}{q(\Pi^G_{X_i})} + N_{\Pi^G_{X_i}=j}\right)} = \hat{\theta}^{\mathrm{EAP}(D)}_{X_i=k|\Pi^G_{X_i}=j}$$

Therefore, we have

$$\begin{aligned} f_1(G, D, \gamma_N) &= \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} \left(\frac{N'}{r_i \, q(\Pi_{X_i}^G)} + N_{X_i=k, \Pi_{X_i}^G=j} \right) \log \hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\mathrm{EAP}(D)} \\ &= \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} \left(\frac{N'}{r_i \, q(\Pi_{X_i}^G)} \right) \log \hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\mathrm{EAP}(D)} \\ &\quad + \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} N_{X_i=k, \Pi_{X_i}^G=j} \, \log \hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\mathrm{EAP}(D)}. \end{aligned} \tag{33}$$

In the first term of (33), $N'/(r_i q(\Pi_{X_i}^G))$ does not depend on N and $\hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\mathrm{EAP}(D)} \in (0, 1)$ for all N, hence that term is $O(1)$. Therefore,

$$f_1(G, D, \gamma_N) = \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} N_{X_i=k, \Pi_{X_i}^G=j} \, \log \hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\mathrm{EAP}(D)} + O(1). \tag{34}$$

Let B be the BN with structure G and EAP-estimated parameters. Using the empirical distribution $\hat{P}(X_i = k, \Pi_{X_i}^G = j) = N_{X_i=k, \Pi_{X_i}^G=j}/N$,

$$\begin{aligned} f_1(G, D, \gamma_N) &= \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} N \cdot \frac{N_{X_i=k, \Pi_{X_i}^G=j}}{N} \cdot \log \hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\mathrm{EAP}(D)} + O(1) \\ &= N \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} \hat{P}(X_i = k, \Pi_{X_i}^G = j) \, \log \hat{\theta}_{X_i=k|\Pi_{X_i}^G=j}^{\mathrm{EAP}(D)} + O(1) \\ &= -N \cdot H\big(\hat{P}(\mathbf{V}), P_B(\mathbf{V})\big) + O(1), \end{aligned} \tag{35}$$

where $H(\hat{P}(\mathbf{V}), P_B(\mathbf{V}))$ is the cross-entropy between the empirical distribution $\hat{P}(\mathbf{V})$ and the model distribution $P_B(\mathbf{V})$ induced by B. Let B^* denotes the BN with structure G^* and EAP-estimated parameters. Similarly, we have $f_1(G^*, D, \gamma_N) = -N \cdot H(\hat{P}(\mathbf{V}), P_{B^*}(\mathbf{V})) + O(1)$. Hence,

$$f_1(G^*, D, \gamma_N) - f_1(G, D, \gamma_N) = N \left(H(\hat{P}(\mathbf{V}), P_B(\mathbf{V})) - H(\hat{P}(\mathbf{V}), P_{B^*}(\mathbf{V})) \right) + O(1).$$

Let $f_2(G, D, \gamma_N)$ denote the second term in (32) as follows:

$$f_2(G, D, \gamma_N) = -\frac{1}{2} \sum_{i=0}^{n} \sum_{j=1}^{q(\Pi_{X_i}^G)} \sum_{k=1}^{r_i} \frac{r_i - 1}{r_i} \, \log \left(1 + \frac{r_i \, q(\Pi_{X_i}^G) \, N_{X_i=k, \Pi_{X_i}^G=j}}{N'} \right).$$

With this notation, the difference in BDeu-NCP scores can be written as

$$\begin{aligned}\text{BDeu-NCP}(G^*,\gamma_N) - \text{BDeu-NCP}(G,\gamma_N) &= f_1(G^*,D,\gamma_N) - f_1(G,D,\gamma_N)\\ &\quad + f_2(G^*,D,\gamma_N) - f_2(G,D,\gamma_N) - \gamma_N\big(\text{NCP}(G^*) - \text{NCP}(G)\big) + O(1)\\ &= N\Big(H(\hat{P}(\mathbf{V}),P_B(\mathbf{V})) - H(\hat{P}(\mathbf{V}),P_{B^*}(\mathbf{V}))\Big)\\ &\quad + \big(f_2(G^*,D,\gamma_N) - f_2(G,D,\gamma_N)\big) - \gamma_N\big(\text{NCP}(G^*) - \text{NCP}(G)\big) + O(1).\end{aligned} \tag{36}$$

In (36), both $f_2(G^*,\gamma_N)$ and $f_2(G,\gamma_N)$ are clearly $O(\log N)$, and we also have $\gamma_N = O(\log N)$ by $\gamma_N = c\log N$. Hence, for sufficiently large N, the difference in BDeu-NCP scores is dominated by $N\big(H(\hat{P}(\mathbf{V}),P_B(\mathbf{V})) - H(\hat{P}(\mathbf{V}),P_{B^*}(\mathbf{V}))\big)$. If this quantity is positive, then the score difference in (36) is positive. As $N\to\infty$, $\hat{P}(\mathbf{V})$ converges to the true distribution $P^{\text{True}}(\mathbf{V})$. Moreover, since G^* is an I-map, $P_{B^*}(\mathbf{V})$ converges to $P^{\text{True}}(\mathbf{V})$ as well. On the other hand, because G is not an I-map, $P_B(\mathbf{V})$ converges to a distribution $P'(\mathbf{V})$ different from $P^{\text{True}}(\mathbf{V})$. Therefore,

$$\begin{aligned}\lim_{N\to\infty} H(\hat{P}(\mathbf{V}),P_B(\mathbf{V})) - H(\hat{P}(\mathbf{V}),P_{B^*}(\mathbf{V})) &= H(P^{\text{True}}(\mathbf{V}),P'(\mathbf{V})) - H(P^{\text{True}}(\mathbf{V}),P^{\text{True}}(\mathbf{V}))\\ &= D_{\text{KL}}\big(P^{\text{True}}(\mathbf{V})\,\|\,P'(\mathbf{V})\big)\\ &> 0 \quad (\because P^{\text{True}}(\mathbf{V}) \neq P'(\mathbf{V})),\end{aligned}$$

where D_{KL} is the Kullback–Leibler divergence. Therefore, $\lim_{N\to\infty} P(H(P^{\text{True}}(\mathbf{V}),P_B(\mathbf{V})) - H(P^{\text{True}}(\mathbf{V}),P^{\text{True}}(\mathbf{V})) > 0) = 1$. Consequently,

$$\lim_{N\to\infty} P(\text{BDeu-NCP}(G^*,D,\gamma_N) > \text{BDeu-NCP}(G,D,\gamma_N)) = 1.$$

(2) G is an I-map.
As $N\to\infty$, $P_B(\mathbf{V})$ converge to the true distribution $P^{\text{True}}(\mathbf{V})$, hence

$$\begin{aligned}\lim_{N\to\infty} H(\hat{P}(\mathbf{V}),P_B(\mathbf{V})) - H(\hat{P}(\mathbf{V}),P_{B^*}(\mathbf{V})) &= H(P^{\text{True}}(\mathbf{V}),P^{\text{True}}(\mathbf{V})) - H(P^{\text{True}}(\mathbf{V}),P^{\text{True}}(\mathbf{V}))\\ &= 0.\end{aligned}$$

Therefore, for sufficiently large N, $f_1(G^*,D,\gamma_N) - f_1(G,D,\gamma_N)$ can be ignored in (36) (Koller and Friedman, 2009). Therefore, we obtain

$$\text{BDeu-NCP}(G^*,\gamma_N) - \text{BDeu-NCP}(G,\gamma_N) = f_2(G^*,\gamma_N) - f_2(G,\gamma_N) - \gamma_N\big(\text{NCP}(G^*) - \text{NCP}(G)\big) + O_p(1).$$

We expand $f_2(G,D,\gamma_N)$ as

$$\begin{aligned}f_2(G,D,\gamma_N) &= -\frac{1}{2}\sum_{i=0}^{n}\sum_{j=1}^{q(\Pi_{X_i}^G)}\sum_{k=1}^{r_i}\frac{r_i-1}{r_i}\log\left(1+\frac{r_i\,q(\Pi_{X_i}^G)\,N_{X_i=k,\Pi_{X_i}^G=j}}{N'}\right)\\ &= -\frac{1}{2}\sum_{i=0}^{n}\frac{r_i-1}{r_i}\sum_{k=1}^{r_i}\sum_{j=1}^{q(\Pi_{X_i}^G)}\log\left(1+\frac{r_i\,q(\Pi_{X_i}^G)\,N_{X_i=k,\Pi_{X_i}^G=j}}{N'}\right)\\ &= -\frac{1}{2}\sum_{i=0}^{n}\frac{r_i-1}{r_i}\sum_{k=1}^{r_i}\sum_{j=1}^{q(\Pi_{X_i}^G)}\log\Big(N'+r_i\,q(\Pi_{X_i}^G)\,N_{X_i=k,\Pi_{X_i}^G=j}\Big) + O(1).\end{aligned}$$

To simplify notation, for fixed i and k we define

$$g_{ik}(G, D, \gamma_N) = \sum_{j=1}^{q(\Pi_{X_i}^G)} \log\left(N' + r_i\, q(\Pi_{X_i}^G)\, N_{X_i=k,\Pi_{X_i}^G=j}\right).$$

We have

$$\lim_{N\to\infty} P(N_{X_i=k,\Pi_{X_i}^G=j} > 0) = 1 \quad \text{for all } i \in \{0,\ldots,n\}, j \in \{1,\ldots,q(\Pi_{X_i}^G)\}, k \in \{1,\ldots,r_i\}.$$

Therefore, for sufficiently large N, we can expand $g_{ik}(G, D, \gamma_N)$ as

$$g_{ik}(G, D, \gamma_N) = \sum_{j=1}^{q(\Pi_{X_i}^G)} \left(\log\big(r_i\, q(\Pi_{X_i}^G)\big) + \log N_{X_i=k,\Pi_{X_i}^G=j} + \log\left(1 + \frac{N'}{r_i\, q(\Pi_{X_i}^G)\, N_{X_i=k,\Pi_{X_i}^G=j}}\right)\right). \tag{37}$$

Inside the sum in (37), the first term is constant for N and the third term is uniformly bounded in N (e.g., by $\log(1 + N')$), hence

$$g_{ik}(G, D, \gamma_N) = \sum_{j=1}^{q(\Pi_{X_i}^G)} \log N_{X_i=k,\Pi_{X_i}^G=j} + O(1).$$

Because $N_{X_i=k,\Pi_{X_i}^G=j} = N \cdot \hat{P}(X_i = k, \Pi_{X_i}^G = j)$,

$$\begin{aligned} g_{ik}(G, D, \gamma_N) &= \sum_{j=1}^{q(\Pi_{X_i}^G)} \log\big(N \cdot \hat{P}(X_i = k, \Pi_{X_i}^G = j)\big) + O(1) \\ &= \sum_{j=1}^{q(\Pi_{X_i}^G)} \log N \;+\; \sum_{j=1}^{q(\Pi_{X_i}^G)} \log \hat{P}(X_i = k, \Pi_{X_i}^G = j) + O(1) \\ &= q(\Pi_{X_i}^G) \log N \;+\; \sum_{j=1}^{q(\Pi_{X_i}^G)} \log \hat{P}(X_i = k, \Pi_{X_i}^G = j) + O(1). \end{aligned} \tag{38}$$

Since the second sum in (38) converges to the constant $\sum_{j=1}^{q(\Pi_{X_i}^G)} \log P^{\text{True}}(X_i = k, \Pi_{X_i}^G = j)$ as $N \to \infty$, we have

$$g_{ik}(G, D, \gamma_N) = q(\Pi_{X_i}^G) \log N + O(1). \tag{39}$$

Therefore,

$$\begin{aligned} f_2(G, D, \gamma_N) &= -\frac{1}{2}\sum_{i=0}^{n}\frac{r_i - 1}{r_i}\sum_{k=1}^{r_i} g_{ik}(G, \gamma_N) + O(1) \\ &= -\frac{1}{2}\sum_{i=0}^{n}\frac{r_i - 1}{r_i}\sum_{k=1}^{r_i}\big(q(\Pi_{X_i}^G)\log N + O(1)\big) + O(1) \\ &= -\frac{1}{2}\sum_{i=0}^{n}\frac{r_i - 1}{r_i}\sum_{k=1}^{r_i} q(\Pi_{X_i}^G)\log N + O(1) \\ &= -\frac{\log N}{2}\sum_{i=0}^{n}(r_i - 1)q(\Pi_{X_i}^G) + O(1). \end{aligned}$$

Similarly,

$$f_2(G^*, D, \gamma_N) = -\frac{\log N}{2}\sum_{i=0}^{n}(r_i - 1)q(\Pi_{X_i}^{G^*}) + O(1),$$

and thus

$$\begin{aligned} &\text{BDeu-NCP}(G^*, D, \gamma_N) - \text{BDeu-NCP}(G, D, \gamma_N) \\ &= f_2(G^*, D, \gamma_N) - f_2(G, D, \gamma_N) - \gamma_N\big(\text{NCP}(G^*) - \text{NCP}(G)\big) + O_p(1) \\ &= -\frac{\log N}{2}\sum_{i=0}^{n}(r_i - 1)\big(q(\Pi_{X_i}^{G^*}) - q(\Pi_{X_i}^G)\big) + \gamma_N\big(\text{NCP}(G) - \text{NCP}(G^*)\big) + O_p(1). \end{aligned} \tag{40}$$

Here, since G^* is an I-map CRDAG that minimizes NCP while G is an I-map CRDAG whose NCP is not minimal, we have $\text{NCP}(G) - \text{NCP}(G^*) \geq 1$. Hence, by (31),

$$\gamma_N\big(\text{NCP}(G) - \text{NCP}(G^*)\big) \geq \gamma_N = \frac{\log N}{2}\left(1 + \sum_{i=0}^{n}(r_i - 1)\left(q(\Pi_{X_i}^{G^*}) - q(\Pi_{X_i}^{G'})\right)\right). \tag{41}$$

From (40) and (41), we obtain

$$\begin{aligned} &\text{BDeu-NCP}(G^*, D, \gamma_N) - \text{BDeu-NCP}(G, D, \gamma_N) \\ \geq\ &-\frac{\log N}{2}\sum_{i=0}^{n}(r_i - 1)\big(q(\Pi_{X_i}^{G^*}) - q(\Pi_{X_i}^G)\big) \\ &+ \frac{\log N}{2}\left(1 + \sum_{i=0}^{n}(r_i - 1)\left(q(\Pi_{X_i}^{G^*}) - q(\Pi_{X_i}^{G'})\right)\right) \ +\ O_p(1) \\ =\ &\frac{\log N}{2}\left(1 + \sum_{i=0}^{n}(r_i - 1)q(\Pi_{X_i}^G) - \sum_{i=0}^{n}(r_i - 1)q(\Pi_{X_i}^{G'})\right) \ +\ O_p(1). \end{aligned} \tag{42}$$

Here, since G' has fewer parameters than G,

$$\sum_{i=0}^{n}(r_i - 1)q(\Pi_{X_i}^G) - \sum_{i=0}^{n}(r_i - 1)q(\Pi_{X_i}^{G'}) \ \geq\ 0. \tag{43}$$

From (42) and (43),

$$\text{BDeu-NCP}(G^*, D, \gamma_N) - \text{BDeu-NCP}(G, D, \gamma_N) \geq \frac{\log N}{2} + O_p(1). \tag{44}$$

The right-hand side of (44) is positive almost surely for all sufficiently large N. Consequently,

$$\lim_{N\to\infty} P(\text{BDeu-NCP}(G^*, D, \gamma_N) \ > \ \text{BDeu-NCP}(G, D, \gamma_N)) = 1$$

■

References

M. Bartlett and J. Cussens. Advances in Bayesian Network Learning Using Integer Programming. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 182–191, 2013.

W. Buntine. Theory Refinement on Bayesian Networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

A. M. Carvalho, P. Adão, and P. Mateus. Efficient Approximation of the Conditional Relative Entropy with Applications to Discriminative Learning of Bayesian Network Classifiers. *Entropy*, 15:2716–2735, 2013.

D. M. Chickering. *Learning Bayesian Networks is NP-Complete*, pages 121–130. Springer, 1996.

D. M. Chickering. Optimal Structure Identification With Greedy Search. *JMLR*, 3:507–554, 2002.

J. Cussens. Bayesian network learning with cutting planes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 153–160, 2012.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7:1–30, 2006.

Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29:131–163, 1997.

R. Greiner and W. Zhou. Structural Extension to Logistic Regression: Discriminative Parameter Learning of Belief Net Classifiers. In *Proceedings of the National Conference on Artificial Intelligence*, pages 167–173, 2002.

D. Grossman and P. Domingos. Learning Bayesian Network classifiers by maximizing conditional likelihood. In *Proceedings of the International Conference on Machine Learning*, pages 361–368, 2004.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20:197–243, 1995.

G. Hommel. A stagewise rejective multiple test procedure based on a modified bonferroni test. *Biometrika*, pages 383–386, 1988.

Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian Network Structure using LP Relaxations. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 358–365, 2010.

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

Z. A. Liao, C. Sharma, J. Cussens, and P. van Beek. Finding all Bayesian network structures within a factor of optimal. In *Proceedings of the AAAI Conference on Artifcial Intelligence*, 2019.

M. Lichman. UCI Machine Learning Repository, 2013. URL `http://archive.ics.uci.edu/ml`.

C. X. Ling and H. Zhang. The Representational Power of Discrete Bayesian Networks. *JMLR*, 3:709–721, 2003.

P. J. F. Lucas. Restricted bayesian network structure learning. 146:217–234, 2004.

B. Malone, C. Yuan, E. A. Hansen, and S. Bridges. Improving the Scalability of Optimal Bayesian Network Learning with External-Memory Frontier Breadth-First Branch and Bound Search. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 479–488, 2011.

Brandon Malone and Changhe Yuan. A Depth-First Branch and Bound Algorithm for Learning Optimal Bayesian Networks. In Madalina Croitoru, Sebastian Rudolph, Stefan Woltran, and Christophe Gonzales, editors, *Graph Structures for Knowledge Representation and Reasoning*, pages 111–122. Springer International Publishing, 2014. ISBN 978-3-319-04534-4.

B. Mihaljević, C. Bielza, and P. Larrañaga. Learning Bayesian network classifiers with completed partially directed acyclic graphs. In *Proceedings of the International Conference on Probabilistic Graphical Models*, pages 272–283, 2018.

M. Minsky. Steps toward Artificial Intelligence. In *Proceedings of the IRE*, pages 8–30, 1961.

J. Pearl. *Models, Reasoning, and Inference*. Cambridge University Press, 2000.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 1558604790.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On Discriminative Bayesian Network Classifiers and Logistic Regression. *Machine Learning*, 59:267–296, 2005.

G. Schwarz. Estimating the Dimension of a Model. *Annals of Statistics*, 6:461–464, 1978.

M. Scutari. Learning Bayesian Networks with the bnlearn R Package. *JSSA*, 35:1–22, 2010.

T. Silander and P. Myllymäki. A Simple Approach for Finding the Globally Optimal Bayesian Network Structure. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 445–452, 2006.

S. Sugahara and M. Ueno. Exact learning augmented naive bayes classifier. *Entropy*, 23, 2021.

S. Sugahara, M. Uto, and M. Ueno. Exact learning augmented naive Bayes classifier. In *Proceedings of the International Conference on Probabilistic Graphical Models*, pages 439–450, 2018.

Shouta Sugahara, Koya Kato, and Maomi Ueno. Learning Bayesian Network Classifiers to Minimize the Class Variable Parameters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38:20540–20549, 2024.

M. Ueno. Learning Networks Determined by the Ratio of Prior and Data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 598–605, 2010.

M. Ueno. Robust learning Bayesian networks for prior belief. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 689–707, 2011.

C. Yuan, H. Lim, and T. Lu. Most Relevant Explanation in Bayesian Networks. *JAIR*, 42: 309–352, 2011.