Transformer Based Bayesian Network Structure Learning from an Information Theory Perspective

Zhiwei Qi[1,2] **Kun Yue**[1,3] **Zhu Yang**[1,3] **Jiahui Wang**[*1,3]

[1]Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, Yunnan, China
[2]School of Education, Yunnan University, Kunming, Yunnan, China
[3]School of Information Science and Engineering, Yunnan University, Kunming, Yunnan, China

Abstract

The pivotal challenge of handling uncertainty in artificial intelligence has prompted a growing focus on learning Bayesian network (BN) structures from data in recent years. Nevertheless, most of the existing methods such as hill-climbing, variational auto-encoder and graph neural network-based algorithms continue to encounter the issues like local optimum and exponential computational complexity, etc. In response to these challenges, we propose a Transformer-based approach for learning the BN structure (T-BNSL) from an information theory perspective. Specifically, we first establish the graph skeleton of a BN by employing conditional independence tests grounded in mutual information. Subsequently, we propose a hill-climbing method with decomposable BIC scoring function to generate potential directed acyclic graphs (DAGs) and evaluate the BIC scores of a subset of these DAGs. Lastly, we use the Transformer framework with a refined attention module to predict the BIC scores of the remaining DAGs, enabling the efficient identification of the DAG with the highest score. Experimental findings demonstrate that our approach significantly surpasses state-of-the-art competitors in terms of accuracy and efficiency by several orders of magnitude when learning the BN structure.

1 INTRODUCTION

Bayesian network (BN) is a directed acyclic graph (DAG) representing dependence relations among random variables with conditional probability tables (CPTs), and has been used in decision making, medical diagnosis, and molecular biology [Pearl, 1986, Yue et al., 2024]. Before a BN is used in these domains, it is necessary to construct the DAG structure in advance by experts or learned from data, which has been paid growing attention in the paradigm of uncertainty in artificial intelligence [Wu et al., 2024, Harviainen et al., 2025]. In this paper, we focus on the latter, which could model uncertain systems automatically by large collections of variables with probabilistic relationships and recover the underlying graphical structure of these systems.

In general, the methods of learning BN structure from data fall into three classes. The first class, constraint-based (CB) methods add, delete and orient edges by a series of conditional independence (CI) tests [Marella and Vicard, 2022]. The second class, score-and-search (SS) methods aim to search over different DAGs for a preferred DAG by maximizing a scoring function [Heckerman et al., 1995, Trösser et al., 2021]. Hybrid methods, as the third class, have gradually become the mainstream of BN structure learning, including CB+SS based and graph neural network (GNN) based methods, etc. Nevertheless, hybrid methods possibly output locally optimum DAGs [Xu et al., 2025b]. GNN based methods have limited expressivity, since the DAG attention mechanism only captures the partial order of DAG like the predecessors A, B, C of D in Figure 1 (a), and ignores the unique information of BN structure, like asymmetry of node degree, partial order of predecessors and successors, and transmission of information flow. Therefore, it is worthwhile to develop a more expressive model tailored for particular features of DAG for BN structure learning.

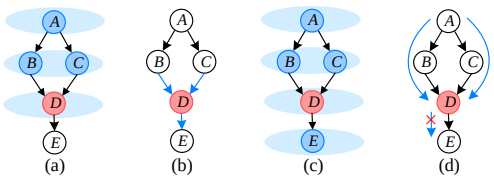

Figure 1: (a) partial order of DAG captured by GNN based methods; (b) - (c) node degrees, partial order and information flow of DAG captured by our Transformer based model.

From an information theory perspective, a BN could be

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:5541–5555.

viewed as a network of information channels, while BN structure learning could be viewed as an optimization problem of trading off structure complexity and mutual information maximization of parent-child node pairs [Koller and Friedman, 2009]. Consequently, we solve the following two critical challenges for finding a globally optimal DAG: (1) how to restrict the search space of DAGs efficiently, and (2) how to maximize the mutual information of each node pair.

To restrict the search space of possible DAGs, we first improve the method for mutual information based CI test to construct a complete partial DAG (CPDAG), by proposing four processes, including drafting, thickening, thinning and orienting processes. To calculate the DAG scores efficiently, we propose a decomposable Bayesian information criterion (BIC) scoring function based on the mutual information of parent-child nodes. Then, we propose a hill-climbing (HC) method to generate candidate DAGs by orienting undirected edges in the CPDAG, and evaluate the BIC scores of a subset of candidate DAGs.

To maximize the mutual information of node pairs in the remaining DAGs, we consider the features in DAG as shown in Figure 1 (b)$\sim$(d), including asymmetry of node degree, partial order of the relations and transmission of information flow compared with the regular graphs. Note that the partial order of each DAG is analogous to the order of words in sentences [Assadi et al., 2025]. This motivates us establish the Transformer model [Xu et al., 2025a] by incorporating the effective encodings of graph attributes and structures in BN. To predict the BIC scores of remaining DAGs efficiently, we design a novel Transformer with tailored attention mechanism to avoid the time-consuming BIC scoring in exponential search space and being trapped in locally optimum.

To summarize, our main contributions are:

- We convert the score-and-search based BN structure learning to a problem of predicting BIC scores by maximizing the mutual information of each node pair.
- We propose the method for mutual information based CI tests to construct a CPDAG, on which we propose the HC method with a decomposable BIC scoring function to generate candidate DAGs with partial scores.
- We propose the attention mechanism of Transformer tailored for the features of node and relation in BN to predict BIC scores of the remaining DAGs.
- We conduct extensive experiments and demonstrate that our method significantly surpasses state-of-the-art competitors in terms of accuracy and efficiency.

2 RELATED WORK

Constraint-based (CB) methods use CI tests to determine the CI relationships between variables, and construct a DAG fitting to data. Spirtes-Glymour-Scheines (SGS) method learns the Markov equivalence class from a complete undirected graph by adjacency, v-structure and orientation propagation [Kitson et al., 2023]. Peter and Clark (PC) method building upon SGS, decides the v-structure by using the sepsets identified in the adjacency phase [Spirtes and Glymour, 1991]. PC-MAX adopts the strategies used in PC-Stable to avoid sensitivity to the order of node processing [Ramsey, 2016]. Unfortunately, these methods can be sensitive to the failures in individual independence tests.

SS methods define a scoring function to evaluate candidate DAGs and then search for a highest-scoring DAG. The HC method explores all neighboring DAGs of the current DAG by adding, removing and reversing an edge in the current DAG, and returns a high-scoring DAG [Heckerman et al., 1995]. The DAG-GFlowNet [Deleu et al., 2022] and GFlowOpt [Yang et al., 2025] methods, approximate the posterior distribution over DAGs with corresponding scores, and find a high-scoring DAG. However, these methods concern the time-consuming scoring process for $2^{O(n^2)}$ candidate DAGs.

Hybrid methods usually combine CB and SS to offer the best characteristics of each. The most common way is to use a CB method to restrict the search space, in which a subsequent SS method finds a DAG with a locally or globally maximal score, like mutual information-guided genetic algorithm (MGA) method [Yan et al., 2022]. The DAG neural network (DAGNN) embeds the nodes to a low-dimensional vector space by using GNN and optimizes the BIC score over the latent space of DAGs [Thost and Chen, 2021]. Nevertheless, MGA easily gets stuck on a locally optimal score, and DAGNN has limited expressivity.

3 PRELIMINARY

From an information theory perspective, a discrete BN $\mathcal{B}$ could be viewed as a network of information channels, where each node viewed as a valve is either active or inactive, the edges as information channels can connect the valves, and the mutual information of two nodes can measure the volume of their information [Salmani and Katoen, 2023], defined as

Definition 1. *A discrete BN $\mathcal{B} = (G, \theta)$ is a DAG $G = \left(X, \bar{E}\right)$ with CPTs θ, satisfying: (1) The set of nodes $X = \{X_i\}_{i=1}^{n}$ represents random variables. Each node has a CPT $\theta_i \in \theta$ $(i = 1, 2, \cdots, n)$ denoting the conditional probability $P\left(X_i | Pa_{X_i}\right)$ of X_i given its parents Pa_{X_i}, or the probability $P\left(X_i\right)$ of X_i without parents. (2) The set of directed edges $\bar{E} = \{e_{ij}\}_{(i,j)=1}^{|\bar{E}|}$ $(i \neq j)$ represents dependence relations among the variables in X.*

Learning a BN from data involves learning the structure G (i.e., the DAG) and parameters θ (i.e., the CPTs). From a

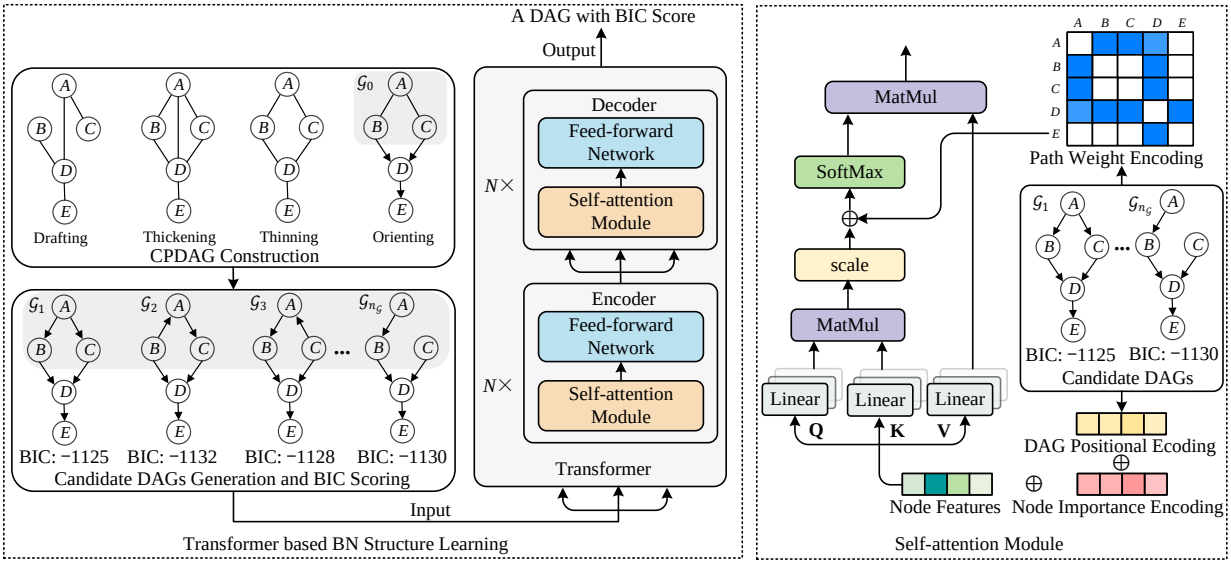

Figure 2: Framework of T-BNSL.

complete data set of $n_{\mathcal{D}}$ instances $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^{n_{\mathcal{D}}}$ over a set of n random variables X, hybrid methods learn an optimal DAG $G = (X, E)$ in two phases [Yan et al., 2022]

- CB phase: using CI tests to construct the graph skeleton to restrict DAG search space.
- SS phase: using scoring functions to find the high-scoring DAG in the restricted search space.

In the CB phase, the most commonly used CI tests for discrete BNs are the G^2 and χ^2 statistical tests and mutual information [Kitson et al., 2023].

Definition 2. *Mutual information measures the amount of information shared between two random variables X and Y, defined as*

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} P(x,y)\log\left(\frac{P(x,y)}{P(x)P(y)}\right), \quad (1)$$

where $P(x)$ and $P(y)$ indicate the marginal distributions of variables X and Y. $P(x,y)$ is the joint probability density, and x and y are the states of X and Y, respectively. If there is no information flow between X and Y, then $I(X;Y) = 0$. If there is a directed edge $X \to Y$ between X and Y, knowing that X eliminates more uncertainty than Y in the information channel $X \to Y$ and provides lots of information for Y [Duan et al., 2024].

Definition 3. *Conditional mutual information is defined as*

$$I(X;Y|Z) = \sum_{x\in X}\sum_{y\in Y}\sum_{z\in Z} P(x,y,z)\log\left(\frac{P(x,y|z)}{P(x|z)P(y|z)}\right). \quad (2)$$

Similarly, if there is no information flow between X and Y given Z, then $I(X;Y|Z) = 0$, that is, X and Y are conditionally independent. If there is information flow between X and Y given Z, then $I(X;Y|Z) > 0$, knowing that the information flow contains a v-structure $X \to Z \leftarrow Y$ [Kitson et al., 2023].

In the SS phase, Bayesian score is used to measure candidate DAGs, which does not require prior parameters and can avoid BN over-fitting [Koller and Friedman, 2009]. By the Bayes rule, we have

$$P(G \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid G)P(G)}{P(\mathcal{D})}, \quad (3)$$

where the denominator is a normalizing factor that does not help distinguish between different structures. Thus, the Bayesian score is defined as:

$$S_B(G,\mathcal{D}) = \log P(\mathcal{D} \mid G) + \log P(G). \quad (4)$$

Since the structure-prior term $P(G)$ in the Bayesian score is almost irrelevant compared to the first term, $P(G)$ can be ignored. If we use a Dirichlet parameter prior for all parameters in a BN $\mathcal{B}$ and perform Laplace's approximation, as $n_{\mathcal{D}} \to \infty$, the marginal likelihood $P(\mathcal{D} \mid G)$ can be approximated by expanding around $\hat{\theta}$ as follows

$$P(\mathcal{D} \mid G) \approx P(\mathcal{D} \mid G, \hat{\theta})P(\hat{\theta} \mid G)\left(\frac{(2\pi)^{d/2}}{\sqrt{|\mathcal{I}(\hat{\theta})|}}\right), \quad (5)$$

where $\mathcal{I}(\hat{\theta})$ is the Fisher information matrix. Taking the logarithm of the approximation $P(\mathcal{D} \mid G)$ as

$$\log P(\mathcal{D} \mid G) \approx \log P(\mathcal{D} \mid G, \hat{\theta}) + \log P(\hat{\theta} \mid G) + \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\mathcal{I}(\hat{\theta})|. \quad (6)$$

As $n_{\mathcal{D}} \to \infty$, the term $\log P(\hat{\theta} \mid G)$ becomes less significant compared to $\log P(\mathcal{D} \mid G, \hat{\theta})$, since the prior $P(\hat{\theta} \mid G)$ does not depend on n_D. Thus, we could obtain

$$\log P(\mathcal{D} \mid G) \approx \log P(\mathcal{D} \mid G, \hat{\theta}) - \frac{\log n_{\mathcal{D}}}{2} \cdot \dim[G] + O(1). \tag{7}$$

The $O(1)$ term represents the remaining constants that become negligible as $n_{\mathcal{D}}$ increases. Hence, as $n_{\mathcal{D}} \to \infty$, we could adopt BIC score $S_{BIC}(G, \mathcal{D})$ to asymptotically approaches the Bayesian score $S_B(G, \mathcal{D})$.

Definition 4. *The BIC score $S_{BIC}(G, \mathcal{D})$ is defined as*

$$\begin{aligned} S_{BIC}(G, \mathcal{D}) &= \log P(\mathcal{D} \mid \hat{\theta}, G) - \frac{\log n_{\mathcal{D}}}{2} \cdot \dim[G] \\ &= \ell\left(\hat{\theta}, \mathcal{D}\right) - \frac{\log n_{\mathcal{D}}}{2} \cdot \dim[G]. \end{aligned} \tag{8}$$

where the first term $\ell\left(\hat{\theta}, \mathcal{D}\right)$ is the logarithm of the likelihood function, and $\hat{\theta}$ denotes the maximum likelihood parameters for G. The second term is the penalty of graph complexity, and $\dim[G]$ is the dimension of BN or the number of independent parameters in BN.

4 METHODOLOGY

4.1 OVERVIEW OF T-BNSL

The framework of our T-BNSL is shown in Figure 2.

- In the CB phase, we view BN as a network of information channels, propose mutual information based CI test method to construct a CPDAG, and use our HC method to generate candidate DAGs with small partial BIC scores on the CPDAG.
- In the SS phase, we extend the self-attention module of Transformer by incorporating unique encodings of the DAGs to predict a large number of remaining DAG scores, and find a single and globally optimal DAG.

4.2 RESTRICTION OF DAG SEARCH SPACE

CPDAG construction. To restrict the search space of DAGs and efficiently construct a graph skeleton named CPDAG shown in Figure 2, an improved method of mutual information based CI test is proposed, including drafting, thickening, thinning and orienting processes on the basis of information flow between node pairs (see Theorems and Algorithms in Appendix A.1 and B.1). Specifically, an information flow (path) between X and Y is open if the nodes are all active valves, whereas is closed given some evidence (all named cut-set) Z, if X and Y are CI separations $Ind(X; Y|Z)$ or d-separation (direction dependence separation) $Dsep(X; Y|Z)$. The cut-set and d-separation are defined as follows [Verma and Pearl, 1991]:

Definition 5. *A path $\mathcal{P}$ from node X to node Y is blocked by cut-set Z, if there is a node W on $\mathcal{P}$ for which one of the following two conditions hold:*

- *W is not a collider and $W \in Z$, i.e., $X \perp Y|Z$, or*
- *W is a collider and neither W or its descendants are not in Z, i.e., $X \perp Y|Z \setminus W$, where W is a node in $\mathcal{P}$ containing two incoming edges (i.e., $X \to W \leftarrow Y$).*

Definition 6. *Two nodes X and Y are d-separated by Z in G (denoted as $Dsep(X; Y|Z)$) iff every path from X to Y is blocked by Z. Two nodes are d-connected (denoted as $Dcon(X; Y|Z)$) if they are not d-separated.*

To use fewer CI tests and smaller condition sets, our method constructs CPDAG $\mathcal{G}_0$ by the following processes:

- **Drafting** generates an initial graph by calculating mutual information of node-pairs $I(X; Y)$. If $I(X; Y)$ is ordered decreasingly, a list L of edges $X - Y$ (i.e., (X, Y)) could be added in an empty graph iff $L = \{(X, Y)|I(X; Y) > \varepsilon\}$.
- **Thickening** adds edges into the draft graph by calculating conditional mutual information of all triples of nodes $I(X; Y|Z)$. According to Definition 6, an edge $X - Y$ could be added into the draft graph iff $I(X; Y|Z') > 0$ and $I(X; Y|Z') \leq I(X; Y|Z)$, where the size of cut-set Z' is no more than that of cut-set Z.
- **Thinning** removes wrongly-added edges in the thickened graph by calculating $I(X; Y|Z)$. According to Definition 6, an edge $X - Y$ could be deleted from the thicken graph iff $I(X; Y|Z') \approx 0$ and $I(X; Y|Z') \leq I(X; Y|Z)$, where the size of cut-set Z' is no more than that of cut-set Z.
- **Orienting** generates directed edges in the thin graph by using the identified colliders. According to Definition 5, the form of a triple of nodes $X - Z - Y$ could be transformed to a v-structure $X \to Z \leftarrow Y$ by orienting two incoming edges, and the collider Z points to their descendants, iff Z is a collider, i.e., $I(X; Y|Z) > 0$.

Candidate DAG generation and scoring. For any three nodes X, Y and Z in $\mathcal{G}_0$, possible candidate structures include the chain ($X \to Z \to Y$ and $X \leftarrow Z \leftarrow Y$), fork ($X \leftarrow Z \to Y$) and collider ($X \to Z \leftarrow Y$) structures. Among them, only the collider structure (also called a v-structure) can make information pass from X to Y when Z is instantiated (also called evidence) according to Definition 5. In other words, only the edges in v-structure and the edges between the collider in v-structure and its descendants could be oriented by using our mutual information based CI test. As the process of orienting edges in CPDAG is shown in Figure 2, only the edges $B \to D \leftarrow C$ and $D \to E$ can be oriented, while the edges in the path $B - A - C$ cannot be oriented.

To orient the remaining edges, such as those in the path $B - A - C$ in the CPDAG $\mathcal{G}_0$, and generate our candidate DAGs $\mathcal{G} = \{\mathcal{G}_i\}_{i=1}^{n_\mathcal{G}}$, there are three possible structures, (1) $B \rightarrow A \rightarrow C$, (2) $B \leftarrow A \leftarrow C$, (3) $B \leftarrow A \rightarrow C$. Thus, we propose the HC method with a decomposable BIC scoring function by performing direction reversal and edge deletion on the CPDAG to generate all possible candidate DAGs with a part of globally optimal BIC scores. To prevent returning to the DAGs that have been recently visited, a tabu list is employed that permits DAG score to decrease, and a globally optimal DAG is returned.

To efficiently calculate the BIC score $S_{BIC}(\mathcal{G}_i, \mathcal{D})$ of $\mathcal{G}_i$, we aim to decompose the logarithm of the likelihood function $\ell(\hat{\theta}, \mathcal{D})$ in Eq. 8 by information theory. Let $\hat{P}(x)$ be the empirical frequency of x in $\mathcal{D}$, $\hat{P}(x, y)$ be the empirical frequency of x and y in $\mathcal{D}$, $M[x] = n_\mathcal{D} \cdot \hat{P}(x)$, and $M[x, y] = n_\mathcal{D} \cdot \hat{P}(x, y)$. By combining all the occurrences of each parameter $\theta_{x_i|u_i}$, the log-likelihood function of $\mathcal{B}$ over $\mathcal{D}$ can be written as

$$\ell(\hat{\theta}, \mathcal{D}) = \sum_{i=1}^{n} \left[\sum_{u_i \in Pa_{X_i}} \sum_{x_i} M\left[x_i, u_i\right] \log \hat{\theta}_{x_i|u_i} \right]. \quad (9)$$

In practice, $\hat{\theta}_{y|x} = \hat{P}(y \mid x)$. According to the proposition of decomposable likelihood score [Koller and Friedman, 2009], we rewrite one terms in the square brackets as

$$\begin{aligned}
&\sum_{u_i} \sum_{x_i} M\left[x_i, u_i\right] \log \hat{\theta}_{x_i|u_i} \\
&= n_\mathcal{D} \sum_{u_i} \sum_{x_i} \hat{P}\left(x_i, u_i\right) \log \hat{P}\left(x_i \mid u_i\right) \\
&= n_\mathcal{D} \sum_{u_i} \sum_{x_i} \hat{P}\left(x_i, u_i\right) \log \frac{\hat{P}\left(x_i, u_i\right)}{\hat{P}\left(u_i\right) \hat{P}\left(x_i\right)} \qquad (10) \\
&\quad + n_\mathcal{D} \sum_{x_i} \left(\sum_{u_i} \hat{P}\left(x_i, u_i\right) \right) \log \hat{P}\left(x_i\right) \\
&= n_\mathcal{D} \cdot I_{\hat{P}}\left(X_i; Pa_i\right) - n_\mathcal{D} \sum_{x_i} \hat{P}\left(x_i\right) \log \frac{1}{\hat{P}\left(x_i\right)} \\
&= n_\mathcal{D} \cdot I_{\hat{P}}\left(X_i; Pa_i\right) - n_\mathcal{D} \cdot H_{\hat{P}}\left(X_i\right),
\end{aligned}$$

where $I_{\hat{P}}\left(X_i; Pa_{X_i}\right)$ is the mutual information between X_i and its parents Pa_{X_i} in $\mathcal{D}$ of $\mathcal{B}$. $H_{\hat{P}}\left(X_i\right)$ is the information entropy of X_i, and $I_{\hat{P}}\left(X_i; Pa_{X_i}\right)$ is 0 if $Pa_{X_i} = \emptyset$. Through arithmetic transformations of the term in Eq. 10, we give the following proposition.

Proposition 1. *The likelihood score could be decomposed as:*

$$\ell(\hat{\theta}, \mathcal{D}) = n_\mathcal{D} \sum_{i=1}^{n} I_{\hat{P}}\left(X_i; Pa_{X_i}\right) - n_\mathcal{D} \sum_{i=1}^{n} H_{\hat{P}}\left(X_i\right), \quad (11)$$

Thus, the BIC score in Eq. 8 can be further decomposed as

$$\begin{aligned}
S_{BIC}(\mathcal{G}_i, \mathcal{D}) &= n_\mathcal{D} \sum_{i=1}^{n} I_{\hat{P}}\left(X_i; Pa_{X_i}\right) \\
&\quad - n_\mathcal{D} \sum_{i=1}^{n} H_{\hat{P}}\left(X_i\right) - \frac{\log n_\mathcal{D}}{2} \cdot \dim[\mathcal{G}_i]. \qquad (12)
\end{aligned}$$

Note that the entropy term $H_{\hat{P}}\left(X_i\right)$ is not dependent of the graph, which thus does not influence the score of DAG and could be ignored. The mutual information and penalty terms jointly determine the score of $\mathcal{G}_i$, while the mutual information term measures the amount of information of the nodes transmitted by their parents and could be further decomposed. Thus, $S_{BIC}(\mathcal{G}_i, D)$ in Eq. 12 is rewritten as

$$\begin{aligned}
S_{BIC}(\mathcal{G}_i, \mathcal{D}) &= \sum_{i=1}^{n} FamScore\left(X_i | Pa_{X_i} : \mathcal{D}\right) \\
&= n_\mathcal{D} \sum_{i=1}^{n} I_{\hat{P}}\left(X_i; Pa_{X_i}\right) - \frac{\log n_\mathcal{D}}{2} \cdot \dim[\mathcal{G}_i], \qquad (13)
\end{aligned}$$

where the family score $FamScore\left(X_i | Pa_{X_i} : \mathcal{D}\right)$ measures how well a set of parents serves as the parents of X in $\mathcal{D}$. By our decomposable BIC scoring method, the computational overhead of DAG scores could be reduced dramatically, since local changes in DAG will not make the remaining DAG scores changed in view of the family scores.

4.3 TRANSFORMER BASED HIGH-SCORING DAG PREDICTION

To predict the BIC scores of remaining DAGs efficiently, the Transformer with tailed attention mechanism is proposed.

Node embedding. Our Transformer has an encoder-decoder structure, where each Transformer layer $l \in \{0, 1, \cdots, n_l\}$ has two main blocks, a self-attention module as the core and a feed-forward network (FFN). Let $\mathbf{y}_i^l$ be the l-layer embedding of X_i in $\mathcal{G}$ and $\mathbf{X}_i$ be the node features of X_i generated from $\mathcal{G}$, satisfying $\mathbf{y}_i^0 = \mathbf{X}_i$. In the self-attention module, the l-layer node embeddings $\mathbf{Y}^l = \left[\left(\mathbf{y}_1^l\right)^\top, \left(\mathbf{y}_2^l\right)^\top, \cdots, \left(\mathbf{y}_n^l\right)^\top \right]^\top$ are projected to query, key and value matrices $\mathbf{Q}^l$, $\mathbf{K}^l$ and $\mathbf{V}^l$, by parameter matrices $\mathbf{W_Q}$, $\mathbf{W_K}$, $\mathbf{W_V}$ respectively, such that $\mathbf{Q}^l = \mathbf{Y}^l\mathbf{W_Q}$, $\mathbf{K}^l = \mathbf{Y}^l\mathbf{W_K}$, $\mathbf{V}^l = \mathbf{Y}^l\mathbf{W_V^l}$. The single-head self-attention $\mathrm{Attn}(\mathbf{Y}^l)$ is

$$\mathbf{A}^l = \frac{\mathbf{Q}^l \left(\mathbf{K}^l\right)^\top}{\sqrt{d}}, \ \mathrm{Attn}(\mathbf{Y}^l) = \mathrm{softmax}\left(\mathbf{A}^l\right)\mathbf{V}^l, \quad (14)$$

where $\mathbf{A}^l$ is a matrix capturing the similarities between queries and keys. $\mathbf{Y}^l \in \mathbb{R}^{n \times d}$, and d is the dimension of

node embeddings. $\mathrm{Attn}(\mathbf{Y}^l) \in \mathbb{R}^{n \times d_{\mathbf{K}}}$, and $d_{\mathbf{K}}$ is the dimension of $\mathbf{Q}^l$ and $\mathbf{K}^l$. For simplicity, we assume $d_{\mathbf{K}} = d$.

To generate $\mathbf{Y}^l$ effectively, we incorporate the node-importance, positional and path-weight encodings of $\mathcal{G}_i$ into the self-attention module, which could capture node attributes and structured relations of $\mathcal{G}_i$, respectively. For each $\mathcal{G}_i$, the indegrees and outdegrees of X_i are different. Hence, the node-importance encoding is designed to measure the importance of X_i with indegrees $deg^-(X_i)$ and outdegrees $deg^+(X_i)$ proposed as the input of self-attention

$$\mathbf{y}_i^{(0)} = \mathbf{X}_i + deg^-(X_i) + deg^+(X_i). \tag{15}$$

The partial order in DAGs can create particular relations between a pair of nodes, meaning that a node's predecessors and successors are most likely more important than other nodes. For any $\mathcal{G}_i$, there is a unique strong partial order $\preccurlyeq$ on V. Specially, $X \preccurlyeq Y$ denotes that there is a directed path from X to Y, where X is a predecessor of Y. All nodes without predecessors are source nodes, and those without successors are target nodes. Based on the partial order of $\mathcal{G}_i$, DAG depth $depth(X_i)$ is defined and $depth(X_i) = 0$ if X_i is a source node, otherwise, $depth(X_i) = 1 + \max_{(X_j,X_i)\in E} depth(X_j)$. Hence, let i_d be the index of the dimension of node embedding, we propose the DAG depth based position encoding method

$$\begin{aligned} PE_{(X_i,2i_d)} &= \sin\left(\frac{depth(X_i)}{1000^{\frac{2i_d}{d}}}\right) \\ PE_{(X_i,2i_d+1)} &= \cos\left(\frac{depth(X_i)}{1000^{\frac{2i_d}{d}}}\right). \end{aligned} \tag{16}$$

On the basis of positional encodings, the node features as the input of self-attention is rewritten as

$$\mathbf{y}_i^{(0)} = \mathbf{X}_i + deg^-(X_i) + deg^+(X_i) + PE_{X_i}. \tag{17}$$

If there is no cut-set on every path from X_i to X_j, and the information paths (flows) $\mathcal{P}$ from X_i to X_j is open, then X_i provides more information for X_j, according to Definition 5. If the information paths from X_i to X_j are closed, then X_i provides no information for X_j. Thus, the path weight from X_i to X_j could be denoted as

$$I_{\mathcal{P}}(X_i; X_j) = \begin{cases} \sum_{i=1}^{n_{\mathcal{P}}-1} I(X_i; Pa_{X_i}), & \text{no cut-set;} \\ 0, & \text{otherwise} \end{cases}. \tag{18}$$

Then, we propose to use $I_{\mathcal{P}}(X_i; X_j)$ to encode directed paths effectively, and measure the graph-structured relations. Hereby, an element $A_{i,j}^l$ of the Query-Key product matrix $\mathbf{A}^l$ in the self-attention is reformulated by adding a bias term $I_{\mathcal{P}}(X_i; X_j)$ as

$$A_{i,j}^l = \frac{\left(\mathbf{y}_i^{(l)}\mathbf{W}_{\mathbf{Q}}\right)\left(\mathbf{y}_j^{(l)}\mathbf{W}_{\mathbf{Q}}\right)^\top}{\sqrt{d}} + I_{\mathcal{P}}(X_i; X_j). \tag{19}$$

Multi-head attention (MHA) inhibits the single-head self-attention, and concatenates multiple instances of Eq. 14 that has shown to be effective in practice [Vaswani et al., 2017, Ni et al., 2023]. To make more efficient convergence of Transformer, we incorporate the layer normalization (LN) before MHA and FFN [Ying et al., 2021, Liu et al., 2024], which jointly compose a Transformer layer. Thus, our Transformer can generate node embeddings $\mathbf{Y}$ at layer l

$$\mathbf{Y}^l = \mathbf{Y}^{(l-1)} + \mathrm{MHA}(\mathrm{LN}(\mathbf{Y}^{(l-1)})), \tag{20}$$

$$\mathbf{Y}^l = \mathbf{Y}^l + \mathrm{FFN}(\mathrm{LN}(\mathbf{Y}^l)), \tag{21}$$

where $\mathbf{Y}^{(l-1)}$ represents the node embeddings and comes from the previous layer decoder output, calculated as

$$\mathbf{Y}^{(l-1)} \leftarrow \mathbf{Y}^{(l-1)} + \mathrm{Attn}(\mathbf{Y}^{l-1}), \tag{22}$$

where '+' represents a residual connection, meaning that the output of the MHA is added to the original input $\mathbf{Y}^{(l-1)}$. This helps preserve information and mitigate the vanishing gradient problem as the depth of Transformer increases.

BIC score prediction. Once the final layer of node embeddings $\mathbf{Y}^{n_l}$ is generated, we use a readout function to concatenate the node embeddings across layers to produce embedding $\mathbf{Y}_{\mathcal{G}_i}$ of $\mathcal{G}_i$. On the basis of the ultimate $\mathbf{Y}_{\mathcal{G}_i}$, a fully-connected layer FC is used to predict the BIC score $\hat{S}_{BIC}(\mathcal{G}_i, \mathcal{D}; \Theta)$ of each nonscoring DAG, where $\mathbf{Y}_{\mathcal{G}_i}$ and $\hat{S}_{BIC}(\mathcal{G}_i, \mathcal{D}; \Theta)$ are written as

$$\begin{aligned} \mathbf{Y}_{\mathcal{G}_i} &= \mathrm{READOUT}\left(\left\{\mathbf{y}_{n_l}^l, X_i \in X\right\}\right) \\ &= \mathrm{FC}\left(\text{Max-Pool}\left(\|_{l=0}^{n_l} \mathbf{y}_i^l\right)\right), \end{aligned} \tag{23}$$

$$\hat{S}_{BIC}(\mathcal{G}_i, \mathcal{D}; \Theta) = \mathbf{W}\mathbf{Y}_{\mathcal{G}_i} + \mathbf{b}, \tag{24}$$

where $\mathbf{y}_{n_l}^l \in \mathbf{Y}^l$, $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters of the fully connected layer. Since the BIC score is discrete, the average mean squared error (MSE) loss aims to minimize the error between the predicted BIC score and ground-truth BIC score of the DAG $\mathcal{G}_i$, denoted by

$$\mathcal{L} = \min_{\Theta} \sum_{i=1}^{n_{\mathcal{G}}} \left(\hat{S}_{BIC}(\mathcal{G}_i, \mathcal{D}; \Theta) - S_{BIC}(\mathcal{G}_i, \mathcal{D})\right)^2, \tag{25}$$

where AdamW is used to iteratively update the parameters Θ of Transformer. At last, we search for the highest scoring DAG as the globally optimal one. The above idea is summarized in Algorithm 1, and the time complexity analysis is shown in Appendix B.2.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETTINGS

Datasets. We adopted five original BNs [1] as seen in Appendix C.1. Each dataset includes a large number of instances sampled from the original BN by forward sampling,

[1] https://www.bnlearn.com/bnrepository/

Algorithm 1 T-BNSL

Input: a complete dataset of instances $\mathcal{D}$
Parameter: model parameters Θ, total training epoch T
Output: globally optimal DAG $\mathcal{G}$

Construct the CPDAG $\mathcal{G}_0$ by drafting, thickening, thinning and orienting processes
Generate $\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_{n_\mathcal{G}}$ with partial BIC scores
for $t = 1$ to T **do**
  $\mathbf{y}_i^{(0)} = \mathbf{X}_i + deg^-(X_i) + deg^+(X_i) + PE_{X_i}$
  for each l in n_l **do**
    Calculate $\mathbf{A}^{l-1}$ by Eq. 19
    $\mathrm{Attn}(\mathbf{Y}^{l-1}) \leftarrow \mathrm{softmax}\left(\mathbf{A}^{l-1}\right)\mathbf{V}^{l-1}$
    $\mathbf{Y}^{(l-1)} \leftarrow \mathbf{Y}^{(l-1)} + \mathrm{Attn}(\mathbf{Y}^{l-1})$
    $\mathbf{Y}^{l} \leftarrow \mathbf{Y}^{(l-1)} + \mathrm{MHA}(\mathrm{LN}(\mathbf{Y}^{(l-1)}))$
    $\mathbf{Y}^{l} \leftarrow \mathbf{Y}^{l} + \mathrm{FFN}(\mathrm{LN}(\mathbf{Y}^{l}))$
  end for
  $\mathbf{Y}_\mathcal{G} \leftarrow \mathrm{READOUT}\left(\mathbf{Y}^l_{n_l}\right)$
  $\hat{S}_{BIC}(\mathcal{G}, \mathcal{D}; \Theta) \leftarrow \mathbf{W}\mathbf{Y}_\mathcal{G} + \mathbf{b}$
  $\mathcal{L} \leftarrow \min_\Theta \sum_{i=1}^{n_\mathcal{G}} \left(\hat{S}_{BIC}(\mathcal{G}_i, \mathcal{D}; \Theta) - S_{BIC}(\mathcal{G}_i, \mathcal{D})\right)^2$
end for
return $\mathcal{G}$ with the highest BIC score

and was split into 50% training and 50% test sets. For each dataset, 10,000 candidate DAGs with their true BIC scores were generated by our CI tests and HC method.

Comparison methods. We adopted three types of state-of-the-art methods for comparison, including (1) PC-MAX [He et al., 2024] as CB method, (2) HC [Heckerman et al., 1995] as SS method, (3) MMHC [Kuipers et al., 2022], DAGNN [Thost and Chen, 2021] and GFlowOpt [Yang et al., 2025] as hybrid methods.

Metrics and implementation. To evaluate the effectiveness of T-BNSL, we adopted mean BIC score, structural hamming distance (SHD), mean absolute error (MAE), ablation study and the ratio of training set. To evaluate the efficiency of T-BNSL, we compared the execution time of different methods. The implementation of our T-BNSL please refer to Appendix C.2.

5.2 EXPERIMENTAL RESULTS

Accuracy of BN structure learning. Our T-BNSL was compared with the competitors by using the same settings and hyperparameters. The mean BIC scores and SHDs of the optimal DAG are reported in Table 1. The result demonstrates that: (a) The BIC scores of our T-BNSL are all higher than that of second-best competitor GFlowOpt, and SHD are all lower than that of GFlowOpt, since our method can generate candidate DAGs with globally optimal BIC scores. (b) The BIC scores of our T-BNSL gradually rise, and SHDs gradually decrease by varying the size of each dataset from 2k to 5k, indicating that larger size of dataset can make correct structure of candidate graph closer to real structure of BN, and make BIC scores easier to distinguish candidate structures correctly. (c) our T-BNSL outperforms HC, MMHC and DAGNN, since these methods easily get stuck in locally optimal, and our method in the CB phase can generate some optimal candidate DAGs, and predict the accurate BIC score in the SS phase.

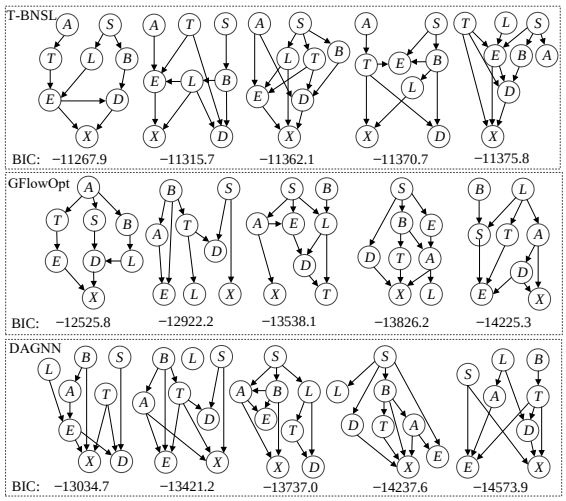

Figure 3: Top 5 DAGs Found by Different Models.

Case study of the predicted DAG. We performed a case study by using DAGNN, GFlowOpt and our T-BNSL. For DAGNN and GFlowOpt, settings and hyperparameters were consistent with the original paper. The top 5 DAGs ranked by their BIC scores are reported in Figure 3. The result demonstrates that: (a) Our T-BNSL searches for a DAG with the highest BIC score -11267.9 that is almost as the same as the true BIC score -11193.0, except adding an extra edge $D \rightarrow X$. (b) The top 5 DAGs found by our method have a much more accurate BIC score than those found by GFlowOpt and DAGNN. This means that our Transformer has better expressivity and captures more long-range dependence relations of node pairs than the competitors.

Ablation study. We made ablation studies on AL-5000 in Table 2. The result demonstrates that the MAE value is minimal when the node-importance, positional and path-weight encodings are all incorporated into the our self-attention module. This means that our designed encoding mechanism could capture the node and structural features.

Impacts of ratio of training set. To better understand the ratio ρ of training set on accuracy of our T-BNSL, ρ was set to 60%, 70% and 80% on Asia BN, respectively. The result demonstrates that the BIC score of our T-BNSL is gradually decreased by varying ρ from 60% to 80%, meaning that our T-BNSL is more accurate when the ratio ρ is larger. For more details please refer to Table 4 in Appendix C.3.

Table 1: Accuracy of BN Structure Learning.

Dataset	Size	Mean BIC Scores and SHDs					
		PC-MAX	HC	MMHC	DAGNN	GFlowOpt	T-BNSL
CH	2k	−2407 (55)	−2189 (45)	−2185 (35)	−2184 (22)	−2183 (18)	**−2179 (13)**
	3k	−1778 (48)	−1775 (38)	−1767 (26)	−1767 (18)	−1760 (14)	**−1759 (11)**
	4k	−1591 (42)	−1498 (29)	−1485 (20)	−1450 (14)	−1437 (12)	**−1421 (10)**
	5k	−990 (31)	−986 (26)	−948 (15)	−942 (11)	−940 (10)	**−938 (9)**
AL	2k	−10822 (128)	−10807 (111)	−10590 (87)	−10510 (73)	−10411 (50)	**−10405 (30)**
	3k	−8572 (125)	−8532 (105)	−8530 (85)	−8517 (65)	−8489 (45)	**−8447 (25)**
	4k	−6778 (115)	−6731 (92)	−6457 (75)	−6426 (53)	−6420 (35)	**−6375 (16)**
	5k	−4629 (105)	−4583 (80)	−4535 (62)	−4422 (45)	−4391 (25)	**−4302 (14)**
BA	2k	−19960 (93)	−19840 (86)	−19835 (71)	−19831 (62)	−19818 (55)	**−19740 (45)**
	3k	−15979 (88)	−15956 (79)	−15931 (64)	−15902 (58)	−15888 (48)	**−15863 (38)**
	4k	−14896 (81)	−12843 (71)	−12827 (61)	−12818 (51)	−11967 (41)	**−11840 (16)**
	5k	−8835 (72)	−8565 (66)	−8475 (54)	−8220 (46)	−8027 (34)	**−7941 (13)**
HA	2k	−12942 (243)	−12940 (202)	−12189 (163)	−12123 (118)	−12122 (81)	**−11192 (55)**
	3k	−10534 (232)	−9963 (190)	−9643 (150)	−9151 (100)	−9067 (70)	**−9051 (48)**
	4k	−7860 (224)	−7267 (180)	−6922 (140)	−6865 (71)	−6809 (60)	**−6753 (42)**
	5k	−5524 (204)	−4934 (170)	−4802 (130)	−4783 (62)	−4554 (55)	**−4524 (38)**
WI	2k	−42911 (384)	−41870 (316)	−39278 (262)	−37697 (205)	−36627 (140)	**−36565 (80)**
	3k	−34655 (365)	−33479 (305)	−30296 (245)	−30133 (180)	−29471 (120)	**−29342 (70)**
	4k	−26136 (350)	−26025 (283)	−24297 (220)	−23185 (118)	−22996 (90)	**−22569 (63)**
	5k	−17863 (347)	−17750 (281)	−17070 (200)	−16839 (113)	−15703 (85)	**−14939 (58)**

Table 2: Ablation Study Results.

Node-importance encoding	Positional encoding	Path-weight encoding	MAE
-	✓	✓	359.7
✓	-	✓	1599.6
✓	✓	-	2027.2
✓	✓	✓	73.9

Efficiency of BN structure learning. We compared our T-BNSL with the competitors, and the total time was reported in Figure 4. The result demonstrates that: (a) Our T-BNSL is faster than the competitors on each dataset, since our T-BNSL could be parallelizable and avoid the time-consuming BIC scoring in exponential search space. (b) Our T-BNSL is about 23% faster than GFlowOpt, since our method restricts the DAG search space, and reduces the time of BIC scoring.

Impacts of Parameters on Efficiency of T-BNSL. To better understand the dimensionality d of embedding vectors and number $n_{\mathcal{D}}$ of instances on the efficiency of T-BNSL, d was set to 30 $\sim$ 60 and $n_{\mathcal{D}}$ was set to 20×10^3 $\sim$ 100×10^3, respectively. T-BNSL was compared with GFlowOpt and DAGNN methods, and the average execution time is reported in Figure 5 (see Appendix C.4). The result demonstrates that: (a) Our T-BNSL is faster than the competitors with the increase of d, since we restrict the search space of DAGs and reduce the time complexity of BIC scoring. (b) Our T-BNSL is faster than the comparison methods by varying $n_{\mathcal{D}}$ from 20×10^3 to 100×10^3, since our parallelizable Transformer could be trained much faster than GFlowOpt and DAGNN.

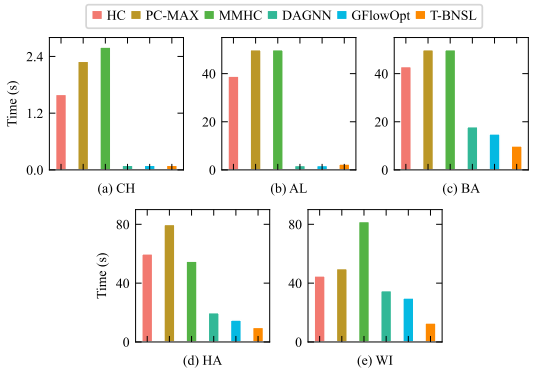

Figure 4: Efficiency of BN Structure Learning.

6 CONCLUSION

In this paper, we propose a Transformer based BN structure learning from an information theory perspective. Experimental results demonstrate that our method surpasses accuracy and efficiency of the comparison methods. In the future, we will generalize our method to learn BN structure for continuous variables. Meanwhile, we will consider extending the BIC scoring function by leveraging the structural entropy of candidate DAGs.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (62567008, U23A20298); Major Science and Technology Special Foundation of Yunnan Province (202502AD080002); Yunnan Fundamental Research Project (202401AT070462); Program of Yunnan Key Laboratory of Intelligent Systems and Computing (202549CE340006); Innovation Project of Caiyun Postdoctoral Program (376936). For any correspondence, please refer to Jiahui Wang.

References

Sepehr Assadi, Gary Hoppenworth, and Nicole Wein. Covering approximate shortest paths with dags. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing* (STOC), pages 2269–2280, Prague, Czechia, 2025.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence* (UAI), pages 518–528, Eindhoven, Netherlands, 2022.

Liang Duan, Xiang Chen, Wenjie Liu, Daliang Liu, Kun Yue, and Angsheng Li. Structural entropy based graph structure learning for node classification. In *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI), pages 8372–8379, 2024.

Juha Harviainen, Kseniya Rychkova, and Mikko Koivisto. Quantum speedups for bayesian network structure learning. In *Proceedings of the 41st Conference on Uncertainty in Artificial Intelligence* (UAI), pages 1640–1647, Rio de Janeiro, Brazil, 2025.

Chuchao He, Peng Wang, LinYu Tian, Ruohai Di, Zidong Wang, and Yu Yang. A novel structure learning method of bayesian networks based on the neighboring complete node ordering search. *Neurocomputing*, 585:127620, 2024.

David Heckerman, Dan Geiger, and David Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

Neville Kenneth Kitson, Anthony C Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. A survey of bayesian network structure learning. *Artificial Intelligence Review*, 56(8):8721–8814, 2023.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Jack Kuipers, Polina Suter, and Giusi Moffa. Efficient sampling and structure learning of bayesian networks. *Journal of Computational and Graphical Statistics*, 31(3): 639–650, 2022.

Chuang Liu, Yibing Zhan, Xueqi Ma, Liang Ding, Dapeng Tao, Jia Wu, Wenbin Hu, and Bo Du. Exploring sparsity in graph transformers. *Neural Networks*, 174:106265, 2024.

Daniela Marella and Paola Vicard. Bayesian network structural learning from complex survey data: a resampling based approach. *Statistical Methods & Applications*, 31 (4):981–1013, 2022.

Jinjie Ni, Rui Mao, Zonglin Yang, Han Lei, and Erik Cambria. Finding the pillars of strength for multi-head attention. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics* (ACL), pages 14526–14540, 2023.

Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

Joseph Ramsey. Improving accuracy and scalability of the pc algorithm by maximizing p-value. *arXiv preprint arXiv:1610.00378*, 2016.

Bahare Salmani and Joost-Pieter Katoen. Finding an ε-close minimal variation of parameters in bayesian networks. In *Proceedings of the 32th International Joint Conference on Artificial Intelligence* (IJCAI), pages 5720–5729, 8 2023. doi: 10.24963/ijcai.2023/635.

Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

Veronika Thost and Jie Chen. Directed acyclic graph neural networks. In *Proceedings of the 9th International Conference International Conference on Learning Representations* (ICLR), May 2021.

Fulya Trösser, Simon de Givry, and George Katsirelos. Improved acyclicity reasoning for bayesian network structure learning with constraint programming. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence* (IJCAI), pages 4250–4257, August 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31th Neural Information Processing Systems* (NIPS), pages 5998–6008, December 2017.

Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence* (UAI), page 255–270, 1991.

Xinran Wu, Kun Yue, Liang Duan, and Xiaodong Fu. Learning a bayesian network with multiple latent variables for implicit relation representation. *Data Mining and Knowledge Discovery*, pages 1–36, 2024.

Ruihan Xu, Haokui Zhang, Yaowei Wang, Wei Zeng, and Shiliang Zhang. Nn-former: Rethinking graph structure in neural architecture representation. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (CVPR), pages 10004–10014, 2025a.

Tong Xu, Simge Küçükyavuz, Ali Shojaie, and Armeen Taeb. An asymptotically optimal coordinate descent algorithm for learning bayesian networks from gaussian models. *Journal of Machine Learning Research*, 26(250): 1–30, 2025b.

Kefei Yan, Wei Fang, Hengyang Lu, Xin Zhang, Jun Sun, and Xiaojun Wu. Mutual information-guided ga for bayesian network structure learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8282–8299, 2022.

Zhu Yang, Kun Yue, Zhiwei Qi, Liang Duan, and Jianyu Li. Gflownet with gradient-based optimization for bayesian network structure learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management* (CIKM), page 3834–3843, Seoul, Republic of Korea, 2025.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Proceedings of the 35th Neural Information Processing Systems* (NIPS), pages 28877–28888, December 2021.

Kun Yue, Zhiwei Qi, Liang Duan, and Zhu Yang. Transformer based bayesian network embedding for efficient multiple probabilistic inferences. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (CIKM), pages 3073–3082, 2024.

Transformer Based Bayesian Network Structure Learning from an Information Theory Perspective (Supplementary Material)

Zhiwei Qi[1,2] **Kun Yue**[1,3] **Zhu Yang**[1,3] **Jiahui Wang**[*1,3]

[1]Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, Yunnan, China
[2]School of Education, Yunnan University, Kunming, Yunnan, China
[3]School of Information Science and Engineering, Yunnan University, Kunming, Yunnan, China

A THEOREM PROOFS

A.1 CORRECTNESS OF MUTUAL INFORMATION BASED CI TEST

Theorem 1. *Given a sufficiently large set of instances from a ground-truth BN, the mutual information based CI test will recover the correct underlying essential DAG.*

We use the following propositions to prove that the mutual information based CI test is correct. Let G be the ground-truth DAG, $\mathcal{G}_{01}$, $\mathcal{G}_{02}$, $\mathcal{G}_{03}$ and $\mathcal{G}_{04}$ be the graphs of drafting, thickening, thinning and orienting processes of CPDAG $\mathcal{G}_0$ construction, respectively.

Proposition 2. *When constructing CPDAG, the graph $\mathcal{G}_{01}$ generated after drafting process contains all the edges of G.*

Proof. This step considers all the edges between any two nodes that are not independent. An edge is not added only if the two nodes are separated by a set of other nodes. Hence, any two nodes that are not directly connected in $\mathcal{G}_{01}$ are conditionally independent in G. □

Lemma 1. *As EdgeNeeded Algorithm 3 (see Appendix B.1) starts with the initial condition-set $S_X \cup S_{X'}$ that can close all the paths of the underlying DAG G between nodes X and Y except the paths connecting X and Y by one collider.*

Proof. By using $S_X \cup S_{X'}$ as the condition-set, we instantiate the nodes in S_X (the neighbors of X on the paths between X and Y) and $S_{X'}$ (the neighbors of S_X that are on the paths between X and Y). Therefore, we can instantiate at least two consecutive nodes of any path that has the length equal to or larger than 3. Since two consecutive nodes of a path cannot both be colliders in the path, and all the paths of the underlying DAG G are in the current graph, we can close all the paths in G between X and Y that have the length equal to or larger than 3. Hence, the only paths that could remain open are those connecting X and Y by one collider. □

Lemma 2. *EdgeNeeded does not open any previously closed X–Y path by removing a node from condition-set C.*

Proof. The EdgeNeeded Algorithm will not remove a node that only opens some paths, such that a removal would necessarily increase the mutual information. We therefore need only prove that removing a node from C cannot simultaneously open some paths and close others. From Lemma 1, we know that initially the only open paths are those connecting X and Y by one collider. Now we consider each node v in C. If v is not a child-node of both X and Y or a descendent of such a child-node, removing it may open some paths but cannot close the paths connecting X and Y by a collider. Thus, suppose that v is a child-node of both X and Y or a descendent of such a child-node. Then, if one of v's descendents is in C, then removing v cannot close the path connecting X and Y by the child-node of both X and Y. If none of v's descendents is in C, removing v may close the path connecting X and Y by the child-node of X or Y but cannot open a path, since the would-be opened path must go through a collider that is a descendent of v. Since none of v's descendents is in C, such a path cannot be opened. □

Lemma 3. *EdgeNeeded can remove all the descendents of both X and Y from condition-set C.*

Proof. Toward a contradiction, suppose that S is the subset of set C containing the descendents of both X and Y that cannot be removed. Then, there must be a node $v \in S$ that is not an ancestor of any other nodes in S. The only reason we would not consider removing v is if removing it increases mutual information. From Lemma 2, we know that removing v will open at least one path. Therefore, node v is not a collider in such a path and this path must go through at least one descendent of v. Since a descendent of v is also a descendent of both X and Y, there must exist at least one descendent of v as a collider in such open path. To make such path open, this collider has to be in S. This contradicts our assumption that v is not an ancestor of any other nodes in S. □

Proposition 3. *Given a $\mathcal{G}$ containing all the edges of the ground-truth DAG G, if two nodes X and Y are independent in G, EdgeNeeded can always separate them in $\mathcal{G}$.*

Proof. From Lemma 1, we know that initially the only open paths are those connecting nodes X and Y by one collider. From Lemmas 2 and 3, we know that the procedure does not open any path when removing nodes from the condition-set C that removes all the descendents of both X and Y that are in C. Therefore, if X and Y are independent in $\mathcal{G}$, the EdgeNeeded Algorithm can separate them by closing all the open paths. □

Proposition 4. *The graph generated after thinning process, $\mathcal{G}_{03}$, contains the same edges as those of G.*

Proof. It is known that $\mathcal{G}_{02}$ contains all the edges of G. An edge is removed in the thinning process only if the pair of nodes is conditionally independent in G, and $\mathcal{G}_{03}$ also contains all the edges of G. From Proposition 2, we also know that if two nodes are independent in G, our algorithm can always separate them in $\mathcal{G}_{03}$. Hence, $\mathcal{G}_{03}$ contains exactly the same edges as G. □

Proposition 5. *Given that graph $\mathcal{G}$ contains the exact same edges as those of the underlying DAG G, all the colliders that can be identified by the orienting process are the real colliders of G.*

Proof. For any structure $X - Y - Z$ where X and Z are not directly connected, the orienting process uses step 1 to check if Y is a collider on the path $X - Y - Z$. From Lemma 3, we know that the final cut-set between X and Z will never include Y if Y is a collider. Thus, step 1 can identify a collider correctly. Since there are no extra-edges in $\mathcal{G}$, step 2 of this process can never orient an edge wrongly. It is also easy to see that the inference of step 3 of the process is correct. □

B ALGORITHMS

B.1 CPDAG CONSTRUCTION ALGORITHM

The four processes of CPDAG construction are drafting, thickening, thinning and orienting, summarized in Algorithm 2.

B.1.1 Time complexity

As is shown in Algorithm 3, the EdgeNeeded Algorithm requires O $\left(n^2\right)$ CI tests. The thickening process can then call EdgeNeeded on at most every pair of nodes, as can thinning process, and thus these processes require O $\left(n^2 \times n^2\right)$ CI tests. Following, as is described in Algorithm 4, the orienting process requires $O\left(n^2\right)$ time. Overall, the CPDAG construction algorithm requires O $\left(n^4\right)$ CI tests.

B.2 TIME COMPLEXITY OF T-BNSL ALGORITHM

Our T-BNSL algorithm includes CB and SS phases. In the CB phase, as is described in Appendix B.1, the CPDAD construction takes $O\left(n^4\right)$ time, and the candidate DAGs generation takes $O\left(n^2\right)$ time. In the SS phase, the Transformer model takes $O\left(T \cdot n_l\right)$ time. Overall, the time complexity of T-BNSL is $O\left(n^4 + T \cdot n_l\right)$. If our T-BNSL has been trained, the score prediction only takes $O\left(n^4\right)$ time.

Algorithm 2 CPDAG Construction

Input: a complete data set of instances $\mathcal{D}$
Parameter: threshold ε
Output: CPDAG $\mathcal{G}_0$

Let $V \leftarrow \{\text{variables in } \mathcal{D}\}, \bar{E} \leftarrow \{\}$
$L \leftarrow \{\langle X, Y\rangle \mid I(X;Y) > \varepsilon\}$ be the list of all pairs of distinct nodes $\langle X, Y\rangle$ where $X, Y \in V$ and $X \neq Y$, with at least ε mutual information.
Sort L into decreasing order, w.r.t., $I(X;Y)$
// Drafting process
for each $\langle X, Y\rangle$ in L **do**
  if there is no adjacency path between X and Y in current graph $(V, \bar{E})$ **then**
    add $\langle X, Y\rangle$ to $\bar{E}$ and remove $\langle X, Y\rangle$ from L
  end if
end for
// Thickening process
for each $\langle X, Y\rangle$ in L **do**
  if EdgeNeeded$(G, X, Y; \mathcal{D}, \varepsilon)$ // Algorithm 3 **then**
    Add $\langle X, Y\rangle$ to $\bar{E}$
  end if
end for
// Thinning process
for each $\langle X, Y\rangle$ in $\bar{E}$ **do**
  if there are indirect paths between X and Y **then**
    $\bar{E}' \leftarrow \bar{E} - \langle X, Y\rangle$
    if $\neg$ EdgeNeeded $(G, X, Y; \mathcal{D}, \varepsilon)$ **then**
      $\bar{E} \leftarrow \bar{E}'$
    end if
  end if
end for
for each $\langle X, Y\rangle$ in $\bar{E}$ **do**
  if X has at least three neighbors other than Y, or Y has at least three neighbors other than X **then**
    $\bar{E}' \leftarrow \bar{E} - \langle X, Y\rangle$
    if $\neg$ EdgeNeeded $(G, X, Y; \mathcal{D}, \varepsilon)$ **then**
      $\bar{E} \leftarrow \bar{E}'$ // Remove $\langle X, Y\rangle$ from E
    end if
  end if
end for
// Orienting process
$\mathcal{G}_0 \leftarrow$OrientEdges($(V, \bar{E})$,$\mathcal{D}$) // Algorithm 4
return $\mathcal{G}_0$

C EXPERIMENT STUDY

C.1 DATASETS

We adopted five original BNs in Table 3, consisting of (1) CHILD (CH) BN from probabilistic expert systems, (2) ALARM (AL) BN from alarm monitoring systems, (3) BARLEY (BA) BN from crop disease modelling, (4) HAILFINDER (HA) BN from hail risk analysis, and (5) WIN95PTS (WI) BN from Windows 95 system troubleshooting.

C.2 IMPLEMENTATION.

We implemented our method in PyTorch 2.4.0 and pgmpy 0.1.26, using the following variables: $n_l = 12$, $d = 512$, $T = 1M$, number of attention heads ($n_h = 32$) and peak learning rate ($2\mathrm{e}-4$). The experiments were conducted on the machine with an Intel 13900KF CPU, 128GB RAM and RTX4090 GPU for 5 times and the average is reported here.

Algorithm 3 EdgeNeeded

Input: a graph G, two nodes X, Y, a complete data set of instances D
Parameter: threshold ε
Output: E

Let $S_X \leftarrow Ngbr(X) \cap AdjPath(X, Y)$ be the neighbors of X on an adjacency path between X and Y; similarly $S_Y \leftarrow Ngbr(Y) \cap AdjPath(X, Y)$
Let $S_{X'} \leftarrow AdjPath(X, Y) \cap (\bigcup_{x \in S_X} Ngbr_G(x) - S_X)$ be the neighbors of the nodes in S_X on the adjacency paths between X and Y, and do not belong to S_X; similarly $S_{Y'} \leftarrow AdjPath(X, Y) \cap (\bigcup_{y \in S_y} Ngbr_G(y) - S_Y)$
Let C be smaller of $\{S_X \cup S_{X'}, S_Y \cup S_{Y'}\}$
$s \leftarrow I_D(X; Y \mid C)$
while $|C| > 1$ **do**
  if $s < \varepsilon$ **then**
    $CutSet \leftarrow CutSet \cup \{\langle \{X, Y\}, C\rangle\}$
  end if
  for each i **do**
    $C_i \leftarrow C \backslash \{$ the ith node of $C\}$, $s_i \leftarrow I(X, Y \mid C_i)$
  end for
  $m \leftarrow \arg\min_i \{s_i\}$
  if $s_m < \varepsilon$ **then**
    $CutSet \leftarrow CutSet \cup \{\langle \{X, Y\}, C_m\rangle\}$
  else if $s_m > s$ **then**
    Continue
  else
    $s \leftarrow s_m, C \leftarrow C_m$
  end if
end while
$\bar{E} \leftarrow \langle X, Y\rangle$
return $\bar{E}$

Algorithm 4 OrientEdges

Input: a graph G, a complete data set of instances D
Output: a CPDAG $\mathcal{G}$

for each three nodes X, Y and Z that X and Y, and Y and Z, are directly connected; and X and Z are not directly connected **do**
  if $\langle \{X, Z\}, C\rangle \in CutSet$ and $Y \notin C$, or $\langle \{X, Z\}, C\rangle \notin CutSet$ **then**
    Let X be a parent of Y and Z be a parent of Y
  end if
end for
for each three nodes X, Y, Z in V, if (i) X is a parent of Y, (ii) Y and Z are adjacent, (iii) X and Z are not adjacent, and (iv) edge (Y, Z) is not oriented, let Y be a parent of Z **do**
  if (i) X is a parent of Y, (ii) Y and Z are adjacent, (iii) X and Z are not adjacent, and (iv) edge (Y, Z) is not oriented **then**
    Let Y be a parent of Z
  end if
end for
for each edge (X, Y) that is not oriented **do**
  if there is a directed path from X to Y **then**
    Let X be a parent of Y
  end if
end for
$\mathcal{G} \leftarrow (V, \bar{E})$
return $\mathcal{G}$

Table 3: Statistics of Datasets.

Datasets	BN	Nodes	Edges	Parms
CH-5000	CH	20	25	230
AL-5000	AL	37	46	509
BA-5000	BA	48	84	114005
HA-5000	HA	56	26	2656
WI-5000	WI	76	112	574

C.3 IMPACTS OF RATIO OF TRAINING SET ON ACCURACY OF BN STRUCTURE LEARNING

To better understand the ratio ρ of training set on accuracy of our T-BNSL, ρ was set to 60%, 70% and 80% on Asia BN dataset, respectively. The average BIC score is reported in Table 4. The result demonstrates that (a) the BIC score of our T-BNSL is gradually decreased by varying ρ from 60% to 80%. (b) As compared to the ground-truth on each BN dataset, the larger the ratio, the more accurate the BIC score of our T-BNSL, since the more accurate the learned parameters of our T-BNSL, the larger the size of training set.

Table 4: Impacts of ρ on Accuracy of T-BNSL.

Datasets	Ratio ρ of Training Set		
	60%	70%	80%
AL-2000	−4554.9	−4535.3	−4523.7
AL-3000	−6837.4	−6823.9	−6806.5
AL-4000	−9141.2	−9135.7	−9109.9
AL-5000	−11301.0	−11281.8	−11266.9

C.4 IMPACTS OF PARAMETERS ON EFFICIENCY OF T-BNSL.

To better understand the dimensionality d of embedding vectors and number $n_{\mathcal{D}}$ of instances on the efficiency of T-BNSL, d was set to $30 \sim 60$ and $n_{\mathcal{D}}$ was set to $20 \times 10^3 \sim 100 \times 10^3$, respectively. T-BNSL was compared with GFlowOpt and DAGNN methods, and the average execution time is reported in Figure 5.

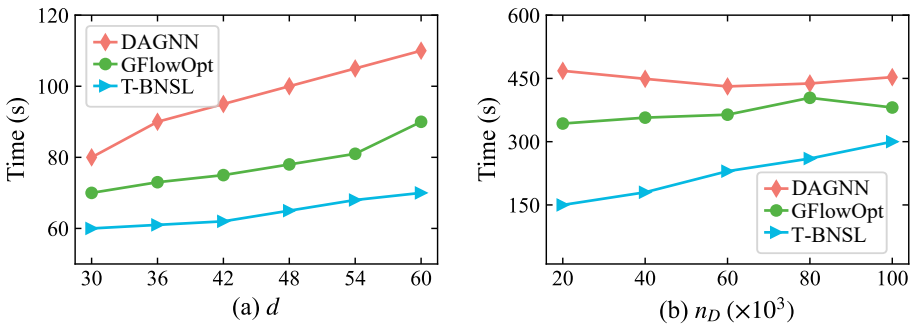

Figure 5: Impacts of Parameters on Efficiency of T-BNSL.