**Annals of Mathematics and Artificial Intelligence manuscript No.**
(will be inserted by the editor)

# Learning Tractable NAT-Modeled Bayesian Networks

**Yang Xiang · Qian Wang**

Mar 2021

**Abstract** Bayesian networks (BNs) encode conditional independence to avoid combinatorial explosion on the number of variables, but are subject to exponential growth of space and inference time on the number of causes per effect variable. Among space-efficient local models, we focus on the Non-Impeding Noisy-AND Tree (NIN-AND Tree or NAT) models, due to their multiple merits, and on NAT-modeled BNs, where each multi-parent variable family may be encoded as a NAT-model. Although BN inference is generally exponential on treewidth, the inference is tractable with NAT-modeled BNs of high treewidth and low density. In this work, we present the first study to learn NAT-modeled BNs from data. We apply the MDL principle to learning NAT-modeled BNs by developing a corresponding scoring function, and we couple it with heuristic structure search. We show that when data satisfy NAT causal independence, high treewidth, and low density structure, learning underlying NAT modeled BNs is feasible.

**Keywords** Bayesian networks · Causal independence models · Probabilistic inference · Local structures · Machine Learning.

## 1 Introduction

Discrete BNs avoid combinatorial explosion on the number of variables by encoding conditional independence in directed acyclic graph (DAG) structures, but space and inference time grow exponentially in the number of causes per effect due to tabular conditional probability tables (CPTs). Space-efficient local models exist, such as noisy-OR, noisy-MAX [7], context-specific independence (CSI) [1], NAT [13], DeMorgan [9], tensor-decomposition [11], and cancellation [12].

We focus on NAT models due to merits of simple causal interactions (reinforcement/undermining), expressiveness (recursive mixture of causal interactions,

Yang Xiang · Qian Wang
School of Computer Science, University of Guelph, Canada
Corresponding author: Yang Xiang
Email: yxiang@uoguelph.ca

multi-valued, ordinal or nominal [15]), generality (generalizing noisy-OR, noisy-MAX, and DeMorgan), and orthogonality to CSI. Although BN inference is generally exponential on treewidth, the inference is tractable with NAT-modeled BNs of high treewidth and low density.

Specifically, the space of a BN (measured by the total number of CPT parameters) is $O(n\ s^{\kappa})$, where $n$ is the number of variables, $s$ bounds domain sizes of variables, and $\kappa$ bounds numbers of causes (parents) per variable. In fully NAT-modeled BNs (see Section 2), variables quantify dependency on parents by NAT models instead of tabular CPTs. Their space is $O(n\ s\ \kappa)$. This efficiency extends to inference time with NAT modeled BNs of high treewidth (bounded by $\kappa$) and low density (measured by percentage of arcs beyond being singly connected) structures.

This work studies learning NAT-model BNs from data. A BN can be compressed into a fully NAT-modeled BN [15]. However, since the source BN must be either manually constructed or learned from data through other methods, the compression approach does not completely solve knowledge acquisition for NAT-modeled BNs.

The main contribution of this work is the first study on learning NAT-modeled BNs directly from data. We apply the MDL principle [10] to learning NAT-modeled BNs to develop a NAT-enabled scoring function, and couple it with heuristic structure search. Our experiment shows that when data satisfy NAT causal independence, high treewidth, and low density structure, it is feasible to learn underlying NAT-modeled BNs that enable inference efficiency and accuracy.

In developing the MDL function, we resolve the following issues: We propose a decomposition of description length for NAT-modeled BNs. We show how to incorporate into description length persistent leaky causes (see Section 4.1) discovered during learning. We reveal the break-down of MDL decomposability, and propose remedy to maintain accuracy of MDL score and learning efficiency. We identify the role of NAT compression in learning NAT-modeled BNs.

The remainder is organized as follows: We review terminology on NAT-modeled BNs in Section 2. Section 3 motivates the task of learning NAT-modeled BNs. Decomposition of MDL scoring function for NAT-modeled BNs is presented in Section 4, with computation of sub-scores elaborated in Sections 4.1 through 4.4. The structure search is described in Section 5 with complexity analysis. Techniques for improving search efficiency are presented in Section 6. Experimental results are reported in Section 7. Two general applications of NAT-modeled BNs can be identified. This work opens the door for possibility of a third, which is discussed in Section 8 along with future research.

## 2 NAT-modeled Bayesian Networks

We review terminology on NAT-modeled BNs.

### 2.1 Causal Variables and Causal Events

A NAT model is defined over an effect $e$ and a set of $\kappa \geq 2$ *uncertain* causes $C = \{c_1, ..., c_\kappa\}$, where $e \in D_e = \{e^0, ..., e^\eta\}$ $(\eta \geq 1)$ and $c_i \in \{c_i^0, ..., c_i^{m_i}\}$ $(i =$

$1, ..., \kappa, m_i \geq 1$). $C$ and $e$ form one family (a child variable plus its parents) in BNs. Values $e^0$ and $c_i^0$ are *inactive*. Other values (may be written as $e^+$ or $c_i^+$) are *active*. Note that $D_e$ needs not be ordered: Although other local models such as noisy-OR and noisy-MAX have been defined over ordinal variables (also referred to as graded variables), NAT models are defined over both ordinal and nominal variables [15]. That causes in a NAT model are uncertain implies the following (they can cause effect to be active, but do not always do so):

$$0 < P(e^k | c_1^{j_1}, ..., c_\kappa^{j_\kappa}) < 1 \quad (k > 0, \exists_i j_i > 0). \tag{1}$$

That $C$ is the set of all causes implies the following:

$$P(e^0 | c_1^0, ..., c_\kappa^0) = 1. \tag{2}$$

A causal event is a *success* or *failure* depending on if $e$ is active up to a given value, is *single-* or *multi-causal* depending on the number of active causes, and is *simple* or *congregate* depending on the value range of $e$. For instance, $P(e^k \leftarrow c_i^j) = P(e^k | c_i^j, c_z^0 : \forall z \neq i)$ $(j > 0)$ is probability of a *simple single-causal success*. $P(e \geq e^k \leftarrow c_1^{j_1}, ..., c_q^{j_q})$

$$= P(e \geq e^k | c_1^{j_1}, ..., c_q^{j_q}, c_z^0 : c_z \in C \setminus X)$$

is probability of a *congregate multi-causal success*, where $j_1, ..., j_q > 0$, $X = \{c_1, ..., c_q\}$ $(q > 1)$, and it may be denoted as $P(e \geq e^k \leftarrow \underline{x}^+)$ where $\underline{x}^+$ corresponds to $X$.

## 2.2 NAT Models

Interactions among causes in a NAT model may be reinforcing or undermining: Let $e^k$ be an active effect value, $R = \{W_1, ..., W_\phi\}$ $(\phi \geq 2)$ be a partition of a set $X \subseteq C$ of causes, $S \subset R$, and $Y = \cup_{W_i \in S} W_i$. Sets of causes in $R$ *reinforce* each other relative to $e^k$, iff

$$\forall S \; P(e \geq e^k \leftarrow \underline{y}^+) \leq P(e \geq e^k \leftarrow \underline{x}^+), \tag{3}$$

where $\underline{y}^+$ corresponds to $Y$. They *undermine* each other iff

$$\forall S \; P(e \geq e^k \leftarrow \underline{y}^+) > P(e \geq e^k \leftarrow \underline{x}^+). \tag{4}$$

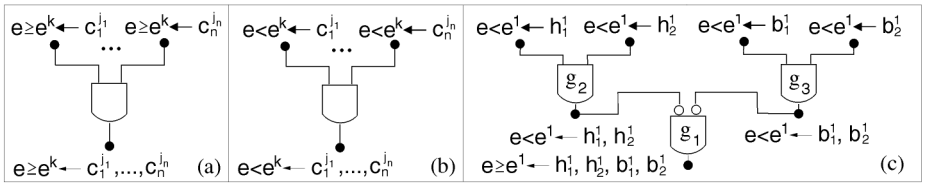


**Fig. 1** Direct (a) and dual (b) NIN-AND gates. (c) NAT.

A NAT consists of multiple NIN-AND (Non-Impeding Noisy-AND) gates. A *direct* gate involves disjoint sets of causes $W_1, ..., W_\phi$. Each input event is a success $e \geq e^k \leftarrow \underline{w}_i^+$ $(i = 1, ..., \phi)$, e.g., Fig. 1 (a) where each $W_i$ is a singleton. The output event is $e \geq e^k \leftarrow \underline{w}_1^+, ..., \underline{w}_\phi^+$ with probability

$$P(e \geq e^k \leftarrow \underline{w}_1^+, ..., \underline{w}_\phi^+) = \prod_{i=1}^{\phi} P(e \geq e^k \leftarrow \underline{w}_i^+), \tag{5}$$

and hence direct gates encode undermining.

Each input event of a *dual* gate is a failure $e < e^k \leftarrow \underline{w}_i^+$, e.g., Fig. 1 (b). The output event is $e < e^k \leftarrow \underline{w}_1^+, ..., \underline{w}_\phi^+$ with probability

$$P(e < e^k \leftarrow \underline{w}_1^+, ..., \underline{w}_\phi^+) = \prod_{i=1}^{\phi} P(e < e^k \leftarrow \underline{w}_i^+), \tag{6}$$

and hence dual gates encode reinforcement.

Fig. 1 (c) shows a NAT, where causes $h_1$ and $h_2$ reinforce, so do $b_1$ and $b_2$, but the two groups undermine each other. Each NAT uniquely defines pairwise causal interaction (PCI) between each pair of causes $c_i$ and $c_j$ $(i \neq j)$, denoted by a PCI bit $\pi(c_i, c_j) \in \{u, r\}$, where $u$ and $r$ denote reinforcing and undermining, respectively. The collection of PCI bits, one per pair of causes, is a *PCI pattern*, which uniquely specifies the corresponding NAT [14].

A NAT is quantified by *single-causals* (probability parameters of root events), in the form $P(e^k \leftarrow c_i^j)$ $(j, k > 0)$. From the NAT and single-causals, $P(e \geq e^1 \leftarrow h_1^1, h_2^1, b_1^1, b_2^1)$, as well as other values of CPT $P(e|h_1, h_2, b_1, b_2)$, are uniquely specified.

### 2.3 Persistent Leaky Causes

The *leaky* cause for an effect $e$ represents all causes of $e$ that are not explicitly named. A leaky cause may be persistent [7,15]. A *non-persistent* leaky cause can be modeled as other causes. In such cases, we denote all causes of effect $e$ by $c_1, ..., c_\kappa$. If the leaky cause is non-persistent, we assume that one of $c_1, ..., c_\kappa$ is the leaky cause.

A *persistent* leaky cause is always active. We integrate all persistent leaky causes of the same effect into a single cause, and denote the leaky cause by $c_0$. In such cases, we denote other causes of effect $e$ by $c_1, ..., c_\kappa$.

### 2.4 NAT-modeled BNs

A BN, where CPTs of some variable families are NAT models, is a *NAT-modeled BN*. We denote a NAT-modeled BN by $\mathcal{N} = (V, G, \Omega, \Theta)$, where $V = \{X_1, X_2, ...\}$ is a finite set of variables. $G$ is a DAG with nodes one-to-one mapped to variables in $V$. A variable plus its parent variables in $G$ form a *family*. $\Omega$ is a set of CPTs one for each variable family whose dependency is quantified by tabular CPT. $\Theta$ is a set of NAT models one for each variable family whose dependency is quantified

by NAT model. The family of every variable in $V$ is covered either by $\Omega$ or by $\Theta$, but not both.

Fig. 2 illustrates a *NAT-modeled BN*. Family of $X_8$ is a NAT model, whose NAT is in (b) (simplified notation). Gate $g_3$ is dual, and $g_1$ and $g_2$ are direct.

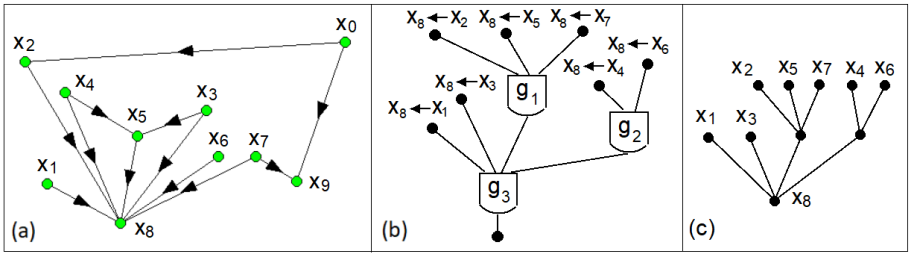


**Fig. 2** (a) DAG of a NAT-model BN. (b) NAT structure over the family of $X_8$. (c) Root labeled tree of NAT in (b).

A BN is *fully* NAT-modeled, if the family dependency of every multi-parent variable is encoded as a NAT model. The above BN is fully NAT-modeled, if families of $X_5, X_8, X_9$ are all NAT-modeled.

### 2.5 Compression of NAT Models

An arbitrary tabular CPT (referred to as *source CPT*) can be approximated by a NAT model, termed *compression*, as follows: The CPT is analyzed to determine causal interaction (reinforcing or undermining) between every pair of causes if possible, with the result being a pair-wise causal interaction (PCI) pattern. From the PCI pattern, compatible candidate NATs are extracted, and are parameterized into NAT models through constrained gradient descent search. The output NAT is selected to minimize a distance measure between source CPT and the NAT CPT.

The above key steps of compression process, PCI pattern recognition, NAT extraction, and NAT parameterization, have the following complexity: Complexity of PCI pattern recognition for a variable of $\kappa$ parents is $O(\kappa^2)$ [15]. For PCI pattern with no missing bits and compatible with a NAT topology, complexity of NAT extraction is $O(\kappa^2\ 2^\kappa)$ [14]. For PCI pattern with $\beta$ missing bits, the complexity grows to $O(\kappa^2\ 2^{\beta+\kappa})$. For PCI pattern incompatible with any NAT topology, further computation is needed to search for a NAT whose PCI pattern is closest. NAT model parameterization can be done by constrained gradient descent [15]. Let $s$ bounds variable domain sizes, and $\tau$ bounds steps in gradient descent. Then a NAT model has up to $\kappa\ s^2$ parameters. Complexity of parameterizing one candidate NAT is $O(\kappa\ s^2\ \tau)$. When the recognized PCI pattern has $\beta$ missing bits, $2^\beta$ NATs are parameterized and the best NAT model is selected from them, yielding the complexity of $O(\kappa\ s^2\ \tau\ 2^\beta)$.

A fully NAT-modeled BN can be obtained from a normal BN by compressing its tabular CPTs into NAT models. As the result, the space complexity of the BN is reduced from $O(n\ s^\kappa)$ to $O(n\ s\ \kappa)$.

### 2.6 BN Structure Learning by MDL

A common approach for learning BN structures from data is to combine heuristic search of structures with a scoring function. The MDL scoring function is based on the MDL principle [10] that views the best model of a dataset as one minimizing sum of encoding lengths of the model and the data. More accurate models reduce data encoding length, but increase model encoding length as they are more complex. Hence, the MDL principle offers a trade-off between accuracy and efficiency.

The MDL scoring function satisfies some desirable conditions, including score equivalence [2] and decomposability. Two BN DAG structures $G$ and $G'$ are *Markov equivalent*, if for every BN based on $G$, there exists a BN based on $G'$ that encodes the same probability distribution, and vice versa. A scoring function of BN DAG structures is *score equivalent*, if it scores Markov equivalent DAG structures identically. A scoring function of BN DAG structures is *decomposable* if it can be expressed as a sum of terms each of which involves only the family of a single variable.

## 3 Learning NAT-modeled BNs

A NAT-modeled BN can be obtained by first learning a BN with tabular CPTs (referred to as *tabular BN* below) from data, and compressing it into a fully NAT-modeled BN. This approach has several limitations:

First, it relies on other methods to acquire a tabular BN, and does not completely resolve acquisition for NAT-modeled BNs. Second, the DAG structure of the resultant NAT-modeled BN is the same as the tabular BN. Since the structure was obtained independently of NAT-modeling, it may not be the most suitable structure for the NAT-modeled BN both efficiency-wise and accuracy-wise. Third, a tabular CPT may not be accurately approximated by a NAT model. The approach does not guide the compression explicitly by trade-off between representational efficiency and accuracy. To overcome these limitations, this work studies learning NAT-modeled BNs directly from data.

A common approach for learning BN structures from data is to combine heuristic search of structures with a scoring function, e.g., BD [6] and MDL. In this work, we extend structure learning of BNs to learning NAT-modeled BNs using a NAT-enabled MDL scoring function to trade off representational efficiency and accuracy. Our work also belongs to structure learning with local structures, e.g., [4, 3]. The main difference is that their work focused on equality constraints such as CSI, with decision trees or decision graphs as local structures. Our current work focuses on inequality constraints such as Eqns. (3) and (4), with NAT models as local structures. Therefore, our work complements existing BN structure learning with local structures.

We develop a NAT-enabled MDL score below, couple with a heuristic structure search, and evaluate feasibility of learning NAT-modeled BNs from data experimentally.

## 4 NAT-Enabled MDL Scoring Function

Since a tabular CPT may not be sufficiently accurately approximated by a NAT model, we make the compression decision by a NAT-enabled MDL score. The MDL function for a tabular BN is additively decomposed into the model description length and the data description length. Both need to be extended to allow NAT modeling. We assume that the dataset $D$ involves $n$ variables $X_1, ..., X_n$ (additional variables may be introduced due to NAT modeling as shown below), and includes $N$ data records. Since the MDL function is score equivalent and decomposable (see below on its violation due to NAT-modeling and the proposed solution), we consider description length over a single family, made of a variable $X_i$ and its parent set $\pi_i$.

To allow trade-off of representational efficiency and accuracy, dependency of $X_i$ on $\pi_i$ may be expressed by tabular CPT or NAT model. With tabular CPT, description length of the $X_i$ family is decomposed into model description length and data description length:

$$DL_{TabCpt} = DL_{Model} + DL_{TabData},$$

where $X_i$ in $DL_{TabCpt}(X_i)$ and other terms are omitted.

Model description length over the family can be further decomposed into DAG description length (relative to DAG structure over the family) and CPT description length (relative to tabular CPT):

$$DL_{TabCpt} = DL_{Dag} + DL_{Cpt} + DL_{TabData}. \tag{7}$$

A NAT-modeled BN has a global structure (the DAG) and local structures (NATs). If the $X_i$ family forms a NAT model, its DAG description length is similar to that of a tabular CPT (except the difference in Section 4.3). Since the NAT CPT is defined by NAT structure and single-causals, CPT description length is decomposed into NAT description length and single-causal description length:

$$DL_{NatMod} = DL_{Dag} + DL_{Nat} + DL_{Sc} + DL_{NatData}. \tag{8}$$

As will be seen, data description length also differs depending on whether tabular CPT or NAT model is used (Section 4.4), and hence the naming of $DL_{TabData}$ and $DL_{NatData}$. The following subsections present these components of description length separately in the order of

$$DL_{Sc}, DL_{Nat}, DL_{Dag}, DL_{NatData}.$$

### 4.1 Single-Causal Description Length

Let $s_i$ denote the domain size of $X_i$, $\kappa_i = |\pi_i|$ denote the number of parents of $X_i$, and $s_{ij}$ denote the domain size of the $j$th parent of $X_i$. If $q_i$ denotes the number of configurations of $\pi_i$, we have $q_i = \prod_{j=1}^{\kappa_i} s_{ij}$. If $X_i$ has a tabular CPT $P(X_i|\pi_i)$, the number of CPT parameters is $(s_i - 1)q_i$. Each parameter is typically encoded with $\frac{1}{2}log_2(N)$ bits [5]. Hence, the CPT description length is

$$DL_{Cpt} = \frac{1}{2}log_2(N)(s_i - 1)\prod_{j=1}^{\kappa_i} s_{ij} \ (bits). \tag{9}$$

If the $X_i$ family is NAT modeled, where $\pi_i$ is the set of all causes of $X_i$ (see below for alternative), the number of single-causals needed to specify the NAT model is

$$(s_i - 1)\sum_{j=1}^{\kappa_i}(s_{ij} - 1).$$

Encoding each parameter with $\frac{1}{2}log_2(N)$ bits, single-causal description length of the $X_i$ family is

$$DL_{Sc} = \frac{1}{2}log_2(N)(s_i - 1)\sum_{j=1}^{\kappa_i}(s_{ij} - 1)\ (bits). \tag{10}$$

The above description length is applicable when $X_i$ family has no persistent leaky cause, but must be extended otherwise. Recall from Section 2.3 that a non-persistent leaky cause can be modeled as other causes. We denote all causes of effect $e$ by $c_1, ..., c_\kappa$, with one of them being the leaky cause. When the $X_i$ family forms a NAT model, we have $e = X_i$ and $\kappa = \kappa_i$. A source CPT with a non-persistent leaky cause has a fully specified $P(e|c_1, ..., c_\kappa)$, where $P(e^0|c_1^0, ..., c_\kappa^0) = 1$ and $P(e^k|c_1^0, ..., c_\kappa^0) = 0$ for $k > 0$. Hence, Eqn. (2) holds.

If effect $e$ has persistent leaky causes, we integrate all persistent leaky causes of $e$ into a single cause $c_0$, and denote other causes of effect $e$ by $c_1, ..., c_\kappa$. Since $c_0$ is persistent, we have $c_0 \in \{c_0^0, c_0^1\}$, and $c_0 = c_0^1$ always holds. Because conditions $(c_0^0, c_1, ..., c_\kappa)$ never hold, and parameters $P(e|c_0^0, c_1, ..., c_\kappa)$ are not empirically available, a source CPT has the form $Q(e|c_1, ..., c_\kappa) = P(e|c_0^1, c_1, ..., c_\kappa)$. Since $c_0$ is an uncertain cause, by Eqn. (1), we have

$$0 < P(e|c_0^1, c_1^0, ..., c_\kappa^0) = Q(e|c_1, ..., c_\kappa) < 1.$$

Hence, Eqn. (2) does not hold with the source CPT $Q(e|c_1, ..., c_\kappa)$, which triggers identification during learning: It signifies that if this $X_i$ family forms a NAT model, it involves a persistent leaky cause. The source CPT $Q(e|c_1, ..., c_\kappa)$ of $\kappa$ causes is compressed into $P_{Nat}(e|c_0, c_1, ..., c_\kappa)$ of $\kappa + 1$ causes.

Due to the extra persistent leaky cause of the NAT model, and that it is binary, additional $s_i - 1$ single-causals are needed to specify the NAT model: $P(e^k \leftarrow c_0^1)\ \ (k > 0)$. Hence, when the family of $X_i$ forms a NAT model with the persistent leaky cause, Eqn. (10) no longer applies. Instead, the single-causal description length is

$$DL_{Sc} = \frac{1}{2}log_2(N)(s_i - 1)(1 + \sum_{j=1}^{\kappa_i}(s_{ij} - 1)). \tag{11}$$

### 4.2 NAT Description Length

We consider two options of NAT encoding. A NAT can be expressed as *root labeled tree* (RLT), where a node represents a root event, or the leaf event, or a gate, preserving the tree topology. The RLT of NAT in Fig. 2 (b) is shown in (c). When the family of $X_i$ forms a NAT model, the NAT can be encoded by encoding RLT, e.g., encoding parent set of each node in the RLT. For RLT of $m$ nodes, encoding

index of each node takes $log_2(m)$ bits. For instance, if $\kappa_i = 3$, the RLT has as fewer as 4 nodes (3 roots and 1 leaf). Encoding index of each node with $log_2(4) = 2$ bits, the RLT can be encoded with $2 * 4 = 8$ bits. If $\kappa_i = 15$, the RLT has as fewer as 16 nodes, and can be encoded with $4 * 16 = 64$ bits.

Alternatively, the RLT can be encoded by encoding its unique PCI pattern. With $\kappa_i = |\pi_i|$ denoting the number of parents of $X_i$, encoding PCI pattern takes $\kappa_i\ (\kappa_i - 1)/2$ bits. When $\kappa_i = 3$, we need 3 bits. When $\kappa_i = 15$, we need $15 * 14/2 = 105$ bits.

The above shows that none of the options dominates the other: For simplicity, we use PCI pattern-based encoding. If the family of $X_i$ forms a NAT model without persistent leaky cause, the NAT description length is

$$DL_{Nat} = \frac{1}{2}\kappa_i(\kappa_i - 1)\ (bits). \tag{12}$$

If the NAT model involves a persistent leaky cause, the number of causes increases to $\kappa_i + 1$. The description length is

$$DL_{Nat} = \frac{1}{2}\kappa_i(\kappa_i + 1)\ (bits). \tag{13}$$

Although we have selected PCI pattern-based encoding for simplicity, we analyze below that the choice has no significant impact: The total number of nodes in a RLT is between $\kappa_i + 1$ and $2\kappa_i - 1$. Hence, a RLT based encoding has a description length between $(\kappa_i + 1)log_2(\kappa_i + 1)$ and $(\kappa_i + 1)log_2(2\kappa_i - 1)$. For $\kappa_i = 15$, it is between 64 bit and 80 bits. If PCI pattern-based encoding is used, the description length is 105 bits by Eqn. (12). Hence, the maximum difference made by the choice is 41 bits.

On the other hand, if a binary $X_i$ family is modeled by a tabular CPT, the CPT description length is $DL_{Cpt} = 163,840$ bits by Eqn. (9), where $N = 1024$ is assumed. If the $X_i$ family is a NAT model, the single-causal description length is $DL_{Sc} = 75$ bits by Eqn. (10). Hence, the difference in NAT description length due to RLT or PCI encoding cannot tip the comparison due to difference between $DL_{Cpt}$ and $DL_{Sc}$. That is, whether the learning outcome for $X_i$ family is a NAT or a tabular CPT cannot be influenced by the choice between RLT and PCI encoding.

What is left is whether the choice between RLT and PCI encoding can significantly influence which NAT will be selected. As shown above, for $\kappa_i = 15$, the difference made by the encoding choice is either 16 bits (RLT) or 0 bit (PCI). The 16 bits cannot tip the comparison between accurate and inaccurate NAT models due to difference in their data description length (presented below).

### 4.3 DAG Description Length

DAG description length over the family of $X_i$ encodes parent set $\pi_i$. Let $n$ be the number of variables in the dataset $D$. It takes $log_2(n)$ bits to encode the index of each node. For tabular BNs, DAG description length of the family of $X_i$ is

$$DL_{Dag} = log_2(n)\ \kappa_i.$$

For NAT-modeled BNs, the number of nodes in the DAG may be greater than $n$, invalidating the above. For each NAT family with a persistent leaky cause, an

extra node is introduced. The extra node has both local and global impact to DAG description length:

Locally, since $X_i$ has an extra parent, factor $\kappa_i$ in $DL_{Dag}$ becomes $1 + \kappa_i$. Globally, with the extra variable, $log_2(n)$ bits are insufficient to encode index of each node. Let $\beta$ be the total number of persistent leaky cause variables (each over a distinct variable family) introduced at a given time during learning. Then DAG description length over the family of $X_i$ is

$$DL_{Dag} = log_2(n + \beta)\ (1 + \kappa_i)\ bits. \tag{14}$$

Since $\beta$ changes as learning proceeds, existence of $\beta$ in $DL_{Dag}$ introduces dependency between description lengths over different variable families, and breaks down decomposability of the MDL function.

To ensure accuracy of MDL score, we deployed the following: When a new NAT family with persistent leaky cause is learned, or such a family learned earlier is invalidated during learning, we apply a global updating of family scores to adjust $\beta$ value in $log_2(n+\beta)$ coefficient: Even when the family of a varaible $X_j$ is unchanged, its $DL_{Dag}$ will be updated into

$$DL_{Dag} = log_2(n + \beta)\ \kappa_j\ bits. \tag{15}$$

This allows decomposability of the MDL function to persist between NAT family updates, and enables efficient score computation.

### 4.4 Data Description Length

When CPT of $X_i$ is tabular, data description length over the family is

$$DL_{TabData} = -\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} s_{ijk}\ log_2\Big(\frac{s_{ijk}}{M_{ij}}\Big),$$

where $s_{ijk}$ counts family configurations $(X_i = k, \pi_i = \pi_{ij})$ in $D$, $M_{ij}$ counts parent configurations $(\pi_i = \pi_{ij})$, and $s_{ijk}/M_{ij}$ estimates $P(X_i = k|\pi_i = \pi_{ij})$.

When family of $X_i$ is a NAT model $\Theta_i$, the above data description length must replace $P(X_i = k|\pi_i = \pi_{ij})$ with $P_{\Theta_i}(X_i = k|\pi_i = \pi_{ij})$ defined by the NAT model CPT. Data description length over the family of $X_i$ is

$$DL_{NatData} = -\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} s_{ijk}\ log_2\Big(P_{\Theta_i}(X_i = k|\pi_i = \pi_{ij})\Big).$$

The above requires fully specifying the NAT model $\Theta_i$. To do so, we first estimate tabular CPT $P(X_i|\pi_i)$ from $D$, and then compress the CPT into $\Theta_i$. Since computation of $DL_{NatData}$ requires compression, it is significantly more costly than $DL_{TabData}$ in learning tabular BNs.

## 5 Heuristic Search and Complexity

### 5.1 Heuristic Search Algorithm

As learning BNs from data is NP-complete, a number of heuristics have been proposed for search of alternative structures. One may start with a complete graph, remove links by conditional independence, and orient the resultant graph. One may also start with an empty graph, add arcs until no further addition improves the score, and then remove arcs until no further removal improves the score. Alternatively, arcs may be added, deleted, or reversed until it is no longer possible to improve the score. The search may also be organized by orderings of variables, rather than by DAGs. A recent summary of search heuristics in BN structure learning can be found in [8]. As the first study of learning NAT-modeled BNs, a heuristic similar to that of [4] is extended in our learning algorithm, referred to as $LearnNatBn$.

**Algorithm 1** $LearnNatBn(D, V)$

*1 preprocess D into a set F of frequencies of complete configurations over V;*
*2 init G to empty DAG, DL(G) to infinity, $\Omega$ = nul, $\Theta$ = nul;*
*3 Done = false;*
*4 while Done = false,*
*5 (G', DL(G'), $\Omega'$, $\Theta'$) = ImproveNatBn(G, DL(G), $\Omega$, $\Theta$, F);*
*6 if $DL(G') < DL(G)$,*
*7 G = G', DL(G) = DL(G'), $\Omega = \Omega'$, $\Theta = \Theta'$;*
*8 else Done = true;*
*9 return NAT-modeled BN (G, $\Omega$, $\Theta$);*

$LearnNatBn$ takes as input a dataset $D$ over a set $V$ of variables. It learns a NAT-modeled BN $\mathcal{N} = (G, \Omega, \Theta)$. $G$ is a DAG possibly over a superset of $V$ (due to persistent leaky causes). $\Omega$ is a set of CPTs one for each variable family whose dependency is quantified by tabular CPT. $\Theta$ is a set of NAT models one for each variable family whose dependency is quantified by NAT model. Note that our notation $\mathcal{N} = (G, \Omega, \Theta)$ here differs slightly from that in Section 2.4, leaving $V$ implicit. This is because the set of variables in a learned NAT-modeled BN may be a proper superset of variables from $D$.

$LearnNatBn$ starts with an empty DAG (with nodes being one-to-one mapped to $V$ and without arcs). The MDL score of a DAG $G$ is denoted as $DL(G)$. Search proceeds in multiple rounds (*while* loop in lines 4 to 8). Each round consists of one execution of $ImproveNatBn$ (Algo. 2), which tries to find a DAG $G'$ that differs from $G$ by one arc, whose score $DL(G')$ is better than $DL(G)$. Search terminates when no such DAG can be found to improve the score.

**Algorithm 2** *ImproveNatBn(G, DL(G), $\Omega$, $\Theta$, F)*

*1 $G^* = G$, $DL(G^*) = DL(G)$, $\Omega^* = \Omega$, and $\Theta^* = \Theta$;*
*2 for each pair of variables $(X_k, X_j)$,*
*3 for each valid operation Op on G over $(X_k, X_j)$,*
*4 apply Op on G to obtain graph G',*
*5 if G' is cyclic, continue;*
*6 for each variable $X_i$ in G',*

*7* *estimate* $P(X_i|\pi_i)$ *from* $F$*;*
*8* *compute tabular CPT based* $DL_{TabCpt}(X_i)$ *over family of* $X_i$ *from* $P(X_i|\pi_i)$*;*
*9* *initialize* $DL(X_i) = DL_{TabCpt}(X_i)$*;*
*10* *for each variable* $X_i$ *in G' where* $|\pi_i| < 2$*, add tabular CPT* $P(X_i|\pi_i)$ *to* $\Omega'$*;*
*11* *for each variable* $X_i$ *in G' where* $|\pi_i| \geq 2$*,*
*12* *compress* $P(X_i|\pi_i)$ *into a NAT model* $\Theta_i = (T_i, SC_i)$*;*
*13* *compute NAT model based* $DL_{NatMod}(X_i)$ *from* $\Theta_i$*;*
*14* *if* $DL_{NatMod}(X_i) < DL_{TabCpt}(X_i)$*,*
*15* *set* $DL(X_i) = DL_{NatMod}(X_i)$*;*
*16* *add NAT model* $\Theta_i$ *to* $\Theta'$*;*
*17* *else add tabular CPT* $P(X_i|\pi_i)$ *to* $\Omega'$*;*
*18* *compute* $DL(G') = \sum_i DL(X_i)$*;*
*19* *if* $DL(G') < DL(G^*)$*,*
*20* *if number of persistent leaky causes in* $G'$ *changed from* $G^*$*,*
*21* *update* $DL_{Dag}$ *for each variable family and* $DL(G')$*;*
*22* $G^* = G'$*,* $DL(G^*) = DL(G')$*,* $\Omega^* = \Omega'$*, and* $\Theta^* = \Theta'$*;*
*23 return* $(G^*, DL(G^*), \Omega^*, \Theta^*)$*;*

For each pair of nodes $\{X_k, X_j\}$ in $G$, $ImproveNatBn$ applies a single-arc based operation $Op$. If $(X_k, X_j)$ is not an arc in $G$, there are two alternatives of $Op$: either arc $(X_k, X_j)$ or $(X_j, X_k)$ is added to $G$. If $(X_k, X_j)$ is an existing arc in $G$, there are also two alternatives of $Op$: either $(X_k, X_j)$ is deleted from $G$, or $(X_k, X_j)$ is reversed into arc $(X_j, X_k)$. The suitable operation is selected in line 3.

For each newly formed variable family in the current DAG, MDL sub-scores are computed for both tabular CPT and NAT model. For $X_i$ with $\kappa_i = |\pi_i|$, the number of alternative NAT models over the $X_i$ family is super-exponential in $\kappa_i$. Instead of computing a MDL sub-score for each NAT model, we compress the tabular CPT (estimated from data) into a NAT model. That is, the search through the NAT space is conducted by compression, and the best NAT model found is MDL-scored. Decision to model the family as tabular CPT or NAT model is made by comparing the two sub-scores.

More specifically, a NAT model over the family of variable $X_i$ is denoted $\Theta_i = (T_i, SC_i)$, where $T_i$ is the RLT and $SC_i$ is the set of single-causals. For each $X_i$, a tabular CPT $P(X_i|\pi_i)$ is estimated from $F$ (line 7). If $X_i$ has less than two parents, its family description length $DL(X_i)$ is determined by the tabular CPT $DL_{TabCpt}(X_i)$ (lines 6 to 10). If $X_i$ has multiple parents (line 11), the tabular CPT is compressed into a NAT model, and the corresponding description length $DL_{NatMod}(X_i)$ is also computed (lines 12 and 13). If $DL_{NatMod}(X_i)$ is better than $DL_{TabCpt}(X_i)$, the family of $X_i$ will be NAT-modeled (lines 14 to 16). Otherwise, the family dependency is quantified by the tabular CPT (lines 9 and 17). If the NAT model is selected that involves persistent leaky cause, a new variable will be included in the current DAG.

Given a variable family, the NAT model has significantly better space efficiency than the tabular CPT, but may not approximate the dependency accurately as the tabular CPT. By evaluating each variable family alternatively according to tabular CPTs and NAT models, the representation that best trades the efficiency with accuracy is selected. It also allows the DAG structures that best take advantage of NAT-modeling to be explored.

The score for the new structure $G'$ is obtained in line 18, due to decomposability. If it improves $DL(G^*)$, the best structure $G^*$ is updated (lines 19 to 22). Lines 20 and 21 handle breakdown of decomposability as discussed in Section 4.3, which can be caused due to introduction or invalidation of persistent leaky cause at line 12 when the NAT model is compressed. The update to $DL_{Dag}$ in line 21 will be performed according to Eqns. (14) and (15). We assume that the update does not tip the comparison in line 19, and will not elaborate the measure needed otherwise.

After each valid operation for each pair of variables has been processed, the best structure is returned.

### 5.2 Complexity Analysis

For complexity of $LearnNatBn$, denote $n = |V|$. $ImproveNatBn$ evaluates $O(n^2)$ links before one is added, removed, or reversed. Each execution of $ImproveNatBn$ adds at most one arc, and at most $O(n^2)$ arcs may be added. Each arc cannot be repeatedly added, reversed, or deleted, and improve the scoring function each time. That is, an arc can be modified no more than a small number of times. Hence, the number of executions of $ImproveNatBn$ by $LearnNatBn$ is $O(n^2)$, and the complexity of $LearnNatBn$ is $O(n^4)$.

Note that NAT-model compression (line 12 of $ImproveNatBn$) involves significant computation. Its complexity is reviewed in Section 2.5, is left implicit in the above analysis, and must be counted for.

Before $LearnNatBn$, $D$ is pre-processed into a set $F$ of frequencies of unique records. $|F|$ is significantly $< |D|$, and complexity of $LearnNatBn$ is linear on $|F|$.

## 6 Improving Search Efficiency

Due to compression, learning NAT-modeled BNs is more costly. To improve efficiency, we apply or develop several techniques. $ImproveNatBn$ evaluates revisions to current DAG $G$. Connection of two variables in $G$ is referred to as *arc* if direction is of concern, and as *link* otherwise. If disconnected, the arc/link is *absent*.

**Limiting Links to Evaluate** $ImproveNatBn$ evaluates up to $n(n-1)$ arcs before one arc is modified. We reduce the number of links to evaluate based on search efficiency as mentioned above, and an additional factor: For many applications, a tractable model that yields approximate posteriors is more useful than an intractable model that yields exact posteriors. NAT-modeled BNs have significant gain in inference efficiency if their structures have high tree-width but low density, where low density typically amounts to less than about 1.5 $n$ arcs. Before $LearnNatBn$ starts, we compute mutual information between each pair of variables, and rank them. $ImproveNatBn$ only evaluates, say, $1.5n$ variable pairs that are top ranked. Its complexity is reduced from $O(n^2)$ to $O(n)$.

**Limiting Arc Addition** $ImproveNatBn$ evaluates each link with 2 arc operations. It is possible to avoid evaluating one or both, without causing error to MDL score. An *isolated link* has degree 1 for each end. If absent, 2 alternative arcs may be added. Since MDL score is independent of the direction, evaluating addition of 1 arc suffices. A DAG of $n$ nodes and $m < n-1$ singly connected links can add

up to $n-1-m$ links and be singly connected. Most of them are absent isolated. Default evaluation has $2(n-1-m)$ arc additions, and a half can be avoided.

A *one-end connected* link has degree 1 for one endpoint and degree 2 or more for the other endpoint (*connected end*). The above saving can be extended to one-end connected links, if it is absent and its connected end has indegree 0. When the link is added, its MDL score contribution is independent of arc direction. Hence, evaluating addition of one arc is sufficient.

**Limiting Arc Deletion and Reversal** A *current isolated arc* is an isolated link of a particular direction in current DAG. By default, $ImproveNatBn$ evaluates both its deletion and reversal. Its deletion cannot improve MDL score, since otherwise the arc would not have been added in the first place. Its reversal cannot improve the score either. Hence, both evaluations can be avoided.

A DAG of $n$ nodes has at most $floor(n/2)$ current isolated arcs. For instance, if $n=3$, the number of isolated arcs is at most $floor(1.5)=1$. If $n=4$, the number of isolated arcs is at most $floor(2)=2$. By default, $ImproveNatBn$ evaluates up to $n/2$ current isolated arcs. Hence, up to $n$ arc operations can be safely avoided.

The above saving can be extended to one-end connected links, if it is absent and its connected end has indegree 0. The DAG resultant from reversing the arc is Markov-equivalent, and has the same MDL score. Hence, reversal of the arc will not improve the score. Since the arc was added to improve the score, its deletion will not improve the score either. Both arc operations can be safely avoided.

**Limiting Variable Families to Evaluate** Since our MDL score is decomposable most of the time (Section 4.3 on restoring decomposability), $ImproveNatBn$ can be more efficient by avoiding evaluating family of every variable in each new DAG $G'$ (lines 6, 10, and 11). Only families of variables affected by the current operation $Op$ need to be evaluated. The maximum number of families affected by an operation is two, when an existing arc is reversed. This reduces the number of families for MDL score evaluation in each $ImproveNatBn$ from $O(n)$ to $O(1)$.

## 7 Experimental Study

We conducted 4 sets of experiments. The 1st set evaluate feasibility of learning NAT-modeled BNs from data. The 2nd set further confirms effectiveness in learning local NAT models. The 3rd set of experiments investigate the possibility to apply structure density control in learning besides MDL scoring, to trade efficiency with inference accuracy. The 4th set of experiments investigate the impact of limiting numbers of links to evaluate during learning, as discussed in Section 6. They were conducted using a Dell Inspiron 7520 with i7-3632QM processor at 2.20GHz and 8GB memory through single-thread computation.

### 7.1 Learning NAT-modeled BNs

To establish feasibility of learning NAT-modeled BNs from data, we generated 30 fully NAT-modeled BNs (Fig. 3) in the 1st set of experiments, referred to as *source BNs*. Each source BN consists of 200 binary or ternary variables. The maximum number of parents per variable is 12. The density of the DAG is controlled by

adding 5% extra arcs beyond being singly-connected. Although inference complexity of general BNs is exponential on their treewidth, complexity of NAT-modeled BNs that have high treewidth and low density is linear. The NAT-modeled BNs generated in this set of experiments belong to this subclass of BNs.

Each source BN is transformed to an equivalent *peer* tabular BN, from which a dataset of size $N = 5000$ is sampled as input to $LearnNatBn$.

Among the 200 variable families in each source BN, between 18 and 28% are NAT-models, and the rest have tabular CPTs. In learned NAT-modeled BNs, between 11 and 18% of variable families are NAT-models.

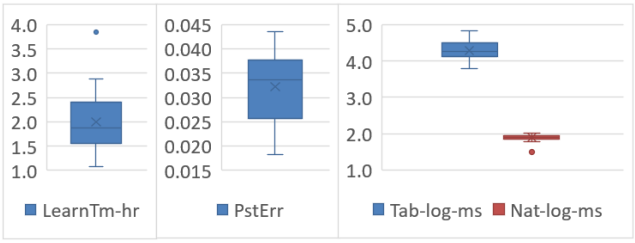


**Fig. 3** Summery of experimental results.

Fig. 3 (left) reports learning time in hours. Learned NAT-modeled BNs are evaluated by accuracy of inference and efficiency gain, relative to the peer BN. Each peer BN is compiled into a junction tree for lazy propagation. Each learned BN is de-causalized [16] and compiled into junction tree. Ten runs of inference are performed on each peer BN and each learned BN, by observing 10% of randomly selected variables.

For inference accuracy, average differences on posterior marginals over all variables (10 runs per BN) are reported in Fig. 3 (middle). Learned NAT-modeled BNs yield sufficiently accurate posteriors with average errors between 0.018 and 0.044.

For efficiency gain in inference, average runtimes (*msec*; 10 runs per BN) in log10 for peer BNs (Tab-log-ms) and learned BNs (Nat-log-ms) are shown in Fig. 3 (right). Learned NAT-modeled BNs are between 110 and 990 times faster in inference.

The experiments suggest that when data satisfy NAT causal independence, and high treewidth, low density structure, it is feasible to learn underlying NAT-modeled BNs that enable inference efficiency and accuracy.

### 7.2 Robust Learning of NAT-modeled BNs

To evaluate robustness of learning, we conducted the 2nd set of experiments, where multiple datasets are sampled from each source BN, a NAT-modeled BN is learnt from each dataset, and the learned BNs are compared.

We generated 10 fully NAT-modeled BNs. Each source BN consists of 60 binary or ternary variables. The maximum number of parents per variable is 12. For most source BNs (9 of them), among 60 variable families in each BN, between 18 and

21% are NAT models. The density of the DAG is controlled by adding 5% extra arcs beyond being singly-connected.

Each source BN is transformed to a peer tabular BN, from which 4 datasets of size $N = 5000$ each are sampled, and a NAT-modeled BN is learnt from each dataset. For most learned BNs (36 out of the 40), between 10 and 15% of variable families are NAT-models.

Five runs of inference are performed on each peer BN and each learned BN, by observing 10% of randomly selected variables. The total number of inference runs is $10 \times (1 + 4) \times 5 = 250$, where 50 runs are performed on the 10 peer BNs, and 200 runs are performed on the 40 learned BNs.

Fig. 4 shows average errors of posterior marginals between peer BNs and learned BNs. The x-axis is indexed by the 200 inference runs with learned BNs, divided into 10 sections (by vertical grid lines). Each section corresponds to 20 runs by 4 BNs learned from datasets with the same source BN. For instance, data points 1 to 20 form 1st section, and show posterior errors of 4 BNs learned from datasets sampled with 1st source BN. Data points 1 to 4 are errors from runs by the 4 learned BNs with the 1st set of observations, data points 5 to 8 are errors by the 4 learned BNs with the 2nd set of observations, and so on. As shown in Fig. 4, for most runs, average posterior errors are between 0.02 and 0.06.

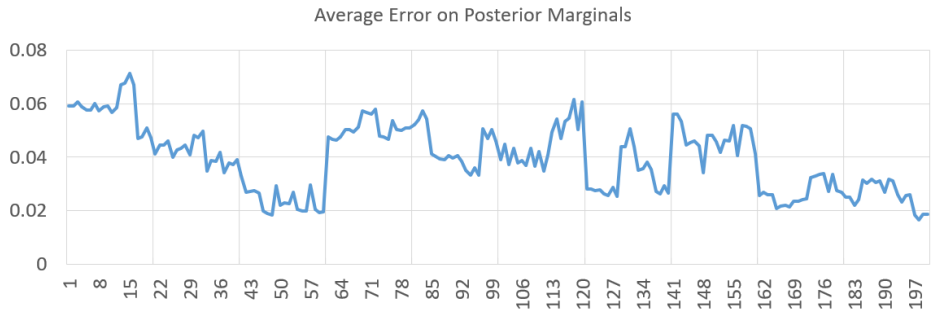


**Fig. 4** Average errors of posterior marginals between source BNs and learned BNs.

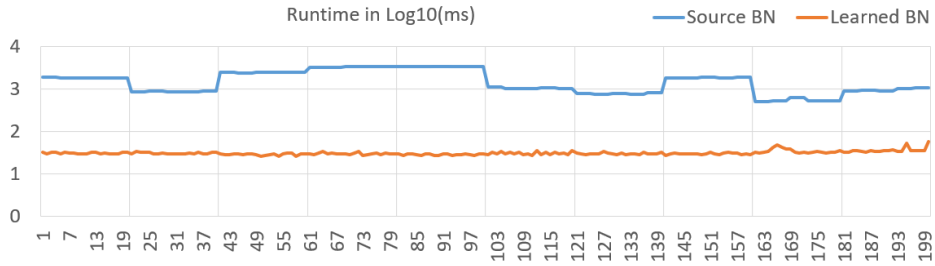


**Fig. 5** Runtime comparision between source peer BNs and learned BNs.

Fig. 5 compares inference runtime between source peer BNs and learned BNs. The x-axis is indexed by 200 inference runs with learned BNs, in the same order as Fig. 4. Each data point for learned BNs is the $log_{10}$ runtime in msec for a particular learned BN with a particular set of observations. The 20 data points of peer BNs in the same section are runtimes from 5 runs of a particular peer BN, one run for each distinct set of observations. For instance, data point 1 to 4 duplicate the same runtime of 1st peer BN with the 1st set of observations. As is shown, runtimes of peer BNs are at least 11 times of that of learned BNs, and are as much as 126 times. This result demonstrates that learned NAT-modeled BNs significantly improve inference efficiency while incurring only small inference errors.

From the perspective of main objective of this set of experiments, the amount of learned NAT-models, posterior errors, and runtimes of NAT-modeled BNs learned from datasets sampled with the same source BN are sufficiently uniform, demonstrating robustness of our learning method.

### 7.3 Randomizing NAT CPTs in Learned NAT-modeled BNs

To further confirm effective learning of NAT-modeled BNs and effectiveness of inference with random observation reported above, we conducted the 3rd set of experiments, where learned NAT-modeled BNs are partially randomized. In particular, each NAT model in the learned NAT-modeled BN is replaced by a random tabular CPT, while the rest of the learned BN remains. Inference accuracy of the resultant BN is then compared with that of the learned BN.

We generated 10 fully NAT-modeled (*source*) BNs, each consisting of 40 binary or ternary variables. The maximum number of parents per variable is 5. The density of the DAG is controlled by adding 5% extra arcs beyond being singly-connected.

From each source BN, a dataset of size $N = 5000$ is sampled as input to $LearnNatBn$, from which a *learned* NAT-modeled BN is obtained. It is then modified by replacing each NAT model with a random tabular CPT, to produce a *modified* BN.

Among the 40 variable families in each source BN, between 20 and 33% are NAT-models, and the rest have tabular CPTs. In learned NAT-modeled BNs, between 5 and 23% of variable families are NAT-models.

Ten runs of inference are performed on each source BN, each learned BN, and each modified BN, by observing 10% of randomly selected variables: a total of $10 \times 10 \times 3 = 300$ inference runs.

For each 3 runs on the corresponding source BN, learned BN, modified BN, and the same set of observations, average difference on posterior marginals over all variables is computed between source BN and learned BN, and between source BN and modified BN. They are referred to as *Learned BN Err* and *Rdmized BN Err*. From the 300 inference runs, 100 pairs of Learned BN Err and Rdmized BN Err are obtained, as shown in Fig. 6.

Out of the 100 pairs of average errors, Rdmized BN Err is larger than Learned BN Err in 93 pairs. Among the 93 pairs, Rdmized BN Err is 200% or above in 19 pairs. The maximum Rdmized BN Err is 505% of the Learned BN Err. Given that only between 5 and 23% of CPTs (over NAT families) are randomized, and

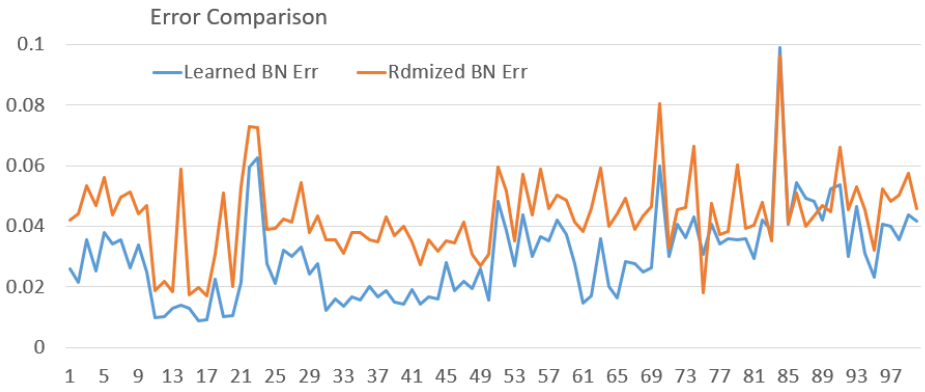


**Fig. 6** Comparision between *Learned BN Err* and *Rdmized BN Err*.

observations are randomly selected, this impact on posterior is significant. This indirectly demonstrates effectiveness of our method in learning NAT models.

### 7.4 Impact of Structure Density Control

As discussed in Section 6, NAT-modeled BNs have significant gain in inference efficiency if their structures have high tree-width but low density. Our MDL score based learning provides a systematic trade-off between model complexity and goodness of fit in learned NAT-modeled BNs. However, the MDL score does not take into account that NAT-modeled BNs are most efficient when they have low structure density. In the 4th set of experiments, we investigate the possibility to apply additional density control in learning, to trade efficiency with inference accuracy.

We generated 5 source BNs, each of 100 variables. The densities of their DAGs are controlled by adding 30% extra arcs beyond being singly-connected. For each source BN, we simulated one dataset. From each dataset, 5 NAT-modeled BNs are learned, controlling percentage of arcs learned beyond being singly-connected to $1\%, 5\%, 10\%, 15\%, 20\%$, respectively. A total of 25 BNs are learned. Inference is performed with each peer tabular BN and each learned BN, using a similar setup as in the 1st set of experiments. A total of $(1+5)*10*5 = 300$ inference runs are performed.

Table 1 summarizes the experimental results. The 5 datasets are indexed 1 to 5, and results from each dataset are contained in one horizontal section. Each section has 3 rows, reporting the results on learning time (LnTm) in seconds, runtime ratio (RtRt) between peer BN and learned BN, and inference error (InfEr) measured by average difference on posterior marginals. Columns 3 to 7 correspond to results from each level of density control (DensCtr).

The bottom section presents means over the results. As density control decrease from 20% to 1%, we see steady decrease of learning time, as well as increase of runtime ratio. It shows that learning becomes more efficient, and inference efficiency gain with learned NAT-modeled BNs becomes more significant.

**Table 1** Summary of experimental results on impact of structure density control

| Dataset | DensCtr | 1% | 5% | 10% | 15% | 20% |
|---|---|---|---|---|---|---|
| 1 | LnTm | 472 | 481 | 488 | 490 | 476 |
| | RtRt | 13 | 11 | 10 | 8 | 8 |
| | InfEr | 0.008 | 0.007 | 0.006 | 0.006 | 0.006 |
| 2 | LnTm | 556 | 611 | 646 | 662 | 653 |
| | RtRt | 63 | 61 | 53 | 52 | 52 |
| | InfEr | 0.005 | 0.005 | 0.005 | 0.005 | 0.005 |
| 3 | LnTm | 592 | 649 | 711 | 705 | 711 |
| | RtRt | 233 | 220 | 214 | 214 | 208 |
| | InfEr | 0.013 | 0.013 | 0.011 | 0.01 | 0.01 |
| 4 | LnTm | 640 | 712 | 799 | 909 | 931 |
| | RtRt | 126 | 118 | 80 | 77 | 77 |
| | InfEr | 0.057 | 0.057 | 0.007 | 0.007 | 0.007 |
| 5 | LnTm | 466 | 491 | 505 | 504 | 553 |
| | RtRt | 37 | 35 | 34 | 31 | 31 |
| | InfEr | 0.008 | 0.008 | 0.006 | 0.006 | 0.006 |
| Mean | $\mu$-LnTm | 545 | 589 | 630 | 654 | 665 |
| | $\mu$-RtRt | 94 | 89 | 78 | 76 | 75 |
| | $\mu$-InfEr | 0.018 | 0.018 | 0.007 | 0.007 | 0.007 |

From the average inference error (bottom row), we see that the mean inference errors are about the same for density control levels $\{1\%, 5\%\}$ and levels $\{10\%, 15\%, 20\%\}$. This is a reflection of the similar pattern in each Section, e.g., results from the 4th dataset. It reveals the following property of the density control: When density control levels are at $\{15\%, 20\%\}$, learning terminates before that percentage of extra arcs are learned as the MDL score no longer improves. Hence, the learning outcome for levels $\{15\%, 20\%\}$ are exactly the same as level 10%. Only when the level further reduces to 5%, it terminates learning before the MDL score does, trading efficiency with accuracy.

This set of experiments shows that density level control provides feasible means for user to influence learning process and maximize efficiency gain from NAT-modeled BNs within user's error tolerance.

### 7.5 Impact of Link Evaluation Control

In the 5th set of experiments, we investigate impact to limit numbers of links for evaluation, as discussed in Section 6. We generated 5 source BNs, each of 200 variables. The densities of their DAGs are controlled by adding 20% extra arcs beyond being singly-connected. From each dataset, we learned 5 NAT-modeled BNs, with density control at level 20%, and limiting number of links to evaluate to the top $n, 2n, 3n, 4n, 5n$, respectively. A total of 25 BNs are learned. Inference is performed with each peer tabular BN and each learned BN, using a similar setup as in the 1st set of experiments. A total of $(1+5) * 10 * 5 = 300$ inference runs are performed.

Table 2 summarizes the experimental results. Columns 3 to 7 correspond to results from each limiting number of links (NumLk). The bottom section presents means over the results. As the limiting number of links increase from $n$ to $3n$, we see steady increase of learning time, with corresponding decrease of runtime

**Table 2** Summary of experimental results on impact of link evaluation control

| Dataset | NumLk | $n$ | $2n$ | $3n$ | $4n$ | $5n$ |
|---|---|---|---|---|---|---|
| 1 | LnTm | 517 | 2526 | 2454 | 2452 | 2434 |
| | RtRt | 407 | 277 | 260 | 259 | 260 |
| | InfEr | 0.046 | 0.015 | 0.015 | 0.015 | 0.015 |
| 2 | LnTm | 745 | 2651 | 4083 | 4134 | 4108 |
| | RtRt | 1223 | 866 | 850 | 864 | 860 |
| | InfEr | 0.034 | 0.006 | 0.005 | 0.005 | 0.005 |
| 3 | LnTm | 655 | 3025 | 5103 | 5131 | 5134 |
| | RtRt | 155 | 96 | 90 | 89 | 86 |
| | InfEr | 0.037 | 0.007 | 0.007 | 0.007 | 0.006 |
| 4 | LnTm | 578 | 2532 | 3900 | 3902 | 3916 |
| | RtRt | 26336 | 8689 | 12439 | 12770 | 12770 |
| | InfEr | 0.045 | 0.006 | 0.006 | 0.006 | 0.006 |
| 5 | LnTm | 711 | 2429 | 2458 | 2422 | 2454 |
| | RtRt | 372 | 301 | 299 | 301 | 299 |
| | InfEr | 0.011 | 0.007 | 0.007 | 0.007 | 0.007 |
| Mean | $\mu$-LnTm | 657 | 2339 | 3227 | 3221 | 3218 |
| | $\mu$-RtRt | 728 | 576 | 561 | 564 | 565 |
| | $\mu$-InfEr | 0.028 | 0.018 | 0.018 | 0.018 | 0.018 |

ratio. It shows that learning becomes less efficient, and the NAT-modeled BNs learned also become less efficient in inference. From the bottom row, we see that the increase from $n$ to $2n$ leads to improved inference accuracy.

However, as the limiting number of links increases from $2n$ to $5n$, there is no significant change in efficiency, nor in inference error. It shows that additional evaluations of low-ranked links do not contribute to the final learned NAT-modeled BNs, and setting the limiting number to about $2n$ has efficiency gain without the cost of losing inference accuracy.

Careful readers may notice that average runtime ratios in Table 2 are about 7 times higher than those in Table 1. Source BNs for Table 1 have 100 variables each, while those for Table 2 have 200 variables each. Their maximum numbers of parents per variable are both $\kappa = 12$. To ensure viable structures, our structure generation software ensures at least one variable with $\kappa$ parents for every $4(\kappa + 1)$ variables. When $\kappa = 12$, every 52 variables has at least one of 12 parents. Hence, source BNs for Table 1 have at least one such variable, but source BNs for Table 2 have at least three such variable. As the result, the latter has about 3 times as many large NAT-models, contributing significantly increased efficiency gain for inference. This result provides further evidence to the strength of NAT-modeled BNs characterized by high tree-width and low density.

## 8 Conclusion and Future Work

The main contribution is the first investigation on learning NAT-modeled BNs. Although this study is not the first on learning BNs with local structures, previous work mainly focused on equality constraints such as CSI, with decision trees or decision graphs as local structures. This work focuses on inequality constraints with NAT models as local structures. Hence, this work complements existing literature on BN learning with local structures. Contributions also include development of

the NAT-enabled MDL function, coupling it with a heuristic search, and empirical study on feasibility of learning NAT-modeled BNs from data.

Two general applications of NAT-modeled BNs are identified in the peer-reviewed literature: First, they offer a tractable subclass of BNs for knowledge representation and acquisition (in line with the recent trend about *tractable models* such as SPNs). Through recursive, reinforcing/undermining local modeling, they reduce space of BNs from $O(n\ s^{\kappa})$ to $O(n\ s\ \kappa)$. For high treewidth, low density BNs, they enable tractable inference through techniques such as de-causalization. Second, they offer a more efficient approximation of intractable BNs through compression, trading accuracy for efficiency.

Note that although arithmetic circuits (ACs) and sum-product networks (SPNs) have linear computational complexity on the size of AC/SPN, when an arbitrary BN is compiled into AC/SPN, it generally incurs exponential blow up in size, unless BN CPTs satisfy certain special conditions. Hence, a probabilistic graphical model encoded as a BN that is not computationally efficient (e.g., $O(n\ s^{\kappa})$ complexity) does not become efficient simply by encoding into AC/SPN (i.e., the $O(n\ s^{\kappa})$ complexity will persist). On the other hand, high treewidth, low density NAT-modeled BNs are computationally efficient ($O(n\ s\ \kappa)$).

The current work opens the door for a third possibility, where NAT-modeled BNs are used directly for modeling data. To realize this option, several issues may be addressed in future research:

Feasibility of NAT-models as alternative for modeling data needs to be evaluated. They are most beneficial when underlying dependency structure has high treewidth and low density. Existing real world BNs often do not fit this profile. For instance, the 9 medium or large BNs in the BN Repository has the maximum number of parents per node of 7. We hypothesize the reason to be difficulty with tabular CPT elicitation (exponential human time) and learning (exponential data). NAT-modeled BNs promise to remove the difficulty. To test the hypothesis, learning NAT-model BNs from real world data needs to be conducted.

Our experiment found that strength of dependency between individual causes and their effect in the same NAT model is far from uniformly distributed. Due to the uneven strength of dependency, a cause in the source NAT model may be excluded during learning. Implication of such exclusion should be evaluated. Deeper understanding of strength of dependency within NAT models is needed, e.g., how NAT topology and single-causal values determine relative strength of dependency for individual causes.

Investigation on learning of NAT-models requires simulation of source models as experimental testbeds. It is desirable that similarity between source and learned BNs positively validates learning. That is, source BNs should be faithful models. Fueled by deeper understanding of the dependency within NAT models, simulations that generate such source BNs should be developed.

Our experimental study tested one heuristic search method. Many alternatives exist, e.g., NAT-enabled scoring is applied at only the last round of search. Further research to compare alternative heuristics relative to quality of output NAT-modeled BNs and efficiency of learning is needed.

Additional issues for future research include integration of alternative encodings relative to $DL_{Nat}$, improved NAT-modeling for small datasets, and comparison with learning algorithms utilizing other local models.

**Acknowledgement**

Financial support from the NSERC Discovery Grant to the first author is acknowledged.

**References**

1. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: Proc. 12th Conf. on Uncertainty in Artificial Intelligence, pp. 115–123 (1996)
2. Chickering, D.: A transformational characterization of equivalent Bayesian network structures. In: Proc. 11th Conf. on Uncertainty in Artificial Intelligence, pp. 87–98 (1995)
3. Chickering, D., Heckerman, D., Meek, C.: A Bayesian approach to learning Bayesian networks with local structure. In: Proc. of 13th Conf. on Uncertainty in Artificial Intelligence, pp. 80–89 (1997)
4. Friedman, N., Goldszmidt, M.: Learning Bayesian networks with local structure. In: Proc. 12th Conf. on Uncertainty in Artificial Intelligence, pp. 252–262. Morgan Kaufmann (1996)
5. Friedman, N., Yakhini, Z.: On the sample complexity of learning Bayesian networks. In: Proc. 12th Conf. on Uncertainty in Artificial Intelligence, pp. 274–282. Morgan Kaufmann (1996)
6. Heckerman, D., Geiger, D., Chickering, D.: Learning Bayesian networks: the combination of knowledge and statistical data. Machine Learning **20**, 197–243 (1995)
7. Henrion, M.: Some practical issues in constructing belief networks. In: L. Kanal, T. Levitt, J. Lemmer (eds.) Uncertainty in Artificial Intelligence 3, pp. 161–173. Elsevier Science Publishers (1989)
8. Lee, C., van Beek, P.: Metaheuristics for score-and-search Bayesian network structure learning. In: Proc 30th Canadian Conf. on Artificial Intelligence, pp. 129–141 (2017)
9. Maaskant, P., Druzdzel, M.: An independence of causal interactions model for opposing influences. In: M. Jaeger, T. Nielsen (eds.) Proc. 4th European Workshop on Probabilistic Graphical Models, pp. 185–192. Hirtshals, Denmark (2008)
10. Rissanen, J.: Modeling by shortest data description. Automatica **14**, 465–471 (1978)
11. Vomlel, J., Tichavsky, P.: An approximate tensor-based inference method applied to the game of Minesweeper. In: Proc. 7th European Workshop on Probabilistic Graphical Models, Springer LNAI 8745, pp. 535–550 (2012)
12. Woudenberg, S., van der Gaag, L., Rademaker, C.: An intercausal cancellation model for Bayesian-network engineering. Inter. J. Approximate Reasoning **63**, 32–47 (2015)
13. Xiang, Y.: Non-impeding noisy-AND tree causal models over multi-valued variables. International J. Approximate Reasoning **53**(7), 988–1002 (2012)
14. Xiang, Y.: Direct causal structure extraction from pairwise interaction patterns in NAT modeling Bayesian networks. Int. J. Approximate Reasoning **105**, 175–193 (2019)
15. Xiang, Y., Jiang, Q.: NAT model based compression of Bayesian network CPTs over multi-valued variables. Computational Intelligence **34**(1), 219–240 (2018)
16. Xiang, Y., Loker, D.: De-causalizing NAT-modeled Bayesian networks for inference efficiency. In: E. Bagheri, J. Cheung (eds.) Canadian AI 2018, LNAI 10832, pp. 17–30. Springer (2018)