Fault Tolerant Direct NAT Structure Extraction from Pairwise Causal Interaction Patterns

Yang Xiang

University of Guelph, Canada
yxiang@uoguelph.ca

Abstract. Non-impeding noisy-And Trees (NATs) provide a general, expressive, and efficient causal model for conditional probability tables (CPTs) in discrete Bayesian networks (BNs). A CPT may be directly expressed as a NAT model or compressed into a NAT model. Once CPTs are NAT-modeled, efficiency of BN inference (both space and time) can be significantly improved. The most important operation in NAT modeling CPTs is extracting NAT structures from interaction patterns between causes. Early method does so through a search tree coupled with a NAT database. A recent advance allows extraction of NAT structures from full, valid causal interaction patterns based on bipartition of causes, without requiring the search tree and the NAT database. In this work, we extend the method to direct NAT structure extraction from partial and invalid causal interaction patterns. This contribution enables direct NAT extraction from all conceivable application scenarios.

Keywords: Graphical models, probabilistic inference, machine learning, Bayesian networks, causal models, non-impeding noisy-AND trees

1 Introduction

Conditional independence encoded in BNs avoids combinatorial explosion in the number of variables. However, BNs are still subject to exponential growth of space and inference time in the number of causes per effect variable in each CPT. A number of space-efficient local models exist, that allow efficient encoding of dependency between an effect and its causes. They include noisy-OR [Pea88], noisy-MAX [Hen89,Die93], context-specific independence (CSI) [BFGK96], recursive noisy-OR [LG04], Non-Impeding Noisy-AND Tree (NIN-AND Tree or NAT) [XJ06], DeMorgan [MD08], tensor-decomposition [VT12], and cancellation model [WvdGR15]. These local models not only reduce the space and time needed to acquire numerical parameters in CPTs, they can also be exploited to significantly reduce inference time, e.g., by exploiting CSI in arithmetic circuits (ACs) and sum-product networks (SPNs) [Dar03,PD11,ZMP15], or by exploiting causal independence in NAT models [XJ16b].

We consider expressing BN CPTs as or compressing them into multi-valued NAT models [Xia12]. Merits of NAT models include being based on simple causal interactions (reinforcement and undermining), expressiveness (recursive mixture, multi-valued), and generality (generalizing noisy-OR, noisy-MAX [XJ16b] and

DeMorgan [Xia12]). In addition, they support much more efficient inference. As shown in [XJ16b], two orders of magnitude speedup in lazy propagation is achieved in NAT-modeled BNs. Since causal independence encoded in a NAT model is orthogonal to CSI, NAT models provide an alternative to CSI for efficient probabilistic inference in BNs.

A NAT model over an effect and n causes consists of a NAT topology and a set of numerical parameters (whose cardinality is linear in n). It compactly represents a BN CPT. In a NAT model, each pair of causes either undermines each other in causing the effect, or reinforcing each other. Hence, the interaction can be specified by one bit with values u (undermining) or r (reinforcing). The collection of such bits defines a pairwise causal interaction (PCI) pattern [XLZ09]. A PCI pattern may be *full* (with one bit for each cause pair) or *partial* (with some missing bits). It has been shown that a full PCI pattern uniquely identifies a NAT [XLZ09,XT14]. This property enables PCI patterns to play an important role for acquisition of NAT topology both in compressing a BN CPT into a NAT model and in learning a BN CPT as a NAT model from data. The corresponding computation takes as input a PCI pattern and returns a compatible (defined below) NAT topology. We term this operation as *NAT structure extraction* from PCI.

For instance, in compressing a target BN CPT into a NAT model, the following method has been applied [XL14,XJ16a]. A partial PCI pattern is first obtained from the target CPT. From the pattern, compatible candidate NATs are extracted through a search tree coupled with a NAT database. Which candidate NAT becomes the final choice is determined by parameterization. For each n value, a NAT database is needed that stores all alternative NATs for n causes. Its size grows super-exponentially in n (see below), and hence it is the source of a computational burden, both offline and online. For instance, it takes 40 hours to generate (offline) the NAT database for $n = 9$ and its search tree [XL14].

An arbitrary bit pattern (either partial or full) may not have a corresponding NAT. Such a pattern is *invalid* (defined below). A recent advance [Xia17] proposed a method for NAT structure extraction from full and valid PCI patterns without needing a search tree and the NAT database. In this paper, we extend the method along two directions. First, we relax the requirement of full PCI patterns so that NAT structures can be extracted from partial PCI patterns. Second, we relax the requirement of valid PCI patterns so that the input can be an invalid pattern and a NAT is extracted whose PCI pattern is closest to the input pattern. These advancements enable NAT structure extraction in all conceivable application scenarios: valid full PCI patterns, valid partial patterns, invalid full patterns, and invalid partial patterns. All of them are through direct extraction, i.e., without need of the search tree and the NAT database.

Section 2 reviews background on NAT models. The task of fault tolerant, direct NAT structure extraction is specified in Section 3. Sections 4 and 5 present theoretical results that the rest of the paper depends on. Direct extraction from full, possibly invalid PCI patterns is covered in Section 6 and extraction from

partial, possibly invalid patterns is presented in Section 7. The experimental results are reported in Section 8.

2 Background

This section briefly reviews background on NAT models. More details can be found in [Xia12]. A NAT model is defined over an effect e and a set of n causes $C = \{c_1, ..., c_n\}$ that are multi-valued and graded, where $e \in \{e^0, ..., e^\eta\}$ $(\eta \geq 1)$ and $c_i \in \{c_i^0, ..., c_i^{m_i}\}$ $(m_i \geq 1)$. C and e form a single family in a BN. Values e^0 and c_i^0 are *inactive*. Other values (may be written as e^+ or c_i^+) are *active* and a higher index means higher intensity (graded).

A causal event is a *success* or *failure* depending on if e is active at a given intensity, is *single-* or *multi-causal* depending on the number of active causes, and is *simple* or *congregate* depending on the value range of e. More specifically,

$$P(e^k \leftarrow c_i^j) = P(e^k | c_i^j, c_z^0 : \forall z \neq i) \quad (j > 0)$$

is the 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 the 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 $P(e \geq e^k \leftarrow \underline{x}^+)$. Interactions among causes may be reinforcing or undermining as defined below.

Definition 1 *Let e^k be an active effect value, $R = \{W_1, ..., W_m\}$ $(m \geq 2)$ 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 $\forall S$ $P(e \geq e^k \leftarrow \underline{y}^+) \leq P(e \geq e^k \leftarrow \underline{x}^+)$. They undermine each other iff $\forall S$ $P(e \geq e^k \leftarrow \underline{y}^+) > P(e \geq e^k \leftarrow \underline{x}^+)$.*

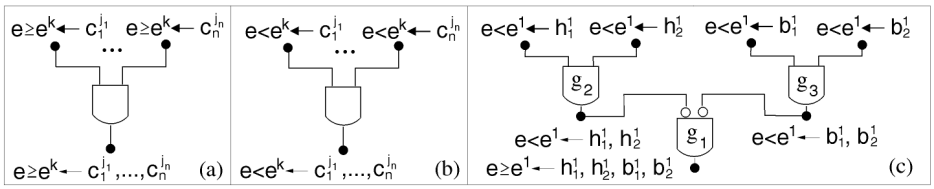

Fig. 1. A direct NIN-AND gate (a), a dual NIN-AND gate (b), and a NAT (c)

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

$$P(e \geq e^k \leftarrow \underline{w}_1^+, ..., \underline{w}_m^+) = \prod_{i=1}^{m} P(e \geq e^k \leftarrow \underline{w}_i^+),$$

which encodes undermining causal interaction. 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 $e < e^k \leftarrow \underline{w}_1^+, ..., \underline{w}_m^+$.

Its probability is

$$P(e < e^k \leftarrow \underline{w}_1^+, ..., \underline{w}_m^+) = \prod_{i=1}^{m} P(e < e^k \leftarrow \underline{w}_i^+),$$

which encodes reinforcement. Fig. 1 (c) shows a NAT, where causes h_1 and h_2 reinforce each other, so do b_1 and b_2, but the two groups undermine each other.

A NAT can be depicted simply by a Root-Labeled-Tree (RLT).

Definition 2 *Let T be a NAT. The RLT of T is a directed graph obtained from T as follows. (1) Delete each gate and direct its inputs to output. (2) Delete each non-root label. (3) Replace each root label by the corresponding cause.*

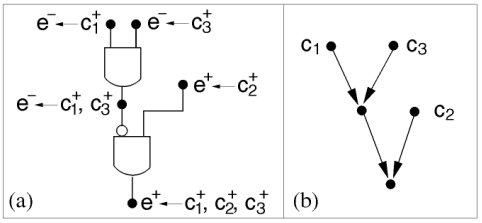

Fig. 2. A NAT (a) and its RLT (b)

Fig. 2 shows a NAT and its RLT. The leaf of RLT corresponds to leaf gate of the NAT. When the leaf gate is dual (or direct), the leaf of RLT is said to be dual (or direct). The leaf gate of a NAT is at level-one. A gate that feeds into the leaf gate is at level-two, and so on. We refer to levels of nodes of a RLT similarly. All gates in the same level have the same type (dual or direct) and gates in adjacent levels differ. An RLT and a leaf type uniquely specifies a NAT.

A NAT T has a single leaf z. For $n \geq 2$, leaf z has at least two parents. Each parent v of z is the leaf of a subtree *induced* by z. If v is a root, then v is a *root parent* of z, and the induced subtree is trivial. In Fig. 2 (b), there are two subtrees induced by the leaf. One subtree is trivial, where c_2 is a root parent of the leaf, and the *root set* of the subtree is $\{c_2\}$. The root set of the other subtree is $\{c_1, c_3\}$.

Each NAT uniquely defines pairwise causal interaction between each pair of causes c_i and c_j $(i \neq j)$, denoted by a PCI bit $\pi(c_i, c_j) \in \{u, r\}$. The value $\pi(c_i, c_j)$ is defined by the common gate of c_i and c_j at the highest level [XLZ09]. The NAT in Fig. 1 (c) has $\pi(h_1, h_2) = r$ since g_2 is dual and $\pi(h_1, b_2) = u$ since g_1 is direct. A collection of PCI bits is a PCI pattern π. If π includes one bit for each cause pair, it is a *full* pattern. Otherwise, it is *partial*.

3 Fault Tolerant Direct NAT Structure Extraction

It has been shown that a full PCI pattern uniquely identifies a NAT [XLZ09,XT14]. This enables PCI patterns to play an important role for acquisition of NAT topology in compressing a BN CPT into a NAT model and in learning a BN CPT

from data as a NAT model. In either case, the input is a PCI pattern and the output is a NAT. We refer to the task as *NAT structure extraction*. Input patterns to the task can be classified as follows. First, we relate two PCI patterns over the same set of causes.

Definition 3 *Let π and ψ be PCI patterns over a set C of causes. If for each pair of causes c_i and c_j $(i \neq j)$ such that both $\pi(c_i, c_j)$ and $\psi(c_i, c_j)$ are defined, $\pi(c_i, c_j) = \psi(c_i, c_j)$ holds, then π and ψ are* **compatible**. *Otherwise, they are* **incompatible**.

Either π or ψ may be partial or full. Compatibility is determined by PCI bits that are defined under both π and ψ. Next, we relate a PCI pattern and a NAT over the same set of causes.

Definition 4 *Let π be a PCI pattern over a set C of causes. Then π is* **valid** *if there exists a NAT over C whose PCI pattern ψ is compatible with π. Otherwise, π is* **invalid**.

In the definition, π may be either partial or full. A full PCI pattern over a set of n causes has $C(n, 2)$ bits. A binary pattern of $C(n, 2)$ bits has $2^{C(n,2)}$ variations, not all of which are necessarily valid.

For $n = 2$, there are 2 NATs. A PCI pattern has $C(2, 2) = 1$ bit. Hence, every PCI pattern is valid. For $n = 3$, there are 8 NATs. A PCI pattern has $C(3, 2) = 3$ bits. Hence, every PCI pattern is valid. For $n = 4$, there are 52 NATs. There are $2^{C(4,2)} = 2^6 = 64$ binary patterns, of which 64 - 52 = 12 patterns are invalid. For $n = 7$, there are $2^{21} = 2,097,152$ binary patterns and 78,416 NATs [XZL09]. The number of invalid full PCI patterns is 2,018,736.

The extraction task has been performed in the context of compressing BN CPTs into NAT models, where PCI patterns are obtained from CPTs and then NATs are extracted [XL14,XJ16a]. The extraction [XL14,XJ16a] relies on a search tree coupled with a NAT database. For each n value, a NAT database stores all alternative NATs over n causes, and the search tree retrieves one or more NATs given a valid PCI pattern [XL14]. We refer to the method as *search tree based* extraction.

The size of the database and the search tree grow super-exponentially in n. Although constructed offline, they are the source of a computational burden. For $n = 9$, there are 25,637,824 NATs and it takes 40 hours to generate the database and the search tree [XL14]. Although NAT models are local models (one BN family per model), and hence n does not grow unbounded due to conditional independence encoded in BNs, it is costly and difficult to generate databases and search trees when n grows beyond 9.

To alleviate these costs, a method is proposed recently [Xia17] for NAT extraction without need of the NAT databases and the search tree. We refer to the method as *direct* extraction. The method requires a full, valid PCI pattern as the input. When a PCI pattern is obtained from a CPT, the pattern is full if the CPT is a NAT model, and is partial otherwise. In this work, we extend the direct method to allow partial input patterns.

When a PCI pattern is obtained from a CPT, there is no guarantee that it is valid. Therefore, the full spectrum of input for NAT extraction includes full and partial, as well as valid and invalid PCI patterns. We refer to NAT structure extraction from invalid PCI patterns as being *fault tolerant*. Existing NAT extraction [XL14,XJ16a] does not explicitly consider the case when input PCI patterns are invalid. Fault tolerant NAT elicitation was considered in [Xia10]. However, it does not provide algorithms for detecting invalid PCI patterns and generating NATs accordingly. In this work, we develop such an algorithm for fault tolerant and direct NAT extraction.

4 Bipartitions of Causes in NAT Models

The direct method for NAT structure extraction initiated in [Xia17] is based on bipartitions of causes. Below, we reformulate some relevant concepts and results from [Xia17] for better clarity and extend them for the purpose of this work.

Definition 5 *Let C ($|C| \geq 2$) be a set of causes, X and Y be non-empty subsets of C where $X \cap Y = \emptyset$ and $X \cup Y = C$, and π be a full PCI pattern over C. Then $\{X, Y\}$ is a* **uniform causal bipartition** *of C under π if one of the following holds.*

1. $\forall x \in X, \forall y \in Y,\ \pi(x, y) = r$
2. $\forall x \in X, \forall y \in Y,\ \pi(x, y) = u$

For $C = \{x, y\}$, π has a single bit. Hence, it is trivially true that $\{\{x\}, \{y\}\}$ forms a uniform causal bipartition. For $C = \{x, y, z\}$, every PCI pattern is valid (see Section 3), and has a NAT T. At least one cause, say x, is the parent of the leaf in T, and $\{\{x\}, \{y, z\}\}$ is a uniform causal bipartition.

Bipartitions in Def. 5 are based on causal interactions. Bipartitions in Def. 6 below are based on NAT topology.

Definition 6 *Let T be a NAT over C. Let $\{X, Y\}$ be a bipartition of C, where $X \neq \emptyset$, $Y \neq \emptyset$, $X \cap Y = \emptyset$, and $X \cup Y = C$. If for each leaf-induced subtree of T and its root set R, either $R \subseteq X$ or $R \subseteq Y$ holds, then $\{X, Y\}$ is a* **subtree-consistent bipartition** *of C with respect to T.*

In Fig. 2, $\{\{c_2\}, \{c_1, c_3\}\}$ is a subtree-consistent bipartition of $C = \{c_1, c_2, c_3\}$, but $\{\{c_1\}, \{c_2, c_3\}\}$ is not.

Theorem 1 below relates the two types of bipartitions defined. It is equivalent to Theorem 1 in [Xia17] but with better clarity.

Theorem 1 *Let T be a NAT over C and π be the PCI pattern of T. Every subtree-consistent bipartition of C is a uniform causal bipartition.*

Theorem 2 below strengthens Theorem 1 with the existence of subtree-consistent bipartitions. It will be used later to justify a main result of this work.

Theorem 2 *Every NAT over a set C ($|C| \geq 2$) of causes has at least one subtree-consistent bipartition of C.*

Proof: Since $|C| \geq 2$, the leaf of T has at least two parents. Let x be such a leaf parent. If x is a root, then $\{\{x\}, C \setminus \{x\}\}$ is a subtree-consistent bipartition. Otherwise, x is the leaf of a subtree. Let X be the root set of the subtree. Then $\{X, C \setminus X\}$ is a subtree-consistent bipartition. □

5 PCI Core and Invalid PCI Patterns

NAT extraction from invalid PCI patterns necessitates operations different from extraction from valid patterns. Activation of such operations in turn necessitates detection of invalid patterns. Below we analyze conditions for such detection. First, we formalize necessary concepts.

Definition 7 *Let π be a full PCI pattern over a set C ($|C| \geq 2$) of causes. If there exists no uniform causal bipartition under π, then π is a* **PCI core** *and C is the* **domain** *of the PCI core.*

From Section 3, there exists no PCI core when $n = 2$ and 3. Following [Xia17], we analyze a PCI pattern equivalently through its PCI matrix, and denote both by π interchangeably. Consider the PCI matrix π in Fig. 3 (left).

	a	b	c	d
a		r	u	r
b	r		r	u
c	u	r		u
d	r	u	u	

	e	a	b	c	d
e		u	u	u	u
a	u		r	u	r
b	u	r		r	u
c	u	u	r		u
d	u	r	u	u	

Fig. 3. PCI matrix over $C = \{a, b, c, d\}$ (left) and one over $C = \{e, a, b, c, d\}$ (right)

Since the row indexed by a is not uniform, $\{\{a\}, \{b, c, d\}\}$ is not a uniform causal partition. In fact, none of the bipartitions $\{X, Y\}$ is when $|X| = 1$. For $X = \{a, d\}$ and $Y = \{b, c\}$, consider cells at the intersection of rows indexed by X and columns indexed by Y. The first row (r, u) in the intersection is non-uniform. Hence, $\{\{a, d\}, \{b, c\}\}$ is not a uniform causal partition. In fact, none of the bipartitions $\{X, Y\}$ is when $|X| = 2$. Hence, π is a PCI core. This shows that the smallest PCI core (over the least number of causes) occurs when $n = 4$.

Definition 8 *Let π be a PCI pattern over C. A PCI pattern ψ over $X \subseteq C$ ($|X| \geq 2$) is a sub-pattern of π if, for every $x, y \in X$, $\psi(x, y) = \pi(x, y)$.*

Note that π is a trivial sub-pattern of itself. Consider the PCI matrix π in Fig. 3 (right). The partition $\{\{e\}, \{a, b, c, d\}\}$ is causally uniform. Hence, π is not a PCI core. If we delete the row and the column indexed by e, the remainder is identical to the matrix in the left. Since the sub-pattern over $\{a, b, c, d\}$ is a PCI core, no other uniform causal partition of C exists under π.

Theorem 3 below reveals a fundamental condition of invalid PCI patterns.

Theorem 3 *A PCI pattern π over C ($|C| \geq 2$) is invalid iff π contains a sub-pattern ψ that is a PCI core.*

Proof: [Sufficiency] Assume that π contains a sub-pattern ψ over $S \subseteq C$ that is a PCI core. Since the smallest PCI core has 4 causes, $|S| \geq 4$. We show that there exists no NAT whose PCI pattern equals to π.

We prove by contradiction. Suppose that a NAT T over C exists with PCI pattern π. By Theorem 2, T has a subtree-consistent bipartition $\{X, Y\}$ of C. Either X and Y split S (possible since $|S| \geq 4$) or they don't. We consider each case below.

(Case 1) If X and Y split S, denote $S_X = X \cap S \neq \emptyset$ and $S_Y = Y \cap S \neq \emptyset$, where $S_X \cup S_Y = S$. Let ψ be the sub-pattern of π over S. By Theorem 1, $\{X, Y\}$ is a uniform casual bipartition of C under π. Hence, $\{S_X, S_Y\}$ is also a uniform casual bipartition of S under ψ: a contradiction to the assumption that ψ is a PCI core.

(Case 2) If X and Y do not split S, then S is contained in one of them, say X. From $|S| \geq 4$, we have $|X| \geq 4$. Since $\{X, Y\}$ is a subtree-consistent bipartition of C, either Y is the root set of a subtree T' induced by the leaf of T, or Y is made of root sets of multiple such subtrees. In either case, we remove each subtree induced by the leaf of T whose root set is contained in Y, and refer to the reduced tree by T'. If the leaf z of T is left with a single parent z' due to the removal, we remove z from T so that z' becomes the leaf of T'. The resultant T' is a well-defined NAT over $X \subset C$ and $|X| \geq 4$.

Since C is finite and the reduction produces a NAT over a proper subset of causes, by processing a subtree-consistent bipartition in T' recursively, Case 1 must be true eventually.

[Necessity] Suppose a PCI pattern π over C does not correspond to any NAT. We show that π contains a sub-pattern ψ that is a PCI core. We prove by contraposition. Assume that π does not contain any PCI core. We show by induction on $|C|$ that a NAT can be constructed with PCI pattern π.

For $|C| = 2$, say, $C = \{x, y\}$, since π is not a PCI core, the only bipartition $\{\{x\}, \{y\}\}$ is a uniform causal bipartition of C. Hence, a tree T with the leaf z and root parents x and y is a NAT over C.

Assume that for $|C| = k \geq 2$, if PCI pattern π over C does not contain a PCI core, then a NAT can be constructed with pattern π.

Consider $|C| = k + 1$ where PCI pattern π over C does not contain a PCI core. Since π is not a PCI core, there exists a uniform causal bipartition $\{X, Y\}$ of C, where $|X| \leq k$ and $|Y| \leq k$. Since $k + 1 \geq 3$, X and Y cannot both be singletons. Either exactly one of them is a singleton (Case a) or none of them is a singleton (Case b). We construct a NAT with pattern π in each case below.

(Case a) Suppose that X is a singleton $\{x\}$. Since π does not contain a PCI core, neither the sub-pattern ψ of π over Y does. Since $|Y| = k$, by inductive assumption, a NAT T_Y can be constructed with PCI pattern ψ. Denote the leaf of T_Y by z.

If z is direct and the uniform causal interaction relative to bipartition $\{\{x\}, Y\}$ is u, add the root parent x to z in T_Y. The resultant tree T is a NAT with PCI

pattern π. The processing is similar if z is dual and the interaction relative to $\{\{x\}, Y\}$ is r.

If z is direct and the causal interaction relative to bipartition $\{\{x\}, Y\}$ is r, create a tree T with leaf v whose two parents are x and z. The resultant tree T is a NAT with PCI pattern π. The processing is similar if z is dual and the interaction relative to $\{\{x\}, Y\}$ is u.

(Case b) Suppose that none of X and Y is singleton. Let π_X (π_Y) be the sub-pattern of π over X (Y). Since π does not contain a PCI core, neither π_X nor π_Y does. Since $|X| \leq k$ ($|Y| \leq k$), by inductive assumption, a NAT T_X (T_Y) can be constructed with PCI pattern π_X (π_Y). Denote the leaf of T_X (T_Y) by z_X (z_Y).

If z_X and z_Y are both direct and the uniform causal interaction relative to bipartition $\{X, Y\}$ is u, merge T_X and T_Y by adding all parents of z_Y as parents of z_X and deleting z_Y. The resultant tree T is a NAT with PCI pattern π. The processing is similar if z_X and z_Y are both dual and the uniform causal interaction relative to bipartition $\{X, Y\}$ is r.

If z_X is direct, z_Y is dual, and the uniform causal interaction relative to bipartition $\{X, Y\}$ is u, merge T_X and T_Y by making z_Y a parent of z_X. The resultant tree T is a NAT with PCI pattern π. The processing is similar for other cases where the types of z_X and z_Y differ. □

Theorem 3 establishes that the necessary and sufficient condition of an invalid PCI pattern π is that either π is a PCI core or a sub-pattern of π is.

6 NAT Extraction with Invalid PCI Pattern Detection

We apply formal results from previous sections to algorithms in [Xia17] to extend their functionality as well as to improve their semantic clarity.

Algorithm 1 below extends InteractBtwSets [Xia17] by improving its semantic clarity. As input, it takes a set C of causes, a PCI matrix π over C, and a proper subset $X \subset C$ from which a bipartition $\{X, Y\}$ (line 1) is defined. It determines if $\{X, Y\}$ is a uniform causal bipartition. If so, it returns the NIN-AND gate type that implements the causal interaction. Otherwise, it returns null.

Algorithm 1 $IsUniformCausalBipartition(C, \pi, X)$

1 $Y = C \setminus X$*;*
2 *if* $\forall x \in X, \forall y \in Y$, $\pi(x, y) = r$ *holds, gatetype = dual;*
3 *else if* $\forall x \in X, \forall y \in Y$, $\pi(x, y) = u$ *holds, gatetype = direct;*
4 *else gatetype = null;*
5 *return gatetype;*

TestPciSetNat below extends SetNatByPci [Xia17] on both functionality and semantic clarity. As input, it takes a set C of causes and a full PCI matrix π over C. Unlike [Xia17] where π is assumed valid, π can be either valid or invalid. TestPciSetNat calls IsUniformCausalBipartition to evaluate alternative

bipartitions. If π is valid, it returns the respective NAT. Otherwise, invalidity of π is detected and the domain of a PCI core is returned instead (lines 21, 25, and 27). $InNat1$ and $InNat2$ are sets of causes added to the current NAT T. The $Subsets$ collects subsets X and Y for each uniform causal bipartition $\{X, Y\}$. An example of matrix reduction (lines 8 and 20) is in Fig. 3, where the matrix in the right over $\{e, a, b, c, d\}$ is *reduced* to the matrix in the left over $\{a, b, c, d\}$. Although NAT is used to refer to T, the actual data structure of T is an RLT (hence, the reference to leaf z).

Algorithm 2 $TestPciSetNat(C, \pi)$

1 init NAT T with leaf z only; type(z) = null; init set $InNat1 = \emptyset$;
2 for each $x \in C$, do
3 if $\forall y \in C \setminus \{x\}, \pi(x, y) = r$ holds,
4 type(z) = dual; add x to T as a parent of z; $InNat1 = InNat1 \cup \{x\}$;
5 else if $\forall y \in C \setminus \{x\}, \pi(x, y) = u$ holds,
6 type(z) = direct; add x to T as a parent of z; $InNat1 = InNat1 \cup \{x\}$;
7 if $InNat1 = C$, return T;

8 reduce (C, π) to (C', ψ) relative to $InNat1$;
9 $InNat2 = \emptyset$; $Subsets = \emptyset$;
10 for $i = 2$ to $|C'|/2$, do
11 for each $X \subset C'$ where $|X| = i$, do
12 gatetype = IsUniformCausalBipartition(C', ψ, X);
13 if gatetype $\neq$ null,
14 if type(z) = null, assign type(z) = gatetype;
15 if gatetype = type(z), $Subsets = Subsets \cup \{X, C' \setminus X\}$;

16 if $Subsets \neq \emptyset$,
17 for each $X \in Subsets$,
18 if $\exists\, V \in Subsets$ such that $X \supseteq V$, remove X from Subsets;
19 for each $X \in Subsets$, do
20 reduce π to ψ over X; $R = TestPciSetNat(X, \psi)$;
21 if $R = X$, return X;
22 add R to T as a subtree induced by z;
23 $InNat2$ = union of subsets in Subsets;
24 if $InNat2 = C'$, return T;
25 if $InNat1 \cup InNat2 = \emptyset$, return C;

26 R = TestPciSetNat(C', ψ);
27 if $R = C'$, return C';
28 add R to T as a subtree induced by z;
29 return T;

TestPciSetNat is sound because whenever π is invalid, TestPciSetNat returns the domain of a PCI core. By Theorem 3, either π is a PCI core, which is detected in line 25 with domain C returned, or π contains a PCI core, which is

detected in lines 21 and 27 with the corresponding domains X and C' returned. TestPciSetNat is complete because whenever π is valid, TestPciSetNat returns a respective NAT. This can be established similarly as Theorem 3 in [Xia17]. We omit detailed analysis on soundness and completeness due to space.

When π is valid, the time complexity of TestPciSetNat is a function of the respective NAT T. Let z be the leaf of T. If every cause in C is a parent of z, only lines 1 to 7 is run, and the complexity is $O(n^2)$. If no cause is a parent of z, lines 1 to 7 are followed by lines 8 to 15. The number of alternative X (line 11) is $2^{n-1} - n - 1$ and evaluation of each X takes $O(n^2/4)$ time. The complexity is $O(n^2\ 2^n)$. This is also the complexity when π is a PCI core.

If π is valid, some causes are the parents of z, the computation time is between $O(n^2)$ and $O(n^2\ 2^n)$. The same holds if π is invalid and contains a PCI core over a proper subset of C. In summary, the time complexity of TestPciSetNat is a function of π and is between $O(n^2)$ and $O(n^2\ 2^n)$. Note that since a NAT model is over a single BN family, n is not unbounded.

7 NAT Extraction from Partial PCI Patterns

The input to TestPciSetNat is a full PCI pattern. The following algorithm allows the input pattern to be full or partial, and valid or invalid. In particular, input of SetNat includes a set C of causes, a PCI pattern π over C, and a set B (possibly empty) of missing PCI bits. Set B is such that if $\pi(x, y)$ is a missing bit, then $(x, y) \in B$.

Set Π collects full PCI patterns that are compatible with π. In line 5, the full PCI pattern ψ is obtained from the partial pattern π by adding the missing bits according to θ. Variable *bsc* is a PCI bit switching counter. When π is invalid, it controls the number of bits in π that will be switched.

Algorithm 3 $SetNat(C, \pi, B)$

1 def = set of defined bits in π;
2 $\Pi = \emptyset$;
3 if $B = \emptyset$, $\Pi = \{\pi\}$;
4 else for each instantiation θ of missing bits in B,
5 complete π by θ into ψ; $\Pi = \Pi \cup \{\psi\}$;
6 for each $\psi \in \Pi$, do
7 R = TestPciSetNat(C, ψ);
8 if R is a NAT, return R;
9 bsc = 1;
10 do
11 for each $\psi \in \Pi$, do
12 for each combination of bsc bits in def, do
13 get τ from ψ by switching these bits;
14 R = TestPciSetNat(C, τ);
15 if R is a NAT, return R;
16 bsc++;
17 end do

When π is full, $B = \emptyset$ and line 3 is run. Otherwise, lines 4 and 5 are run. Each full pattern in Π is processed in lines 6 to 8. If π is valid, one ψ will succeed and the respective NAT will be returned.

Otherwise, π is invalid. Lines 9 to 17 switch some bits in π for each ψ. The number of bits to be switched starts from 1 and increases as needed. Hence, a NAT with the minimum number of PCI bits that differ from π (least incompatible) will be returned.

8 Experiment

We evaluated the algorithms by 16 batches of experiments (see Table 1), running in a ThinkPad X230. Each batch extracts NATs from 100 PCI patterns. In

Table 1. Summary of experimental batches

Index	n	Valid	Full	#Bits switched	#Missing bits	Runtime $\hat{\mu}$ msec	Runtime $\hat{\sigma}$ msec
1	8	yes	yes	0	0	0.16	1.59
2	12	yes	yes	0	0	0.82	3.42
3	16	yes	yes	0	0	21.83	21.93
4	20	yes	yes	0	0	520.59	437.37
5	8	may not	yes	1	0	0.36	2.27
6	12	may not	yes	1	0	22.19	30.24
7	16	may not	yes	1	0	1117.95	1498.54
8	20	may not	yes	1	0	63439.55	70736.72
9	8	yes	no	0	1	0.19	1.60
10	12	yes	no	0	1	1.32	4.27
11	16	yes	no	0	1	31.60	30.46
12	20	yes	no	0	1	855.87	821.17
13	8	may not	no	1	1	0.64	3.09
14	12	may not	no	1	1	34.39	47.41
15	16	may not	no	1	1	2187.95	2999.55
16	20	may not	no	1	1	96184.07	122835.54

batches 1 to 4, each input PCI pattern is derived from a randomly generated NAT with $n = 8, 12, 16, 20$, respectively. Hence, each pattern is full and valid.

In the remaining batches, each input pattern is derived from a random NAT and is modified in addition. In batches 5 to 8, the pattern is modified by randomly selecting a bit and switching its value. Hence, the pattern is full, but may be invalid. In batches 9 to 12, a randomly selected bit is dropped from each pattern. Hence, the pattern is partial but valid. In batches 13 to 16, for each pattern, one bit is dropped and the value of another bit is switched. Hence, the pattern is partial and may be invalid. The experiment setup includes all combinations of fullness and validity of input patterns, and spans a wide range of n values.

Being able to conduct the experiment at $n = 12, 16, 20$ is itself a demonstration of the advantage of the proposed algorithms. The number of NATs for $n = 9$ is 25,637,824 [XZL09]. Generation of the NAT database and search tree take about 40 hours [XL14]. The number of NATs for $n = 10$ is 564,275,648. It would take at least 880 hours to generate the NAT database and search tree.

After each NAT is extracted, its PCI pattern is compared with the input pattern. For batches 1 to 4 and 9 to 12, each NAT pattern is compatible with the input pattern. For batches 5 to 8 and 13 to 16, the extracted NAT pattern differs from the input by no more than 1 bit. Hence, our algorithms successfully extract NATs in all possible types of scenarios. Due to space, we skip a more elaborative report and analysis of the experimental results.

9 Conclusion

The main contribution of this paper is a collection of algorithms for direct NAT extraction from partial or invalid PCI patterns, founded on formal analysis. They allow NAT structure extraction in all conceivable scenarios, and enable NAT modeling to be applied more effectively in compressing BN CPTs and in learning compact BN CPTs from data. Integrating these algorithms with the existing CPT compression algorithms is an immediate future work.

Our experiments showed that an incorrect PCI bit in the input pattern is much more costly than a missing PCI bit in NAT extraction. Further research will be devoted to improve efficiency of extraction from invalid PCI patterns.

Acknowledgement

Financial support from the NSERC Discovery Grant is acknowledged.

References

[BFGK96] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 115–123, 1996.

[Dar03] A. Darwiche. A differential approach to inference in Bayesian networks. *J. ACM*, 50(3):280–305, 2003.

[Die93] F.J. Diez. Parameter adjustment in Bayes networks: The generalized noisy OR-gate. In D. Heckerman and A. Mamdani, editors, *Proc. 9th Conf. on Uncertainty in Artificial Intelligence*, pages 99–105. Morgan Kaufmann, 1993.

[Hen89] M. Henrion. Some practical issues in constructing belief networks. In L.N. Kanal, T.S. Levitt, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 161–173. Elsevier Science Publishers, 1989.

[LG04] J.F. Lemmer and D.E. Gossink. Recursive noisy OR - a rule for estimating complex probabilistic interactions. *IEEE Trans. on System, Man and Cybernetics, Part B*, 34(6):2252–2261, Dec 2004.

[MD08] P.P. Maaskant and M.J. Druzdzel. An independence of causal interactions model for opposing influences. In M. Jaeger and T.D. Nielsen, editors, *Proc. 4th European Workshop on Probabilistic Graphical Models*, pages 185–192, Hirtshals, Denmark, 2008.

[PD11] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, page 25512558, 2011.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[VT12] J. Vomlel and P. Tichavsky. An approximate tensor-based inference method applied to the game of Minesweeper. In *Proc. 7th European Workshop on Probabilistic Graphical Models, Springer LNAI 8745*, pages 535–550, 2012.

[WvdGR15] S. Woudenberg, L.C. van der Gaag, and C. Rademaker. An intercausal cancellation model for Bayesian-network engineering. *Inter. J. Approximate Reasoning*, 63:3247, 2015.

[Xia10] Yang Xiang. Acquisition and computation issues with NIN-AND tree models. In P. Myllymaki, T. Roos, and T. Jaakkola, editors, *Proc. 5th European Workshop on Probabilistic Graphical Models*, pages 281–289, Finland, 2010.

[Xia12] Yang Xiang. Non-impeding noisy-AND tree causal models over multi-valued variables. *International J. Approximate Reasoning*, 53(7):988–1002, Oct 2012.

[Xia17] Yang Xiang. Extraction of NAT causal structures based on bipartition. In *Proc. 30th Inter. Florida Artificial Intelligence Research Society Conf.* Accepted and in print, 2017.

[XJ06] Yang Xiang and Ning Jia. Modeling causal reinforcement and undermining with noisy-AND trees. In L. Lamontagne and M. Marchand, editors, *Advances in Artificial Intelligence, LNAI 4013*, pages 171–182. Springer-Verlag, 2006.

[XJ16a] Yang Xiang and Qian Jiang. Compression of general Bayesian net CPTs. In R. Khoury and C. Drummond, editors, *Advances in Artificial Intelligence, LNAI 9673*, pages 285–297. Springer, 2016.

[XJ16b] Yang Xiang and Yiting Jin. Multiplicative factorization of multi-valued NIN-AND tree models. In Z. Markov and I. Russell, editors, *Proc. 29th Inter. Florida Artificial Intelligence Research Society Conf.*, pages 680–685. AAAI Press, 2016.

[XL14] Yang Xiang and Qing Liu. Compression of Bayesian networks with NIN-AND tree modeling. In L.C. vander Gaag and A.J. Feelders, editors, *Probabilistic Graphical Models, LNAI 8754*, pages 551–566. Springer, 2014.

[XLZ09] Yang Xiang, Yu Li, and Jingyu Zhu. Towards effective elicitation of NIN-AND tree causal models. In L. Godo and A. Pugliese, editors, *Inter. Conf. on Scalable Uncertainty Management (SUM 2009), LNCS 5785*, pages 282–296. Springer-Verlag Berlin Heidelberg, 2009.

[XT14] Yang Xiang and Minh Truong. Acquisition of causal models for local distributions in Bayesian networks. *IEEE Trans. Cybernetics*, 44(9):1591–1604, 2014.

[XZL09] Yang Xiang, Jingyu Zhu, and Yu Li. Enumerating unlabeled and root labeled trees for causal model acquisition. In Y. Gao and N. Japkowicz, editors, *Advances in Artificial Intelligence, LNAI 5549*, pages 158–170. Springer, 2009.

[ZMP15] H. Zhao, M. Melibari, and P. Poupart. On the relationship between sum-product networks and Bayesian networks. In *Proc. 32nd Inter. Conf. Machine Learning*, 2015.