Acquisition and Computation Issues with NIN-AND Tree Models

Yang Xiang
University of Guelph, Canada

Abstract

Most techniques to improve efficiency of conditional probability table (CPT) acquisition for Bayesian network (BN) can only represent reinforcing causal interactions. The non-impeding noisy-AND (NIN-AND) tree is the first causal model that explicitly expresses reinforcement, undermining, and their mixture, while its acquisition is of linear complexity. We address three issues on acquisition and computation with these models. In particular, we propose methods to improve computation of conditional probability from a model, to improve the efficiency of CPT computation from these models, and to address NIN-AND tree acquisition by elicitation of pairwise causal interactions.

1 Introduction

To acquire a BN, for each non-root node, a CPT needs to be specified. When the BN is constructed along the causal direction, a CPT is the distribution of an effect conditioned on its n causes. In general, the complexity of CPT assessment is exponential on n. A number of techniques have been proposed to make the assessment more efficient. Noisy-OR (Pearl, 1988) is the most well known that reduces this complexity to linear. A number of extensions have also been proposed such as (Heckerman and Breese, 1996; Galan and Diez, 2000; Lemmer and Gossink, 2004). However, noisy-OR, noisy-AND (Galan and Diez, 2000), as well as related techniques, can only represent causal interactions that are reinforcing (Xiang and Jia, 2007).

The NIN-AND tree (Xiang and Jia, 2007) is a recently proposed technique for efficiently modeling and acquiring CPT.[1] It inherits the reinforcing behavior of noisy-OR (Pearl, 1988). It inherits some features of noisy-AND (Galan and Diez, 2000) while moves away from its impeding behavior (see (Xiang and Jia, 2007)) to allow modeling of undermining. It also inherits the flexibility of recursive noisy-OR (Lemmer and Gossink, 2004) to allow probabilities of multi-causal events as input. As the result, NIN-AND tree provides the first causal model that explicitly expresses reinforcing and undermining causal interactions, as well as their mixture. Its acquisition involves elicitation of probability parameters whose number is linear on n, and a tree topology whose size is linear on n.

This paper addresses three technical issues regarding acquisition of NIN-AND tree models and computations performed on them. First, given an NIN-AND tree model for an effect and some of its present causes, the probability of the effect can be computed, conditioned on that these causes are present and the remaining causes are absent . We propose a new algorithm to compute this probability from an NIN-AND tree model, that improves upon the alternative in (Xiang and Jia, 2007). The new algorithm takes advantage of minimal NIN-AND tree models and enables more efficient CPT computation.

Second, according to a method in (Xiang and Jia, 2007), the exponential number of probability parameters in a CPT can each be computed from a suitable NIN-AND tree model. Although all such models can be obtained by modifying a common base model, an exponential number of alternative NIN-AND tree models must be created in the process. We propose a new method that can directly compute all probability parameters of a CPT from the base model, sav-

[1] Being unaware of this work and its precursor (Xiang and Jia, 2006), (Maaskant and Druzdzel, 2008) independently presented special cases of NIN-AND tree models.

ing the computation of creating the exponential number of NIN-AND trees.

Third, acquisition of the tree topology is a critical step in specifying an NIN-AND tree model. Three methods have been proposed: direct specification (Xiang and Jia, 2007), two-step menu selection (Xiang et al., 2009b), and identification by pairwise causal interaction (Xiang et al., 2009a). An expert-specified pairwise causal interaction function may have no corresponding NIN-AND tree model. We propose a technique to address this issue.

In this paper, we focus on binary effect and causes. For generalization of NIN-AND tree models to multi-valued effect and cause variables, see (Xiang, 2010).

The remainder of the paper is organized as follows: The background on NIN-AND tree models is introduced in Section 2. Computation of conditional probability from an NIN-AND tree model is addressed in Section 3. The method for computing CPT without exponential generation of NIN-AND trees is presented in Section 4. How to facilitate expert in NIN-AND tree acquisition by pairwise causal interaction is described in Section 5.

2 Background

This section is mostly based on (Xiang and Jia, 2007). An *uncertain cause* is a cause that can produce an effect but does not always do so. Denote a binary effect variable by e and a set of binary cause variables of e by $X = \{c_1, ..., c_n\}$. Denote $e = true$ by e^+ and $e = false$ by e^-. Similarly, for each cause c_i, denote $c_i = true$ by c_i^+ and $c_i = false$ by c_i^-. Denote the set of *all causes* (including a leaky cause) of e by C.

A *singular causal event* refers to an event that a cause c_i caused its effect e to occur successfully when all other causes of e are absent. Denote this causal event by $e^+ \leftarrow c_i^+$ and its probability by $P(e^+ \leftarrow c_i^+)$. The singular causal failure event, where e is false when c_i is true and all other causes of e are false, is denoted by $e^+ \not\leftarrow c_i^+$. Denote the multi-causal event that a set $X = \{c_1, ..., c_n\}$ $(n > 1)$ of causes caused e by $e^+ \leftarrow c_1^+, ..., c_n^+$ or $e^+ \leftarrow \underline{x}^+$.

Causes reinforce each other if collectively they are at least as effective in causing the effect as some acting by themselves. If collectively they are less effective, then they undermine each other. For $C = \{c_1, c_2\}$, if c_1 and c_2 undermine each other, all the following hold:

$$P(e^+|c_1^-, c_2^-) = 0, P(e^+|c_1^+, c_2^-) > 0, P(e^+|c_1^-, c_2^+) > 0,$$

$$P(e^+|c_1^+, c_2^+) < min(P(e^+|c_1^+, c_2^-), P(e^+|c_1^-, c_2^+)).$$

Reinforcement and undermining occur between individual as well as sets of variables. Variables within each set can be reinforcing, while the sets can undermine each other. Hence, W_i in Def. 1 is not necessarily a singleton.

Definition 1. *Let $R = \{W_1, W_2, ...\}$ be a partition of a set X of causes, $R' \subset R$ be any proper subset of R, and $Y = \cup_{W_i \in R'} W_i$. Sets of causes in R reinforce each other, iff $\forall R'$ $P(e^+ \leftarrow \underline{y}^+) \leq P(e^+ \leftarrow \underline{x}^+)$. Sets of causes in R undermine each other, iff $\forall R'$ $P(e^+ \leftarrow \underline{y}^+) > P(e^+ \leftarrow \underline{x}^+)$.*

Disjoint sets of causes $W_1, ..., W_m$ satisfy *failure conjunction* iff $e^+ \not\leftarrow \underline{w}_1^+, ..., \underline{w}_m^+ = \wedge_{i=1}^m (e^+ \not\leftarrow \underline{w}_i^+)$. That is, collective failure is attributed to individual failures. They also satisfy *failure independence* iff $P(\wedge_{i=1}^m (e^+ \not\leftarrow \underline{w}_i^+)) = \prod_{i=1}^m P(e^+ \not\leftarrow \underline{w}_i^+)$. Disjoint sets of causes $W_1, ..., W_m$ satisfy *success conjunction* iff $e^+ \leftarrow \underline{w}_1^+, ..., \underline{w}_m^+ = \wedge_{i=1}^m (e^+ \leftarrow \underline{w}_i^+)$. That is, collective success requires individual effectiveness. They also satisfy *success independence* iff $P(\wedge_{i=1}^m (e^+ \leftarrow \underline{w}_i^+)) = \prod_{i=1}^m P(e^+ \leftarrow \underline{w}_i^+)$.

Causes are reinforcing whenever they satisfy failure conjunction and independence, and they are undermining whenever they satisfy success conjunction and independence. Hence, undermining can be modeled by a direct NIN-AND gate (Fig. 1, left), and reinforcement by a dual NIN-AND gate (right). Complex mixture of reinforcement and undermining can be modeled by an NIN-AND tree defined below.

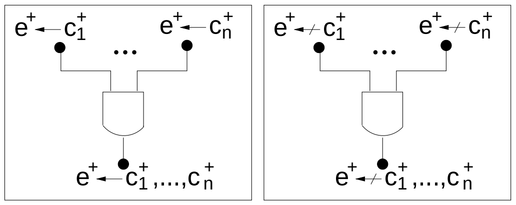

$e^+ \leftarrow c_1^+$
$e^+ \leftarrow c_n^+$
...
$e^+ \leftarrow c_1^+,...,c_n^+$
$e^+ \not\leftarrow c_1^+$
$e^+ \not\leftarrow c_n^+$
...
$e^+ \not\leftarrow c_1^+,...,c_n^+$

Figure 1: Direct (left) and dual (right) NIN-AND gates

Definition 2. *An* `NIN-AND tree` *T is a directed tree for effect e and a set $X = \{c_1, ..., c_n\}$ of occurring causes.*

1. *There are two types of nodes,* `event` *nodes (a black oval of in-degree ≤ 1 and out-degree ≤ 1) and* `gate` *nodes (a NIN-AND gate) of in-degree ≥ 2 and out-degree 1.*

2. *There are two types of links, each connecting an event and a gate along input-to-output direction of gates,* `forward` *links (a line) and* `negation` *links (with a white oval at gate end).*

3. *Each terminal node is an event labelled by a causal event $e^+ \leftarrow \underline{y}^+$ or $e^+ \not\leftarrow \underline{y}^+$. There is a single* `leaf` *(no child) with $\underline{y}^+ = \underline{x}^+$, connecting to the* `leaf gate`*. For each* `root` *(no parent; indexed by i), $\underline{y}_i^+ \subset \underline{x}^+$, $\underline{y}_j^+ \cap \underline{y}_k^+ = \emptyset$ for $j \neq k$, and $\bigcup_i \underline{y}_i^+ = \underline{x}^+$.*

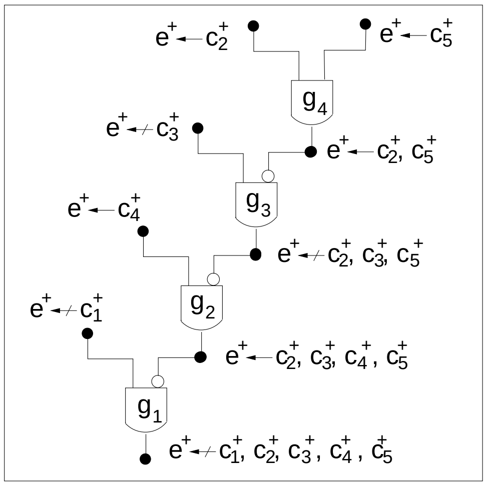
4. *For inputs to a direct gate g, each is either connected by a forward link to a node labelled $e^+ \leftarrow \underline{y}^+$, or by a negation link to a node labelled $e^+ \not\leftarrow \underline{y}^+$. Gate g outputs by a forward link to a node labelled $e^+ \leftarrow \cup_i \underline{y}_i^+$.*

5. *For inputs to a dual gate g, each is either connected by a forward link to a node labelled $e^+ \not\leftarrow \underline{y}^+$, or by a negation link to a node labelled $e^+ \leftarrow \underline{y}^+$. Gate g outputs by a forward link to a node labelled $e^+ \not\leftarrow \cup_i \underline{y}_i^+$.*

Fig. 2 shows an NIN-AND tree, where $X = \{c_1, ..., c_5\}$. The leaf gate g_1 is dual, and so is g_3. The remaining gates are direct. Causes c_2 and c_5 are undermining each other, but they reinforce c_3 (and vice versa). Collectively, the three undermines c_4. The four of them reinforces c_1.

A root node may be labelled by a singular or multi-causal event. In this paper, we assume that it is singular. When we refer to a node in T by v, we overload the symbol v to refer also to the causal event that labels the node.

Definition 3. *An NIN-AND tree model M is a quadruple (X, e, T, B). X is a set of uncertain causes of effect e. T is an NIN-AND tree for e and X. B is a set of parameters, one for each root in T and being a potential in $(0, 1]$ over the corresponding causal event.*



$e^+ \leftarrow c_2^+$
$e^+ \leftarrow c_5^+$
g_4
$e^+ \not\leftarrow c_3^+$
$e^+ \leftarrow c_2^+, c_5^+$
g_3
$e^+ \leftarrow c_4^+$
$e^+ \not\leftarrow c_2^+, c_3^+, c_5^+$
$e^+ \not\leftarrow c_1^+$
g_2
$e^+ \leftarrow c_2^+, c_3^+, c_4^+, c_5^+$
g_1
$e^+ \not\leftarrow c_1^+, c_2^+, c_3^+, c_4^+, c_5^+$

Figure 2: An NIN-AND tree

For a root v in T, when parameter $B(v) < 1$, it represents probability $P(v)$. When $B(v) = 1$, it plays a special role to be described below. T is a *full* NIN-AND tree if $X = C$, otherwise T is *partial*. Model M is defined to be full or partial similarly.

Model $M = (X, e, T, B)$ can be obtained by eliciting tree T plus $|X|$ singular causal probabilities. Probability $P(e^+|\underline{x}^+, \underline{y}^-)$, where $C = X \cup Y$ and $X \cap Y = \emptyset$, can then be derived as $P(e^+ \leftarrow \underline{x}^+)$.

Definition 4. *An NIN-AND tree T is* `minimal` *if, whenever a gate g feeds directly into another gate g', the types (direct or dual) of g and g' differ.*

The NIN-AND tree in Fig. 2 is minimal. Model $M = (X, e, T, B)$ is minimal if T is minimal.

3 Probability of Leaf Causal Event

Algorithm 1 below computes probability of the leaf event of a minimal NIN-AND tree model M. All nodes mentioned are event nodes. We refer to the number of gate nodes from an event node v to the leaf as the *level* of v. If event node w feeds into a gate that connects to event node v, we simply refer to w as the parent of v.

Algorithm 1. *GetLeafEventProb(M)*
Input: A minimal NIN-AND tree model M;

```
1 for each non-root node $v$, set $B(v) = 1$;
2 $L$ = max level of any root in $T$;
3 for $lev = L - 1$ to 0, do
4    for each non-root node $v$ at level $lev$, do
5       for each parent $w$ of $v$, do
6          if $w$ is a root, $B(v) = B(v) * B(w)$;
7          else if $B(w) < 1$,
8             $B(v) = B(v) * (1 - B(w))$;
9 return $B(x)$ where $x$ is the leaf;
```

Consider model $M = (X, e, T, B)$, where T is shown in Fig. 2 and B is defined as follows:

$$P(e^+ \not\leftarrow c_1^+) = 0.2, \quad P(e^+ \leftarrow c_2^+) = 0.85,$$
$$P(e^+ \not\leftarrow c_3^+) = 0.3, \quad P(e^+ \leftarrow c_4^+) = 0.65,$$
$$P(e^+ \leftarrow c_5^+) = 0.75$$

GetLeafEventProb(M) returns $P(e^+ \not\leftarrow c_1^+, ..., c_5^+) = 0.0841375$, that is, $P(e^+ \leftarrow c_1^+, ..., c_5^+) = 0.9158625$.

The following proposition shows that if all parameters in M are root event probabilities, GetLeafEventProb(M) computes the leaf event probability exactly.

Proposition 1. *Let M be a minimal NIN-AND tree model with the leaf node x and with $B(v) = P(v) < 1$ for each root v. Then $B(x)$ returned from $GetLeafEventProb(M)$ is equal to $P(x)$ as determined by root event probabilities and causal interactions encoded in the tree topology of M.*

Proof: We prove by induction on the maximal level L of any root. If $L = 1$, all parents of the leaf x are roots. The loop beginning at line 3 iterates exactly once and so does the loop at line 4 (relative to x). Each parent w of leaf x has $B(w) = P(w)$ from specification of M. Hence, we have $B(x) = \prod_w P(w)$ from line 6. Since each $P(w) < 1$, we have $B(x) < 1$.

If the leaf gate g is direct, by Def. 2 (4), each w is a causal success and so is x. T implies that causes satisfy success conjunction and independence. Hence, $B(x)$ correctly models their undermining relation, and $B(x) = P(x)$. If g is dual, by Def. 2 (5), each w is a causal failure and so is x. T implies that causes satisfy failure conjunction and independence. Hence, $B(x)$ correctly models their reinforcing relation, and $B(x) = P(x)$.

Assuming that the proposition holds with $L \leq k$ for $k \geq 1$, consider the case $L = k + 1$. Let y be a node at level 1. If y is a root, we have $B(y) = P(y)$ from specification of M.

If y is a non-root, it defines a subtree T' of T with the leaf being y, and a corresponding submodel M'. The loop at line 3 iterates $k+1$ times during $GetLeafEventProb(M)$. The computation in the first k iterations that involves nodes in T' is exactly the same computation performed by $GetLeafEventProb(M')$. Since for M', $L' \leq k$, by inductive hypothesis, $B(y)$ returned by $GetLeafEventProb(M')$ is exactly $P(y)$. Hence, at the end of k'th iteration during $GetLeafEventProb(M)$, we have $B(y) = P(y) < 1$.

The above argument holds for each node y at level 1. At the $k+1$'th iteration (loop at line 3) of $GetLeafEventProb(M)$, node v is the leaf x. If y is a root, $B(y)$ affects $B(x)$ through line 6. This is correct since y and x are either both causal successes or both causal failures. If y is a non-root, since $B(y) < 1$, test in line 7 will succeed, and $B(y)$ affects $B(x)$ through line 8. This is correct since M is minimal. Either y is a causal failure and x is a causal success, or y is a causal success and x is a causal failure. Hence, $B(x)$ is computed by the correct product according to the type of leaf gate (and its implied causal interaction), and we have $B(x) = P(x)$.

□

Computation of leaf event probability is an essential component of NIN-AND tree modeling. The algorithm above improves upon that in (Xiang and Jia, 2007). It takes advantage of a minimal model, while the latter does not. It is iterative while the latter is recursive. Hence, when the number of input events of a gate is upper bounded, its complexity is $O(n)$, while that of the latter is $O(n^2)$. This saving is amplified during CPT computation, as it needs to be preformed $O(2^n)$ times, as shown below.

4 CPT Computation without Model Regeneration

NIN-AND tree models can be used to acquire a CPT $P(e|C)$ efficiently. As proposed in (Xiang and Jia, 2007), the model $M = (C, e, T, B)$ is elicited first, from which $P(e^+|\underline{c}^+)$ can be computed as $P(e^+ \leftarrow \underline{c}^+)$. For each $P(e^+|\underline{x}^+, \underline{y}^-)$, where $X \subset C$ and $Y = C \setminus X$, a model $M' = (X, e, T', B')$ is created by modifying T into T'. The modification involves removing roots corresponding to absent causes as well as downstream event and gate nodes, and arrives at a significantly different tree structure. $P(e^+|\underline{x}^+, \underline{y}^-)$ is then computed from M' as $P(e^+ \leftarrow \underline{x}^+)$. Hence, this method requires generation of $O(2^{|C|})$ NIN-AND tree models M'.

For instance, suppose $C = \{c_1, ..., c_5\}$ and T is as shown in Fig. 2. To compute $P(e^+|c_1^+, c_2^+, c_3^+, c_4^-, c_5^+) = P(e^+ \leftarrow c_1^+, c_2^+, c_3^+, c_5^+)$, T is modified by deleting event node $e^+ \leftarrow c_4^+$ and gate g_2 to create T' as shown in Fig. 3 (a). T' is not minimal. To use the equivalent minimal NIN-AND tree, the model shown in (b) needs to be created.

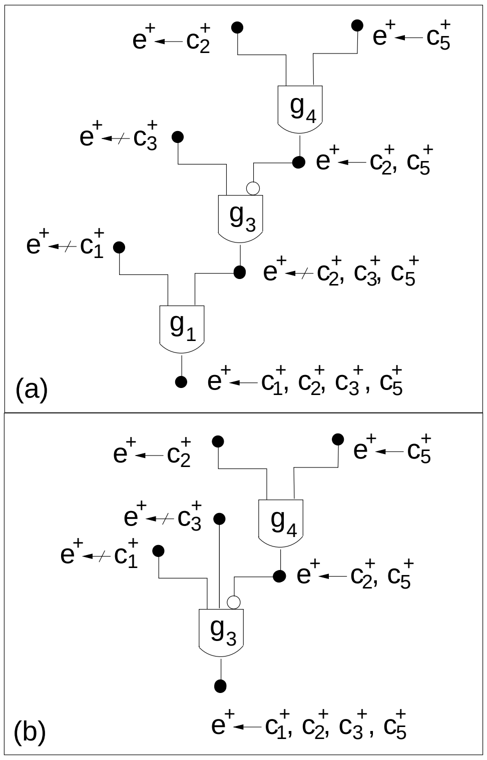

$e^+ \leftarrow c_2^+$
$e^+ \leftarrow c_5^+$
g_4
$e^+ \not\leftarrow c_3^+$
$e^+ \leftarrow c_2^+, c_5^+$
g_3
$e^+ \not\leftarrow c_1^+$
$e^+ \not\leftarrow c_2^+, c_3^+, c_5^+$
g_1
$e^+ \leftarrow c_1^+, c_2^+, c_3^+, c_5^+$
(a)
$e^+ \leftarrow c_2^+$
$e^+ \leftarrow c_5^+$
$e^+ \not\leftarrow c_3^+$
g_4
$e^+ \not\leftarrow c_1^+$
$e^+ \leftarrow c_2^+, c_5^+$
g_3
$e^+ \leftarrow c_1^+, c_2^+, c_3^+, c_5^+$
(b)

Figure 3: NIN-AND trees modified from Fig. 2 when c_4 is absent

We propose an alternative method below for CPT computation which does not require generation of an exponential number of models.

Algorithm 2. *GetCPTByNinAndTree(M)*
Input: A minimal, full NIN-AND tree model M;

```
1  M' = M;
2  $P(e^+|\underline{c}^+) = GetLeafEventProb(M')$;
3  $P(e^+|\underline{c}^-) = 0$;
4  for each non-empty $X \subset C$, do
5     $Y = C \setminus X$;
6     for each root $v$ in $T$, do
7        if causal event at $v$ involves $z \in X$,
8           set $B'(v) = B(v)$;
9        else set $B'(v) = 1$;
10    $P(e^+|\underline{x}^+, \underline{y}^-) = GetLeafEventProb(M')$;
11 return $P(e^+|C)$;
```

$GetCPTByNinAndTree(M)$ uses a duplicated model M' of M for computing all probabilities in $P(e^+|C)$. $P(e^+|\underline{c}^+)$, where all causes are present, is derived from unmodified M'. $P(e^+|\underline{c}^-)$, where all causes are absent, is trivially set, as C is the entire set of causes of e.

All other probabilities in $P(e^+|C)$ are in the form $P(e^+|\underline{x}^+, \underline{y}^-)$, and are computed by the loop started in line 4, one per iteration. In the iteration relative to a given $P(e^+|\underline{x}^+, \underline{y}^-)$, the potential for each root in M' is reset by the loop started in line 6. For each root whose causal event concerns the cause z, if $\underline{x}^+$ includes z, the potential associated with the root is set to the potential in the corresponding root in M. Otherwise, $\underline{y}^-$ must include z, and the potential of the root is set to 1. We consider the effect of this assignment below.

According to Algorithm 1, root potentials are not modified. From line 6 of Algorithm 1, if the potential of a root v is 1, it has no impact to the potential of its child node, and hence has no impact to the value of $P(e^+|\underline{x}^+, \underline{y}^-)$. When a cause z is included in $\underline{y}^-$, it is absent. Hence, this is exactly the effect expected.

Furthermore, consider the child node u of the root v. If all its parents of u are roots like v, $B(u)$ will remain at the value 1 due to line 1 of Algorithm 1. Hence, $B(u)$ will have no impact to the value of $P(e^+|\underline{x}^+, \underline{y}^-)$. This is exactly the effect expected when causes appearing upstream to u are all absent.

On the other hand, if any root parent w of

u involves a cause z that is included in $\underline{x}^+$, it must be the case $B(w) < 1$ (Algorithm 2, line 8). By line 6 of Algorithm 1, $B(w)$ affects $B(u)$ correctly as a factor.

By an inductive argument, any non-root w with ancestors that involve causes in $\underline{x}^+$, it must be the case $B(w) < 1$. By lines 7 and 8 of Algorithm 1, $B(w)$ will affect $B(u)$ correctly as a factor, where u is the child of w. This leads to the following theorem whose proof can be phrased based on the above analysis.

Theorem 1. *Let M be a minimal, full NIN-AND tree model with $B(v) = P(v) < 1$ for each root v. Then $P(e^+|C)$ returned by GetCPT-ByNinAndTree(M) is exact relative to root event probabilities and causal interactions encoded in the tree topology of M.*

Note that $GetCPTByNinAndTree(M)$ needs only modify the root potentials of M' for the computation of each $P(e^+|\underline{x}^+, \underline{y}^-)$. Hence, the computation to generate an exponential number of NIN-AND tree models is saved without affecting exactness.

5 Tree Structure Acquisition by Pairwise Causal Interaction

To acquire a minimal NIN-AND tree model $M = (C, e, T, B)$, the tree structure T must be obtained. It has been shown (Xiang et al., 2009a) that T defines a *pairwise causal interaction* function *pci* from pairs of distinct causes $\{c_i, c_j\} \subset C$, where $i \neq j$, to the set $\{rif, udm\}$, where rif stands for reinforcing and udm stands for undermining. Table 1 shows an example.

Table 1: The *pci* function defined by NIN-AND tree in Fig. 2

c_i	c_j	$pci(c_i, c_j)$	c_i	c_j	$pci(c_i, c_j)$
c_1	c_2	rif	c_2	c_4	udm
c_1	c_3	rif	c_2	c_5	udm
c_1	c_4	rif	c_3	c_4	udm
c_1	c_5	rif	c_3	c_5	rif
c_2	c_3	rif	c_4	c_5	udm

Furthermore, for $|C| \leq 10$, it has been computationally verified (Xiang et al., 2009a) that, the *pci* function of T is unique among *pci* functions of alternative minimal NIN-AND trees over C and e. Therefore, given the *pci* function of T, the tree structure can be identified uniquely. For instance, given Table 1, Fig. 2 is the unique full minimal NIN-AND tree. This fact suggests that, instead of eliciting T directly from expert, it can be obtained alternatively by eliciting the corresponding *pci* function. Denote $n = |C|$, the number of pairwise causal interactions to be elicited is $n(n-1)/2$. We investigate this approach below.

First of all, does every *pci* function correspond to a NIN-AND tree? Since given $n = |C|$, the number of pairwise causal interactions is $n(n-1)/2$, the number of alternative *pci* function for n causes is $2^{n(n-1)/2}$.

On the other hand, every NIN-AND tree defines the same *pci* function as its minimal NIN-AND tree. Hence, it is sufficient to count the number of minimal NIN-AND trees. Unfortunately, no closed formula is known for the number of minimal NIN-AND trees over n causes. Instead, the number can be obtained by enumeration of minimal NIN-AND trees given n (Xiang et al., 2009b; Xiang et al., 2009a).

Table 2 compares the number of alternative *pci* functions and that of minimal NIN-AND trees given n. The rows are indexed by n. The second column lists the number of *pci* functions. The third column lists the number of minimal NIN-AND trees. The last column is the ratio of the number in the second column over that in the third column.

Table 2: Comparison of the number of *pci* functions and that of minimal NIN-AND trees

n	No. pci func	No. trees	Ratio
3	8	8	1
4	64	52	1.2
5	1024	472	2.2
6	32768	5504	6
7	2097152	78416	26.7
8	268435456	1320064	203
9	68719476736	25637824	2680

We refer to a *pci* function that can be implemented by an NIN-AND tree as *feasible*. Otherwise, it is *infeasible*. Hence, Table 2 shows that other than $n = 3$, there are more infeasible *pci* functions than feasible ones. It follows that in acquisition of an NIN-AND tree by elicitation of the corresponding *pci* function, an expert could specify an infeasible *pci* function. This could happen for at least two reasons. The expert could misspecify some pairwise interaction (human error), or the specified function is not expressible by an NIN-AND tree model.

From the human error perspective, having more infeasible *pci* functions is desirable, because a misspecified *pci* function can thus be detected rather than being regarded as feasible and causing an unintended NIN-AND tree to be returned unnoticed.

Furthermore, as n grows, the ratio between number of *pci* functions and number of minimal NIN-AND trees grows very rapidly. This is also desirable because, as n grows, a misspecified *pci* function is even more likely to be detected. This counter-balances the increased chance of human error due to the increased number of pairwise causal interactions to be elicited as n grows.

Next, we consider what an NIN-AND tree acquisition software should do when expert specifies an infeasible *pci* function. Rejecting the function is a simple measure. However, it would be more helpful to assist expert in correcting the mistake in the case of human error, and in providing an approximation in the case of inexpressibility. We propose such a method below.

Definition 5. *Let C be a set of causes and Pci be a pci function over C. Let ρ be an ordering of pairs of causes in C. Let Pci_ρ be the sequence of Pci values ordered according to ρ. Let $BPci_\rho$ be the binary string obtained from Pci_ρ by replacing each rif value with 1 and each udm value with 0. Then $BPci_\rho$ is the binary string expression of Pci relative to ordering ρ.*

Definition 6. *Let C be a set of causes, and Pci and Pci' be two pci functions over C. Let $BPci_\rho$ and $BPci'_\rho$ be the binary string expressions of Pci and Pci', relative to ordering ρ of causes in C. Then the following sum is the distance between Pci and Pci':*

$$Dist(Pci, Pci') = \sum_{i=1}^{n(n-1)/2} |BPci_\rho[i] - BPci'_\rho[i]|,$$

where $BPci_\rho[i]$ refers to the ith bit of $BPci_\rho$.

Definition 7. *Let Pci be a pci function over C. The minimum distance of Pci from feasible pci functions over C is*

$$MinDist(Pci) = \min_T Dist(Pci, Pci^T),$$

where Pci^T is the pci function defined by an NIN-AND tree T.

Note $MinDist(Pci) = 0$ if Pci is feasible. Otherwise, we have $MinDist(Pci) > 0$, and there exists a set of minimal NIN-AND trees whose *pci* functions differ from Pci by distance $MinDist(Pci)$. When an expert-specified Pci is detected as infeasible, we propose to return in response either this set of minimal NIN-AND trees or the pairwise interactions implied by them with those interactions that differ from Pci highlighted. This response should be useful for interactive error correction by expert and for choosing the best approximate NIN-AND tree.

For example, suppose that expert specified the *pci* function in Table 1 with an error $pci(c_2, c_4) = rif$. The resultant function Pci is infeasible. It can be shown that $MinDist(Pci) = 1$ and there are six minimal NIN-AND trees whose *pci* functions differ from Pci by distance 1. One of them is Fig. 2.

We exhaustively tested all *pci* functions for $4 \le n \le 7$ (Test for $n = 7$ took about 50 hours on a 8-core workstation). For each n value, we define the following *maximal minimum distance* between individual *pci* function and feasible *pci* functions over n causes:

$$MaxMinDist(n) = \max_{Pci} MinDist(Pci),$$

where Pci is a *pci* function over n causes. Table 3 shows this distance for $4 \le n \le 7$.

This result shows that for $n = 7$, if a function Pci is infeasible, a minimal NIN-AND tree exists whose *pci* function differs from Pci by no more than four pairwise causal interactions.

Table 3: Maximal minimum distance between individual *pci* function and feasible ones

n	4	5	6	7
$MaxMinDist(n)$	1	2	2	4

Our experiments also show that for many infeasible *pci* functions, a minimal NIN-AND tree exists whose *pci* function differs from the infeasible *pci* function by one pairwise interaction.

For an infeasible *pci* function Pci, there exist generally multiple minimal NIN-AND trees whose *pci* functions differ from Pci by distance $MinDist(Pci)$. We define the *maximum number of minimum distance trees* for *pci* functions over n causes as follows:

$$MaxNumMinDistTree(n) = \max_{Pci} |MinDistTree(Pci)|,$$

where Pci is a *pci* function over n causes, and $MinDistTree(Pci)$ is the set of minimal NIN-AND trees whose *pci* functions differ from Pci by distance $MinDist(Pci)$. Table 3 shows this number for $4 \leq n \leq 7$.

Table 4: Maximum number of minimum distance trees for *pci* functions over n causes

n	4	5	6	7
$MaxNumMinDistTree(n)$	6	10	15	49

Our experiments also show that for many infeasible *pci* functions, $|MinDistTree(Pci)|$ is a small number (less than 10).

These results suggest that acquisition of minimal NIN-AND trees through elicitation of *pci* function is feasible.

6 Conclusion

We presented an algorithm that improves the efficiency for computing conditional probability from an NIN-AND tree model. We proposed another algorithm that computes CPT from a full minimal NIN-AND tree model without having to generate an exponential number of models. Finally, we presented a technique that allows interactive acquisition of NIN-AND trees from pairwise causal interactions.

Future work includes suitability of NIN-AND trees as approximations of arbitrary CPTs, their exploitation in inference, human acquisition testing, and acquisition from learning.

Acknowledgement

Financial support from NSERC Discovery Grant is acknowledged. I thank anonymous reviewers for their comments.

References

S.F. Galan and F.J. Diez. 2000. Modeling dynamic causal interaction with Bayesian networks: temporal noisy gates. In *Proc. 2nd Inter. Workshop on Causal Networks*, pages 1–5.

D. Heckerman and J.S. Breese. 1996. Causal independence for probabilistic assessment and inference using Bayesian networks. *IEEE Trans. on System, Man and Cybernetics*, 26(6):826–831.

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

P.P. Maaskant and M.J. Druzdzel. 2008. 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.

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

Y. Xiang and N. Jia. 2006. 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.

Y. Xiang and N. Jia. 2007. Modeling causal reinforcement and undermining for efficient cpt elicitation. *IEEE Trans. Knowledge and Data Engineering*, 19(12):1708–1718.

Y. Xiang, Y. Li, and J. Zhu. 2009a. Towards effective elicitation of NIN-AND tree causal models. In L. Godo and A. Pugliese, editors, *Inter. Conf. on Scalable Uncertainty Management (SUM 2009), LNAI 5785*, pages 282–296. Springer-Verlag Berlin Heidelberg.

Y. Xiang, J. Zhu, and Y. Li. 2009b. 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.

Y. Xiang. 2010. Generalized non-impeding noisy-AND trees. In *Proc. 23th Inter. Florida Artificial Intelligence Research Society Conf.*, pages 555–560.