Mixing ICI and CSI Models for More Efficient Probabilistic Inference

Michael Roher and Yang Xiang

University of Guelph, Canada

Abstract. Conditional probability tables (CPTs) in Bayesian Networks (BNs) have exponential space on the family size. Local models based on independence of causal influence (ICI) or context-specific independence (CSI) have been applied separately to improve the efficiency. We propose a framework to mix both local models in the same BN for improved efficiency. In particular, we show that ICI and CSI are orthogonal, and each is unable to express the other efficiently and accurately. We propose a formalism to encode both types of local models in the same BN, and to convert it into a homogenous representation to support exact inference. We report experimental evaluation where significant efficiency gain is obtained in exact inference.

Keywords: Bayesian networks · Probabilistic inference · Causal independence models · Context-specific independence models

1 Introduction

Discrete Bayesian networks (BNs) [6] exploit conditional independence among variables through directed acyclic graph (DAG) structures, and only quantify dependence of variables on their parents by conditional probability tables (CPTs). As tabular CPTs have exponential space, which extends to inference complexity, local models have been applied for further efficiency. Some exploit independence of causal influence (ICI), e.g., noisy-OR [6], noisy-MAX [4], DeMorgan [5], Non-Impeding Noisy-AND Tree (NIN-AND Tree or NAT) [11], and cancellation model [10]. Other local models exploit context-specific independence (CSI), e.g., CPT-trees [1], rule-based CSI [8], and algebraic decision diagrams [2].

These methods exploit ICI or CSI, but not both. Since ICI and CSI apply to individual families of variables in BNs, they can co-exist in an environment (see Section 7). In such cases, methods that exploit only one type of local models lose the opportunity afforded by also exploiting the other type.

We propose a framework that exploits both ICI and CSI for more efficient inference in BNs. When both exist, we apply NAT local models for ICI and CPT-tree local models for CSI, encoding both in the same BN. We convert each type of local models accordingly to obtain a homogeneous runtime representation for more efficient inference.

The remainder is organized as follows: Section 2 reviews background on NAT and CSI. We establish their orthogonality in Section 3. We analyze alternatives for mixing NAT and CSI in Section 4, and specifies our choice formalism. Section 5 formalizes CPT-tree transformation, and Section 6 combines it with NAT de-causalization [13] to obtain a homogeneous runtime representation. We report experimental results in Section 7.

2 Background

2.1 NAT Modelling of ICI

We review NAT modelling (see [11, 13] for more details). A NAT model encodes dependency of an effect e on a set of uncertain causes $C = \{c_1, ..., c_n\}$, where $e \in \{e^0, ..., e^\nu\}$ $(\nu \geq 1)$ and $c_i \in \{c_i^0, ..., c_i^{m_i}\}$ $(i = 1, ..., n;\ m_i \geq 1)$. The effect and cause are inactive at e^0 and c_i^0, and are active at other values (may be written as e^+ or c_i^+) where higher indices may denote higher intensity. C and e form a family in BNs, where C is the parent set of e.

A causal event is a *success* or *failure* depending on if e is produced up to a certain value, is *single-* or *multi-causal* depending on the number of active causes, and is *simple* or *congregate* depending on the number of active effect values. A simple single-causal success is an event that cause c_i of value c_j^i $(j > 0)$ renders e to occur at value e^k $(k > 0)$, when other causes are inactive. Its probability is denoted $P(e^k \leftarrow c_i^j) = P(e^k | c_i^j, c_z^0 : \forall z \neq i, j > 0)$. A congregate multi-causal success is an event where a set of active causes $X = \{c_1, ..., c_q\}$ caused e to occur at e^k $(k > 0)$ or higher intensity. Its probability is denoted $P(e \geq e^k \leftarrow c_1^{j_1}, ..., c_q^{j_q}) = P(e \geq e^k | c_1^{j_1}, ..., c_q^{j_q}, c_m^0 : c_m \in C \setminus X)$, where $j_i > 0$ for $i = 1, ..., q$, or $P(e \geq e^k \leftarrow \underline{x}^+)$ for simplicity.

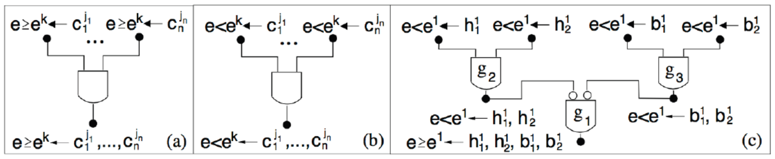
$e \geq e^k \leftarrow c_1^{j_1}$
$e \geq e^k \leftarrow c_n^{j_n}$
$e \geq e^k \leftarrow c_1^{j_1}, ..., c_n^{j_n}$
(a)
$e < e^k \leftarrow c_1^{j_1}$
$e < e^k \leftarrow c_n^{j_n}$
$e < e^k \leftarrow c_1^{j_1}, ..., c_n^{j_n}$
(b)
$e < e^1 \leftarrow h_1^1$
$e < e^1 \leftarrow h_2^1$
$e < e^1 \leftarrow b_1^1$
$e < e^1 \leftarrow b_2^1$
g_2
g_3
g_1
$e < e^1 \leftarrow h_1^1, h_2^1$
$e < e^1 \leftarrow b_1^1, b_2^1$
$e \geq e^1 \leftarrow h_1^1, h_2^1, b_1^1, b_2^1$
(c)

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

A NAT is composed of two types of NIN-AND gates, each over disjoint sets of causes $W_1, ..., W_q$. An input event of a *direct gate* (Fig. 1 (a)) is a causal success $e \geq e^k \leftarrow \underline{w_i}^+$, and the output event is $e \geq e^k \leftarrow \underline{w_1}^+, ..., \underline{w_q}^+$. An input of a *dual gate* (Fig. 1 (b)) is causal failure $e < e^k \leftarrow \underline{w}_i^+$, and the output event is $e < e^k \leftarrow \underline{w_1}^+, ..., \underline{w_q}^+$. Probability of output event is the product of input event probabilities.

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

A direct gate encodes undermining interactions and a dual gate encodes reinforcing interactions. They are combined in a NAT to express complex interactions among causes. Fig. 1 (c) shows a NAT with 3 gates. Causes h_1 and h_2 reinforce each other, and so do b_1 and b_2. The two groups undermine each other.

A BN is *NAT-modelled* if the CPT of each variable of 2 or more parents is a NAT model. Its space is linear: $O(N\ \kappa\ n)$, where N is the number of variables, κ bounds variable domain sizes, and n bounds the number of parents per variable.

Common inference methods for BNs do not directly apply to NAT-modelled BNs. *Normalizing* NAT models to full tabular CPTs loses efficiency of NAT-modelling. Techniques to support efficient inference include *multiplicative factorization*, where NAT-modelled BNs are converted to equivalent, efficient Markov networks, and *de-causalization*, where they are converted to equivalent, efficient tabular BNs. For NAT-modelled BNs with high treewidth and low density (measured by percentage of links beyond being singly connected), two orders of magnitude speedup in inference has been demonstrated.

2.2 CPT-tree Modelling of CSI

We review CSI (see [1] for more details). For a BN variable, a *context* is an assignment of values to some parents. For disjoint sets of variables X, Y, Z, and Cxt, X and Y are *contextually independent* given Z and context $Cxt = cxt$, denoted $I_c(X; Y|Z, Cxt = cxt)$, if $P(X|Z, cxt, Y) = P(X|Z, cxt)$ whenever $P(Z, cxt, Y) > 0$.

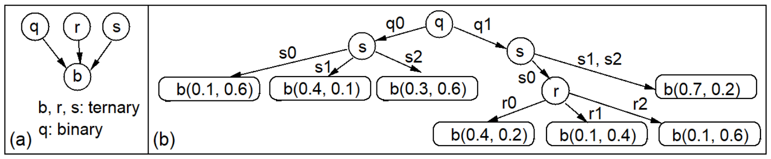

Fig. 2. (a) A BN family. (b) CPT-Tree for the family.

When CSI exists in a BN family, the CPT contain similar values. The BN family in Fig. 2 (a) admits $I_c(b; r|s, q = q_0)$ and $I_c(b; r|q = q_1, s \in \{s_1, s_2\})$. Its CPT has 14 parameters, though $P(b|q, r, s)$ generally has 36.

A CPT with CSI can be specified as CPT-tree (b)[1]. A CPT-tree for variable x and parents $\pi(x)$ is directed from the root. Each non-leaf is a variable in $\pi(x)$. Each path from the root to a leaf is a context, and the leaf specifies the conditional probability distribution (CPD) of x, given the context. The CSI above are expressed by the left subtree and the rightmost branch, respectively.

We refer to BNs where some families are modelled by CPT-trees as *CPT-tree-modelled BNs*. Common inference methods for BNs do not directly apply to CPT-tree-modelled BNs. Techniques that support inference with CPT-tree-modelled BNs include network transformation and clustering [1], cutset conditioning [1], and variable elimination [7].

[1] The example generalizes CPT-trees in [1] slightly as explained in Section 5.

3 Orthogonality of NAT and CSI Models

A fundamental question that may undermine efforts to exploit a mixture of NAT and CSI is whether the local models are orthogonal. A negative answer renders the effort invalid, since one type of local models can be encoded by the other. For instance, alternative ICIs, noisy-OR, noisy-MAX, and DeMorgan, are all special NAT models. Below, we empirically answer the question positively.

First, we show that CSI generally cannot be exactly expressed as NAT models. A batch of 100 seed CPTs $P(x_0|x_1, x_2, x_3, x_4, x_5)$ are simulated, where variables have the same domain $\{1, 2, 3, 4, 5\}$.

Given a seed CPT P and a CSI, we generate a CSI CPT P^* as follows: For $I_c(x_0; x_1, x_2, x_3|x_4, x_5 = 5)$, P^* must satisfy $P^*(x_0|x_1, x_2, x_3, x_4, x_5 = 5) = P^*(x_0|x'_1, x'_2, x'_3, x_4, x_5 = 5)$. For each distinct assignment (x_0, x_4), we arbitrarily assign (x_1, x_2, x_3), retrieve value $P(x_0|x_1, x_2, x_3, x_4, x_5 = 5)$ from P, and assign to every term $P^*(x_0|x'_1, x'_2, x'_3, x_4, x_5 = 5)$.

We specified 3 alternative CSIs (Table 1). They allow different space reduction (2nd col.). Using the above method, we generated 3 CSI CPTs for each of 100 seed CPTs (a total of 400 source CPTs). We then compress each source CPT into NAT model, and evaluate the average Kullback-Leibler and Euclidean distances between the NAT and source CPTs (4th and 5th cols.).

Table 1. Summary of experiments on representing CSI CPTs as NAT models.

CSI Statement	# Src Para	# NAT Para	KL	ED
No CSI	12,500	80	0.738	0.219
$I_c(x_0; x_4, x_5\|x_1 = 1, x_2 = 2, x_3 \in \{3, 4\})$	12,304	80	0.710	0.214
$I_c(x_0; x_2, x_3, x_4, x_5\|x_1 = 1)$	10,004	80	0.692	0.210
$I_c(x_0; x_2, x_3, x_4, x_5\|x_1 \in \{1, 2, 3, 4\})$	2,504	80	0.501	0.176

Table 1 reveals that source CPTs take much more space than resultant NAT models (> 30 times). On the other hand, NAT models reduce space 30 to 150 times, but with errors. Though errors decrease as the numbers of source CPT parameters, NAT models generally cannot express CSI CPTs exactly.

Next, we show that NAT models generally cannot be suitably expressed as CSI models. Given a NAT CPT, if its probabilities can be grouped into tight clusters (small distance between member values), and the total number of such clusters is significantly less than the number of NAT parameters, then it pays to express the NAT CPT as a CSI model. A smaller number of clusters means space saving, and tight clusters mean small approximation errors. Based on this idea, given a NAT CPT and a distance bound δ (e.g., $\delta = 0.02$), we group values in the CPT into a set Ψ of clusters, such that the following conditions hold:

1. For each cluster $Q \in \Psi$ and each pair of values $p, q \in Q$, $|p - q| \leq \delta$.
2. For each two clusters $Q, R \in \Psi$, let $min_Q, min_R, max_Q, max_R$ be extreme values in Q and R, respectively. Either $max_Q < min_R$ or $max_R < min_Q$.
3. For clusters $Q, R \in \Psi$ where $max_Q < min_R$, we have $min_R - max_Q > \delta$.

Condition 1 bounds inner distance within each cluster. Condition 2 orders clusters by member values. Condition 3 bounds inter-cluster distance.

The number of clusters obtained is a lower-bound of the number of parameters when the NAT CPT is approximated by a CSI model. This is because values in the same cluster may refer to incompatible contexts, and cannot be encoded by the same CPT-tree leaf. We split such clusters as needed.

The clustering is applied to 100 generated NAT CPTs, each over a family of 5 parent variables. All variables are binary, with 32 parameters per CPT. Results with $\delta = 0.02$ are shown in Fig. 3. Each bar counts NAT CPTs that produced a particular number of clusters. As values in a cluster can be encoded by a single value with error $< \delta$, the number of clusters indicates the number of parameters needed if those NAT CPTs are encoded as CPT-trees. As is shown, all CPT-trees require at least 17 parameters, while the NAT CPT only needs 5.

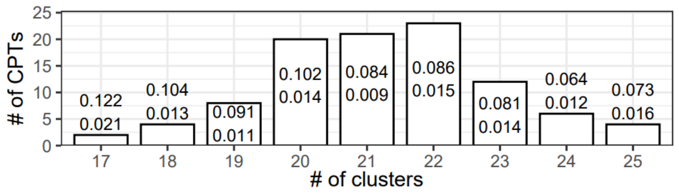

Fig. 3. Experiment results on representing NAT CPTs as CSI models.

CSI modelling errors are evaluated as follows: Compute the *centroid* of each cluster as the mean of its values, and use it as the CPT-tree parameter. The error to model a NAT CPT as CPT-tree is the Euclidean distance between the two CPTs. In Fig. 3, average modelling error for CPTs in each bar is at the top, with the standard deviation below it.

To summarize, it is generally not possible to encode CSI CPTs exactly as NAT models. When NAT CPTs are represented as CSI models, it generally not only introduces error, but also increases the number of parameters required. These evidences suggest that NAT models and CSI models are orthogonal.

4 Mixed NAT-CSI Bayesian Networks

ICI and CSI can each be exploited to improve space and inference efficiency in BNs. To our knowledge, no prior study considered inference on BNs that take advantage of both simultaneously. For that purpose, we resolve issues below:

First, we observe that ICI and CSI are applicable to individual families in BNs. Therefore, both can co-exist in an environment as well as in a BN: Some family of variables follow ICI local models and others follow CSI local models.

Second, suitable representation is needed for each type of local models. For ICI, we focus on NAT models for several reasons: They express both reinforcing and undermining interactions. They can mix such interactions recursively among cause subsets. They apply to multi-valued, ordinal, and nominal variables. They generalize other ICI models including noisy-OR, noisy-MAX, and DeMorgan, while having the same linear space.

For CSI, several formalisms are available. Rule bases and algebraic decision diagrams (ADDs) have been used for inference by variable elimination [7, 2].

When used to answer multiple queries (over multiple unobserved variables), the compilation requires knowledge of evidence prior to inference. Loops in ADDs tend to increase treewidth of the resultant structure. As we aim at computing posteriors of all unobserved variables, with arbitrary evidence at inference time, we selected CPT-trees [1] to encode CSI.

We define the representation of choice as *mixed NAT-CSI Bayesian net* (MNCBN) (an example is given in Section 6):

Definition 1 *A MNCBN is a BN (V, D, P) over a set V of variables with dependency structure DAG D. The set P of CPTs is composed of one CPT per variable in V, partitioned into (TC, NM, CT), where TC is a set of tabular CPTs, NM is a set of NAT models, and CT is a set of CPT-trees.*

Third, neither NAT models nor CPT-trees support common BN inference algorithms directly. Each type of local models admits alternative processings before inference. NAT modelled BNs admit multiplicative factorization or de-causalization. CPT-tree-modelled BNs admit network transformation, cutset conditioning, or variable elimination. To prepare MNCBNs for inference, we have chosen to compile them into a homogeneous representation, by combining de-causalization for NAT models and network transformation for CPT-trees, as both convert local models into equivalent BN segments that tend to reduce treewidth. This choice assumes no prior knowledge on evidence and supports computing posteriors over all unobserved variables. In comparison, multiplicative factorization for NAT models and cutset conditioning with CPT-trees do not support a homogeneous representation. We demonstrate the choice in Section 6.

5 Formalizing CPT-tree Transformation

We apply network (CPT-tree) transformation to convert CPT-trees in MNCBNs to BN segments. Although the idea is illustrated in [1] with a simple example over binary variables, to the best of our knowledge, no general algorithm has been formalized. We specify an algorithm suite, formalizing processings on multi-valued variables, set-valued CPT-tree branches, and multiplexer CPTs.

Let $dom(x)$ be the domain of variable x. A CPT-tree arc outgoing from node t may be labeled by a subset of $dom(t)$. A path from the root to a leaf, including such arcs, defines a set of contexts, e.g., the rightmost branch in Fig. 2 (b). CPT-trees with such set-valued arcs generalize those in [1], and allow more efficient CSI encoding.

Algo. `SetDagSeg` takes a CPT-Tree T over variable x and parents $\pi(x)$, and builds a BN segment with auxiliary variables, all of which have domain $dom(x)$. Each node of T is at a level with the root at level 0, and each child of the root is at level 1. Transformation is driven by topology of T from level 0 onwards. For each node t in T, denote the path from root to t by $path(t)$. The output is a DAG G with a single leaf x, and G is constructed from x upwards.

Algorithm 1 *SetDagSeg*$(x, \pi(x), T)$

```
1  initialize empty graph G with nodes $\{x\} \cup \pi(x)$;
2  denote the root of T by $\rho$ and set $path(\rho) = \{\}$;
3  label x in G as $x_{path(\rho)}$;
4  for level $L = 0$ to max level in T,
5     for each node t in T at level L with path(t);
6        find node v in G that is labelled $x_{path(t)}$ and add arc $t \rightarrow v$ in G;
7        if each child of t in T is leaf, continue;
8        denote partition of dom(t) by arcs outgoing from t as $\{sd_1, ..., sd_m\}$;
9        for i=1 to m,
10          add node y to G with domain dom(x) and label it $x_{\{path(t), t \in sd_i\}}$;
11          add arc $y \rightarrow v$ in G;
12 return G;
```

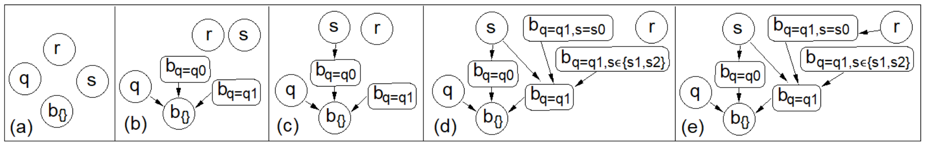

Fig. 4. Transformation of the CPT-Tree in Fig. 2.

For CPT-tree T in Fig. 2, `SetDagSeg` begins with G in Fig. 4 (a). For level $L = 0$ of T, q in T is processed with resultant G in (b). For $L = 1$, 1st instance of s in T is processed as in (c), followed by 2nd instance as in (d), where arcs outgoing from s partition *dom*(s) into subdomains $\{s_0\}$ and $\{s_1, s_2\}$. For $L = 2$, r in T is processed to produce (e).

Node v in line 6 may not be processed by the *for* loop in line 9. If it is, multiple parents are added to v, and v is called a *multiplexer*. In Fig. 4 (e), $b_{\{\}}$ and $b_{q=q_1}$ are the only multiplexers.

Next, Algo. `AssignCpt` assigns a CPT to each node in G except those in $\pi(x)$. They include $x_{\{\}}$ and nodes added by `SetDagSeg` line 10, divided as follows. Type 1: They are added in line 10 and never processed after by line 6. Hence, they remain roots, e.g., $b_{q=q_1, s \in \{s_1, s_2\}}$. Type 2: They are processed in line 6 as v and by the *for* loop in line 9. They are multiplexers such as $b_{\{\}}$ and $b_{q=q_1}$. Type 3: They are the remaining nodes that are processed in line 6 as v, passed test in line 7, and skipped the *for* loop in line 9, e.g., $b_{q=q_0}$ and $b_{q=q_1, s=s_0}$.

For Type 1 node v in G, traverse its path in T from root to a leaf, and assign its CPD to v. For $b_{q=q_1, s \in \{s_1, s_2\}}$ in Fig. 4 (e), follow path $(q = q_1, s \in \{s_1, s_2\})$ in Fig. 2 (b), and assign $P(b_{q=q_1, s \in \{s_1, s_2\}})$ *over* $(b_0, b_1, b_2) = (0.7, 0.2, 0.1)$.

For Type 3 node v, traverse its path in T to a node t. Since t passed test in line 7, each child of t is a leaf that specifies a CPD. Assemble them to form a CPT and assign to v. For $b_{q=q_0}$, follow path $(q = q_0)$ in Fig. 2 (b) to node s, retrieve CPDs, and assign CPT: $P(b_{q=q_0}|s_0) = (0.1, 0.6, 0.3)$, $P(b_{q=q_0}|s_1) = ...$, $P(b_{q=q_0}|s_2) = ...$ Algo. `AssignCpt` formalizes the above, where CPTs for Type 2 nodes are from `SetSwitchCpt` presented below.

Algorithm 2 *AssignCpt*$(x, \pi(x), T, G)$

```
1  for each node $v$ in $G$,
2    if $v \in \pi(x)$, continue;
3    if $v$ is Type 1 with $path(v)$,
4      traverse $path(v)$ in $T$ to leaf $t$;
5      retrieve CPD parameters at $t$ and assign the CPD to $v$;
6    else if $v$ is Type 3 with $path(v)$,
7      traverse $path(v)$ in $T$ to node $t$;
8      for each child $z$ of $t$ in $T$, retrieve CPD parameters at $z$;
9      assemble the CPDs into CPT and assign to $v$;
10   else
11     denote the unique parent of $v$ from $\pi(x)$ by $y$;
12     call $SetSwitchCpt(v, y, T, G)$ and assign the CPT returned to $v$;
13 return $G$;
```

Family of a Type 2 node v is created in `SetDagSeg` in line 6 (1st parent) and 11 (other parents). The 1st parent y is from $\pi(x)$, and is identified in line 11 of `AssignCpt`. Each other parent is auxiliary. CPT at v is to deterministically set v value according to an auxiliary parent. The right parent is decided by y value and path label of the parent. This is specified in Algo. `SetSwitchCpt`.

Algorithm 3 *SetSwitchCpt*(v, y, T, G)

```
1  initialize CPT $P(v|y, u_1, ..., u_k)$ where $\{y, u_1, ..., u_k\}$ is parent set of $v$ in $G$;
2  for each assignment $(v = v', y = y', u_1 = u'_1, ..., u_k = u'_k)$,
3    find $u_i$ in $\{u_1, .., u_k\}$ whose path label $= (path(v), y \in sd_i)$ and $y' \in sd_i$;
4    if $v' = u'_i$, $P(v'|y', u'_1, ..., u'_k) = 1$;
5    else $P(v'|y', u'_1, ..., u'_k) = 0$;
6  return $P(v|y, u_1, ..., u_k)$;
```

Consider CPT for $b_{q=q_1}$ with parent $s \in \pi(x)$ (Fig. 4 (e)). To set value $P(b_{q=q_1} = b_0 | s = s_1, b_{q=q_1,s=s_0} = b_0, b_{q=q_1,s\in\{s_1,s_2\}} = b_1)$, variable $b_{q=q_1,s\in\{s_1,s_2\}}$ is selected since $s = s_1$ satisfies its path label. Value $b_{q=q_1,s\in\{s_1,s_2\}} = b_1$ is then compared with $b_{q=q_1} = b_0$. Since they do not match, it results in value 0 for the above probability.

6 Inference with Mixed NAT-CSI Bayesian Networks

We outline and demo a framework for exact inference with MNCBNs, that exploit both types of local models for improved efficiency.

First, the MNCBN is converted to standard BN: Each NAT model is decausalized into an efficiency preserving BN segment. Each CPT-tree is transformed as described in Section 4. The result is a homogeneous, standard BN, encoding the same joint probability distribution (JPD), with a treewidth lower than that of MNCBN (if sparse). We refer to it as a *de-causalized and transformed BN* (DTBN). The DTBN supports exact inference with any common method.

Consider a MNCBN with the DAG in Fig. 5 (a). Family of g is a NAT model (b) and family of h is modelled by CPT-tree (c). All variables are ternary. Fig. 6 shows DAG of the DTBN. The BN segment de-causalizing g family is dashed. In the dashed region, variables outside $\{g\} \cup \pi(g)$ are auxiliary. If all CPTs are tabular, the MNCBN has a total CPT size of 4506, where the largest CPT has size 2187. The DTBN has a total CPT size of 1267, where the largest CPT has size 243.

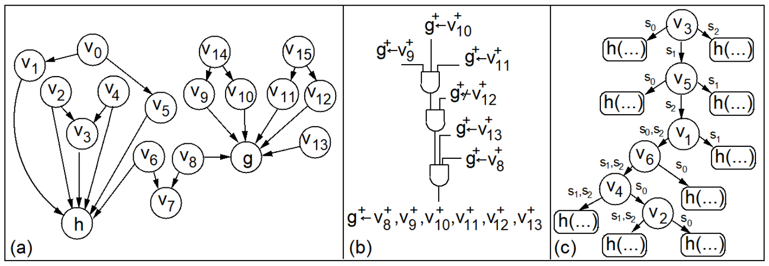

Fig. 5. (a) BN DAG. (b) NAT model for family of g. (c) CPT-tree over family of h.

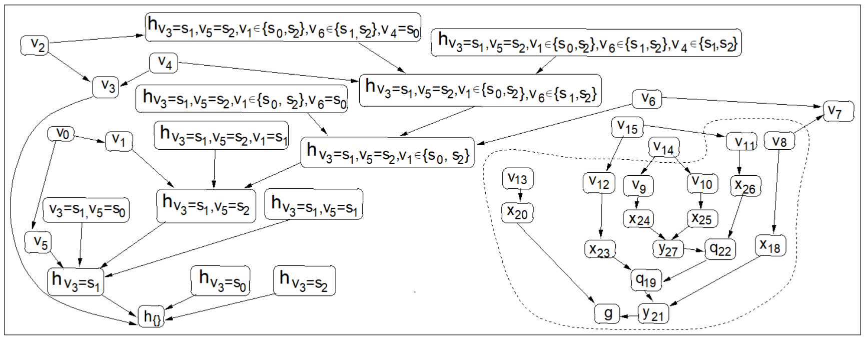

Fig. 6. The DTBN.

7 Experiments

Our experimental study aims to (i) confirm co-existence of NAT and CSI in real world BNs, (ii) evaluate computational gain by mixing NAT and CSI local models, and (iii) compare effectiveness of NAT and CSI models.

The 1st experiment confirms co-existence of NAT and CSI local models. NAT modelling has been applied to approximate 8 real world BNs from the *bnlearn* repository with reasonable inference errors [12]. Here, we test the 2 binary BNs from the 8, *Andes* and *Win95pts*, for CSI local models. We apply clustering in Section 3 to each CPT where the variable has 2 or more parents. Similar probabilities are grouped, subject to upper bound on inner-cluster distance and lower bound on inter-cluster distance. We use bound $\delta = 0.02$.

The results are shown in Table 2. The maximum number of parameters per CPT (4th col.) is 64 (6 parents) and 128 (7 parents), respectively. The maximum number of clusters found per CPT (5th col.) is 3 and 6, respectively. Hence, a significant amount of CSI exists in these CPTs. If modelled as CPT-trees, the Euclidean error for Win95pts is 0.041 (6th col.). The error for Andes is 0: For each cluster, all values are identical.

Table 2. Summary of results from clustering *Andes* and *Win95pts* BNs.

BN	#Node	#Fmly Proced	Max #Para/CPT	Max #Clus/CPT	Eu Dist
Andes	223	50	64	3	0
Win95pts	76	24	128	6	0.041

CSIs have also been identified by others from biological datasets, other BNs in *bnlearn*, and UCI datasets, e.g., [3, 9]. These studies, the result above, and the NAT modelling study [12] suggest collectively co-existence of NAT and CSI local models in practice.

The 2nd experiment evaluates computational gain by mixing NAT and CSI models. We simulated MNCBNs each of 100 variables (binary or ternary), where 50% of families of 2 parents or more are NAT-modelled and the remaining 50% are CSI-modelled. The largest number of parents per node is 12, and each MNCBN has at least 2 such families. At least one such family is NAT-modelled, and at least one is CPT-tree modelled. Structural density of MNCBNs is controlled at $d = 5\%$ or 10% more links beyond being singly connected.

When each variable is unique in a CPT-tree, its transformation has no loop. Duplicated variables, e.g., s in Fig. 2, induce loops (see Fig. 4 (e)), and increases treewidth of the transformed structure. We control the number of variable duplications at $k = 0, 2, 4, 7, 10$. For each combination of (d, k), 30 BNs are generated. Hence, a total of 300 distinct MNCBNs are generated. Each MNCBN is converted into 4 standard BNs (encoding the same JPD) by methods D+N, D+T, N+N, and N+T, where D refers to De-causalizing NAT models, T refers to Transforming CPT-trees, and N referes to Normalizing NAT models and CPT-trees.

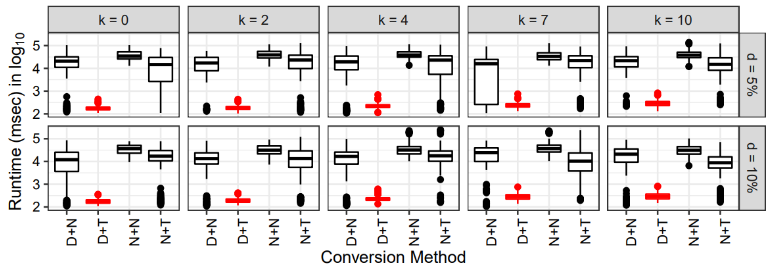

Fig. 7. Summary of inference runtimes

Each resultant BN is compiled for inference by lazy propagation (LP) (See Section 8 for rationale). Each BN has 10 inference runs, each with different observations over 10 randomly selected variables. Inference runs by BNs from the same MNCBN yielded the same posterior marginals (exact). Runtimes (2.9GHz desktop) are shown in Fig. 7.

In all (d, k) combinations, N+N is the slowest. Both D+N and N+T are advantageous, even though they only exploit one type of local models. D+T is on average two orders of magnitude faster than alternatives, demonstrating clear computational advantage of exploiting both NAT and CSI in MNCBNs.

Relative performance of D+N and N+T is indiscernible in Fig. 7, partly due to existence of normalized CPTs. To evaluate relative gain from alternative local models, the 3rd experiment generated BNs in two steps: In the 1st step, only DAGs are generated with 200 variables each (binary or ternary). The largest number of parents per node is 12, and each DAG has at least 4 such families. We use $d = 5\%, 10\%$ and $k = 0, 2, 4, 7, 10$, a total of 10 combinations. For each combination, we simulated 30 DAGs, resulting in 300 distinct DAGs. In the 2nd step, a pair of BNs are created from each DAG: NM-BN where each family of 2 parents or more is NAT-modelled, and CM-BN where such families are modelled as CPT-trees. Hence, the pair share the DAG, and differ in JPDs.

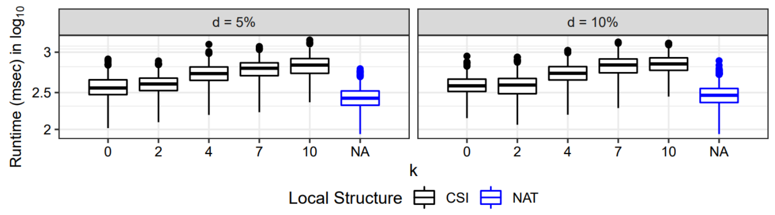

Fig. 8. Summary of inference runtimes by NM-BNs and CM-BNs.

We de-causalize NM-BNs and transform CM-BNs for lazy propagation. Ten inference runs are performed on each BN with random observations over 20 variables. The runtimes are shown in Fig. 8.

Runtimes of NM-BNs are the least, even relative to the most efficient CM-BNs ($k = 0$). Let m be domain size of child variable of CPT-tree family. Assuming single-valued CPT-tree arcs, each multiplexer has $m + 1$ parents. If CPT-tree has $k > 0$, transformation has loops. On the other hand, every node in de-causalized BN segment has at most 2 parents, and the segment is always loop-free. Hence, NAT modelled BNs are generally more efficient than CPT-tree modelled BNs, as confirmed by the experiment.

8 Conclusion and Remarks

The main contribution is a framework to mix ICI and CSI local models in BNs for more efficient inference. They are shown to be orthogonal, and hence neither subsumes the other. We have shown that NAT models and CPT-trees are suitable ways for mixing, and combining de-causalization and transformation enables a homogenous representation for exact inference. We report significant speedup in inference relative to exploitation of only one type of local models.

Two questions were received from peer review, to which we respond below. Due to space, we omit relevant references. (1) On why not adopt sum-product networks (SPNs): Although exact inference in BNs is NP-hard and that in SPNs is linear, when a general BN is compiled into a SPN, it incurs an exponential blow-up. Hence, SPNs are one way to explore special conditions in BNs, e.g., CSI, but not the only way, as this work shows. (2) On why not adopt simple propagation (SP): Published work on SP reported that SP is not always faster than LP. Other BN inference methods also exist. This work shows significant gains in inference efficiency by mixing ICI and CSI, and our performance comparison only necessitates identical inference method on BNs with and without mixing. There is no need for the fastest method, nor do we claim that LP is so.

Acknowledgement

Financial support from NSERC Discovery Grant to 2nd 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. Chavira, M., Darwiche, A.: Compiling Bayesian networks using variable elimination. In: Proc. 20th IJCAI. pp. 2443–2449 (2007)
3. Friedman, N., Yakhini, Z.: On the sample complexity of learning Bayesian networks. In: Proc. 12th Conf. on UAI. pp. 274–282. Morgan Kaufmann (1996)
4. Henrion, M.: Some practical issues in constructing belief networks. In: Kanal, L., Levitt, T., Lemmer, J. (eds.) Uncertainty in Artificial Intelligence 3, pp. 161–173. Elsevier Science Publishers (1989)
5. Maaskant, P., Druzdzel, M.: An independence of causal interactions model for opposing influences. In: Jaeger, M., Nielsen, T. (eds.) Proc. 4th European Workshop on Probabilistic Graphical Models. pp. 185–192. Hirtshals, Denmark (2008)
6. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
7. Poole, D.: Probabilistic partial evaluation: exploiting rule structure in probabilistic inference. In: Proc. 15th IJCAI. pp. 1284–1291 (1997)
8. Poole, D., Mackworth, A., Goebel, R.: Computational Intelligence: A Logical Approach. Oxford University Press (1998)
9. Talvitie, T., Eggeling, R., Koivisto, M.: Learning Bayesian networks with local structure, mixed variables, and exact algorithms. International Journal of Approximate Reasoning **115**, 69–95 (2019)
10. Woudenberg, S., van der Gaag, L., Rademaker, C.: An intercausal cancellation model for Bayesian-network engineering. Inter. J. Approximate Reasoning **63**, 32–47 (2015)
11. Xiang, Y.: Non-impeding noisy-AND tree causal models over multi-valued variables. International J. Approximate Reasoning **53**(7), 988–1002 (Oct 2012)
12. Xiang, Y., Baird, B.: Compressing Bayesian networks: Swarm-based descent, efficiency, and posterior accuracy. In: Bagheri, E., Cheung, J. (eds.) Canadian AI 2018, LNAI 10832, pp. 3–16. Springer (2018)
13. Xiang, Y., Loker, D.: De-causalizing NAT-modeled Bayesian networks for inference efficiency. In: Bagheri, E., Cheung, J. (eds.) Canadian AI 2018, LNAI 10832, pp. 17–30. Springer (2018)