Theoretical Foundations for Parent Divorcing Transformations in Bayesian Networks

Daniel J. Rosenkrantz[1] Madhav V. Marathe[2] Zirou Qiu[2] S. S. Ravi[1]

Abstract

Parent divorcing is a commonly used technique to reduce the complexity of Bayesian models. In particular, this transformation decreases the number of parents of some nodes in a given Bayesian Network (BN). Such transformations must be done in such a way that solutions to inference problems for the original BN can be readily obtained from the solutions to the new BN. Despite its wide use in practice, there have been no attempts to formally analyze these transformations. In this work, we present a first step towards the development of theoretical foundations for the use of divorce transformations in BNs and establish analytical results. Specifically, we develop a formalism that captures a structural relationship between the original BN and the new BN. Using this formalism, we present an algorithm for parent divorcing which ensures that the treewidth of the modified BN is within a constant factor of that of the original BN. We also present an algorithm that carries out parent divorcing while preserving the domain size of each node. Further, we present lower bound results to prove that there are BNs for which parent divorcing transformations give an exponential increase in the domain size or lead to an exponentially large new BN.

1 Introduction and Motivation

A **Bayesian Network** (BN) is a commonly used graphical model to represent dependencies among a collection of stochastic variables [13, 22, 32, 34, 41]. Formally, a BN is a *directed acyclic graph* (dag) $G(V, E)$ in which each node $v \in V$ represents a stochastic variable x_v; a directed edge (u, v) in E indicates that variable x_v depends on x_u. In addition, each node v is associated with a **conditional probability table** (CPT) which gives the probability distribution of x_v conditioned on the variables on which x_v depends. Precise definitions concerning BNs are given in Section 2. Motivated by applications in medical diagnosis, weather forecasting and risk management, a number of different inference problems for BNs have been considered in the literature [13, 22, 29, 32, 34, 41].

We refer to the finite set of values assumed by a stochastic variable v as the **domain** of v. For example, if each node of a BN $G(V, E)$ represents a stochastic Boolean variable, then the domain of each node is $\{0, 1\}$. In this case, the CPT for a node v with d parents has 2^d rows. When d is large, the corresponding CPT is very large. To overcome this problem, an approach called **parent divorcing** is used in practice to deal with nodes with a large number of parents. This approach, introduced in [33], modifies a given BN in the

[1]Biocomplexity Institute, University of Virginia, Charlottesville, VA 22904 and Computer Science Department, University at Albany – SUNY, Albany, NY 12222, USA. Email: {drosenkrantz, ssravi0}@gmail.com

[2]Biocomplexity Institute and the Department of Computer Science, University of Virginia, Charlottesville, VA 22904, USA. Email: {marathe,zq5au}@virginia.edu

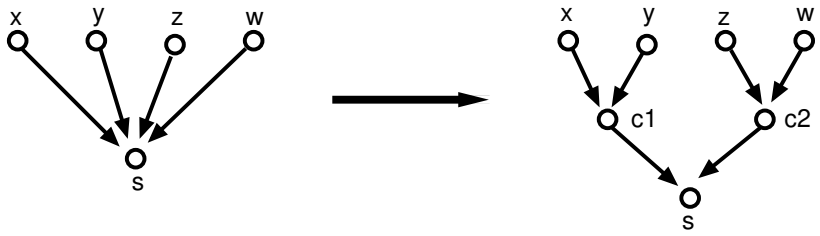


Figure 1: An Example for Parent Divorcing Approach

following manner: when a node s has a large number of parents (i.e., the **indegree** of s is large), the subgraph consisting of the node and its parents is replaced by a directed tree in which each node has a small indegree. An example for this approach, where the subgraph consisting of node s and its four parents (nodes x, y, z and w) is replaced by another subgraph in which node s and the two new nodes $c1$ and $c2$ have only two parents, appears in Figure 1.

References that present examples of parent divorcing include [5, 16, 20, 33, 35, 36, 44]. Since these references provide only informal descriptions of the divorce transformations, it is hard to reason formally about the relationships between the original BN and the modified BN. In particular, such informal descriptions cannot be used to prove rigorously that a solution to an inference problem for a given BN can be obtained from the solution to the corresponding problem on the modified BN. Further, divorcing transformations can significantly affect structural properties of a BN. One example of such a structural property is the treewidth of a BN. It is well known [13, 22, 26, 41] that many inference problems can be solved in time that is exponential in the treewidth of the underlying BN. Thus, if divorcing operations increase the treewidth significantly, solving inference problems may become impractical.

Our contributions. To the best of our knowledge, foundational aspects of parent divorcing transformations have not been addressed in the literature. (Henceforth, we will use "divorcing transformations" to mean "parent divorcing transformations".) Such foundations are crucial in developing divorcing transformations that *provably* satisfy two properties: (i) solutions to inference problems on a given BN can be obtained efficiently from the solutions to inference problems on the modified BN and (ii) the transformations preserve appropriate structural properties of the given BN. We present a formalism and several basic divorcing transformations which can be combined appropriately to develop algorithms that provably satisfy the two properties mentioned above. Our approach also enables the development of efficient algorithms for inference problems for special classes of BNs. Our contributions are summarized below.

1. Formalizing Divorce Transformations: We develop (Section 2.4) two formalisms (called **embedding** and **functional embedding**) that allow us to rigorously capture relationships between the original BN and the new BN resulting from the divorce transformations. In particular, these formalisms ensure that inference problems for the original BN are reducible to the inference problems for the new BN (Theorem 2.4).

2. A parent divorcing algorithm that preserves treewidth to within a constant factor: We present (Section 3) basic divorcing transformations and use them in developing an algorithm that provably transforms a given BN G_1 of treewidth t into a new BN G_2 such that all of the following properties hold: (a) G_1 is functionally embedded in G_2, (b) each node of G_2 has at most two parents, and (c) the treewidth of G_2 is at

most $3t + 3$.

3. A parent divorcing algorithm that preserves domain size: We also present (Section 4) an algorithm that transforms a given BN G_1 into a new BN G_2 such that all of the following properties hold: (a) G_1 is embedded in G_2, (b) each node of G_2 has at most two parents, and (c) the maximum domain size in G_2 is equal to that in G_1.

4. An efficient inference algorithm for a special class of BNs: Our techniques lead to an efficient algorithm (Section 5) for inference problems on (unmoralized[3]) treewidth-bounded BNs whose CPTs are r-symmetric[4] for some fixed integer $r \geq 1$. The efficient solvability of inference problems for this class of BNs was established in [37]. Our methods provide an alternative proof of that result.

5. Lower bounds: The treewidth preserving algorithm mentioned in Item 2 above may lead to exponentially large domain sizes for some new nodes. Also, the algorithm mentioned in Item 3 above may add an exponential number of new nodes. We present lower bound results (Section 6) to prove that such increases cannot be avoided in general when the original BN and the new BN are required to have the same set of source nodes (i.e., nodes with indegree zero). This natural restriction on source nodes is satisfied in the parent divorcing examples presented in many references (see e.g., [5, 33, 35, 36]).

2 Preliminaries

2.1 Bayesian Networks and Inference Problems

A Bayesian network (BN) is a directed acyclic graph $G(V, E)$, where nodes represent stochastic domain variables and directed edges represent dependencies between variables. When there is a directed edge $(u, v) \in E$, we say that u is a **parent** of v. The **indegree** of a node v is the number of parents of v.

At each node v, there is a **conditional probability table** (CPT) T_v which specifies the probability values for the variable v, conditioned on the parents of v. Assuming the domain to be $\{0, 1\}$, for a node v with indegree t, there are 2^t different combinations of Boolean values for the parents of v. For each such combination, the table specifies the probability of v being 1 (or 0) conditioned on the parents assuming the given combination of values. Thus, the number of entries in T_v is 2^t. For a node v with indegree 0, the table T_v specifies simply the probability of v assuming the value 1 (or 0).

Our formulation of the computational problems for BNs follows the presentation in [7]. For a node (also called **variable**) v, let $\mathcal{P}(v)$ denote the set of parents of v. Given a BN $G(V, E)$, a **configuration** c_V is an assignment of values to each variable in V. Given a subset $O \subseteq V$, a **partial configuration** (also called an **observation**) c_O on O specifies a value for each variable in O. Given a configuration c_V (or an observation c_O), we use $c_V(v)$ ($c_O(v)$) to denote the value of variable v in that configuration (observation). We also extend this notation to subsets of variables. Thus, given a configuration c_V (or an observation c_O), and a subset $W \subseteq V$ ($W \subseteq O$), $c_V(W)$ ($c_O(W)$) denotes the combination of values assigned to the variables in W.

[3] The definition of moralized BNs appears in Section 2.2.

[4] The definition of r-symmetric CPTs appears in Section 5.

Given a configuration c_V, its probability $\Pr\{c_V\}$ is given by

$$\Pr\{c_V\} = \prod_{v \in V} \Pr\{c_V(v) \mid c_V(\mathcal{P}(v))\}.$$

A configuration c_V is an **extension** of an observation c_O if for each variable v that is assigned a value in c_O, $c_V(v) = c_O(v)$. Thus, an extension of an observation c_O is obtained by specifying values for all the variables that are not assigned a value in c_O.

We now define some inference problems defined in the literature in the context of BNs. In all of these problems, we are given an observation c_O on a set of nodes $O \subseteq V$. A basic problem in this context, namely **Probability Computation Problem** (denoted by PROB), is the following: given an observation c_O, find the probability of c_O, that is, the sum of the probabilities of all configurations that are extensions of c_O. In the **Inference Problem** (denoted by INF), we are also given a variable v; the goal is to compute $\Pr\{v = \alpha \mid c_O\}$, that is, the probability that v assumes the value α (where α is a value in the domain of v) conditioned on the observation c_O. As is well known, the INF problem can be solved using an algorithm for the PROB problem as a subroutine [13, 22]. In the **Most Probable Explanation Problem** (denoted by MPE), the goal is to find an extension of c_O which has the maximum probability among all the extensions of c_O. Other inference problems are discussed in [13, 14, 26].

2.2 Tree Decompositions

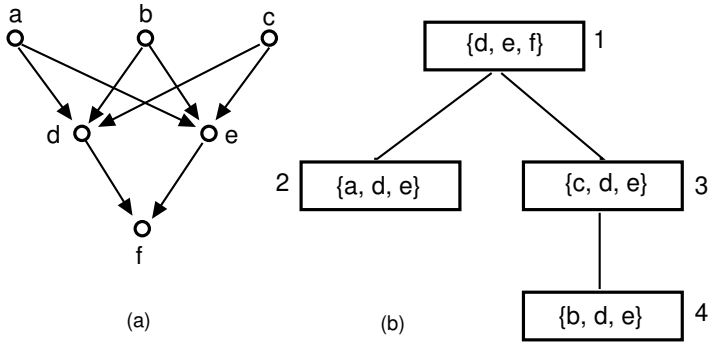
We now recall the standard definition of *tree decomposition* and *treewidth* from [8].

Definition 2.1. *(a) Given a BN $G(V, E)$, a* **tree decomposition** *of G is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a family of subsets of V and $T = (I, F)$ is an undirected tree with the following properties: (i) $\bigcup_{i \in I} X_i = V$. (ii) For every directed edge $e = (v, w) \in E$, there is a subset X_i, $i \in I$, with $v \in X_i$ and $w \in X_i$. (iii) For all $i, j, k \in I$, if j lies on the path from i to k in T, then $X_i \bigcap X_k \subseteq X_j$.*

(b) The **treewidth** *of a tree decomposition $(\{X_i \mid i \in I\}, T)$ is $\max_{i \in I}\{|X_i| - 1\}$. The treewidth of a graph is the minimum over the treewidths of all its tree decompositions.*

Many algorithms for inference problems on BNs assume that the corresponding graph is *moralized*, that is, the parents of any node are connected together as a clique (e.g., [13, 22, 27]). In particular, the treewidth of a BN is assumed to be that of its moralized version. As observed in [37], moralization significantly increases the treewidth when there are nodes with high indegree. Therefore, throughout this paper, we assume that the BNs are not moralized.

We also assume that the given tree decomposition T of a BN is a binary tree; that is, each node of T has at most two children [6]. The following additional terminology regarding nodes in tree decompositions is from [4]. Let T be the given tree decomposition of a BN G. For a given node i of T, the nodes of G in X_i are called **explicit nodes** of i. If a given explicit node v of i is also an explicit node of the parent of i, then v is referred to as an **inherited node** of i; and if v does not occur in the parent of i, then v is called an **originating node** of i. We refer to the set of all explicit nodes occurring in the subtree of T rooted at i that



Tree Node (i)	Explicit Nodes (X_i)	Inherited Nodes (Y_i)	Originating Nodes (Z_i)	Hidden Nodes (W_i)
1	$\{d, e, f\}$	$\varnothing$	$\{d, e, f\}$	$\{a, b, c\}$
2	$\{a, d, e\}$	$\{d, e\}$	$\{a\}$	$\varnothing$
3	$\{c, d, e\}$	$\{d, e\}$	$\{c\}$	$\{b\}$
4	$\{b, d, e\}$	$\{d, e\}$	$\{b\}$	$\varnothing$

Figure 2: An Example to Illustrate Tree Decomposition and Related Terminology

are not explicit nodes of i as **hidden nodes** of i. An example of a BN G, a tree decomposition T for G and the various sets of nodes associated with each node of T are shown in Figure 2.

2.3 Related Work

In general, inference problems for BNs are known to be computationally intractable (see e.g., [1,2,11,12,39]). So, researchers have presented efficient heuristics for obtaining fast solutions in practice (e.g., [10, 15, 30]). Researchers have also identified restricted versions of inference problems that can be solved efficiently. For example, an efficient algorithm is known for inference problems for treewidth bounded (moralized) BNs [27]. As shown in [37], inference problems for treewidth-bounded unmoralized BNs with no restriction on indegrees can also be solved efficiently provided the CPTS are restricted to r-symmetric functions [4] for some fixed r. Other references that consider inference problems for BNs include [3,9,15,19,26,28,44]. The necessity of bounded treewidth to obtain efficient inference algorithms is shown in [25]. A more general form of inference (called lifted inference) has also been investigated [21, 31]. Some recent papers have addressed issues such as structure learning in and evaluation of Bayesian models (e.g., [17, 24]).

The parent divorcing approach in Bayesian networks has been used in a wide variety of applications. For example, Ducamp ([16]) uses the parent divorcing approach in optimizing the compilation probabilistic production rules for graphical models. Basnet et al. ([5]) use the approach to reduce the complexity of a Bayesian network used to perform risk analysis in remotely piloting naval vessels. Kraisangka and Druzdela ([23]) employ a complexity reduction technique based on parent divorcing in the application of regression-based analysis methods. Xiang and Loker ([43]) apply an approach called causalization for reducing the number of model parameters of certain forms of Bayesian networks. Like parent divorcing, their method also

reduces the number of parents of a node through suitable transformations of the underlying dag. Wu and Ruggeri ([42]) consider a partitioning approach for directed acyclic graphs of Bayesian networks to reduce the model complexity in a manner similar to that of parent divorcing.

As mentioned earlier, prior work on parent divorcing has not address the development of formalisms that can be used to reason rigorously about the relationships between the given BN and the new BN. An approach called child-friendly divorcing is presented in [36]. The goal is to partition the set of parents of a node into groups so that the nodes in each group are similar in terms of graph theoretic properties such as degree and connectivity. It is shown through experiments that the approach is beneficial in the context of hierarchy learning in BNs. The use of parent divorcing in learning the structure of a BN is considered in [40]. These papers do not address the issue of preserving structural parameters such as treewidth.

2.4 BN Embeddings

We present a first step towards the development of theoretical foundations for divorce transformations in BNs. This requires a formalization of (i) divorce transformations and (ii) the relationship between a given BN and the BN resulting from the transformations. Moreover, such a formalism should also ensure that inference problems on a given BN can be reduced to the corresponding inference problems on the new BN. We now provide one such formalization through the following notion of *embeddings*.

Definition 2.2. *BN $G_1(V_1, E_1)$ is **embedded** in BN $G_2(V_2, E_2)$ if (a) $V_1 \subseteq V_2$, (b) every node in V_1 has the same domain in G_1 and G_2, and (c) for every configuration c_{V_1} of G_1, there is an extension c_{V_2} of G_2 such that the configuration probability $Pr\{c_{V_1}\}$ in G_1 equals the configuration probability $Pr\{c_{V_2}\}$ in G_2.*

The following lemma points out some properties of embeddings which will be used in proving the main result of this section (Theorem 2.4).

Lemma 2.3. *Suppose BN $G_1(V_1, E_1)$ is embedded in BN $G_2(V_2, E_2)$. Then the following properties hold.*

*1. For any configuration c_{V_1} of G_1, if the configuration probability $Pr\{c_{V_1}\}$ in G_1 is nonzero, then there is a **unique** configuration c_{V_2} of G_2 s.t. c_{V_2} is an extension of c_{V_1} and $Pr\{c_{V_1}\}$ in G_1 equals $Pr\{c_{V_2}\}$ in G_2.*

2. For any configuration c_{V_1} of G_1, if the configuration probability $Pr\{c_{V_1}\}$ in G_1 is zero, then for every extension c_{V_2} of c_{V_1}, $Pr\{c_{V_2}\}$ in G_2 is zero.

Proof: To prove Part 1, let S_1 be the set of all configurations of G_1 such that the probability of each configuration in S_1 is nonzero. Since G_1 is embedded in G_2, by Definition 2.2, for each configuration c_{v_1} in S_1, there is a configuration c_{v_2} of G_2 such that $\Pr\{c_{V_1}\}$ in G_1 equals $\Pr\{c_{V_2}\}$ in G_2. For each configuration c_{v_1} in S_1, choose such a corresponding configuration in G_2, and let S_2 denote the resulting set of configurations of G_2. Since the configurations in S_1 are all distinct, the configurations in S_2 are also distinct; thus, $|S_1| = |S_2|$. Note that the sum of probability values of the configurations in S_1 equals 1; hence, the sum of probability values of the configurations in S_2 is also 1. In other words, for each configuration c_{V_2} of G_2 which is not in S_2, $\Pr\{c_{V_2}\} = 0$. Therefore, for any configuration c_{V_1} of G_1, the *only* extension with nonzero probability in G_2 appears in S_2. This completes the proof of Part 1.

To prove Part 2, consider the sets S_1 and S_2 defined in the proof of Part 1. The proof of Part 1 showed that each configuration c_{V_2} with nonzero probability in G_2 appears in S_2. Consider any configuration c_{V_1} with zero probability in G_1. By the definition of S_1, c_{v_1} does not appear in S_1. Further, since each configuration in S_2 is the extension of some configuration in S_1, it follows that no extension of c_{V_1} appears in S_2. In other words, for each extension of c_{V_1}, the probability in G_2 is zero, and this completes the proof of Part 2. ■

The following theorem shows that the notion of embedding formally reduces inference problems on a BN G_1 to those on another BN G_2 as long as G_1 is embedded in G_2.

Theorem 2.4. *Suppose BN G_1 is embedded in G_2. Then, solutions to any of the inference problems* PROB, INF *and* MPE *on G_1 can be obtained from the solutions to the corresponding problem on G_2.*

Proof: We will present the proof for the PROB problem. The proofs for the other two problems are similar.

Let c_{O_1} be an observation on BN G_1. To complete the proof for the PROB problem, we show that $\Pr\{c_{O_1}\}$ in G_1 equals $\Pr\{c_{O_1}\}$ in G_2. We have two cases.

Case 1: $\Pr\{c_{O_1}\}$ in G_1 is nonzero.

Let S_1 be the set of all configurations of G_1 such that each configuration c_{V_1} in S_1 is an extension of c_{O_1} and $\Pr\{c_{V_1}\}$ in G_1 is nonzero. Let $p_1 = \Pr\{c_{O_1}\}$ in G_1. By definition,

$$p_1 = \sum_{c_{V_1} \in S_1} \Pr\{c_{V_1}\}.$$

By Part 1 of Lemma 2.3, for each configuration c_{v_1} of G_1 with $\Pr\{C_{V_1}\} \neq 0$, there is a *unique* configuration c_{v_2} of G_2 such that c_{v_2} is an extension c_{v_1} and $\Pr\{c_{V_1}\}$ in G_1 equals $\Pr\{c_{V_2}\}$ in G_2. For every other extension of c_{v_1} into a configuration of G_2, the corresponding probability in G_2 is zero. Since $\Pr\{C_{O_1}\}$ in G_2 is the sum of the probabilities of all configurations of G_2 which are extensions of c_{O_1}, it follows that $\Pr\{c_{O_1}\}$ in G_2 is equal to $p_1 = \Pr\{C_{V_1}\}$ in G_1.

Case 2: $\Pr\{c_{O_1}\}$ in G_1 is zero.

Thus, the probability of every configuration c_{V_1} of G_1 which is an extension of c_{O_1} is also zero. By Part 2 of Lemma 2.3, the probability in G_2 of every extension of c_{v_1} is also zero. Since $\Pr\{C_{O_1}\}$ in G_2 is the sum of the probabilities of all configurations of G_2 that extends c_{O_1}, it follows that $\Pr\{c_{O_1}\}$ in G_2 is zero. Hence, the probability of the observation c_{O_1} in G_1 is equal to the probability of the configuration c_{O_1} in G_1. ■

We now present a special form of the embedded-in relationship which facilitates the construction of the CPTs for the new BN.

Definition 2.5. *A node v of a BN is **functional** if the values of its parents completely determine its value.*

Thus, when a node v is functional, for each entry of the CPT (which corresponds to one combination of values for the parents of v), there is a unique value α in the domain of v such that $\Pr\{v = \alpha\} = 1$; for all other values in v's domain, the corresponding probability is zero.

Definition 2.6. *BN $G_1(V_1, E_1)$ is **functionally embedded** in BN $G_2(V_2, E_2)$ if (a) G_1 is embedded in G_2 and (b) every node in $V_2 - V_1$ is functional.*

Another property of embeddings is stated below.

Observation 2.7. *The* embedded-in *and* functionally embedded-in *relations on BNs are transitive.* ■

This observation enables us to produce an embedding of a BN through a chain of embedding steps.

3 Treewidth Preserving Divorcing

In this section, we give an algorithm for transforming a given BN G into a BN G' such that G is functionally embedded in G', each node of G' has at most two parents, and the treewidth of G' is at most a constant times the treewidth of G. The sizes of the domains of the added nodes in G' may be larger than those of the domains of the nodes in G. (As shown by the lower bound results of Section 6, such an increase cannot be avoided in general.) Starting with the graph G of a BN and a tree decomposition T for G, the algorithm constructs in polynomial time the graph G' for the new BN and a tree decomposition T' for G'. This section uses the terminology regarding tree decompositions introduced in Section 2.2.

Let T denote a decomposition tree of a BN G. We select one of the nodes of T as a root node, and view T as a rooted tree. As mentioned earlier, we assume that T is a binary tree; that is, each node of T has at most two children.

We say that a node v of G is **safe** if it has at most two parents in G, and is **unsafe** if it has more than two parents. The **divorce degree** of an unsafe node v is two less than the number of parents of v. The **divorce degree** of G is the sum of the divorce degrees of all the unsafe nodes in G.

For node i of T, and node u in X_i (the set of explicit nodes of i), we say that node u is **purgeable** in i if u is an originating node of i, u is safe, and every node of G for which u is a parent is safe.

For node i of T, and nodes u and v in X_i, we say that node u is a **divorceable parent** of v in i if v is unsafe, u is a parent of v, and at least one other parent of v is in X_i. For node i of T, and nodes u, w and v in X_i, we say that (u, w, v) is a **divorceable triple** in i if u and w are distinct divorceable parents of v in i.

For node i of T, and node u in X_i, we say that node u is a **prepurgeable** parent in i if u is an originating node of i, u is safe, u is a parent of exactly one unsafe node, and u is a divorceable parent of this unsafe node in i. For node i of T, and nodes u and v in X_i, we say that node u is a **dangling** parent of v in i if u is an originating node of i, u is safe, v is unsafe, u is a parent of v, and u is the only parent node of v in X_i.

Let $G(V, E)$ be a BN, with rooted decomposition tree T. Let i be a node of T. Suppose that u and w are distinct divorceable parents of v in node i. The **divorce transformation** $\tau^i_{u,w,v}$ of (G, T) produces a modified BN G' and rooted tree T', as follows. We add a new node v' to V. We modify E by replacing the two directed edges (u, v) and (w, v) with the three directed edges (u, v'), (w, v'), and (v', v). In general, the domain of the stochastic variable $x_{v'}$ for new node v' is made the cross product of the domains of the stochastic variables x_u and x_w. (In practice, the semantics of the CPT for v in G may permit a smaller domain for $x_{v'}$.)

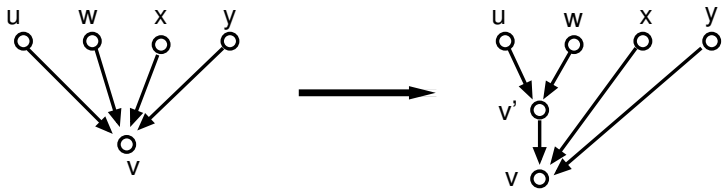


Figure 3: An Example for Divorce Transformation

The conditional probability table for $x_{v'}$ is purely functional, with the values of x_u and x_w determining the value of $x_{v'}$. The conditional probability table for x_v is appropriately modified to use the value of $x_{v'}$ instead of the values of x_u and x_w. The only modification of tree T is that v' is added to X_i. An example of the divorce transformation is shown in Figure 3.

We note several properties of the divorce transformation: (1) BN G is functionally embedded in BN G'. (2) The change to G replaces two parents of v with the new parent v', and so reduces the divorce degree of the BN by one. (3) The new node v' is safe and is an originating node of i in T', so v' is either purgeable in i, a dangling parent of v in i, or a prepurgeable parent in i.

Suppose i contains at least one purgeable node. The **purge transformation** ρ^i of (G, T) leaves BN G unchanged, but produces a modified rooted tree T', as follows. A new node i' is added to T. If node i was the root of T, then i' becomes the root of T', and i' is made the parent of i. If i had a parent, say j, in T, then in T', i' is inserted between i and j. Node set $X_{i'}$ consists of all the non-purgeable nodes in X_i.

Suppose that i contains at least one dangling parent in X_i, and i is not the root node of T. The **promotion transformation** μ^i of (G, T) leaves BN G unchanged, but produces a modified rooted tree T', as follows. Let j be the parent node of i in T. Then all the dangling parents in X_i are added to X_j. Note that after this transformation, the nodes that have been added to X_j are no longer originating nodes of i, so node i no longer contains any dangling parents.

Note: Of the three transformations mentioned above, only the divorce transformation changes the given BN; the other two transformations change only the tree decomposition.

Our next result shows how these transformations can be used to construct a new BN in which a given BN is functionally embedded while the treewidth is increased only by a small constant factor.

Theorem 3.1. *There is a polynomial time algorithm that given a BN G_1 and a tree decomposition T_1 of treewidth t for G_1, constructs a BN G_2 and a tree decomposition of treewidth at most $3t + 3$ for G_2, such that G_1 is functionally embedded in G_2, each node of G_2 has at most two parents, and the number of added nodes in G_2 is less than than number of edges in G_1.*

Proof: The algorithm proceeds as follows. At each step of the algorithm, there is a current BN G with tree T, and these objects are possibly transformed into a modified BN G' and tree T', which become the current G and T for the next step. Initially, $G = G_1$ and $T = T_1$. The algorithm halts when all nodes of the current BN are safe. When the algorithm halts, the current G and T become the algorithm's output G_2 and T_2.

The algorithm processes the nodes of the tree T in bottom-up order. Let i be an unprocessed node of

the current tree, such that the children (if any) of i have been processed. The processing of i depends on properties of the nodes in X_i (the set of explicit nodes of i) as follows.

1. Suppose that X_i contains at least one purgeable node. In this case, the algorithm applies the purge transformation ρ^i. This transformation leaves G unchanged, but produces a modified tree T', which becomes the current tree T for the next step of the algorithm. The processing of node i is considered completed, but the new tree node created by the purge transformation is considered unprocessed.

2. Suppose that X_i contains no purgeable nodes, but contains a prepurgeable parent. In this case, the algorithm selects a divorceable triple, say (u, w, v), that contains a prepurgeable parent in i. The divorce transformation $\tau^i_{u,w,v}$ is applied, producing a G' and T', which become the new G and T. Note that the prepurgeable parent in the triple becomes a purgeable node in i after the transformation. The algorithm then processes node i again. (Note that Case 1 will be applicable to this reprocessing of i.)

3. Suppose that X_i contains no purgeable nodes, and no prepurgeable parents, but contains a divorceable triple. In this case, the algorithm selects a divorceable triple in i, say (u, w, v). The divorce transformation $\tau^i_{u,w,v}$ is applied, producing a G' and T', which become the new G and T. The algorithm processes node i again.

4. Suppose that X_i contains no purgeable nodes and no divorceable triples, but contains a dangling parent. In this case, the promotion transformation μ^i is applied. This transformation leaves G unchanged, but produces a modified tree T', which becomes the current tree T for the next step of the algorithm. The processing of node i is considered completed.

5. Suppose that X_i contains no purgeable nodes, no divorceable parents, and no dangling parents. In this case, no transformation is applied, and the processing of node i is considered completed.

Theorem 3.1 now follows from the following claims.

Claim 3.1: The algorithm halts and produces the graph G_2 of the new BN and a tree decomposition of G_2 in polynomial time.

Proof of Claim 3.1: Each application of a divorce transformation reduces the divorce degree of the BN by one. So, the number of divorce transformations applied by the algorithm is at most the divorce degree of G_1. The only type of transformation that increases the number of nodes in the decomposition tree is the purge transformation. If a purge transformation is applied to a given tree node, say i, it creates an additional tree node, i'. The purge transformation completes the processing of node i. Moreover, at the time the processing of node i' begins, node i' contains no purgeable nodes. If the processing of i' does not entail a divorce transformation, then the processing of i will not entail a purge transformation. Thus, the number of nodes in the current tree at any time during the algorithm is at most the sum of the divorce degree of G_1 and twice the number of nodes in G_1. Each tree node has at most one purge or promotion transformation applied to it. So, the number of steps taken by the algorithm is linear in the sum of the number of nodes in G_1 and the divorce degree of G_1. Thus the algorithm halts, and produces G_2 and its tree decomposition in polynomial time.

We now argue that all nodes of G_2 are safe. Consider a given node i of the current tree during the algorithm, after the processing of node i has been completed. It can be seen by induction on the number of steps of the algorithm that have been executed so far, that all originating and hidden nodes of i are safe. Therefore, if all the nodes of the tree are processed by the algorithm, all nodes in the final BN are safe. If the algorithm halts before processing all the nodes in the tree, it has halted because all nodes are safe. In either case, all nodes of the final BN G_2 are safe.

The only type of transformation that modifies the BN is a divorce transformation. Each application of a divorce transformation changes the current BN G to a new BN G', such that G is functionally embedded in G'. Since the *functionally embedded-in* relationship is transitive, G_1 is functionally embedded in G_2.

We now consider the treewidth of the BN G_2 produced by the algorithm. Because G_1 has treewidth t, each node of T_1 contains at most $t + 1$ nodes of G. Each of these nodes is possibly unsafe.

Consider a given node i of a tree. The promotion transformation applied to a child of i can add at most $t + 1$ nodes to X_i. Node i has at most two children. Thus, when processing of i begins, X_i contains at most $3(t + 1)$ nodes. A divorce transformation applied to i adds exactly one node to X_i. □

Claim 3.2: The algorithm produces a BN with treewidth at most $3t + 3$.

Proof of Claim 3.2: We prove the claim by establishing the following Subclaim by induction.

Subclaim: At each step of the algorithm presented in the proof of Theorem 3.1, the current BN and tree decomposition satisfy the following five properties.

1. For every tree node k, X_k contains at most $3t + 3$ nodes.
2. For every tree node k, X_k contains at most $t + 1$ unsafe nodes.
3. For every tree node k, X_k contains at most $t + 1$ nodes that are dangling parents in k.
4. For every tree node k for which processing has not yet been completed, X_k contains at most $2(t + 1)$ nodes that are not purgeable nodes in k, prepurgeable parents in k, or dangling parents in k.
5. For every tree node k for which processing has not yet been completed, X_k contains at most $3(t + 1)$ nodes that are not purgeable nodes in k.

Proof of Subclaim: We use induction on the number of steps taken by the algorithm. For the basis case, note that each node of the initial tree T_1 contains at most $t + 1$ nodes, so T_1 satisfies all five properties.

Now, consider Property (2). None of the three types of transformation used by the algorithm creates a new unsafe node, changes a node from safe to unsafe, adds an unsafe node to a tree node, or creates a new tree node with more unsafe nodes than the maximum number of unsafe nodes in any current tree node. Therefore, Property (2) is preserved at every step of the algorithm.

Now, consider Property (3). Each dangling parent in a given tree node k is the parent of some unsafe node v in X_k, and is the only parent of v in X_k. From Property (2), there are at most $t + 1$ unsafe nodes in

X_k. Therefore, there are at most $t+1$ dangling parents in X_k, so Property (3) is preserved at every step of the algorithm.

Now, consider Property (4). Since each node of the initial tree T_1 contains at most $t+1$ nodes, the initial tree satisfies Property (4).

Suppose the algorithm applies a purge transformation ρ^i and that Property (4) holds before the transformation. After the transformation, node i is considered processed, and new tree node i' has been created. Set $X_{i'}$ consists of all the nonpurgeable nodes in X_i, and every node in X_i that was classified as a purgeable parent or dangling parent in i is also classified as a purgeable parent or dangling parent in i'. So, Property (4) holds after the transformation.

Suppose the algorithm applies a divorce transformation $\tau^i_{u,w,v}$ and that Property (4) holds before the transformation. The transformation adds a new node v' to X_i. This new node v' is either purgeable in i, a dangling parent in i, or a prepurgeable parent in i. For each X_k the transformation might change the classification of some nodes in X_k. In particular, a given prepurgeable or dangling node in X_k might possibly now be classified as a purgeable node in X_k. Any given purgeable node in X_k will not have its classification changed. So, Property (4) holds after the transformation.

Suppose the algorithm applies a promotion transformation μ^i and that Property (4) holds before the transformation. Let j be the parent node of i. Note that after the transformation, the processing of node i is considered completed, so node i does not have to satisfy the condition in Property (4). The transformation adds the nodes that were dangling parents in i to X_j. Each of these added nodes is the parent of at least one unsafe node in X_j. Suppose that of these nodes that are added to X_j, α nodes are the parent of more than one unsafe node in X_j, and β are the parent of exactly one unsafe node in X_j. Note that after the transformation, each of the β nodes is either a prepurgeable parent in j or a dangling parent in j. Since t contains at most $t+1$ unsafe nodes, we have that $2\alpha+\beta \le t+1$. Consequently, $\alpha \le (t+1)/2$. So, the transformation adds at most $\alpha \le (t+1)/2$ nodes to X_j that are neither prepurgeable parents nor dangling parents in j. Note that because the algorithm processes tree nodes in a bottom-up order, node j was also a node in G_1, and the processing of node j has not yet begun. Set X_j contained at most $t+1$ nodes in G_1. Since j has at most two children, at most two promotion transformations can add nodes to X_j, so the number of nodes added to X_j by promotion transformations that are neither prepurgeable parents nor dangling parents in j is at most $t+1$. So, Property (4) holds after any promotion transformation.

Therefore, Property (4) is preserved at every step of the algorithm.

Now, consider Property (5). The only way in which a given unprocessed node i can contain more than $3(t+1)$ nodes in X_i is for X_i to contain $3(t+1)$ nodes at some step in the algorithm, and then have a divorce transformation $\tau^i_{u,w,v}$ add an additional node to X_i. Suppose that Property (5) holds before the transformation carried out in the current step. Suppose that at the current step in the algorithm, node i is being processed, and X_i contains $3(t+1)$ nodes. From Property (4), at most $2(t+1)$ nodes in X_i are not purgeable nodes in i, prepurgeable parents in i, or dangling parents in i. So, X_i contains at least $t+1$ nodes that are purgeable nodes in i, prepurgeable parents in i, or dangling parents in i. If X_i contains a purgeable node, then a purge transformation would be applied. If X_i contains no purgeable nodes, but contains a prepurgeable parent, then

a divorce transformation would be applied that changes at least one prepurgeable parent in i into a purgeable node in i. If X_i contains no purgeable nodes or prepurgeable parents, then it must contain $t+1$ dangling parents. Since X_i contains at most $t+1$ unsafe nodes, X_i contains no divorceable parents, a promotion parent is applied. So, Property (5) holds after the transformation, and is preserved at every step of the algorithm.

Now, consider Property (1). Suppose that Property (1) holds before the transformation. The only way in which a given unprocessed node i can contain more than $3t+4$ nodes in X_i is for X_i to contain $3t+4$ nodes at some step in the algorithm, and then have a divorce transformation $\tau^i_{u,w,v}$ add an additional node to X_i. Suppose that at the current step of the algorithm, tree node i is being processed, and X_i contains $3t+4$ nodes. From Property (5), X_i contains at least one purgeable node, so at the current step of the algorithm, the purge transformation ρ^i will be applied. This transformation will create a new tree node i' with at most $3(t+1)$ nodes in $X_{i'}$. Therefore, Property (1) is preserved at every step of the algorithm. This completes the proof of the Subclaim and also that of Claim 3.2. □

As stated earlier, Theorem 3.1 follows from Claims 3.1 and 3.2. ■

4 Divorcing While Preserving Domain Size

In this section, we consider the problem of transforming a given BN G into a new BN G' such that G is embedded in G', each node of G' has at most two parents, and each new node in G' has the same domain as some node of G. The algorithm applies to arbitrary BNs, but has the drawback that the number of added nodes may be exponential in the maximum number of parents of the nodes of G. (As will be shown in Section 6, this increase cannot be avoided in general.) The algorithm makes no attempt to preserve treewidth.

We first note that any BN edge incident on a node whose domain has cardinality 1 can be deleted. Also, for each BN node domain of cardinality at least two, we designate some element of the domain as 0, and some other element of the domain as 1. For every unsafe node v of the given BN, the algorithm creates a set of added nodes. We call some of the added nodes **configuration-gathering nodes**, and some as **evaluation nodes**. Each of these added nodes has the same domain as v. The configuration-gathering nodes are functional nodes, and their conditional probability tables only assign nonzero probabilities to the domain values 0 and 1.

Theorem 4.1. *There is an algorithm that given a BN G_1, constructs a BN G_2 such that G_1 is embedded in G_2, each node of G_2 has at most two parents, and each node in G_2 that is not in G_1 has the same domain as some node of G_1.*

Proof: Consider a given unsafe node v of G_1. Suppose that the parents of v in G_1 are $u_1, u_2 \ldots u_k$, where $k \geq 3$. Let P^j, $1 \leq j \leq k$ denote the set of nodes $\{u_1, u_2 \ldots u_j\}$. Let $\mathcal{D}_i$, $1 \leq i \leq k$ denote the domain of u_i. Let $\mathcal{D}_v$ denote the domain of v.

Let $\mathcal{D}^i$, $1 \leq i \leq k$ denote the cross product $\mathcal{D}_1 \times \mathcal{D}_2 \cdots \mathcal{D}_i$. For a given $\alpha \in \mathcal{D}^i$, where $3 \leq i \leq k$, let $\hat{\alpha}$ denote the element of $\mathcal{D}^{i-1}$ obtained from α by deleting its last component.

For each (j, α), where $2 \leq j \leq k$, and $\alpha \in \mathcal{D}^j$, G_2 contains a configuration-gathering node, which we denote as $v_{j,\alpha}$. In addition, for each $\alpha \in \mathcal{D}^k$, G_2 contains an evaluation node, which we denote as v^α.

Each node of the form $v_{2,\alpha}$ contains two parents: u_1 and u_2. The conditional probability table for $v_{2,\alpha}$ specifies that the value of this functional node is 1 if the values of nodes u_1 and u_2 equal the two components of α. Otherwise, the value of this node is 0.

Each node of the form $v_{j,\alpha}$, where $3 \leq j \leq k$, contains two parents: $v_{j-1,\hat{\alpha}}$ and u_j. The conditional probability table for $v_{j,\alpha}$, where $3 \leq j \leq k$, specifies that the value of this functional node is 1 if the value of node $v_{j-1,\hat{\alpha}}$ is 1 and the value of node u_j equals the value of the last component of α. Otherwise, the value of this node is 0.

We can consider the elements of $\mathcal{D}^k$ to be lexicographically ordered. Under this order, let α_0 denote the first element of $\mathcal{D}^k$, and let α_f denote the last element of $\mathcal{D}^k$. For any element $\alpha \in \mathcal{D}^k$, other than α_0, let $p(\alpha)$ denote the lexicographic predecessor of α in D^k.

Node v^{α_0} has one parent: node v_{k,α_0}. The conditional probability table for v^{α_0} specifies that if its parent node equals 1, then the probability of each value of D_v is the probability of that value in the row for α_0 in the conditional probability table for v in G_1. Otherwise, the probability of value 0 is 1.

Each node of the form v^{α}, where α is not α_0, contains two parents: $v^{p(\alpha)}$ and $v_{k,\alpha}$. The conditional probability table for v^{α} specifies that if the value of parent node $v_{k,\alpha}$ equals 1 then the probability of each value of D_v is the probability of that value in the row for α in the conditional probability table for v in G_1. Otherwise, v^{α} has the same value as parent $v^{p(\alpha)}$.

Node v has one parent: v^{α_f}. Node v is a functional node, and its conditional probability table specifies that its value equals the value of its parent node.

Now we consider the configuration of G_2 resulting from an evaluation of the BN.

It can be seen, by induction on j, that the value of a node $v_{j,\alpha}$ is 1 iff the values of the nodes in P_j equal α. Consequently, of the nodes of the form $v_{k,\alpha}$, exactly one has value 1, namely the node for which α corresponds to the values of the nodes in P_k.

Let α' denote this value of D^k (corresponding to the values of the nodes in P_k). The following can be seen by induction on the lexicographic ordering of the elements of D_k. All nodes of the form v^{α}, where α precedes α', have value 0, with probability 1. Node $v^{\alpha'}$ has each value of D_v with the probability specified in the row for α' in the conditional probability table for v in G_1. All nodes of the form v^{α}, where α' precedes α, and node v itself, have the same value as node $v^{\alpha'}$.

We now argue that G_1 is embedded in G_2. Let c_{V_1} be a given configuration of G_1. Let c_{V_2} be the following extension of c_{V_1} to V_2. Consider a given unsafe node v in G_1 for which additional nodes have been added in G_2. Let α' denote the value of D^k corresponding to the values of the nodes in P_k. In c_{V_2}, all nodes of the form v^{α}, where α precedes α', have value 0. Node $v^{\alpha'}$, nodes of the form v^{α}, where α' precedes α, and node v all have the value $c_{V_1}(v)$. Note that the probability of these node values in G_2, given the values of the nodes in P_k, is the same as the probability that that the value of v equals $c_{V_1}(v)$, given the values of $c_{V_1}(u_1)$, $c_{V_1}(u_2) \ldots c_{V_1}(u_k)$.

Thus the configuration probability $\Pr\{c_{V_1}\}$ in G_1 equals the configuration probability $\Pr\{c_{V_2}\}$ in G_2, so G_1 is embedded in G_2. ■

5 Divorcing Transformations in BNs with r-Symmetric CPTs

In this section, we present an efficient algorithm for parent divorcing in BNs with r-symmetric CPTs [4] for any fixed r. This algorithm preserves the treewidth to within a constant factor and also ensures that the maximum domain size of the new BN is bounded by a polynomial function of the size of the given BN. As will be shown, this parent divorcing algorithm leads to an efficient algorithm for inference problems on (unmoralized) BNs whose CPTs are r-symmetric functions, thus providing alternative proof of a result established in [37]. We begin with the definition of the class of r-symmetric functions from [4].

Definition 5.1. *Let $r \geq 1$ be a* fixed *integer. Let q be an integer ≥ 1. A function f from $\{0,1\}^q$ to the set of real values in $[0,1]$ is said to be* ***r-symmetric*** *if the set of inputs can be partitioned into r classes such that the value of f depends only on the number of 1-valued inputs in each class.*

A 1-symmetric function is also referred to as a **symmetric function** in the literature [4]. For any $r \geq 1$, it can be seen that any r-symmetric function f of q variables can be concisely described by specifying $O(q^r)$ probability values. Since r is fixed, the size of the specification of any r-symmetric function is a polynomial in the size of the BN. When a node has a bounded indegree, say d, the corresponding CPT can be thought of as a d-symmetric function, where each of the d classes contains exactly one input. For any fixed integer $r \geq 1$, an r-symmetric BN is one in which the CPT of each node is an r'-symmetric function for some $r' \leq r$.

We now discuss divorce transformations for r-symmetric BNs. Consider a given BN node v of such a BN. We assume that all the parents of v in a given class of the r-symmetric function for v have the same domain. If not, we can for notational purposes, envision some of the domains enlarged if necessary so that the domains of all the parents in each class are the same.

A **signature** for v is a function that maps each pair consisting of a class, and a non-zero domain value for that class, into a nonnegative integer, such that for each class ν for v, the sum of the signature values for class ν does not exceed the number of parents of v that are in ν. Note that for any fixed bound Δ on domain cardinality and fixed r, the number of signatures for v is a polynomial in the number of parents of v.

We say that a given signature is **feasible** for a set of parents P' of v, if for each class ν for v, the sum of the signature values for class ν does not exceed the number of members of P' that are in ν. For a given assignment α to the nodes in P', we say that signature σ **summarizes** α if for each pair (ν, x) in the domain of σ, the value of $\sigma(\nu, x)$ = the number of members of class ν that are assigned value x by α.

We now discuss how to construct domains in the algorithm in Theorem 3.1, so as to take advantage of r-symmetry. The only type of transformation in the algorithm that modifies the current BN is a divorce transformation. Consider a given divorce transformation $\tau^i_{u,w,v}$. Node v is an unsafe node of the original BN G_1. The transformation creates a new node v', which is made a parent of v, with new edge (v', v) replacing the edges that made u and w parents of v. We let the domain of v' be the set of signatures for v. The conditional probability table for $x_{v'}$ is purely functional, with the value of $x_{v'}$ being the component-wise sum of the signatures for v corresponding to the values of x_u and x_w. The conditional probability table for x_v is appropriately modified to use the value of $x_{v'}$ instead of the values of x_u and x_w. Using this technique in the divorce transformation yields the following result.

Theorem 5.2. *For any fixed bound d on domain cardinality and fixed r, there is a polynomial-time algorithm that given a BN G_1 and a tree decomposition T_1 of treewidth t for G_1, constructs a BN G_2 and a tree decomposition of treewidth at most $3t + 3$ for G_2, such that G_1 is functionally embedded in G_2, each node of G_2 has at most two parents, the number of added nodes in G_2 is less than than number of edges in G_1, and the domain size of the added nodes is polynomial in the maximum node degree of G_1.* ■

Theorem 5.2 shows that parent divorcing can be done on BNs with r-symmetric CPTs so that treewidth is preserved to within a constant factor and the domain size is increased only by a polynomial factor. We now show how this theorem in conjunction with known algorithms for inference problems for treewidth bounded BNs leads to an alternative proof of the following theorem established in [37].

Theorem 5.3. *Inference problems for unmoralized treewidth-bounded BNs in which the maximum domain size is bounded and each CPT is given as an r-symmetric function for some fixed $r \geq 1$ can be solved in polynomial time.*

Proof: Consider an unmoralized and treewidth-bounded BN G with n nodes where the maximum domain size is bounded. Thus, the maximum indegree is at most $n - 1$. We first apply the algorithm mentioned in Theorem 5.2 to G to obtain a new treewidth-bounded BN G' where each node has indegree at most two and the maximum domain size is $O(n^k)$ for some fixed integer k. Several inference algorithms with a running time of $O(D^t N)$ are known for treewidth-bounded BNs, where D is the maximum domain size, t is the treewidth and N is the number of nodes [18,38,41]. We can use any such algorithm on G' to solve inference problems. Since the number of nodes in G' and the maximum domain size in G' are polynomial in n (the number of nodes of G), the running time of the inference algorithm is a polynomial in n. Further, since G is functionally embedded in G', we can efficiently obtain a solution to an inference problem for G from the corresponding solution to G'. ■

6 Lower Bound Results

In the previous sections, we presented parent divorcing algorithms that considered structural properties such as treewidth (Section 3) and the maximum domain size (Section 4). In the former case, it was pointed out that the domain sizes of the new nodes in the resulting BN may become exponentially large. In the latter case, it was observed that the the number of new nodes in the resulting BN may be exponentially large. In this section, we present lower bound results to show that such increases cannot be avoided in general.

6.1 Lower Bounds for General BNs

We first show that functional embedding subject to a fixed bound on the number of parents requires an exponential increase in domain size, even if there is no constraint on treewidth.

Theorem 6.1. *For every $n > 1$, there is a BN G_n, whose treewidth is 1 and all of whose nodes are binary-valued, such that for every BN G' such that G_n is functionally embedded in G', G_n and G' have the same set*

of source nodes, and b is an upper bound on the number of parents of every node of G', there is at least one node of G' whose domain size is at least $2^{(n-1)/b}$.

Proof: G_n contains a single sink node y (i.e., node of outdegree 0) and $n-1$ source nodes (i.e., nodes of indegree 0) x_i, $0 \le i \le n-2$. There is an edge from each source node to the sink node. The CPT (conditional probability table) for each source node x_i specifies that the probability that variable x_i equals 1 is $1/2$. The CPT for node y is as follows. For each assignment α of values to the source nodes, let $\hat{\alpha}$ denote the integer corresponding to interpreting α as the bits of a number written in binary. Note that the value of $\hat{\alpha}$ ranges between 0 and $2^{n-1}-1$. The row for α in the CPT for y specifies that the probability that variable y equals 1 is $(\hat{\alpha}+1)/2^n$.

Now consider G'. Because G_n is functionally embedded in G', each assignment α to the source nodes of G' selects a row of the CPT for y in G'. If two distinct assignments were to select the same row of this CPT, they would assign the same probability for y taking on the value 1 in G'. But in G_n, each assignment α selects a unique probability for y to assume the value 1. Thus, each of the 2^{n-1} assignments to the source nodes of G' must select a distinct row of the CPT for y. Let d denote the maximum domain size of G'. The CPT for y in G' contains at most d^b rows. Thus $d^b \ge 2^{n-1}$; in other words, $d \ge 2^{(n-1)/b}$. ■

Our next theorem shows that embedding subject to fixed bounds on the number of parents and domain size requires an exponential number of new nodes, even if there is no constraint on treewidth. To prove the theorem, we begin with some definitions.

Definition 6.2. *A CPT* **entry** *is the specification in the CPT for a BN node v of the the probability of each value in the domain of v, given specific values for each of the parents of v. We call a given entry* **selective** *if at least two of these probability values are nonzero.*

Note that if BN node v has b_v parents, with respective domain sizes $d_1, d_2, \ldots, d_{b_v}$, then the number of entries in the CPT for v is $\prod_{i=1}^{b_v} d_i$.

Definition 6.3. *(1) We say that a configuration c_V of a BN $G(V,E)$ is* **viable** *if its probability is nonzero. (2) For a given viable configuration c_V of a BN $G(V,E)$, we say that a given entry for a node v is* **active** *for c_V if the combination of values of the parents of v in c_V correspond to that entry.*

The following observations are direct consequences of Definition 2.2 and Lemma 2.3.

Observation 6.4. *If BN G is embedded in BN G', then G and G' have the same number of viable configurations.* ■

Observation 6.5. *If BN G is embedded in BN G', and v is a source node of both G and G', then the CPT for v in G is identical to the CPT for v in G'.* ■

We are now ready to state and prove our lower bound on the number of new nodes in a BN after parent divorcing.

Theorem 6.6. *For every $n > 1$, there is a BN $G_n(V, E)$, whose treewidth is 1 and all of whose nodes are binary-valued, such that for every BN $G'(V', E')$ such that G_n is embedded in G', G_n and G' have the same set of source nodes, b is an upper bound on the number of parents of every node of G', and d is an upper bound on the domain size of the nodes in G', there are at least $\frac{2^{n-1}}{d^b} - 1$ nodes in G' that are not in G_n.*

Proof: Consider the BN G_n, specified in the proof of Theorem 6.1. For any given assignment α to the source nodes of G_n, there are exactly two viable configurations of G_n that are extensions of α. Thus, by Observation 6.4, there are exactly two viable configurations of G' that are extensions of α.

Consider a given viable configuration $c_{V'}$ of G' that is an extension of some assignment α to the source nodes of G'. Since there are exactly two viable configurations of G' that are extensions of α, among the non-source nodes of G', there is **exactly one** non-source node active entry for $c_{V'}$ that is selective. Moreover, this active entry, say for node v, has exactly two nonzero probability values for v. From Observation 6.5, the probability of $c_{V'}$ is the product of $(1/2)^{n-1}$ and the probability value selected by this selective entry.

For a given assignment α to the source nodes of G', let α' denote the viable configuration of G' where the source nodes have the values corresponding to α, and node y has the value 1. Let us say that two probability values are **complementary** if their sum is 1. For distinct α_1 and α_2, the probabilities of α'_1 and α'_2 are the product of $(1/2)^{n-1}$ and two probability values that are distinct and non-complementary. Thus, the non-source node selective active entry for α'_1 is distinct from the non-source node selective active entry for α'_2.

Consequently, there are at least 2^{n-1} distinct non-source node selective entries in the CPTs for G'. Each non-source node of G' contains at most d^b entries. Let q denote the number of nodes in G' that are not in G_n. G' contains $q + 1$ non-source nodes. Therefore, $(q + 1)d^b \geq 2^{n-1}$. In other words, $q \geq \frac{2^{n-1}}{d^b} - 1$. ■

6.2 Lower Bounds for BNs with Symmetric CPTs

Here, we show that the above techniques can also be used to prove polynomial lower bounds for divorce transformations in BNs whose CPTs are 1-symmetric (i.e., symmetric). First, the following result provides a polynomial lower bound on domain size for symmetric CPTs.

Theorem 6.7. *For every $n > 1$, there is a BN G_n, whose treewidth is 1, all of whose nodes are binary-valued, and all of whose CPTs are symmetric, such that for every BN G' such that G_n is functionally embedded in G', G_n and G' have the same set of source nodes, and b is an upper bound on the number of parents of every node of G', there is at least one node of G' with domain size $\geq (n-1)^{1/b}$.*

Proof: Let BN G_n be the same as that constructed in the proof of Theorem 6.1, except that for each assignment α of values to the source nodes, we now let $\hat{\alpha}$ denote the integer corresponding to interpreting α as the bits of a number written in unary, i.e. $\hat{\alpha}$ is the number of 1s in α. With this change, the value of $\hat{\alpha}$ ranges between 0 and $n - 1$. The row for α in the CPT for y is changed to specify that the probability that variable y equals 1 is $\frac{\hat{\alpha}+1}{2n}$.

Now consider G'. Each of the $n - 1$ possible values of $\hat{\alpha}$ must select a distinct row of the CPT for y. Let d denote the maximum domain size of G'. The CPT for y in G' has $\leq d^b$ rows. Thus $d^b \geq n - 1$ or

$d \geq (n-1)^{1/b}$. ■

The following result provides a linear lower bound on the number of new nodes for symmetric CPTs.

Theorem 6.8. *For every $n > 1$, there is a BN $G_n(V, E)$, whose treewidth is 1, all of whose nodes are binary-valued, and all of whose CPTs are symmetric, such that for every BN $G'(V', E')$ such that G_n is embedded in G', G_n and G' have the same set of source nodes, b is an upper bound on the number of parents of every node of G', and d is an upper bound on the domain size of the nodes in G', there are at least $\frac{n-1}{d^b} - 1$ nodes in G' that are not in G_n.*

Proof: Consider the BN G_n, specified in the proof of Theorem 6.7 and the $n-1$ assignments α to the source nodes of G_n, such that their corresponding $n-1$ values of $\hat{\alpha}$ are distinct. Each of these $n-1$ assignments has a distinct non-source node selective entry in the CPTs for G'. So, letting q denote the number of nodes in G' that are not in G_n, we have $(q+1)\, d^b \geq n-1$. So, $q \geq \frac{n-1}{d^b} - 1$. ■

We note that the above lower bound results are for BNs with 1-symmetric CPTs, and they hold even when there is no restriction on the treewidth of the resulting BN. Our parent divorcing algorithm discussed in the proof of Theorem 5.2 allows BNs with r-symmetric CPTs for any fixed $r \geq 1$ and preserves treewidth to within a constant factor while ensuring that the maximum domain size increases only by a polynomial amount and the number of new nodes added is linear in the size of the given BN.

7 Directions for Future Work

We conclude by mentioning some directions for future work. First, it is of interest to improve our treewidth-preserving algorithm (Section 3) so that the treewidth increases by a factor smaller than 3. Second, it would be useful to improve our algorithm (Section 4) that preserves the maximum domain size so that it also preserves treewidth to within a small constant factor. Finally, it is also of interest to devise other formalisms (similar to our embedding formalism) for parent divorcing that can be used to reason rigorously about the relationships between the given BN and the modified BN.

Acknowledgments: This work was partially supported by University of Virginia Strategic Investment Fund Award SIF160, and NSF grants CMMI-1745207 (EAGER), OAC-1916805 (CINES) and CCF-1918656 (Expeditions).

References

[1] A. M. Abdelbar and S. M. Hedetniemi. Approximating MAPs for belief networks is NP-hard and other theorems. *Artificial Intelligence*, 102(1):21–38, 1998.

[2] A. M. Abdelbar, S. T. Hedetniemi, and S. M. Hedetniemi. The complexity of approximating MAPs for belief networks with bounded probabilities. *Artificial Intelligence*, 124(2):283–288, 2000.

[3] F. Bacchus, S. Dalmao, and T. Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *Proc. 44th Annual FOCS*, pages 340–351, 2003.

[4] C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and M. Thakur. Predecessor existence problems for finite discrete dynamical systems. *Theoretical Computer Science*, 386:3–37, Oct. 2007.

[5] S. Basent, A. BahooTooroody, M. Chaal, J. Lahtinen, V. Balbot, and O. A. Valdez-Banda. Risk analysis methodology using STPA-based Bayesian network applied to remote pilotage operation. *Ocean Enfineering*, 270:1–18, 2023.

[6] H. Bodlaender. Treewidth: Algorithmic techniques and results. *Proc. 22nd Symposium on Mathematical Foundations of Computer Science*, pages 29–36, 1997.

[7] H. Bodlaender. Probabilistic networks: Inference and other problems. Lecture Slides – Institute for Programming Research and Algorithmics, 2004.

[8] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–22, 1993.

[9] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. UAI 1996*, pages 115–123, 1996.

[10] M. Chavira. *Beyond Treewidth in Probabilistic Inference*. PhD thesis, Computer Science Dept., UCLA, 2007.

[11] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[12] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, 1993.

[13] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, New York, NY, 2009.

[14] C. P. de Campos. New complexity results for MAP in Bayesian networks. In *Proc. IJCAI*, pages 2100–2106, 2011.

[15] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

[16] Gaspard Ducamp. PROCOP: Probabilistic rules compilation and optimisation. Ph.D. thesis, 2021.

[17] T. Gao, K. P. Fadnis, and M. Campbell. Local-to-global Bayesian network structure learning. In *Proc. ICML*, pages 1193–1202, 2017.

[18] V. G. Gogate. *Sampling Algorithms for Probabilistic Graphical Models with Determinism*. PhD thesis, University of California Irvine, 2009.

[19] F. Jensen, S. Lauritzen, and K. Olesen. Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, 4:269–282, 1990.

[20] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, New York, NY, 2001.

[21] A. K. Jha, V. Gogate, A. Meliou, and D. Suciu. Lifted inference seen from the other side: The tractable features. In *Proc. NIPS*, pages 973–981, 2010.

[22] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, MA, 2009.

[23] Jidapa Kraisangka and Marek J. Druzdzel. Making large Cox's proportional hazard models tractable in Bayesian networks. In Alessandro Antonucci, Giorgio Corani, and Cassio Polpo de Campos, editors, *Probabilistic Graphical Models - Eighth International Conference, PGM 2016, Lugano, Switzerland, September 6-9, 2016. Proceedings*, volume 52 of *JMLR Workshop and Conference Proceedings*, pages 252–263. JMLR.org, 2016.

[24] A. Kucukelbir, Y. Wang, and D. M. Blei. Evaluating Bayesian models with posterior dispersion indices. In *Proc. ICML*, pages 1925–1934, 2017.

[25] J. Kwisthout, H. L. Bodlaender, and L. C. van der Gaag. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *ECAI*, pages 237–242, 2010.

[26] J. Kwisthout and C. P. de Campos. Lecture notes: Computational complexity of Bayesian networks. Tutorial at UAI, 2015.

[27] S. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *J. Royal Statistical Society, Series B*, 50(2):157–224, 1988.

[28] J. Lee, C. Heaukulani, Z. Ghahramani, L. F. James, and S. Choi. Bayesian inference on random simple graphs with power law degree distributions. In *Proc. ICML*, pages 2004–2013, 2017.

[29] J. G. Lindley and M. R. Blackburn. Architecting very large, complex Bayesian network simulations for practical airworthiness risk assessment applications. *Procedia Computer Science*, 114:37–46, 2017.

[30] R. Mateescu. *AND/OR Search Spaces for Graphical Models*. PhD thesis, Computer Science Dept., UC Irvine, 2007.

[31] B. Milch, L. S. Zettlemoyer, K. Kresting, M. Haimes, and L. P. Kaelbling. Lifted probabilistic inference with counting formulas. In *Proc. AAAI*, pages 1062–1068, 2008.

[32] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.

[33] K. G. Olesen, U. Kjaerulff, F. Jensen, F. V. Jensen, B. Falck, S. Andreassen, and S. K. Andersen. A MUNIN Network for the Median Nerve – A Case Study on Loops. *Applied Artificial Intelligence*, 3(2-3):385–403, Oct. 1989.

[34] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann Publishers, San Mateo, CA, 1988.

[35] E. Rajabally, P. Sen, S. Whittle, and J. Dalton. Aids to Bayesian belief network construction. In *Proc. Second IEEE International Conference on Intelligent Systems*, pages 457–461, June 2004.

[36] F. Röhrbein, J. Eggert, and E. Körner. Child-friendly divorcing: Incremental hierarchy learning in Bayesian networks. In *Proc. Intl. Joint Conf. on Neural Networks*, pages 2711–2716, 2010.

[37] D. J. Rosenkrantz, M. V. Marathe, S. S. Ravi, and A. K. Vullikanti. Bayesian inference in treewidth-bounded graphical models without indegree constraints. In *Proc. 30th Conference on Uncertainty in AI (UAI 2014)*, pages 702–711, July 2014.

[38] S. Sarawagi. Probabilistic graphical models. Slides from VLDB Tutorial, 2007.

[39] S. E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410, 1994.

[40] U. von Waldow and F. Röhrbein. Structure learning in Bayesian networks with parent divorcing. In *EAPCogSci*, volume 1419 of *CEUR Workshop Proceedings*, pages 146–151. CEUR-WS.org, 2015.

[41] M. J. Wainright and M. I. Jordan. Graphical models, exponential families and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

[42] P. P. Wu, F. Ruggeri, and K. Mengersen. Optimal partitioning of directed acyclic graphs with dependent costs between clusters. arXiv:2308.03970v1, 2023.

[43] Y. Xiang and D. Loker. Trans-causalizing NAT-modeled Bayesian networks. *IEEE Trams. Cybernetics*, 52(5):3553–3566, 2022.

[44] N. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.