Controlled Generation of Hard and Easy Bayesian Networks: Impact on Maximal Clique Size in Tree Clustering

Ole J. Mengshoel
RIACS
NASA Ames Research Center
Mail Stop 269-3
Moffett Field, CA 94035
omengshoel@riacs.edu

David C. Wilkins
Center for the Study of Language and Information
Stanford University
Stanford, CA 94305
dwilkins@stanford.edu

Dan Roth
Department of Computer Science
University of Illinois, Urbana-Champaign
201 N. Goodwin
Urbana, IL 61801
danr@cs.uiuc.edu

Abstract

This article presents and analyzes algorithms that systematically generate random Bayesian networks of varying difficulty levels, with respect to inference using tree clustering. The results are relevant to research on efficient Bayesian network inference, such as computing a most probable explanation or belief updating, since they allow controlled experimentation to determine the impact of improvements to inference algorithms. The results are also relevant to research on machine learning of Bayesian networks, since they support controlled generation of a large number of data sets at a given difficulty level. Our generation algorithms, called BPART and MPART, support controlled but random construction of bipartite and multipartite Bayesian networks. The Bayesian network parameters that we vary are the total number of nodes, degree of connectivity, the ratio of the number of non-root nodes to the number of root nodes, regularity of the underlying graph, and characteristics of the conditional probability tables. The main dependent parameter is the size of the maximal clique as generated by tree clustering. This article presents extensive empirical analysis using the HUGIN tree clustering approach as well as theoretical analysis related to the random generation of Bayesian networks using BPART and MPART.

1 Introduction

Essentially all inference problems studied using the Bayesian network (BN) formalism are known to be computationally hard in the general case [14, 60, 66]. Given the central role of BNs in a wide range of automated reasoning applications, for example in medical diagnosis [3, 43, 67], probabilistic risk analysis [9,45], language understanding [10,12], intelligent data analysis [40,54,61], error correction coding [27,28,48, 49], and biological pedigree analysis [68], developing efficient algorithms for these inference problems is an important research problem. The performance of exact Bayesian network inference algorithms — including tree clustering algorithms [2,33,38,39,46,65], conditioning algorithms [16,17,22,32,57,58,64], and elimination algorithms [19, 47, 72] — depends on the treewidth or the optimal maximal clique size of a BN's induced clique tree [5,17,20,21]. Treewidth was initially a theoretical concept related to graph minors [59]; it has more recently been established that the notion of treewidth plays a key role in the analysis of algorithms [8,20,44].

A significant component of research on inference in BNs has to be experimental and rely on the use of BN instances. Similar experiments are needed and have indeed been performed for other problems, including the satisfiability problem (SAT) [11,13,26,55,62,63]. For SAT, it has been established empirically that there

is a phase transition in the probability of satisfiability of an instance drawn from a certain distribution [55]. This phase transition phenomenon has been found to be closely related to a parameter describing the constrainedness of instances, namely the ratio between the number of variables V and the number of clauses C, denoted the C/V-ratio. Interestingly, it has been found that algorithmic hardness also varies with the C/V-ratio, at least for certain algorithms [55]. As the C/V-ratio is varied, there is a variation in problem difficulty (or hardness), as measured in mean or median inference time for certain algorithms across a sample of problems. Maximal hardness for several algorithms occurs in the phase transition region.

Experimental work in Bayesian network inference can also be performed using randomly generated instances. In this article, we investigate the following research questions: How should BNs for experimentation be randomly generated, such that their computational hardness can be understood, analyzed, and controlled? More specifically, is it fruitful to generalize the C/V-ratio from SAT to a BN setting? If it is, what is the relationship between the C/V-ratio and treewidth or maximal clique size?

Answering these research questions is important for several reasons. Generating problem instances randomly, a common practice in the BN community [7, 17, 34, 35, 41, 56, 69, 70], may result in easy inference problems that do not present a challenge to inference algorithms [4, 11, 25], even though worst case complexity results show that both exact and approximate MPE computation is NP-hard [1, 66]. In this article we extend previous research on randomly generating BN instances and present an experimental paradigm for systematically generating increasingly hard random Bayesian network instances for tree clustering. We describe two algorithms for controlled generation of BNs, the bipartite (BPART) and multipartite (MPART) construction algorithms, and prove several properties for the BNs that they construct. For the BPART case, this includes the distribution over the root node out-degrees and the minimum out-degree as well as the (small) probability that an irregular BN is also regular. For MPART networks [41] we analyze the relationship to BPART BNs and in particular present a formula for the probability that an MPART BN is bipartite. We characterize properties of the BN generation algorithms in order to better understand the factors that in turn contribute to the hardness of inference, so that thorough benchmarking and comparison of algorithms can be performed.

The inference approach we focus on, tree clustering as implemented in the HUGIN algorithm, was introduced as a belief updating algorithm [46], and was later extended to encompass belief revision [18]. Thus, in the tree clustering approach, computing marginal distributions and most probable explanations (MPEs) are closely related. In particular, they both depend on the total clique tree size as well as the maximal clique size of a BN's clique tree. In a BN, let V be the number of root nodes and C the number of non-root nodes. We show that the C/V-ratio is a key parameter for BN inference hardness, as it is for SAT [11, 55]. Analytically, we provide a conservative lower bound on total clique tree size and introduce a new class of BNs, only-child BNs, for which we give sufficient conditions for Hamiltonicity and longest cycle. Formation of cycles, including Hamiltonian cycles, is important because they often need fill-in edges in order for a triangulated graph to be constructed, and cycles thus significantly contribute to clique tree size.

Even when the topology is restricted to the BPART or MPART types, we identify several input parameters that can be varied when randomly generating BNs. We empirically study a few of these parameters in detail and show how changing them affects properties of the generated BNs which again can increase computational hardness for tree clustering. For both the BPART and MPART constructions, generating random networks may result in very easy instances, but a careful selection of the parameters along the dimensions we discuss, even while keeping the size of the networks fixed, gradually increases the complexity of inference and results in networks that the tree clustering algorithm cannot handle. A main empirical result is that the C/V-ratio can be used to predict an upper bound on the treewidth (or optimal maximal clique size) of the induced clique trees for samples of BPART and MPART BNs. Our selection of families of hard networks extends research on generating hard instances for the satisfiability problem [4, 11, 25, 55] as well as existing research in the BN community [34, 41, 70]. Increasing the C/V-ratio causes, for certain values for C and V, an approximately linear increase in the upper bound on treewidth or the number of nodes in the largest clique. In other words, we obtain an easy-hard-harder pattern for tree clustering algorithms including HUGIN, which contrasts with the easy-hard-easy pattern observed for SAT formulas using the Davis-Putnam algorithm [55]. Experimenters may thus use the C/V-ratio directly, instead of or as a complement to maximal clique size or treewidth. In addition to the C/V-ratio, we study the regularity of a BN's underlying graph and the distributional nature of conditional probability tables.

The rest of this article is organized as follows. Section 2 introduces Bayesian network definitions and

notation as well as the MPE problem. In Section 3 we briefly describe inference and in particular tree clustering and the HUGIN algorithm as well as the concepts of maximal clique size and treewidth. Section 4 discusses the use of application BNs and randomly generated BNs for experimentation. In particular, Bayesian networks generated by the BPART algorithm as well as the MPART algorithm are presented and analyzed; there are also results on their relationship. Section 5 discusses the interaction between properties of randomly generated BNs and their hardness for tree clustering algorithms and HUGIN in particular. In Section 6 we turn to the experimental part of the article, with experimental results for BPART and MPART networks, using the state-of-the-art inference system HUGIN to study characteristics of the maximal clique sizes generated as well as inference times. Section 7 concludes and discusses future work.

Earlier versions of this research have been reported previously [51, 52]. In closely related work we have developed and investigated a stochastic local search approach to computing MPEs and compared it to tree clustering for varying C/V-ratios [51, 53].

2 Preliminaries

A Bayesian network (BN) represents a multi-variate probability distribution as a directed acyclic graph (DAG), where the nodes represent random variables.

Definition 1 (Directed acyclic graph (DAG)) *Let $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{E})$ be a directed acyclic graph (DAG) with nodes $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ and edges $\boldsymbol{E} = \{E_1, \ldots, E_m\}$. An ordered tuple $E_i = (Y, X)$, where $1 \leq i \leq m$ and $X, Y \in \boldsymbol{X}$, represents a directed edge from Y to X. Here, Π_X denotes the parents of X: $\Pi_X = \{Y \mid (Y, X) \in \boldsymbol{E}\}$. Similarly, Ψ_X denotes the children of X: $\Psi_X = \{Z \mid (X, Z) \in \boldsymbol{E}\}$. The out-degree and in-degree of a node X is $o(X) = |\Psi_X|$ and $i(X) = |\Pi_X|$ respectively. The minimal non-zero out-degree of any node in $\boldsymbol{G}$ is denoted $\delta_o(\boldsymbol{G})$ and the minimal non-zero in-degree of any node in $\boldsymbol{G}$ is denoted $\delta_i(\boldsymbol{G})$; $n(\boldsymbol{G}) = |\boldsymbol{X}|$ is the number of nodes in $\boldsymbol{G}$.*

The following characterization of graphs in general and BNs in particular turns out to be fruitful when analyzing the performance of inference algorithms on BNs.

Definition 2 (Root node, non-root node, leaf node) *Let $\boldsymbol{G}$ be a non-empty DAG and let X be a node in $\boldsymbol{G}$. If $i(X) = 0$ then X is a root node. If $i(X) > 0$ then X is a non-root node. If $i(X) > 0$ and $o(X) = 0$ then X is a leaf node.*

Any non-empty DAG $\boldsymbol{G}$ has at least one root node so $V \geqslant 1$ and the C/V-ratio is always well-defined for non-empty DAGs according to Definition 2. Only non-empty graphs are considered in the rest of this article. In the important special case of bipartite DAGs, which we formally introduce below, the C/V-ratio is the ratio of the number of leaf nodes to the number of root nodes.

Definition 3 (Bipartite DAG) *Let $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{E})$ be a DAG. If $\boldsymbol{X}$ can be split into partite sets $\boldsymbol{V} = \{X \in \boldsymbol{X} \mid i(X) = 0\}$ (the root nodes) and $\boldsymbol{C} = \{X \in \boldsymbol{X} \mid i(X) > 0\}$ (the leaf nodes) such that any $(V, C) \in \boldsymbol{E}$ is such that $V \in \boldsymbol{V}$ and $C \in \boldsymbol{C}$ then $\boldsymbol{G}$ is a bipartite DAG; $\mathcal{B}$ is the set of all bipartite DAGs.*

In tree clustering algorithms, which we return to in Section 3, a directed graph (BN) is transformed into an undirected graph (a clique tree) over which inference is performed.

Definition 4 (Undirected graph) *Let $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{E})$ be an undirected graph with nodes $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ and edges $\boldsymbol{E} = \{E_1, \ldots, E_m\}$. An undirected edge $E_i = \{X, Y\}$ is a set where $1 \leq i \leq m$ and $X, Y \in \boldsymbol{X}$. The set of adjacent (or neighbor) nodes of a node X is denoted $a(X) = \{Y \mid \{X, Y\} \in \boldsymbol{E}\}$ while its degree, $d(X) = |a(X)|$, is the number of neighbors X has in $\boldsymbol{G}$. Finally, $\delta(\boldsymbol{G})$ is the minimal degree of all nodes in $\boldsymbol{G}$ and $n(\boldsymbol{G}) = |\boldsymbol{X}|$ is the number of nodes in $\boldsymbol{G}$.*

In this article we will usually not distinguish, in BNs, between a graph node and the corresponding random variable. For the purpose of this article we also focus exclusively on labelled graphs — graphs with distinguishable vertices — both in the directed and the undirected cases. When parts of a graph (BN) are studied, the following notion of an induced subgraph is useful.

Definition 5 (Induced subgraph) *Let $\boldsymbol{X}$ be the nodes in a directed graph. The product $\boldsymbol{X} \times \boldsymbol{X}$ is defined as $\{(X_i, X_j) \mid X_i, X_j \in \boldsymbol{X}\}$. Let $\boldsymbol{Y}$ be the nodes in an undirected graph. The product $\boldsymbol{Y} \times \boldsymbol{Y}$ is defined as $\{\{Y_i, Y_j\} \mid Y_i, Y_j \in \boldsymbol{Y}\}$. Let $\boldsymbol{G} = (\boldsymbol{Z}, \boldsymbol{E})$ be a (directed or undirected) graph. The induced subgraph $\boldsymbol{G}[\boldsymbol{W}] = (\boldsymbol{W}, \boldsymbol{E}[\boldsymbol{W}])$ is a graph with nodes $\boldsymbol{W} \subseteq \boldsymbol{Z}$ and edges $\boldsymbol{E}[\boldsymbol{W}] = (\boldsymbol{W} \times \boldsymbol{W}) \cap \boldsymbol{E}$.*

We also extend the graph notation and definitions to BNs, understanding that it applies to the graph part of the formal definition of BNs, which follows. First, we have the following definition.

Definition 6 (BN node) *A discrete BN node X is a random variable with a discrete, finite state space $\Omega_X = \{x_1, \ldots, x_k\}$.*

While BN nodes can also be continuous, this article is restricted to the discrete case and we will take "BN node" to mean "discrete BN node" in the following.

Definition 7 (Bayesian network) *A Bayesian network is a tuple $\beta = (\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{P})$, where $(\boldsymbol{X}, \boldsymbol{E})$ is a DAG with an associated set of conditional probability distributions $\boldsymbol{P} = \{\Pr(X_1 \mid \Pi_{X_1}), \ \ldots, \ \Pr(X_n \mid \Pi_{X_n})\}$. Here, $\Pr(X_i \mid \Pi_{X_i})$ is the conditional probability distribution for $X_i \in \boldsymbol{X}$. Further, let π_{X_i} represent the instantiation of the parents Π_{X_i} of X_i. The independence assumptions encoded in $(\boldsymbol{X}, \boldsymbol{E})$ imply the joint probability distribution*

$$\Pr(\boldsymbol{x}) = \Pr(x_1, \ \ldots, \ x_n) = \Pr(X_1 = x_1, \ \ldots, \ X_n = x_n) = \prod_{i=1}^{n} \Pr(x_i \mid \pi_{X_i}). \tag{1}$$

Bayesian networks are also known as belief networks, Bayesian belief networks, or probabilistic networks; a conditional probability distribution $\Pr(X_i \mid \Pi_{X_i})$ is also known as a conditional probability table (CPT).

Sometimes a BN is provided with *observations* or *evidence* by setting or clamping m nodes $\{O_1, \ldots, O_m\}$ to known states $\boldsymbol{o} = \{O_1 = o_1, \ldots, O_m = o_m\} = \{o_1, \ldots, o_m\}$. These nodes are called *observation nodes* and need to be considered when computing a most probable explanation, which is defined below.

Definition 8 (Explanation) *Consider a BN $\beta = (\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{P})$ with $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ and observations $\boldsymbol{o} = \{o_1, \ldots, o_m\}$ for $m \leq n$. An explanation $\boldsymbol{x}$ assigns states to all non-evidence nodes $\{X_{m+1}, \ldots, X_n\}$: $\boldsymbol{x} = \{x_{m+1}, \ldots, x_n\} = \{X_{m+1} = x_{m+1}, \ldots, X_n = x_n\}$.*

When discussing an explanation $\boldsymbol{x}$, the BN β for which $\boldsymbol{x}$ is an explanation is easily understood and therefore left implicit. Among explanations, the u most probable ones are of particular interest.

Definition 9 (Most probable explanation (MPE)) *Let $\boldsymbol{x}$ range over all explanations in a BN β. Finding a most probable explanation (MPE) in β is the problem of computing an explanation $\boldsymbol{x}^*$ such that $\Pr(\boldsymbol{x}^*) \geq \Pr(\boldsymbol{x})$. The u most probable explanations is $\boldsymbol{X}^* = \{\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_u^*\}$ where $\Pr(\boldsymbol{x}^*) = \Pr(\boldsymbol{x}_1^*) = \cdots = \Pr(\boldsymbol{x}_u^*)$ and $\Pr(\boldsymbol{x}_i^*) \geq \Pr(\boldsymbol{x})$ for $1 \leq i \leq u$.*

Here, $u = |\boldsymbol{X}^*|$ is simply the number of MPEs in a BN and no other explanation has higher probability than $\boldsymbol{x}^* \in \boldsymbol{X}^*$. Since there might be $u > 1$ MPEs with the same probability we say "an" or "one" MPE rather than "the" MPE. As is common, we compute just one MPE $\boldsymbol{x}^*$ even when multiple MPEs exist in a BN. Following Pearl we sometimes denote computing an MPE as belief revision, while computing the marginal distribution over a BN node is also denoted belief updating [58].

It has been shown that exact MPE computation is NP-hard [66]. The problem of relative approximation of an MPE is to find an assignment with probability close to that of an MPE to within a small ratio. This problem has also been proven to be NP-hard [1]. Belief updating is computationally hard also [14, 60].

3 Inference in Bayesian Networks

In addition to the arguments referring to the mapping from SAT, the claim that we have a way to generate hard and easy BNs for the inference task needs to be supported analytically and experimentally by considering one or more BN inference algorithms. BN inference algorithms can be classified as exact or approximate.

Exact BN inference algorithms are the main focus in this article and include tree clustering algorithms [2, 33, 38, 39, 46, 65], conditioning algorithms [16, 17, 22, 32, 57, 58, 64], elimination algorithms [19, 47, 72], and hybrid exact methods [20].

For the purpose of this article, we study in detail one of the most prominent inference approaches — the tree clustering approach and more specifically the HUGIN algorithm.[1] Tree clustering is further discussed in Section 3.1. Section 3.2 briefly discusses other exact BN inference algorithms.

3.1 Inference by Tree Clustering: The Role of Maximal Clique Size

Tree clustering is currently one of the major approaches to inference in multiply connected Bayesian networks [58]. Like other tree clustering algorithms, the HUGIN algorithm employs two phases: a compilation (or clustering) phase and a propagation (or run-time) phase [2, 37, 39, 46]. During compilation, a Bayesian network is transformed into cliques organized in a clique tree. During propagation, evidence is propagated in the clique tree, leading to belief updating or belief revision computations as appropriate.

A clique (or junction) tree is constructed from a Bayesian network in the following way by the HUGIN algorithm. First, an initial moral graph β' is constructed by making an undirected copy of β and then augmenting it as follows. Let X systematically range over all nodes in β. For each node X, HUGIN adds to β' an edge between each pair of nodes in Π_X if no such edge already exists in β'. Second, HUGIN triangulates the moral graph β', creating a triangulated graph β''. Triangulation amounts to adding fill-in edges to the moral graph β' such that no chordless cycle of length greater than three exists. Third, a clique tree β''' is created from the triangulated graph β''.[2] This clique tree — which consists of cliques and separators — must exhibit the property that for any two clique nodes F and H in the tree, all nodes between them contain $F \cap H$. In β''', where both cliques and separators have belief tables associated with them, the joint probability $\Pr(\boldsymbol{X})$ is the product of the clique belief tables divided by separator belief tables.

The following quantities are important for characterizing computation in the clique tree [46].

Definition 10 (Clique tree parameters) *Let $\boldsymbol{\Gamma}$ be the set of cliques in the clique tree β''', created from a BN β using tree clustering. The state space size of a clique H in β''', g, is defined as*

$$g = |\Omega_H| = \prod_{X \in H} |\Omega_X|, \tag{2}$$

where X is a node in β. The maximal number of nodes in a clique in β''', h, is defined as

$$h = \sup_{H \in \boldsymbol{\Gamma}} |H|. \tag{3}$$

The total clique tree size (or total state space size) k of β''' is defined as

$$k = \sum_{H \in \boldsymbol{\Gamma}} |\Omega_H|, \tag{4}$$

while ℓ, the maximal clique size (or maximal state space size of a clique) in β''', is

$$\ell = \sup_{H \in \boldsymbol{\Gamma}} |\Omega_H|. \tag{5}$$

A functional notation, for example $k\left(\beta'''\right)$ or $\ell\left(\beta'''\right)$, is sometimes used in this article in order to make β''' explicit. For the optimal (minimal) values, h^*, k^*, and ℓ^* are used for h, k, and ℓ respectively. When the above parameters are considered random variables, the letters $\mathbb{G}$, $\mathbb{H}$, $\mathbb{K}$, and $\mathbb{L}$ are used.

Some of our investigations are restricted to nodes with two states, $S = 2$, and in this case the state space size g of a clique (2) simplifies to

$$g = |\Omega_H| = \prod_{X \in H} |\Omega_X| = S^{|H|} = 2^{|H|}.$$

[1] For the sake of simplicity, we generally do not distinguish between (i) the HUGIN algorithm and (ii) the HUGIN system, namely a software implementation of the HUGIN algorithm. In general, this article discusses the HUGIN system in the experimental parts of the article, and the HUGIN algorithm elsewhere.

[2] Some tree clustering algorithms, but not HUGIN, employ an intermediate step right before clique tree construction. This intermediate step creates a junction graph or system of cliques from the triangulated graph.

When all BN nodes have S states the maximal number h of nodes in a clique is the same as the number of nodes in a clique of maximal size ℓ. Consequently, we do not distinguish between these two quantities in this article, even though in general they need to be kept distinct. It is also easy to see that

$$\ell = S^h = 2^h. \tag{6}$$

When a BN is highly connected, as in some of the networks considered in this article, the cliques in the clique tree become very large, thus making tree clustering inference slow. A crucial step in the process of creating a clique tree from a Bayesian network is triangulation — the construction of a triangulated moral graph β''. Triangulation determines g, h, k, and ℓ. Optimal triangulation including the computation of h^* is unfortunately known to be NP-hard, but there are heuristic algorithms such as MinimumFillInWeight, MinimumFillInSize, MinimumCliqueWeight, MinimumCliqueSize that compute upper bounds on h^*, k^*, and ℓ^* and in practice perform triangulation quite well [33, 37, 42].

Hugin was introduced as a belief updating algorithm [46], and was later extended to MPE computation (belief revision) [18], using essentially the same clique tree β''' in both cases. Thus, in the Hugin approach, computing marginal distributions and computing MPEs are closely related. There are two main algorithmic differences: First, when computing an MPE $\boldsymbol{x}^* \in \boldsymbol{X}^*$, maximization is performed, while in belief updating, summation is performed. For our purposes, this step has essentially the same performance in both cases. Second, Hugin belief revision must, in cases of multiple most probable explanations $|\boldsymbol{X}^*| > 1$, perform propagation several times [50]. On the other hand, one propagation is sufficient in Hugin belief updating. Of these two differences, the latter has a more significant impact on the computational cost of propagation and is further discussed in Section 5.6.

3.2 Inference, Maximal Clique Size, and Treewidth

The complexity of most exact Bayesian network inference algorithms — including tree clustering algorithms, conditioning algorithms, and elimination algorithms — has been found to depend on treewidth ϖ^* or on optimal maximal clique size h^*, where $\varpi^* = h^* - 1$ [20, 46]. Time and space complexity for tree clustering is exponential in the treewidth of the clique tree. Conditioning algorithms [16, 17, 22, 32, 57, 58, 64] transform a multiply connected graph into several singly connected graphs by introducing cycle cutsets, and perform computations over each singly connected graph. The time complexity of conditioning, for a minimal cycle cutset of size c, has been bounded from below by treewidth ϖ^* in the inequality $\varpi^* \leq c + 1$ [8]. The time complexity of elimination is closely related to that of tree clustering, and also depends on the treewidth ϖ^* [5, 21]. Finally, there are hybrid algorithms — combining tree clustering, conditioning, and elimination — that trade off space- and time-complexity and again there is a dependency on treewidth [17, 20]. In one such hybrid algorithm it is possible to move from $O(n)$ space and $O\left(n \exp(\varpi^* \log n)\right)$ time to $O(n \exp(\varpi^*))$ space and $O\left(n \exp(\varpi^*)\right)$ time in a gradual fashion [17].

Unfortunately, computing treewidth ϖ^* for a graph is in itself computationally hard. In particular, the problem of determining whether the treewidth of a given graph is bounded by an integer k has been shown to be NP-complete [6]. However, it is possible to empirically establish lower bounds for treewidth as well as upper bounds, computed using heuristics, for treewidth in polynomial time [44]. Optimal triangulation is closely related to computing treewidth, and triangulation heuristics play a key role in tree clustering algorithms, as discussed above and further investigated in the experiments in Section 6.

4 Bayesian Networks for Experimentation and Benchmarking

There are several ways to experiment with inference algorithms using BNs. In this section we discuss the two main classes of BNs used for experimentation in the literature: application BNs and randomly generated BNs. While essential, we argue that application BNs also have limitations. It is non-trivial, for application BNs, to understand how different BN parameters interact and contribute to inference complexity. Using randomly generated BNs, we can start addressing those problems, but still need to make sure that the BNs generated are such that inference hardness can — at least to some extent — be controlled and predicted.

In Section 4.1 we define subsets of the set of Bayesian networks. Section 4.2 briefly discusses BNs from applications. In Section 4.3 we discuss our two approaches to randomly generating Bayesian networks, the BPART algorithm and the MPART algorithm.

	Child-regular set $\mathcal{U}_{\mathrm{CR}}$: Non-leaf nodes have the same number of children	**Child-irregular set $\mathcal{U}_{\mathrm{CI}}$:** Non-leaf nodes typically have different number of children
Parent-regular set $\mathcal{U}_{\mathrm{PR}}$: Non-root nodes have the same number of parents	**Class A: Parent-regular and Child-regular** $\mathcal{U}_A := \mathcal{U}_{\mathrm{PR}} \cap \mathcal{U}_{\mathrm{CR}}$ Classical Gallager codes [28, 49] Regular kCNF Read-ℓ formulas Regular multipartite graphs	**Class B: Parent-regular and Child-irregular** $\mathcal{U}_B := \mathcal{U}_{\mathrm{PR}} \cap \mathcal{U}_{\mathrm{CI}}$ Modern Gallager codes [48, 49] Irregular kCNF formulas [55] Irregular multipartite graphs Biological pedigrees [68]
Parent-irregular set $\mathcal{U}_{\mathrm{PI}}$: Non-root nodes typically have different number of parents	**Class C: Parent-irregular and Child-regular** $\mathcal{U}_C := \mathcal{U}_{\mathrm{PI}} \cap \mathcal{U}_{\mathrm{CR}}$ Read-ℓ formulas	**Class D: Parent-irregular and Child-irregular** $\mathcal{U}_D := \mathcal{U}_{\mathrm{PI}} \cap \mathcal{U}_{\mathrm{CI}}$ Many application BNs Mixed CNF formulas

Table 1: An informal presentation of directed graphs, including Bayesian networks, along the two orthogonal dimensions of child-regularity and parent-regularity, leading to the following four classes. Class A: Parent-regular and child-regular; Class B: Parent-regular and child-irregular; Class C: Parent-irregular and child-regular; and Class D: Parent-irregular and child-irregular. Class D is unconstrained.

4.1 Classes of Bayesian Networks

It turns out that the regularity of a BN's underlying graph varies between applications and also has a major impact on maximal clique size and thus on BN inference times. In order to discuss the effect of graph regularity on BN inference, we introduce the following terminology which applies to directed graphs in general.

Definition 11 (Regularity) *Let $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{E})$ be a directed graph. If any two nodes $X, Y \in \boldsymbol{X}$ with in-degrees $i(X), i(Y) > 0$ have the same number of parents $i(X) = i(Y)$ then we say that $\boldsymbol{G}$ is parent-regular, or $\boldsymbol{G} \in \mathcal{U}_{PR}$. If $\boldsymbol{G}$ is not parent-regular then it is parent-non-regular, or $\boldsymbol{G} \in \mathcal{U}_{PN}$, with $\mathcal{U}_{PR} \cap \mathcal{U}_{PN} = \varnothing$. If $\boldsymbol{G}$ is either parent-regular or parent-non-regular then it is parent-irregular, or $\boldsymbol{G} \in \mathcal{U}_{PI}$, where $\mathcal{U}_{PI} := \mathcal{U}_{PR} \cup \mathcal{U}_{PN}$. If any two nodes $X, Y \in \boldsymbol{X}$ with out-degrees $o(X), o(Y) > 0$ have the same number of children $o(X) = o(Y)$ then $\boldsymbol{G}$ is child-regular, or $\boldsymbol{G} \in \mathcal{U}_{CR}$. If $\boldsymbol{G}$ is not child-regular then it is child-non-regular, or $\mathcal{U}_{CN}$, with $\mathcal{U}_{CR} \cap \mathcal{U}_{CN} = \varnothing$. If $\boldsymbol{G}$ is either child-regular or child-non-regular then it is child-irregular: $\boldsymbol{G} \in \mathcal{U}_{CI}$ where $\mathcal{U}_{CI} := \mathcal{U}_{CR} \cup \mathcal{U}_{CN}$.*

We note that the above definitions allow a non-trivial graph $\boldsymbol{G}$ to be both parent regular and parent-irregular or child-regular and child-irregular: $\boldsymbol{G} \in \mathcal{U}_{\mathrm{PI}} \cap \mathcal{U}_{\mathrm{PR}}$ or $\boldsymbol{G} \in \mathcal{U}_{\mathrm{CI}} \cap \mathcal{U}_{\mathrm{CR}}$, where $\mathcal{U}_{\mathrm{PI}} \cap \mathcal{U}_{\mathrm{PR}}$ as well as $\mathcal{U}_{\mathrm{CI}} \cap \mathcal{U}_{\mathrm{CR}}$ are non-empty. This turns out to simplify the construction algorithms along with their analysis as we will see in Section 4.3.4. We will also see that the probability of a BN that is an element of $\mathcal{U}_{\mathrm{CI}}$ is also an element of $\mathcal{U}_{\mathrm{CR}}$, in other words the size of $\mathcal{U}_{\mathrm{CI}} \cap \mathcal{U}_{\mathrm{CR}}$, is extremely small for the constructions and parameter values considered there.

Based on the regularity concepts introduced in Definition 11, the following four classes of directed graphs (and BNs) are identified.

Definition 12 (Class A, Class B, Class C, Class D directed graphs) *A directed graph $\boldsymbol{G}$ is a Class A (or "regular") graph if it is child-regular and parent-regular: $\boldsymbol{G} \in \mathcal{U}_A$ where $\mathcal{U}_A := \mathcal{U}_{CR} \cap \mathcal{U}_{PR}$. If $\boldsymbol{G}$ is child-irregular and parent-regular then it is a Class B (or "irregular") graph: $\boldsymbol{G} \in \mathcal{U}_B$ where $\mathcal{U}_B := \mathcal{U}_{CI} \cap \mathcal{U}_{PR}$. If $\boldsymbol{G}$ is child-regular and parent-irregular then it is a Class C graph: $\boldsymbol{G} \in \mathcal{U}_C$ where $\mathcal{U}_C := \mathcal{U}_{CR} \cap \mathcal{U}_{PI}$. If $\boldsymbol{G}$ is child-irregular and parent-irregular then it is a Class D (or unconstrained) graph: $\boldsymbol{G} \in \mathcal{U}_D$ where $\mathcal{U}_D := \mathcal{U}_{CI} \cap \mathcal{U}_{PI}$. The sets of all Class A, Class B, Class C and Class D bipartite graphs (BNs) are respectively introduced as follows: $\mathcal{B}_A := \mathcal{U}_A \cap \mathcal{B}$, $\mathcal{B}_B := \mathcal{U}_B \cap \mathcal{B}$, $\mathcal{B}_C := \mathcal{U}_C \cap \mathcal{B}$, and $\mathcal{B}_D := \mathcal{U}_D \cap \mathcal{B}$.*

Table 1 summarizes some families of BNs, including BNs for error correction coding [27, 28, 48, 49], and how they can be classified into our four classes $\mathcal{U}_A$, $\mathcal{U}_B$, $\mathcal{U}_C$, and $\mathcal{U}_D$. The following example and Figure 1

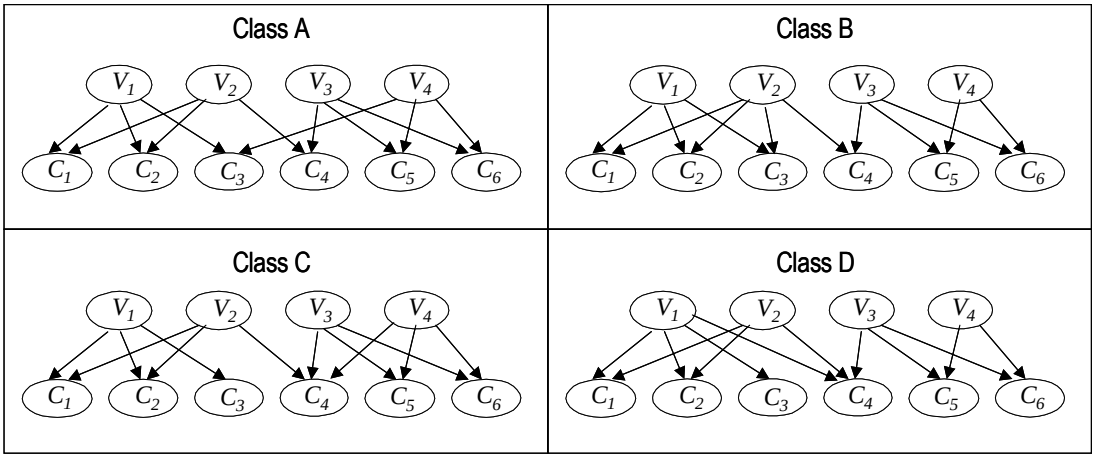

Figure 1: Examples of Class A, Class B, Class C, and Class D bipartite Bayesian networks (BNs). The Class A BN is parent-regular and child-regular (a "regular" BN); the Class B BN is parent-regular and child-non-regular and thus child-irregular (an "irregular" BN); the Class C BN is parent-non-regular (thus parent-irregular) and child-regular; and the Class D BN is parent-non-regular (thus parent-irregular) as well as child-non-regular (thus child-irregular) and therefore unconstrained.

illustrate the classes of networks presented in Definition 12 and Table 1.

Example 13 *Figure 1 contains examples of* Class A, Class B, Class C*, and* Class D *BNs.*

In propositional logic, the notion of read-ℓ means that a variable is used or "read" ℓ times in the clauses of a formula. This concept is used in Table 1. Also, the development in information theory from classical Gallager codes to modern Gallager codes fits into our framework as presented in Table 1. Gallager's original codes [28] can be encoded as Class A BNs according to our terminology. Modern Gallager codes [48], on the other hand, correspond to Class B BNs. Table 1 also includes biological pedigree BNs [68], which are typically also Class B BNs. In a BN representing a pedigree, non-root nodes typically have two parents, but the number of children per non-leaf node can vary.

Note that regularity can easily be made more gradual than in the framework presented in Table 1. For example, one could use the variance in in-degree and out-degree as a measure of regularity. With this more general measure, high variance means irregular, low variance means regular. In this article, however, we are concerned with two extreme cases and leave other variations for future work. We investigate, under a minor relaxation introduced in Definition 16, the effect of regularity by considering Class A BNs as well as Class B BNs. We denote the former regular and the latter irregular BNs when there is no chance of confusion.

4.2 Bayesian Networks from Applications

BN inference algorithms may be studied empirically by evaluating their performance on one or several BNs from applications [41, 51, 56]. For example, BNs may be taken from Friedman's Bayesian Network Repository at `http://www.cs.huji.ac.il/labs/compbio/Repository/`. Application BNs are obviously very important when performing experimental studies. However, we believe that it is difficult to understand the performance of a BN inference algorithm by studying only application BNs. First, there is a problem of dimensionality in that application BNs vary considerably in their many topological and distributional parameters. It is therefore unclear how much one can learn from pooling BNs from different applications. Second, the inference times vary significantly between application BNs, and there is in general no clear correlation between any of the BN parameters and the inference times [51]. Third, the number of BNs per application is most often very small — typically there is one BN per application. Restricting oneself to BNs from one application is thus not desirable: It is very difficult to obtain good statistics using small samples.

There is also a more fundamental limitation associated with the use of application BNs as the "gold standard" for performance. Some application BNs might have been fine-tuned to give adequate performance using existing inference algorithms. It seems very valuable to construct BNs that are not biased in this way, in order to thoroughly characterize existing algorithms as well as lay the groundwork for novel algorithms and more challenging applications.

4.3 Bayesian Network Generation Algorithms

A potential solution to some of the limitations associated with using application BNs for empirical research is to randomly generate BN instances [7, 17, 34, 41, 56, 69, 70]. Using randomly generated BNs, one can create as many BNs as needed to provide a significant evaluation. This approach reduces the problem of dimensionality, since one may vary the BNs generated along just one or a few dimensions at a time.

Issues related to randomly generating BNs are addressed in the remainder of this section. In Section 4.3.1 we present the parameters we have used to (partly) control the process of randomly generating Bayesian networks. In Section 4.3.2, Section 4.3.3, and Section 4.3.4 we present the BPART algorithm used to generate a certain class of bipartite networks, for which the mapping from the satisfiability problem (SAT) is fairly direct. In Section 4.3.5 we discuss the MPART construction, which we show to be related to the BPART construction and, as a result, can be studied from a similar point of view. In Section 4.3.6 we discuss the MPART and BPART constructions as well as related work.

4.3.1 Input Parameters for Bayesian Network Generation Algorithms

Many parameters might be varied when randomly generating Bayesian networks. The following parameters, which cover both topological and distributional issues and whose impact on inferential hardness for tree clustering is further discussed in Section 5, correspond to the input parameters of the BPART and MPART algorithms presented later in this section:

- Number of root nodes V in BN: The range of this integer is 1 to ∞; the default value is 30. For the special case of SAT-like BNs (formally defined in Section 4.3.2), root nodes correspond to variables in conjunctive normal form (CNF) formulas.

- Number of non-root nodes C in BN: The range of this integer is 0 to ∞; the default value is 90. For SAT-like BNs (Section 4.3.2), non-root nodes are leaf nodes and correspond to clauses in CNF formulas.

- Conditional probability table (CPT) type Q and F for BN root and non-root nodes respectively: The choices are deterministic (`or`, `and`, and `xor`), `uniform`, and `random`; the default value for root nodes is `uniform` while it is `or` for non-root nodes. Section 5.6 contains further discussion of CPTs. For experimentation in this article, `or` non-root CPTs and `uniform` root CPTs are employed except in Section 6.5 where all CPTs are `random`.

- Child-regularity R of BN: The choices are Class B child-irregular ($R =$ **false**) or Class A child-regular ($R =$ **true**). The default value is child-irregular, or $R =$ **false.** Four classes Class A, Class B, Class C, and Class D of BNs of varying regularity have been identified; in this article we consider parent-regular BNs in detail. Section 5.4 further analyzes the child-regular case $R =$ **true**; Section 6.4 presents experimental results for $R =$ **true**, the remaining experiments in Section 6 focus on the child-irregular case and use $R =$ **false**.

- The number of states for a node X in the BN, $S = |\Omega_X|$. The range of this integer is 1 to ∞; the default value is $S = 2$ (boolean nodes). For boolean nodes, the states may without loss of generality be called 0 and 1 or **false** and **true** respectively. Experiments in Section 6 are restricted to $S = 2$.

- The in-degree or number of parents P for each non-root node X in the BN: Given our focus on parent-regular BNs, the number of parents P for each non-root node X is a parameter: $P = i(X) = |\Pi_X|$. The range of this integer is $1 \leq P \leq V$; the default value is $P = 3$. Experiments in Section 6 use $P = 3$ except in Section 6.5 where $P = 2$ is used.

```
BPART(Q, F, V, C, S, R, P)
 Input:  Q   conditional probability table (CPT) type, root nodes
         F   CPT type, non-root nodes
         V   number of "variables" (root nodes)
         C   number of "clauses" (leaf or non-root nodes)
         S   number of states per node
         R   create regular BN - true or false
         P   number of parents of clauses (non-root nodes)
 Output: β   Bayesian network
begin
    β ← CreateBN()
    AddLayer(β, V, 0, S, false) {First, add layer of root nodes - V variables}
    AddLayer(β, C, P, S, R) {Second, add layer of child nodes - C clauses}
    for i ← 1 to V + C
        node ← GetNode(β, i)
        if RootNode(node) then
            SetDistribution(node, S, Q) {Set CPT of root node}
        then
            SetDistribution(node, S, F) {Set CPT of non-root node}
        end
    end
    return β
end
```

Figure 2: The BPART algorithm for constructing synthetic, bipartite BNs. The input parameters *Q, F, V, C, S, R* and *P* are used to create different variants of BPART networks. The BPART algorithm creates classical, irregular SAT-like BNs when it is invoked with the paramenters Q = `uniform`, F = `or`, $S = 2$, R = **false**, and $P = 2$; in other words as follows: BPART*(*`uniform`*,* `or`*, V, C, 2,* **false***, 2)*.

In the following we will always assume that the constraints for V, C, Q, F, R, S, and P as presented above are all satisfied. As an example, the default values $V = 30$, $C = 90$, Q = `uniform`, F = `or`, R = **false**, $S = 2$, and $P = 3$ make up a valid set of input parameters. These default values give a specific signature for, say, the BPART algorithm, namely BPART(`uniform`, `or`, 30, 90, *2,* **false**, *3*).

The following quantities can easily be derived from the input parameters presented above:

- The total number of BN nodes is $N = C + V$.
- From V and C the C/V-ratio can be obtained: The range is 0 to ∞; the default value is $C/V = 3$. Since $V \geq 1$, the C/V-ratio is always well-defined. See Section 5.2 for further discussions of the C/V-ratio; experiments in Section 6 use C/V-ratios varying from $C/V = 0.75$ to $C/V = 3.4$.
- Given C and P, the total number of BN edges is: $E = C \times P$, giving $E/V = C \times P/V$. Since $V \geq 1$, the E/V-ratio is always well-defined. The E/V-ratio generalizes the C/V-ratio; we generally fix P and use C/V-ratio in this article but E/V shows up in analytical results in Section 5.4.2.

To be consistent with the existing research literature on randomly generating problem instances in the areas of satisfiability and constraint satisfaction, we use upper-case V to denote the number of root nodes and upper-case C to denote the number of non-root nodes in a BN. Neither V nor C are nodes or random variables in a Bayesian network, even though upper-case letters are also used to represent these concepts.

4.3.2 The BPART Network Generation Algorithm: A Synthetic Bipartite Construction

For many NP-hard problems, simply generating random instances in an undiscriminating fashion has sometimes resulted in fairly easy problem instances [4, 11, 25]. This problem has been addressed in the context of

Function name	Description
ADD($X, \boldsymbol{X}$)	Adds node X to the set of nodes $\boldsymbol{X}$; sets $\boldsymbol{X} \leftarrow \{X\} \cup \boldsymbol{X}$.
ADDPARENT(X, Y)	If possible, adds Y to Π_X and returns **true**, else returns **false**.
CHOOSEFEWESTCHILDREN($\boldsymbol{Y}$)	Randomly chooses and returns appropriate node among nodes $\boldsymbol{Y}$.
CREATEBN()	Returns a new, empty BN.
CREATENODE(β)	Returns a newly created node in BN β.
GETLEAFNODES(β)	Returns nodes without any children in BN β.
GETNODE(β, i)	Returns the i-th node in the BN β, assuming some node ordering.
GETNUMBEROFCHILDREN(X)	Returns the number of children for BN node X.
GETNUMBEROFNODES($\boldsymbol{X}$)	Returns the number of BN nodes in the set of nodes $\boldsymbol{X}$.
RANDOMINT(L, H)	Returns, uniformly at random, a natural number in the interval $[L, H]$.
ROOTNODE(X)	Returns **true** if X is a root node, else returns **false**.
SETDISTRIBUTION(X, S, F)	Sets distribution of BN node X, with S states, to CPT of type F.
SETNUMBEROFSTATES(X, S)	Creates S states in the input BN node X.

Table 2: Algorithms used by the Bayesian network construction algorithms. Possible values of the parameter F in SETDISTRIBUTION and details of CHOOSEFEWESTCHILDREN are discussed in the text.

satisfiability, in the seminal work of Mitchell, Selman, and Levesque [55], where it was shown how to generate hard instances for 3SAT. Here we show how these ideas can be generalized and used to generate, as it turns out, hard instances for belief revision and belief updating when using tree clustering. In this section we present our approach to randomly generating bipartite BNs of varying hardness.

Figure 2 presents our BPART construction algorithm which generates SAT-like Bayesian networks as a special case. When BPART is invoked using the following signature, it creates SAT-like BNs.

Definition 14 (SAT-like BN) *Let* $\beta \leftarrow$ BPART*(**uniform**, `or`,* V, C, 2, ***false**,* P*). The BN* β *is SAT-like.*

The BPART algorithm can also construct more general BNs as reflected in the following definition.

Definition 15 (BPART BNs) *The set of all BNs generated by* BPART *is defined as* $\mathcal{U}_{\text{BPART}} = \{\beta \mid \beta \leftarrow \text{BPART}(Q, F, V, C, S, R, P)\}$. *The set of regular and irregular* BPART *BNs are respectively denoted* $\mathcal{U}^r_{\text{BPART}} = \{\beta \mid \beta \leftarrow \text{BPART}(Q, F, V, C, S,$ **true**$, P)\}$ *and* $\mathcal{U}^i_{\text{BPART}} = \{\beta \mid \beta \leftarrow \text{BPART}(Q, F, V, C, S,$ **false**$, P)\}$.

Building on an existing construction [14,60], the basic idea is to generalize from a conjunctive normal form (CNF) formula and generate a Bayesian network for which an MPE corresponds to a satisfying assignment of the formula. For a SAT-like BN, given a 3CNF formula $f = \bigwedge_{i=1}^{m} C_i$ with clauses $C_i = X_{i_1} \vee X_{i_2} \vee X_{i_3}$, one can construct a bipartite Bayesian network, in which one layer of nodes (the root nodes $\boldsymbol{X}$) corresponds to the variables and a second layer (the leaf nodes $\boldsymbol{C}$) corresponds to the clauses. A variable $X_j \in \boldsymbol{X}$ has an edge directed toward $C_i \in \boldsymbol{C}$ iff X_j occurs in the clause C_i. Clause nodes are clamped during inference. The conditional probability tables are set so that $\Pr(f = 1) > 0$ iff the assignment to the X_j's satisfies the 3CNF formula. It is easy to see that this happens if for all i, the conditional probability table associated with the node C_i simulates an `or` gate of three inputs. To generate a BN that corresponds to a non-monotone CNF, the CPTs need to be generalized accordingly in a straight forward way. It is easy to verify that an MPE $\boldsymbol{x}^*$ — an assignment of values to the X_j's — has a positive probability iff it satisfies the corresponding 3CNF formula [14, 60]. There may be many satisfying assignments, all with the same probability, making them all MPEs.

There are several ways to generate random BNs corresponding to CNF formulas. Our BPART approach, presented in Figure 2, is based on the following policy: work with V variables and C clauses; generate the clauses by selecting variables uniformly into clauses and negate each variable with probability $p = 0.5$ [55]. Subroutines used by BPART include ADDLAYER (see Figure 3), CHOOSEFEWESTCHILDREN, and SETDISTRIBUTION (see Table 2). Turning to BPART in Figure 2, CREATEBN first creates a new, empty BN β. ADDLAYER($\beta, V, 0, S,$ **false**) then adds to β a layer of V root nodes, where each root node has S states. Next, ADDLAYER(β, C, P, S, R) adds to β the C non-root (or leaf) nodes. Each leaf node has P

```
AddLayer(β, M, P, S, R)
 Input:  β   BN to which new layer is added
         M   number of new nodes to create in the new layer
         P   number of parent nodes to assign to a new node
         S   number of states per new node
         R   true if regular BN layer is to be created, else false
 Output: β   Bayesian network with layer of nodes added
begin
    parents ← GetLeafNodes(β)
    for i ← 1 to M
        X_i ← CreateNode(β)
        SetNumberOfStates(X_i, S)
        c ← 0
        while c < P {The new node X_i is given parents}
            if not R then {The irregular case R = false}
                j ← RandomInt(1, GetNumberOfNodes(parents))
                parent ← parents[j]
            else {The regular case R = true}
                parent ← ChooseFewestChildren(parents) {Pick parent with fewest children}
            end
            success ← AddParent(X_i, parent) {false if parent among X_i's parents already}
            if success then c ← c + 1 end
        end
    end
    return β
end
```

Figure 3: The function AddLayer adds a layer consisting of M nodes to the Bayesian network β. AddLayer is invoked by BPART.

parents, chosen among the V root nodes. The nature of a leaf node's parent selection process, which takes place in AddLayer, is controlled by the regularity parameter $R \in \{\textbf{true}, \textbf{false}\}$. Figure 3 presents how a layer in the BPART construction is added by the AddLayer procedure. AddLayer, which returns the BN β with a new layer of M nodes added to it, works differently for the regular ($R = \textbf{true}$) and the irregular ($R = \textbf{false}$) cases as discussed further in Section 4.3.3 and Section 4.3.4 respectively.

The CPTs of all nodes are constructed in the top level of BPART (Figure 2). Non-root node CPTs are determined by the input parameter F, with $F \in \{\texttt{or}, \texttt{and}, \texttt{xor}, \texttt{uniform}, \texttt{random}\}$. Similarly, root node CPTs are determined by $Q \in \{\texttt{or}, \texttt{and}, \texttt{xor}, \texttt{uniform}, \texttt{random}\}$. In a deterministic CPT (where CPTs are `or`, `and`, or `xor`), all entries are either 0 or 1. In a `uniform` CPT of a node X with $|\Omega_X| = S$ states, the probability mass for a node state for a given parent instantiation is $1/S$. In a `random` CPT, the probabilities are first picked from a uniform random $U(0, 1)$ distribution, and then normalized to make sure that the conditional probabilities for a given parent instantiation sum to 1. These types cover the CPTs that are required to provide an exact mapping from SAT problems to corresponding BNs as well as idealizations of CPTs that might occur in some applications. In Section 5.6, we discuss the relationship between these CPTs and hardness for inference.

There are two related but slightly different perspectives on BPART's signature. The first perspective is to regard it as a parametrized probability distribution $\mathbb{B}$ over BNs — as in $\mathbb{B} \sim$ BPART(`uniform`, `or`, 30, 90, *2*, **false**, *3*). The second perspective is to regard the signature as an assignment that generates one sample β from the probability distribution $\mathbb{B}$: The assignment $\beta \leftarrow$ BPART(`uniform`, `or`, 30, 90, *2*, **false**, *3*) will, under reasonable assumptions regarding the periodicity of the pseudo-random number generator, create a different BN β each time BPART is invoked. In general, the former perspective is taken in this section as well as in Section 5, while the latter perspective is taken in the experimental part of the article, Section 6, where we sample from $\mathbb{B}$.

The BPART and MPART construction algorithms presented in this section can construct irregular or regular BNs. The setting $R =$ **true** gives a regular BN, while $R =$ **false** gives an irregular BN. We now discuss these two cases separately.

4.3.3 The BPART Regular Case: $R =$ true

In the AddLayer procedure, if the new layer of nodes is to be regular (so $R =$ **true**), the ChooseFewestChildren procedure is invoked as shown in Figure 3. ChooseFewestChildren selects a node X among its input nodes $\boldsymbol{Y}$ such that no other input node has fewer children. By doing this, the algorithm ensures that the parent layer is child-regular or "close to" child-regular. More formally, it is ensured that the BN has an underlying relaxed Class A graph, as defined below.

We now introduce concepts closely related to Definition 11 and Definition 12, in order to make the notion of regularity more widely applicable without losing the essence of regularity.

Definition 16 (Relaxed Class A directed graph) *Consider a directed graph $\boldsymbol{G}$ with V non-leaf nodes and E edges. If any non-leaf node X_i where $i \in \{1, \ldots, V\}$ has $o(X_i) = \lceil \frac{E}{V} \rceil$ or $o(X_i) = \lfloor \frac{E}{V} \rfloor$ children, then the notation $o(X_i) \approx \frac{E}{V}$ is used and we say that $\boldsymbol{G}$ is relaxed child-regular. If a directed graph $\boldsymbol{G}$ is relaxed child-regular and parent-regular, $\boldsymbol{G}$ is a relaxed Class A directed graph. The set of relaxed Class A graphs is $\mathcal{U}_{A^*}$; the set of bipartite relaxed Class A graphs is $\mathcal{B}_{A^*} := \mathcal{U}_{A^*} \cap \mathcal{B}$.*

The parameter value $R =$ **true** controls the number of children (the out-degree) of BPART root nodes as presented in the following theorem and corollary.

Theorem 17 *Suppose the* BPART *algorithm is called with $R =$ **true**. If F edges have been distributed during the BN construction process of* BPART*, then $o(X) \approx \frac{F}{V}$ for any root node X.*

Proof. Suppose it were not the case. This means that there exists at least one root node X where $|\Psi_X| > \lceil \frac{F}{V} \rceil$ or $|\Psi_X| < \lfloor \frac{F}{V} \rfloor$. Consider the case $|\Psi_X| > \lceil \frac{F}{V} \rceil$. This means that there exists at least one root node Y with $\lfloor \frac{F}{V} \rfloor$ (or fewer) children. Assume, for the purpose of contradiction, that all root nodes excluding X have $\lceil \frac{F}{V} \rceil$ children. Since $\lceil \frac{F}{V} \rceil \geq \frac{F}{V}$, the number of children (edges) for these $(V-1)$ root nodes is

$$(V-1)\left\lceil \frac{F}{V} \right\rceil \geq (V-1)\frac{F}{V} = F - \frac{F}{V}.$$

Since the number of children for X is $|\Psi_X|$, the total number of edges in the BN is $F - \frac{F}{V} + |\Psi_X|$, which is not possible since $|\Psi_X| > \lceil \frac{F}{V} \rceil \geq \frac{F}{V}$ and there is a contradiction. Consequently, there exists at least one root node Y with $\lfloor \frac{F}{V} \rfloor$ edges. However, this fact contradicts how ChooseFewestChildren operates. ChooseFewestChildren always picks, for a leaf node, a parent with fewest children. However, the fact that X now has $|\Psi_X| > \lceil \frac{F}{V} \rceil$ implies that at some earlier stage X was picked by ChooseFewestChildren while having $\lceil \frac{F}{V} \rceil$ children. At the same time, there must have been a node Y with $|\Psi_Y| \leq \lfloor \frac{F}{V} \rfloor$, and so ChooseFewestChildren should have chosen Y. This is a contradiction. The proof for the case $|\Psi_X| < \lceil \frac{F}{V} \rceil$ is similar. ■

Using Theorem 17, we formally characterize the output of the BPART algorithm in Figure 2.

Corollary 18 *With input parameter $R =$ **true**, the* BPART *algorithm creates bipartite relaxed Class A BNs: $\mathcal{U}^r_{\text{BPART}} \subset \mathcal{B}_{A^*}$.*

Proof. Use $F = CP$ in Theorem 17 and apply Definition 16. ■

Unless otherwise noted, we do not distinguish between relaxed Class A BNs and Class A BNs in the following, and denote both as Class A BNs. In particular, we shall say that BPART, given input parameter $R =$ **true**, creates Class A BNs even though this is true, strictly speaking according to Definition 11, only in the special case where $\lceil \frac{CP}{V} \rceil = \lfloor \frac{CP}{V} \rfloor$ (see Corollary 18). The notion of regularity and its impact on inference hardness is further discussed in Section 5.4, and we turn now to the irregular case.

4.3.4 The BPART Irregular Case: $R =$ **false**

If $R =$ **false**, parent nodes are chosen uniformly at random without replacement in the AddLayer procedure (see Figure 3). It is easy to show that bipartite Class B BNs are generated.

Theorem 19 *If $R =$ **false**, the* BPART *algorithm creates bipartite Class B BNs: $\mathcal{U}^i_{\text{BPART}} \subset \mathcal{B}_B$.*

In any BN, the parents of a non-root node must be distinct. Consequently, for a given BPART leaf node, AddLayer selects among the $|\boldsymbol{V}| = V$ root nodes without replacement and we obtain the following.

Theorem 20 (Exact child distribution) *Consider a BN with root nodes $\boldsymbol{V}$ and leaf nodes $\boldsymbol{C}$ created using* BPART *with input parameter $R =$ **false**. For any $X \in \boldsymbol{V}$, let the number of children be $\mathbb{N} = |\Psi_X|$, and let b be the binomial distribution. It is the case that $\mathbb{N} \sim b\,(C, P/V)$.*

Proof. Using the V root nodes, form $\binom{V}{P}$ distinct super-nodes, where each super-node W_i contains P different root nodes $\{V_{i,1}, ..., V_{i,P}\}$, where $V_{i,j} \in \boldsymbol{V}$ for $1 \leq i \leq \binom{V}{P}$ and $1 \leq j \leq P$. Clearly, BPART performs C Bernoulli trials among these $\binom{V}{P}$ super-nodes. One Bernoulli trial is, for an arbitrary node $X \in \boldsymbol{V}$, considered a success if X is an element of the chosen super-node. There are $\binom{V-1}{P-1}$ super-nodes in which X is an element. Consequently, the probability of Bernoulli trial success for X is

$$p = \frac{\binom{V-1}{P-1}}{\binom{V}{P}} = \frac{P}{V},$$

and since there are C trials we obtain a binomial distribution $b(C, P/V)$. ■

In the following theorem we introduce a simplifying assumption, further justified below, that parent selection in AddLayer is made with replacement even though we assumed differently in Theorem 20.

Theorem 21 (Approximate child distribution) *Consider a BN with root nodes $\boldsymbol{V}$ and leaf nodes $\boldsymbol{C}$ created using* BPART *with $R =$ **false**. As an approximation, suppose, for any $Y \in \boldsymbol{C}$, that each of Y's P parents is picked independently and uniformly at random among $\boldsymbol{V}$. For any $X \in \boldsymbol{V}$, let the number of children be $\mathbb{N} = |\Psi_X|$, and let b be the binomial distribution. It is the case that $\mathbb{N} \sim b\,(CP, 1/V)$.*

Proof. Consider a BN β with root nodes $\boldsymbol{V}$ created using the BPART algorithm with input $R =$ **false**. In AddLayer, which is invoked by the BPART algorithm, the i-th leaf node $Y_i \in \boldsymbol{C}$ selects P parent nodes. As an approximation, assume that each parent is picked independently. Now consider one particular root node $X_k \in \boldsymbol{V}$. Call it a success if, for the selection of the j-th parent by the i-th leaf node Y_i, X_k gets picked as a parent node, else call it a failure. Let $\mathbb{I}_{i,j}$ be an indicator random variable for this trial. By assumption, each selection of a root node is an independent Bernoulli trial with probability of success $p = 1/V$, and we obtain a sequence of Bernoulli random variables $\mathbb{I}_{1,1}$, ..., $\mathbb{I}_{1,P}$, ..., $\mathbb{I}_{i,1}$, ..., $\mathbb{I}_{i,P}$, ..., $\mathbb{I}_{C,1}$, ..., $\mathbb{I}_{C,P}$. The number of times that X_k is picked is a random variable $\mathbb{N} = \sum_{i=1}^{C} \sum_{j=1}^{P} \mathbb{I}_{i,j}$. Clearly, $\mathbb{N}$ has a binomial distribution $b\,(n, p)$ with $n = CP$ trials and probability of success $p = 1/V$. ■

The approximating assumption of independence in Theorem 21 is justified as follows. As $V \to \infty$, the random parent selection process in AddLayer approaches drawing independently with replacement, since the probability of picking the same root node twice or more tends to zero.

Having introduced and analyzed the BPART construction, we return to a discussion of Definition 11 and the classes $\mathcal{U}_{\text{CI}}$, $\mathcal{U}_{\text{CR}}$, and $\mathcal{U}_{\text{CN}}$. As an example, suppose for $\beta \in \mathcal{U}^i_{\text{BPART}}$ that each child node has three parents. Clearly, $\beta \in \mathcal{U}_{\text{CI}}$ and typically $\beta \in \mathcal{U}_{\text{CN}}$ also. We say "typically" because it could happen that parents are randomly picked such that the graph ends up being child-regular "by chance", or $\beta \in \mathcal{U}_{\text{CR}}$. As an example, each root node in β could end up with, say, exactly six children, assuming there are twice as many leaf nodes as root nodes. However, as we will see in Theorem 22 and Example 23, the probability of $\beta \in \mathcal{U}_{\text{CR}}$ is extremely small for $\beta \in \mathcal{U}^i_{\text{BPART}}$ when β is generated using parameter values for P, V, and C of the order of magnitude used in the experimental part of this article.

Theorem 22 *Consider a BN $\beta \in \mathcal{U}_{CI}$ with root nodes $\boldsymbol{V}$ and leaf nodes $\boldsymbol{C}$ created using* BPART *with input parameter $R =$ **false**. Suppose for any $Y \in \boldsymbol{C}$ that each of Y's P parents is picked independently and uniformly at random from $\boldsymbol{V}$. Also assume that $k := \frac{CP}{V}$ is an integer. It is the case that*

$$\Pr(\beta \in \mathcal{U}_{CR}) = \frac{CP!}{\left(\frac{CP}{V}!\right)^V} \left(\frac{1}{V}\right)^{CP}. \tag{7}$$

Proof. The desired joint distribution over root nodes $\boldsymbol{V}$ is clearly multinomial, giving

$$\begin{aligned} \Pr(\beta \in \mathcal{U}_{\mathrm{CR}}) &= \Pr(\mathbb{N}_1 = k, \ldots, \mathbb{N}_V = k) \\ &= \frac{n!}{(k!)^V} \times p^n. \end{aligned}$$

Using Theorem 21 we have $p = 1/V$ and $n = CP$; therefore (7) follows. ■

Unfortunately, it is not easy to see what happens when $CP \to \infty$ in (7). However, by putting $n := CP$ and using Stirling's formula to approximate the factorial function in (7), we get

$$\begin{aligned} \Pr(\beta \in \mathcal{U}_{\mathrm{CR}}) &\approx \frac{\sqrt{2\pi n} n^n e^{-n}}{\left(\sqrt{2\pi \frac{n}{V}}\right)^V \left(\frac{n}{V}\right)^n e^{-n}} \left(\frac{1}{V}\right)^n \\ &= \sqrt{\frac{V^V}{(2\pi n)^{V-1}}}, \end{aligned} \tag{8}$$

from which it is easy to see that $\lim_{n\to\infty} \Pr(\beta \in \mathcal{U}_{\mathrm{CR}}) = 0$. In words, when a child-irregular BPART BN $\beta \in \mathcal{U}_{\mathrm{CI}}$ is generated, β is also child-non-regular with "high" probability $\Pr(\beta \in \mathcal{U}_{\mathrm{CN}}) = 1 - \varepsilon$, where $\varepsilon := \Pr(\beta \in \mathcal{U}_{\mathrm{CR}})$ is characterized by (7) or (8). Here is an example.

Example 23 *Consider a BPART BN $\beta \in \mathcal{U}_{CI}$ constructed using parameters $R =$ **false**, $P = 3$, $V = 30$, and $C = 90$. Using Theorem 22 we obtain $\Pr(\beta \in \mathcal{U}_{CR}) \approx 1.61 \times 10^{-25}$, a very small probability indeed. Using (8) gives the approximation $\Pr(\beta \in \mathcal{U}_{CR}) \approx 2.130 \times 10^{-25}$.*

We now turn to a related but different question. How can the minimum out-degrees of root nodes in an irregular BPART BN β be characterized? This question is answered in Theorem 25 below, using the fact that parents of a leaf node are picked at random in the BPART algorithm. Before stating the result, we formally define minimal root node out-degree, which is a random variable $\mathbb{M}$.

Definition 24 (Minimum out-degree) *Let $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ be a set of n BN nodes with randomly distributed children, and let $\mathbb{N}_i \sim o(X_i)$. The minimum out-degree random variable $\mathbb{M}$ is defined as $\mathbb{M} = \min(\mathbb{N}_1, \ldots, \mathbb{N}_n)$ and abbreviated as $\mathbb{M} = \min(\boldsymbol{X})$.*

In the following, $\mathbb{M}^i$ is used for the irregular case ($R =$ **false**), $\mathbb{M}^r$ for the regular case ($R =$ **true**).

Theorem 25 (Expected minimum out-degree in BPART) *Consider a BN β with root nodes $\boldsymbol{V}$ and leaf nodes $\boldsymbol{C}$ created using* BPART *with $R =$ **false**. Suppose, for any $C_j \in \boldsymbol{C}$, that C_j's P parents are picked independently and uniformly at random among $\boldsymbol{V}$. Let $\mathbb{M}^i = \min(\boldsymbol{V})$. The expectation $E(\mathbb{M}^i)$ is*

$$E(\mathbb{M}^i) = \sum_{j=1}^{\lfloor \frac{CP}{V} \rfloor} \left(\left(1 - B(j-1; CP, \frac{1}{V})\right)^V \right), \tag{9}$$

where $B(k; n, p)$ is the cumulative binomial distribution function $\Pr(\mathbb{X} \leq k)$, where $\mathbb{X} \sim b(n, p)$.

Proof. The probability that the value of some $\mathbb{N}_i$ is k or greater is

$$\Pr(\mathbb{N}_i \geq k) = 1 - \Pr(\mathbb{N}_i < k) = 1 - B(k-1; n, p), \tag{10}$$

where B is the cumulative binomial distribution. (Recall that Theorem 21 uses the binomial distribution when R = **false**.) Considering all V root nodes in β, for the minimum to be k, all V nodes need to be k or greater: $\Pr\left(\mathbb{M}^i \geq k\right) = \Pr\left(\mathbb{N}_1 \geq k, \mathbb{N}_2 \geq k, \ldots, \mathbb{N}_V \geq k\right)$. Since, as an approximation, root nodes are assumed to be picked independently and uniformly at random, there is independence between $\mathbb{N}_1, \ldots, \mathbb{N}_V$. By introducing $\mathbb{N} = \mathbb{N}_1 = \ldots = \mathbb{N}_V$ as well as (10) we obtain

$$\begin{aligned}\Pr\left(\mathbb{M}^i \geq k\right) &= \prod_{i=1}^{V} \Pr\left(\mathbb{N}_i \geq k\right) \\ &= \left(1 - \Pr(\mathbb{N} \leq k-1)\right)^V \\ &= \left(1 - B(k-1; n, p)\right)^V . \end{aligned} \tag{11}$$

Given a random variable $\mathbb{M}^i$ with possible values $\{0, 1, \ldots, m\}$, the tail sum formula for expectation is

$$E(\mathbb{M}^i) = \sum_{j=1}^{m} \Pr\left(\mathbb{M}^i \geq j\right) \tag{12}$$

which by combining (11) and (12) yields

$$E(\mathbb{M}^i) = \sum_{j=1}^{m} \left(\left(1 - B(j-1; n, p)\right)^V \right).$$

By substituting in $m = \left\lfloor \frac{CP}{V} \right\rfloor$, $n = CP$ and $p = \frac{1}{V}$ we obtain (9) as desired. ■

The following example illustrates Theorem 25 using parameter values from experiments in this article.

Example 26 (Expected minimum out-degree in BPART) *Consider a* BPART *network constructed using input parameters $C = 60$, $V = 30$, $P = 3$, and $R =$* **false**. *Using Theorem 25, we obtain $E(\mathbb{M}^i) \approx 1.72$. If instead we use $C = 102$, while the other parameters stay the same, we obtain $E(\mathbb{M}^i) \approx 4.39$.*

While Equation 9 can be used to compute values for $E(\mathbb{M}^i)$ as shown in Example 26, it is unfortunately not obvious how $E(\mathbb{M}^i)$ changes when the C/V-ratio changes. However, based on Theorem 21 we may use the binomial distribution $b\left(CP, 1/V\right)$ with $\hat{E}(\mathbb{M}^i, a) = \mu - a\sigma$ as an approximation to $E(\mathbb{M}^i)$, where a is the number of standard deviations. A more explicit form is thus obtained:

$$\hat{E}(\mathbb{M}^i, a) = \mu - a\sigma = \frac{CP}{V} - a\sqrt{\frac{CP(V-1)}{V^2}} = \frac{CP - a\sqrt{CP(V-1)}}{V}. \tag{13}$$

The impact of an increasing C/V-ratio can now be observed from Equation 13: For a fixed number of standard deviations a, when the C/V-ratio increases due to an increase in C, it is clear that $\hat{E}(\mathbb{M}^i, a)$ in Equation 13 will increase as well. Table 3 provides, by means of example, some insight into the difference between the irregular and regular cases. The difference in the expectations of the minimal degrees, $E(\mathbb{M}^r)$ versus $E(\mathbb{M}^i)$, is quite dramatic. Table 3 also compares Equation 13 with Equation 9 for $V = 30$, $P = 3$, C varying from 60 to 102, and $a = 1$ or $a = 2$ standard deviations. Here, $E(\mathbb{M}^i)$ is bounded as follows: $\hat{E}(\mathbb{M}^i, 2) < E(\mathbb{M}^i) < \hat{E}(\mathbb{M}^i, 1)$; $\hat{E}(\mathbb{M}^i, 1)$ provides a conservative upper bound.

4.3.5 The MPART Network Generation Algorithm: A Synthetic Multipartite Construction

Are the patterns of hard and easy restricted to the BPART construction? If not, then what is construction-specific, and what is general across constructions? In order to explore these questions, we investigated a different but related algorithm for generating random Bayesian networks, the MPART construction. MPART is closely related to an approach of Kask and Dechter [41]. The procedure they describe can be viewed as follows. Choose these three parameters: N - the number of nodes in the network; V - the number of root nodes and P - the number of parents of a non-root node. Construct a network as follows: Index the nodes from 1 to N, and iterate from the C-th node X_C. At the i-th step, process the i-th node X_i, with $1 \leq i \leq C$, and pick, uniformly at random, P parents among the nodes indexed from $i+1$ to N. Repeat until all C non-root nodes $\{X_1, \ldots, X_C\}$ have been assigned parents.

C/V	2.0	2.2	2.4	2.6	2.8	3.0	3.2	3.4
Class A: $E(\mathbb{M}^r)$	6	6	7	7	8	9	9	10
Class B: $\hat{E}(\mathbb{M}^i, 1)$	3.59	4.07	4.56	5.05	5.55	6.05	6.55	7.06
Class B: $E(\mathbb{M}^i)$	1.72	2.07	2.44	2.81	3.20	3.59	3.99	4.39
Class B: $\hat{E}(\mathbb{M}^i, 2)$	1.18	1.55	1.92	2.31	2.70	3.10	3.51	3.92

Table 3: These results are relevant for BPART BNs with $V = 30, P = 3$, and C varying from 60 to 102. The table shows the expectation of minimum out-degrees $E(\mathbb{M}^r)$ in Class A (regular) BNs and $E(\mathbb{M}^i)$ in Class B (irregular) BNs. For the irregular case, the approximation $\hat{E}(\mathbb{M}^i, b) = \mu - a\sigma$ for $a = 1$ and $a = 2$ standard deviations is also displayed.

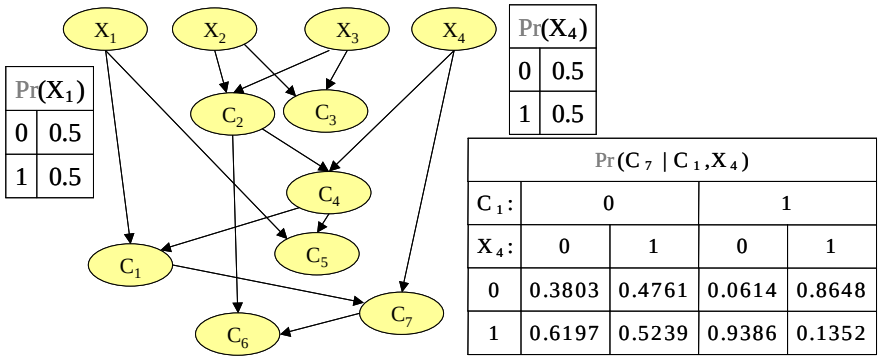

Figure 4: Example Bayesian network generated by the MPART algorithm. This BN has four root nodes $\{X_1, ..., X_4\}$ and seven non-root nodes $\{C_1, ..., C_7\}$. Among the non-root nodes, only C_3 could have appeared in a BPART network; the remaining non-root nodes either have at least one non-root child or parent. For instance, C_1 has non-root C_4 as parent and C_5 as child.

Example 27 *An example* MPART *BN is shown in Figure 4.*

The MPART and BPART constructions are compared in Figure 5. The essential difference is that MPART BNs allow non-root nodes to have other non-root nodes as parents, while this is not allowed in BPART networks. Essentially, the MPART algorithm is similar to the BPART algorithm except a slight variation on BPART's AddLayer algorithm (AddLayer is presented in Figure 3). Instead of BPART's AddLayer statement **"if** success **then** $c \leftarrow c + 1$", MPART sequentially uses the statements "**if** success **then** $c \leftarrow c + 1$" and "Add(node, parents)". The signature of MPART mirrors BPART's signature as presented in Figure 2 and is MPART(Q, F, V, C, S, R, P).

Definition 28 (MPART BNs) *The set of all BNs generated by* MPART *is* $\mathcal{U}_{\text{MPART}} = \{\beta \mid \beta \leftarrow \text{MPART}(Q, F, V, C, S, R, P)\}$. *The sets of regular and irregular* MPART *BNs are respectively defined as* $\mathcal{U}^r_{\text{MPART}} = \{\beta \mid \beta \leftarrow \text{MPART}(Q, F, V, C, S, \textbf{true}, P)\}$ *and* $\mathcal{U}^i_{\text{MPART}} = \{\beta \mid \beta \leftarrow \text{MPART}(Q, F, V, C, S, \textbf{false}, P)\}$.

This multipartite construction algorithm typically creates BNs with a tree-like topology. However, if the number of non-root nodes $C = N - V$ is much smaller than V, the graph may be bipartite or "close to" bipartite. To reflect two different types of MPART BNs, we introduce the following terminology.

Definition 29 *Let an* MPART *BN* β *have root nodes* $\boldsymbol{V}$ *and non-root nodes* $\boldsymbol{X} = \{X_1, \ldots, X_C\}$ *with* $C = |\boldsymbol{C}|$. *We define the bipartite subset of* $\mathcal{U}^i_{\text{MPART}}$ *as* $\mathcal{B}^i_{\text{MPART}} = \mathcal{B} \cap \mathcal{U}^i_{\text{MPART}} = \{\beta \mid \Pi_{X_1} \subseteq \boldsymbol{V}, \ldots, \Pi_{X_C} \subseteq \boldsymbol{V}\} \cap \mathcal{U}^i_{\text{MPART}}$.

Given this terminology, we show in the following theorems that the MPART construction is a generalization of the BPART construction in the case of irregular MPART BNs.

Theorem 30 *Irregular* BPART *BNs are a subset of irregular* MPART *BNs:* $\mathcal{U}^i_{\text{BPART}} \subset \mathcal{U}^i_{\text{MPART}}$.

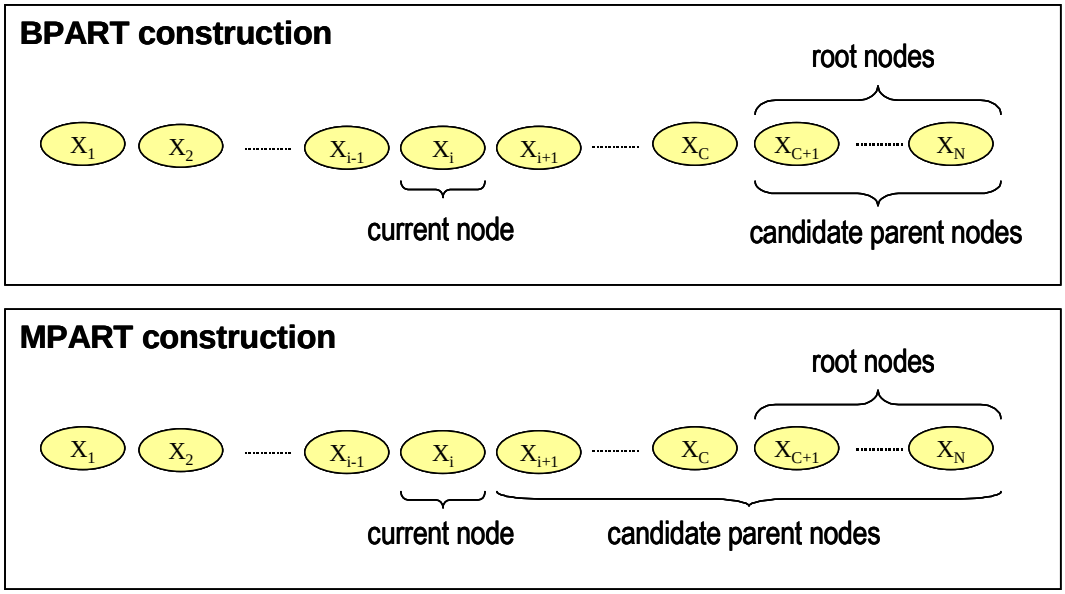

Figure 5: Generation of Bayesian networks using the BPART and MPART constructions. Edges between nodes are, for simplicity, omitted. In both constructions, nodes are treated sequentially. Let X_i be the BN node currently being processed. For X_i, a fixed number of P parent nodes are randomly selected. For BPART, parent nodes for X_i are chosen among the root nodes $\{X_{C+1}, ..., X_N\}$, while they are chosen among the nodes $\{X_{i+1}, ..., X_N\}$ for MPART. The MPART construction is a generalization of the BPART construction: When it so happens that all parent nodes for an MPART BN are picked from the root nodes, the BN could also have been generated by BPART.

Proof. First, we show that if $\beta \in \mathcal{U}^i_{\text{BPART}}$ then $\beta \in \mathcal{U}^i_{\text{MPART}}$, and thus that $\mathcal{U}^i_{\text{BPART}} \subseteq \mathcal{U}^i_{\text{MPART}}$. Assume, for purposes of contradiction, that $\beta \in \mathcal{U}^i_{\text{BPART}}$ and $\beta \notin \mathcal{U}^i_{\text{MPART}}$. For this to happen, there must exist a node X in β with Π_X constructed by BPART such that Π_X can not be constructed by MPART. However, this is not possible, since the candidate set for Π_X in BPART is a subset of that in MPART. This follows from the structure of the MPART construction algorithm, and in particular the statement ADD(node, parents) which adds "node" to the candidate set "parents". This statement in the MPART algorithm is lacking in the BPART algorithm. Second, we exhibit a $\beta \in \mathcal{U}^i_{\text{MPART}}$ such that $\beta \notin \mathcal{U}^i_{\text{BPART}}$, and therefore $\mathcal{U}^i_{\text{BPART}} \not\supseteq \mathcal{U}^i_{\text{MPART}}$. Consider the three-node chain β with nodes $\{X_1, X_2, X_3\}$, and edges $\{(X_1, X_2), (X_2, X_3)\}$. Clearly, $\beta \notin \mathcal{U}^i_{\text{BPART}}$ since β is not bipartite. However, $\beta \in \mathcal{U}^i_{\text{MPART}}$ since β may be constructed by the MPART construction algorithm using the signature MPART(Q, F, V, C, S, R, P) with $V = 1$ and $C = 2$. ■

Theorem 31 *Irregular* BPART *BNs are the same as irregular bipartite* MPART *BNs:* $\mathcal{U}^i_{\text{BPART}} = \mathcal{B}^i_{\text{MPART}}$.

Proof. We need to show that (i) $\mathcal{U}^i_{\text{BPART}} \subseteq \mathcal{B}^i_{\text{MPART}}$ and (ii) $\mathcal{U}^i_{\text{BPART}} \supseteq \mathcal{B}^i_{\text{MPART}}$. From the proof of Theorem 30 we have $\mathcal{U}^i_{\text{BPART}} \subset \mathcal{U}^i_{\text{MPART}}$, which gives (i) as follows:

$$\begin{aligned} \mathcal{U}^i_{\text{BPART}} \cap \mathcal{B} &\subseteq \mathcal{U}^i_{\text{MPART}} \cap \mathcal{B} \\ \mathcal{U}^i_{\text{BPART}} &\subseteq \mathcal{B}^i_{\text{MPART}} \ . \end{aligned}$$

For (ii), consider a BN $\beta \in \mathcal{B}^i_{\text{MPART}}$. Since β is bipartite, the MPART statement ADD(node, parents) had no effect on the structure of β, and could have been left out. In that case we have, by construction, the BPART algorithm and clearly $\beta \in \mathcal{U}^i_{\text{BPART}}$. ■

Finally, we provide the probability that a randomly generated irregular MPART BN will be bipartite, and in particular an irregular BPART BN.

Theorem 32 *Let $\beta \in \mathcal{U}^i_{\text{MPART}}$. For $P \geq 1$ and $C \geq 1$, the probability of the event $\beta \in \mathcal{B}^i_{\text{MPART}}$ is*

$$\Pr\left(\beta \in \mathcal{B}^i_{\text{MPART}}\right) = \prod_{i=1}^{C} \prod_{j=0}^{P-1} \left(\frac{V-j}{C+V-i-j}\right).$$

Proof. Let $\boldsymbol{C}$ be the set of C non-root nodes and let $X_i \in \boldsymbol{C}$ be the node in β currently processed by the MPART algorithm. For X_i, parents are picked among $\{X_{i+1}, \ldots, X_C\} \cup \{X_{C+1}, \ldots, X_N\}$; the non-root nodes already processed and all root nodes. Clearly, X_i's parents Π_{X_i} must all be picked among the root nodes $\boldsymbol{V} = \{X_{C+1}, \ldots, X_N\}$ for the event $\beta \in \mathcal{B}^i_{\text{MPART}}$ to take place. For MPART's first pick of a parent Y for X_i there are $k = C - i$ non-root nodes $\{X_{i+1}, \ldots, X_C\}$ to avoid. For $R = \textbf{false}$, MPART's selection distribution is uniform, giving a probability of success for X_i of $\Pr(Y \in \boldsymbol{V}) = \frac{V}{k+V}$. The nodes in Π_{X_i} need to be distinct but are otherwise picked independently and the multiplication principle can be applied, giving

$$\Pr\left(\Pi_{X_i} \subseteq \boldsymbol{V}\right) = \left(\frac{V}{k+V}\right) \times \left(\frac{V-1}{k+V-1}\right) \times \cdots \times \left(\frac{V-P+1}{k+V-P+1}\right) = \prod_{j=0}^{P-1} \left(\frac{V-j}{k+V-j}\right). \tag{14}$$

For $\beta \in \mathcal{B}^i_{\text{MPART}}$ to occur, a condition similar to (14) needs to hold for all C non-root nodes; by Definition 29 we get $\Pr\left(\beta \in \mathcal{B}^i_{\text{MPART}}\right) = \Pr\left(\Pi_{X_1} \subseteq \boldsymbol{V},\ \ldots,\ \Pi_{X_C} \subseteq \boldsymbol{V}\right)$. Since the selection of parents for X_i is independent of the selection of parents for X_j, for $i \neq j$, the multiplication principle applies again and we obtain for non-root nodes $\{X_1, \ldots, X_C\}$ in β

$$\Pr\left(\Pi_{X_1} \subseteq \boldsymbol{V},\ \ldots,\ \Pi_{X_C} \subseteq \boldsymbol{V}\right) = \prod_{i=1}^{C} \Pr\left(\Pi_{X_i} \subseteq \boldsymbol{V}\right). \tag{15}$$

Substituting (14) into (15) and using the fact that $k = C - i$ gives

$$\Pr\left(\beta \in \mathcal{B}^i_{\text{MPART}}\right) = \prod_{i=1}^{C} \Pr\left(\Pi_{X_i} \subseteq \boldsymbol{V}\right) = \prod_{i=1}^{C} \prod_{j=0}^{P-1} \left(\frac{V-j}{k+V-j}\right) = \prod_{i=1}^{C} \prod_{j=0}^{P-1} \left(\frac{V-j}{C+V-i-j}\right).$$

Since $P \geq 1$ and $C \geq 1$ by assumption, $0 \leq \Pr\left(\beta \in \mathcal{B}^i_{\text{MPART}}\right) \leq 1$ and is thus a well-defined probability. ■

A reader might ask how our MPART algorithm relates to the work of Kask and Dechter [41]. MPART is based on their approach and the topologies of the BNs generated are quite similar. There are some minor differences, including some details in the algorithms; the CPTs generated; and the clamping of evidence. Our main contribution is not MPART itself but rather the following: The observation that BPART is a special case of MPART as characterized in Theorem 30, Theorem 31, and Theorem 32; the fact that C/V-ratio is key also for MPART; and finally the fact that MPART BNs generated using $C/V \leq 0.75$ turn out to be relatively easy for tree clustering as further investigated in Section 6.5. The reason for our interest in $C/V \leq 0.75$ is that this inequality holds for the BNs investigated by Kask and Dechter [41].

4.3.6 Discussion and Related Work

The BPART construction is, as already mentioned, a generalization of the random generation of problem instances for the satisfiability problem (SAT) [55]. Satisfiability might seem like a limited problem to consider since (i) it is a decision problem rather than an optimization problem like the MPE problem (see Definition 9) and (ii) it gives a particular bipartite BN topology. Before discussing these possible concerns, we note that the BPART algorithm can generate Bayesian networks that are *not* SAT-like, using parameter settings such as $F = \texttt{random}$ or $S > 2$.

Concerning (i), even though satisfiability is obviously a decision problem, decision problems are special cases of optimization problems. For instance, it has proven fruitful to view the SAT decision problem as the MAXSAT optimization problem [15, 30]. MAXSAT is an optimization problem where one maximizes the number of satisfied clauses. So even though satisfiability is a decision problem, there is a strong connection to an optimization problem.

Concerning (ii), we argue that the bipartite topology is interesting in its own right and inference algorithms have been specifically designed for bipartite BNs [31]. Important classes of application BNs, including

medical diagnosis BNs such as the QMR-DT BN [67] and information theory BNs [27, 28], are essentially bipartite.[3] Bipartite medical BNs typically model diseases and symptoms — diseases are root nodes, symptoms are leaf nodes [43, 67]. These BNs may be used to compute an MPE over the disease nodes given known symptoms. Bipartite information theory BNs are used for coding and decoding in the presence of noisy transmission [27,28]. Two classes of information theory Bayesian networks are Hamming code BNs and low density parity code BNs, and for both there is a close relationship to BPART networks in general and SAT-like BNs in particular. The fact that "traditional" information theory BNs [28] have a fixed number of children per root node corresponds to a BPART parameter value $R =$ **true** in Figure 2. A parameter value $R =$ **false** gives "modern" Gallager codes [48].

The BPART topology also provides a well-understood stepping stone towards other topologies, and in particular it is a component in multi-partite BNs. Furthermore, as we detail in the MPART construction above, one can regard the leaf nodes in a BPART BN as corresponding to the non-root nodes in an arbitrary, non-bipartite BN such as an MPART BN.

Finally, we note that our approach is independent of and complementary to related research which relies on Markov chain convergence in order to randomly sample BNs [34, 35]. While the use of a Markov chain is different from our approach, there is some similarity between their use of heuristic width [35] and our use of heuristics for clique tree optimization. However, there is an important difference between these two approaches. While we generate BNs randomly without directly controlling maximal clique size or total clique tree size, Ide et al. enforce a constraint on heuristic width as part of the random BN generation process [35]. This makes their approach more general but also potentially slower and more difficult to analyze compared to our approach.

5 Analysis of Hard and Easy Synthetic Networks

This section synthesizes the discussion, in Section 4, of parameters of randomly generated BNs and the presentation of tree clustering in Section 3. In particular, we discuss how varying some of these BN parameters might affect the performance of tree clustering inference. This discussion motivates the experiments performed in Section 6 and justifies our expectations regarding the effect of varying different parameters. Section 6 also provides further quantitative details regarding how clique sizes and inference times increase with the C/V-ratio and when different triangulation heuristics are used.

The rest of this section is organized as follows. In Section 5.1 we relate our focus on the C/V-ratio to previous research. The following few sections discuss topological issues. In Section 5.2 we identify a conservative lower bound on clique tree size. Qualitatively, this lower bound gives an easy-hard-harder pattern, with increasing C/V-ratio, for BN inference using tree clustering. Section 5.3 introduces the structural concept of only-child BN as well as results used in the following. Section 5.4 focuses on cycle formation in the moral graphs of Regular Class A BNs. The reason for our analysis of cycles in general and Hamiltonian cycles in particular is that cycles in the moral graph force tree clustering (including HUGIN) to add fill-in edges to the moral graph, which again significantly impacts the size of the clique tree and thus inference time. In Section 5.5, we argue that regular (Class A) BNs should be harder than irregular (Class B) BNs based on insights regarding cycle formation. The final section, Section 5.6, discusses how the nature of CPTs can impact tree clustering inference.

5.1 Previous Research and the C/V-ratio

For the problem of solving SAT instances, a *phase transition* phenomenon has been observed for the probability of satisfiability [11, 55]. This phase transition phenomenon has been studied by controlling the ratio C/V between the number of variables V and the number of clauses C in a CNF formula, generating problem instances, and experimentally observing the resulting probability of satisfiability [55]. Through extensive experimentation it has been established that for large 3CNF formulas the phase transition occurs for $C/V \approx 4.25$. For smaller instances, the phase transition is at higher C/V values. For example, for $V = 20$ the transition is at $C/V \approx 4.55$. Interestingly, it has been found that the computational cost of

[3] The QMR-DT BN is computationally challenging, but is unfortunately not publicly available and is consequently not used for experimentation here.

finding a satisfying instantiation is correlated with the probability of satisfiability [11,55]. For instance, when using the Davis-Putnam algorithm to search for satisfying assignments, there is an easy-hard-easy pattern for SAT. The hardest instances are found in the region around the phase transition, the *hard region*, of the easy-hard-easy pattern [13, 26, 55, 71].

The nature of the hardness pattern depends on the algorithmic approach investigated, as has been clearly recognized for SAT [55, our emphasis]. Similar to the results for SAT, there is a need for investigations of synthetically generated BPART and MPART networks using a BN inference algorithm such as HUGIN. This is what we do in this section. It should be noted that we are not aiming to establish a phase transition exactly like the one established for SAT; rather our goal is to develop an approach to systematically generate BNs of varying and predictable hardness.

5.2 A Lower Bound

In the BPART and MPART constructions, the C/V-ratio is controlled by varying the values of the input parameters C and V.

Via the construction described in Section 4.3.2 and the discussion in Section 5.1, we hypothesized that the importance of the C/V-ratio carries over to BPART BNs. Given the way we have generated the SAT-like networks corresponding to generation of CNF formulas, we have mapped V to be the number of root nodes in a BN and C to be the number of non-root nodes in the BN. When limiting our attention to SAT-like BNs, the problem is exactly the same as the SAT problem, so one might on first thought expect very similar result. However, in previous work the Davis-Putnam algorithm, a recursive splitting approach [30], was used [55]. The Davis-Putnam algorithm is quite different from tree clustering algorithms including HUGIN. Consequently, while an easy-hard-easy pattern has been observed for SAT formulas using Davis-Putnam [55], there is a different pattern for tree clustering.[4] Our reasoning, which is summarized below in Theorem 34, is based on the following definitions.

Definition 33 (Clique tree metrics) *Let β''' be a clique tree constructed from a BN β by tree clustering. The total size of cliques containing only root nodes from β is defined as $k_R\left(\beta'''\right)$. The total size of cliques containing a mixture of root nodes and non-root nodes from β is defined as $k_M\left(\beta'''\right)$.*

Clearly, no cliques contain leaf nodes only, and thus $k\left(\beta'''\right) = k_M\left(\beta'''\right) + k_R\left(\beta'''\right)$. For BPART in particular there are two types of cliques: Cliques with root nodes only (with total size k_R), and cliques with leaf nodes and root nodes (total size k_M). The total size of cliques with root nodes only, $k_R\left(\beta'''\right)$, turns out to be important in the experimental part of this article; we first focus on $k_M\left(\beta'''\right)$ below.

Theorem 34 (Lower bound) *Consider a BN β with C non-root nodes, each with $P \geq 0$ parents. Let $S = |\Omega_X| \geq 2$ for all nodes X in β. Then, for β's clique tree β''', $k_M\left(\beta'''\right) \geq CS^{P+1}$.*

Proof. Suppose the m mixed node cliques in the clique tree are enumerated from 1 to m. Let the number of root nodes and non-root nodes in the i-th such clique be denoted $V(i)$ and $C(i)$ respectively, where $V(i) \geq P$ and $C(i) \geq 1$. Clearly, $S^{C(i)+V(i)} \geq S^{C(i)+P} \geq C(i)S^{P+1}$. The last inequality follows because it is easy to show, for $S \geq 2$, $P \geq 0$ and $C(i) \geq 1$, that $S^{C(i)+P} \geq C(i)S^{P+1}$. The total size is therefore

$$k_M\left(\beta'''\right) = \sum_{i=1}^{m} S^{C(i)+V(i)} \geq \sum_{i=1}^{m} C(i)S^{P+1} = S^{P+1}\sum_{i=1}^{m} C(i) \geq CS^{P+1},$$

where the last inequality follows because $\sum_{i=1}^{m} C(i) \geq C$. ■

An informal explanation of this result is as follows: By construction, each non-root node C_i in β has P parents and each node in β has S states. Recall from Section 3.1 that as a result of moralization, a non-root node C_i and its parents Π_{C_i} always end up in the same clique. After moralization, a given non-root node C_i will therefore belong to a clique whose size is at least S^{P+1}. As stated in Theorem 34, one can as a lower

[4] For Davis-Putnam, SAT formulas become easier with a very high C/V-ratio, $C/V > 4.25$, due to the overwhelming proportion of unsatisfiable instances where the Davis-Putnam procedure encounters inconsistencies and can prune the search space. Suppose there is an inconsistency for a BN when employing Davis-Putnam. When using tree clustering on the same BN, this inconsistency is not detected until during the propagation phase, which is where the CPT values come into play.

bound argue that this is true for all C non-root nodes, thus the total size of β' after moralization is at least CS^{P+1}. Since edges are potentially added but never deleted in subsequent steps of tree clustering, CS^{P+1} is a lower bound for the total size $k(\beta''')$ of the clique tree β'''. Clearly, this space complexity lower bound implies a time complexity lower bound of CS^{P+1} as well.

Given Theorem 34 and our focus on C/V, how may one make C/V increase? If we hold V constant, then C needs to be increasing, and consequently CS^{P+1} increases. Therefore, the HUGIN compilation and propagation times will typically increase, too. An argument for an *easy-hard-harder* pattern can therefore be made based on HUGIN's moralization step only, and considering all non-root nodes in BNs (including BPART and MPART BNs). In other words, when considering moralization only, one would for tree clustering expect an *easy-hard-harder pattern with increasing C/V-ratio when V is kept constant and C is increased.* This expectation is confirmed in experiments in Section 6.

5.3 Only-Child BNs

For a subclass of Class A BNs, namely only-child BNs (see Definition 36), we can show Hamiltonicity (see Theorem 42). In preparation for that result, we introduce a few definitions that apply to BNs in general. Intuitively, two BN nodes X and Y are q-siblings if they have q parents in common.

Definition 35 (q-siblings) *Let q be a non-negative integer. Consider two nodes X and Y in a directed acyclic graph (BN). If $|\Pi_X \cap \Pi_Y| = q$, then X and Y are q-siblings.*

Based on the concept of q-siblings, the notion of "only (common) child" is introduced below. Informally, a node X is an only-child node if its parents Π_X do not have any other children in common.

Definition 36 (Only-child) *Let $\boldsymbol{X}$ be the nodes in a directed graph (BN). A node X is an* only-child *node if, for all $Y \in \boldsymbol{X} - \{X\}$, X and Y are k-siblings with $k < 2$, where k is a non-negative integer. A graph (BN) is an* only-child graph (BN) *if all nodes are only-child nodes.*

We hypothesize that the notion of only-child provides a good approximation to the BNs constructed by BPART and MPART in the following sense: When the number of root nodes V is "large", both BPART and MPART will "often" construct a node that is an only-child, as long as the number of non-root nodes C is "not too large" compared to V. Stated differently, a "high" proportion of the children will be only-children, given "reasonable" assumptions on the number of non-root nodes and root nodes. We believe that these ideas can be further formalized, and leave this for future work.

The notion of only-child is not, in general, valid in application BNs and we have not built the assumption into the BPART and MPART algorithms either. However, the notion of only-child paves the way for a formal analysis based on filtering of clique trees, which we now introduce.

Definition 37 (Filtering) *Let β''' be an undirected graph $\left(\boldsymbol{X}''', \boldsymbol{E}'''\right)$ that is a clique tree. The filtering of clique nodes $\boldsymbol{X}'''$ using nodes $\boldsymbol{V}$ is defined as $\boldsymbol{X}''' \langle \boldsymbol{V} \rangle = \boldsymbol{X}''' \cap \mathcal{P}(\boldsymbol{V})$, where $\mathcal{P}(\boldsymbol{V})$ is the power set of $\boldsymbol{V}$.*

Given a filtered set of nodes, one can construct an induced subgraph (Definition 5) of the clique tree β''' as follows: First, consider the tuple of clique tree nodes and edges $\left(\boldsymbol{X}''', \boldsymbol{E}'''\right)$ in β''', and form $\boldsymbol{Y}''' \leftarrow \boldsymbol{X}''' \langle \boldsymbol{V} \rangle$. Second, form the induced subgraph $\boldsymbol{Z}''' \leftarrow \beta''' \left[\boldsymbol{Y}'''\right]$ by using the clique tree nodes $\boldsymbol{Y}'''$. This process, where we define $\beta'''[\boldsymbol{V}] = \beta''' \left[\boldsymbol{X}''' \langle \boldsymbol{V} \rangle\right]$, is illustrated below. For simplicity we say, for example, $V_1V_2C_1$ rather than $\{V_1, V_2, C_1\}$ to represent a clique tree node containing the BN nodes V_1, V_2, and C_1. The difference between $\boldsymbol{Y}'''$ and $\beta''' \left[\boldsymbol{Y}'''\right]$ is that the former is a set of nodes while the latter is a graph. In Example 38 the notions of only-child, induced subgraph, and filtering are brought together.

Example 38 (Only-child BN) *Figure 6 shows a bipartite only-child BN β as well as the induced subgraphs $\beta'[\boldsymbol{V}]$, $\beta''[\boldsymbol{V}]$, and $\beta'''[\boldsymbol{V}]$ of β', β'', and β''' respectively. This is an only-child BN because no leaf node C_i has more than one parent in common with another leaf node C_j for any $i \neq j$. There is a loop $(V_5, V_7, V_6, V_4, V_2, V_3, V_1)$ in the moral subgraph $\beta'[\boldsymbol{V}]$, which again causes fill-in edges (V_5, V_6), (V_5, V_4), (V_1, V_4), and (V_1, V_2) in the triangulated subgraph $\beta''[\boldsymbol{V}]$ and cliques as shown in the clique subtree $\beta'''[\boldsymbol{V}]$.*

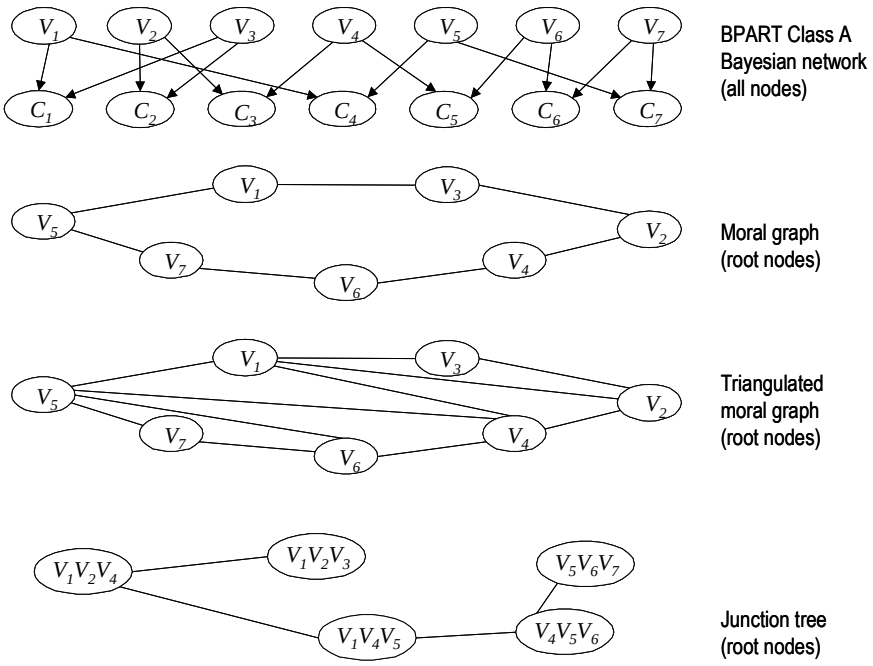

Figure 6: An example of compiling an only-child BPART BN. Only the subgraph induced by the BN root nodes, $\boldsymbol{V}$, are shown for the moral graph, the triangulated moral graph, and the junction (or clique) tree. This is an only-child BN because no leaf node C_i has, for all $i \neq j$, more than one common parent with another leaf node C_j. For instance, $|\Pi_{C_1} \cap \Pi_{C_2}| = |\{V_1, V_3\} \cap \{V_2, V_3\}| = 1$.

In Example 38 there is a Hamiltonian cycle in the moral graph for the root nodes, which again causes four fill-in edges and five cliques in the subgraph of the junction tree induced by the BN root nodes.

The notion of only-child is useful when we analyze how moralization affects the non-leaf nodes in a BN. In particular, the only-child definition is used in the following theorem.

Theorem 39 (Degree of moralized only-child BNs) *Let $\boldsymbol{C}$ be the non-root nodes in a* BPART *only-child BN β (Definition 36), and let $\boldsymbol{V}$ be the root nodes. If $P = |\Pi_C| \geq 2$ for all $C \in \boldsymbol{C}$, then any $V \in \boldsymbol{V}$ has degree $d(V) = (P-1)|\Psi_V|$ in the subgraph $\gamma' = \beta'[\boldsymbol{V}]$ of the moralized graph β' induced by $\boldsymbol{V}$.*

Proof. There are three cases: $|\Psi_V| = 0, |\Psi_V| = 1$, and $|\Psi_V| \geq 2$. For the first two cases, it holds true for $V \in \boldsymbol{V}$ that $d(V) = 0$ and $d(V) = P - 1$ respectively and so the formula is correct. We now consider the case of $|\Psi_V| \geq 2$. Since each $C \in \boldsymbol{C}$ has P parents, moralization ensures that each parent $V \in \Pi_C$ has at most $(P-1)$ neighbors in the moralized graph β' due to C. Consider arbitrary $C_i, C_j \in \boldsymbol{C}$ such that $C_i, C_j \in \Psi_V$, where $i \neq j$. By assumption, each $C \in \boldsymbol{C}$ is an only-child. Due to the assumed only-child property, $(\Pi_{C_i} - V) \cap (\Pi_{C_j} - V) = \emptyset$ for any $V \in \boldsymbol{V}$. Each of C_i and C_j consequently give $(P-1)$ neighbors for V in the graph $\gamma' = \beta'[\boldsymbol{V}]$ induced by $\boldsymbol{V}$ in the moralized graph β'. This holds for any $C_i, C_j \in \Psi_V$, giving a total of $d(V) = (P-1)o(V) = (P-1)|\Psi_V|$. ■

An example of applying Theorem 39 follows.

Example 40 (Degree of moralized only-child BNs) *Consider Figure 6 again, and in particular the subgraph $\gamma' = \beta'[\boldsymbol{V}]$ induced by the BN's root nodes $\boldsymbol{V} = \{V_i \mid 1 \leq i \leq 7\}$. (The second graph from the top in Figure 6 is γ'.) For all $V_i \in \boldsymbol{V}$, $d(V_i) = (P-1)|\Pi_C| = 2$ as predicted by Theorem 39.*

5.4 Regular BNs

In BPART and MPART, setting $R =$ **true** creates regular BNs. In this section, the focus is on how regularity impacts cycles in the moralized graph of a BN. Long, undirected cycles (loops) in the moralized

graph are one of the main factors causing large maximal clique sizes. Consequently, a key question for tree clustering is the impact of the moralization and triangulation steps in terms of cycles. We focus on the longest cycle as well as the extreme case of cycles visiting all nodes exactly once, *Hamiltonian cycles*.

5.4.1 Regular Class A BNs

The issue of the regularity of the underlying graph of a Bayesian network has received some attention in information theory [27, 28, 48]. Gallager's original codes, which are denoted classical Gallager codes in Table 1, require each root node to have the same number of children and each leaf node to have the same number of parents. They are therefore of the Class A type [28].

Recently, a compelling argument has been provided for why lifting the Class A regularity constraints may be beneficial when computing the MPE in BNs for decoding [48]. In particular, when using iterated belief propagation to compute the MPE given a codeword transmitted over a noisy channel, irregular BNs have been found to perform better than regular ones [48]. The intuition is that, using iterated belief propagation, high degree root nodes may tend to get quicker to the "right" setting, given that they need to satisfy more constraints. This leads to a "wave effect" that helps lower degree root nodes find their "right" setting. For belief propagation it is therefore beneficial to have a mixture of high-degree and low-degree ("low regularity") root nodes in information theory BNs. These observations raise the question of how BN regularity might impact tree clustering. In the following we provide further insight into this question by considering cycle formation; some related work exists [29].

5.4.2 Regular Only-Child BNs

We now turn to cycle formation in the moralized graph of only-child BNs. Example 38 and Figure 6 illustrate the importance of Hamiltonian cycles. The following Hamiltonicity result by Jackson [36] is used below.

Theorem 41 (Hamiltonicity, Jackson) *Let $\boldsymbol{G}$ be an undirected graph where all vertices have degree $\delta(\boldsymbol{G}) = d(\boldsymbol{G}) = k$. If $d(\boldsymbol{G}) \geq \frac{n(\boldsymbol{G})}{3}$ then $\boldsymbol{G}$ is Hamiltonian.*

In Theorem 42 we apply Theorem 41 to a BN's moral graph β'.

Theorem 42 (Hamiltonicity of BPART only-child BN) *Consider an only-child* BPART *BN β constructed with $R = \mathbf{true}$, let β' be the moralized graph of β, and let $\gamma' = \beta'[\boldsymbol{V}]$ be the subgraph in β' induced by the root nodes $\boldsymbol{V}$ of β. If*

$$(P-1)\left\lfloor \frac{CP}{V} \right\rfloor \geq \frac{V}{3} \tag{16}$$

then γ' is Hamiltonian.

Proof. The theorem follows from Theorem 41, Corollary 18, and Theorem 39. Consider the subgraph γ' induced by the root nodes $\boldsymbol{V}$ in the moralized graph β', $\gamma' = \beta'[\boldsymbol{V}]$. Since β is an only-child BN it is known from Theorem 39 that for a root node $X \in \boldsymbol{V}'$, where $\gamma' = (\boldsymbol{V}', \boldsymbol{E}')$, $d(X) = (P-1)|\Psi_X|$. Due to the BPART BN's regularity, Corollary 18 applies and $|\Psi_X| \approx \frac{CP}{V}$. Consequently, each root node X from the BN has, in γ', the degree $d(X) = (P-1)\left\lfloor \frac{CP}{V} \right\rfloor$ or $d(X) = (P-1)\left\lceil \frac{CP}{V} \right\rceil$. For the purposes of this proof we drop edges for nodes with $(P-1)\left\lceil \frac{CP}{V} \right\rceil$ edges, if any, thus constructing γ'' where all nodes have degree $d(\gamma'') = (P-1)\left\lfloor \frac{C(P-1)}{V} \right\rfloor$. Since γ'' is a regular graph, we can apply Theorem 41 with $n(\gamma') = n(\gamma'') = V$ and obtain (16). ■

We note how, for constant P, the ratio C/V plays a prominent role in Theorem 42, specifically in the expression $\left\lfloor \frac{CP}{V} \right\rfloor$. And, as the following example illustrates, the C/V-ratio does not need to be very high before a Hamiltonian cycle is guaranteed.

Example 43 (Hamiltonicity of BPART only-child BN) *Consider an only-child* BPART *network with $C = 90$, $V = 30$, $P = 3$, and $R = \mathbf{true}$. In particular, consider the undirected graph γ' induced by the root nodes $\boldsymbol{V}$ in β, after moralization to β'. Using Theorem 42, we obtain $(P-1)\lfloor CP/V \rfloor = 18$, while $n(\gamma')/3 = 10$, so a Hamiltonian cycle is guaranteed.*

There are cases where one might not be able to show that a Hamiltonian cycle must exist, but one can compute the longest cycle $c(\gamma')$ and apply the following theorem due to Dirac [23].

Theorem 44 (Longest cycle, Dirac) *Let $c(\boldsymbol{G})$ be the length of the longest cycle of an undirected graph $\boldsymbol{G}$. If $\boldsymbol{G}$ is 2-connected then $c(\boldsymbol{G}) \geq \min(n(\boldsymbol{G}), 2\delta(\boldsymbol{G}))$.*

The longest cycle is determined by the following result when the underlying BN β is of a certain type.

Corollary 45 (Longest cycle in BPART only-child BN) *Consider an only-child* BPART *BN β constructed with $R =$ **true**, and let $\boldsymbol{V}$ be the root nodes in β. Further, let β' be the moralized graph of β, and consider $\gamma' = \beta'[\boldsymbol{V}]$, the subgraph of β' induced by $\boldsymbol{V}$. If γ' is 2-connected then*

$$c(\gamma') \geq \min\left(V, 2(P-1)\left\lfloor \frac{CP}{V} \right\rfloor\right). \tag{17}$$

Proof. Theorem 44 applies with $n = V = |\boldsymbol{V}|$ and $\delta(\boldsymbol{G}) \geq (P-1)\left\lfloor \frac{CP}{V} \right\rfloor$; the result follows. ■

Again, we see that the C/V-ratio is part of a lower bound (17), here for the longest cycle $c(\gamma')$.

Example 46 *Consider a 2-connected undirected graph $\gamma' = \beta'[\boldsymbol{V}]$ constructed, using tree clustering, from an only-child* BPART *BN β created using input parameters $R =$ **true**, $P = 3, V = 30$, and $C = 60$. In this case, Corollary 45 applies and the longest cycle is $c(\gamma') = \min(30, 24) = 24$.*

While only the longest cycle $c(\gamma')$ is mentioned in the corollary and example above, it is clear that there may be several cycles, of varying length, in a moralized BN. Due to the limited number of root nodes V these cycles are likely to interact, increasing the chances of fill-in edges, leading to larger cliques and larger maximal clique sizes.

5.5 Irregular BNs

So far, the role of the C/V-ratio in regular Class A BNs has been established. We now turn to irregular Class B BNs, which are generated by setting $R =$ **false** when invoking BPART and MPART. How do Class A regular ($R =$ **true**) BPART BNs compare to Class B irregular ($R =$ **false**) BPART BNs, specifically with regard to the Hamiltonicity of the moralized graph as induced by the root nodes of the BN? The following theorem addresses this question by considering random variables $\mathbb{M}^i$ and $\mathbb{M}^r$ representing the minimal root node out-degree in the BPART irregular and regular case respectively (see Section 4.3.4). Using expectations, we consider undirected graphs $\boldsymbol{G}^r$ and $\boldsymbol{G}^i$, both induced over the moralized graphs using the root nodes of the respective Class A and Class B BPART networks.

Theorem 47 *Let $\boldsymbol{V}^r$ and $\boldsymbol{V}^i$ be root nodes for only-child* BPART *BNs α and β generated using $R =$ **true** and $R =$ **false** respectively, and put $\mathbb{M}^r = \min(\boldsymbol{V}^r)$ and $\mathbb{M}^i = \min(\boldsymbol{V}^i)$. Let $\boldsymbol{G}^r = \alpha'[\boldsymbol{V}^r]$ and $\boldsymbol{G}^i = \beta'[\boldsymbol{V}^i]$. Further, for $b > 0$, let $d(\boldsymbol{G}^r) := bE(\mathbb{M}^r)$ and $d(\boldsymbol{G}^i) := bE(\mathbb{M}^i)$ respectively and assume that there exist* BPART *input parameters for any $E(\mathbb{M}^r)$ where $0 \leq E(\mathbb{M}^r) < C$. There exist* BPART *input parameter values (excluding for R) such that $\boldsymbol{G}^r$ is Hamiltonian while $\boldsymbol{G}^i$ is not necessarily Hamiltonian.*

Proof. For $R =$ **true**, set the other input parameter values to BPART such that $E(\mathbb{M}^r) = \frac{V}{3(P-1)}$, which according to Theorem 39 gives $d(\boldsymbol{G}^r) = \frac{V}{3}$. Using Theorem 41, $\boldsymbol{G}^r$ is Hamiltonian. For $R =$ **false**, suppose that the same BPART input parameters values are used as for the $R =$ **true** case (except for R). Clearly, $E(\mathbb{M}^i) < E(\mathbb{M}^r)$ and consequently $d(\boldsymbol{G}^i) < d(\boldsymbol{G}^r)$ and $d(\boldsymbol{G}^i) < \frac{V}{3}$ and as a result $\boldsymbol{G}^i$ is not necessarily Hamiltonian according to Theorem 41. ■

Theorem 47 says that, for certain BPART input parameter values excluding R's, the induced subgraph $\boldsymbol{G}^r$ constructed from a regular BPART BN is Hamiltonian, while there is no such guarantee for $\boldsymbol{G}^i$ constructed from an irregular BPART BN.

The result above and the analysis in Section 4.3.4 suggest that regular Class A BPART networks are, for given values of the input parameters excluding R, more likely to have cycles that need fill-in edges than irregular Class B BPART networks. This again makes the inference problem harder for tree clustering in the Class A BPART case. Further insight is provided by the experiments discussed in Section 6.4, where we consider the effect of regularity on the hardness of tree clustering in terms of maximal clique sizes and inference times.

5.6 Hardness and Conditional Probability Tables

To investigate the effect of different conditional probability tables (CPTs), we consider, as described in Section 4.3.2, three CPT types: deterministic CPTs (`or`, `and`, and `xor`), `uniform` CPTs, and `random` CPTs. In the BPART and MPART algorithms, CPTs are controlled using the Q and F input parameters.

The values of CPTs have, with a few notable exceptions discussed below, little significance during tree clustering's *compilation phase*, since this phase is primarily impacted by a BN's topology. The exceptions include the two techniques of zero-compression and approximation, techniques that also take numerical CPT values into account [24]. These techniques are not used or investigated in this article; we leave this for future research. In the remainder of this section, we focus on tree clustering's *propagation phase*.

5.6.1 Random Root Nodes and Random Non-Root Nodes

The nature of a BN's CPT impacts the number of MPEs, which we formally define as follows in the case of randomly generated BNs.

Definition 48 (Number of MPEs) *Let β be a randomly generated BN with MPEs $\boldsymbol{X}^* = \{\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_u^*\}$. The number of MPEs in β is defined as a random variable $\mathbb{U} = |\boldsymbol{X}^*| = |\{\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_u^*\}|$.*

For both BPART and MPART BNs with `random` CPTs there will be very few MPEs, typically one, or $E(\mathbb{U}) \approx 1$. The reason for this is that it is very unlikely that two different explanations will have exactly the same probability, when all conditional probabilities are sampled from a continuous distribution as is done when $Q =$ `random` and $F =$ `random`. The existence of one MPE means that just one propagation in the clique tree β''' is required. The time required for one clique tree propagation increases with the C/V-ratio, on average, due to increased clique sizes, as argued earlier in this section.

5.6.2 Uniform Root Nodes and Deterministic Non-Root Nodes

We consider now the case of `uniform` root nodes and boolean leaf `or`-nodes, or $Q =$ `uniform` and $F =$ `or`. For these SAT-like BPART BNs there are lessons to be learned from the extensive research on SAT and constraint satisfaction problems using search algorithms [11, 13, 26, 71]. Specifically, an approximately log-linear relationship between the C/V-ratio and the expected number of solutions $E(\mathbb{U})$ has been empirically established, with for example $\hat{E}(\mathbb{U}) \approx 1000$ for $C/V = 3$ and $\hat{E}(\mathbb{U}) \approx 11.5$ for $C/V = 4.67$ for the case of $V = 24$ variables and satisfiable problem instances [71]. In other words, as the C/V-ratio increases the number of solutions decreases on average. While other characteristics of search space structure, such as the size variability of global minima [26] and the decrease in the number of local minima with the C/V-ratio [71] are important, it is clear that the drop-off in $E(\mathbb{U})$ as a function of the C/V-ratio is prominent.

For SAT-like BNs the number of MPEs would depend on the C/V-ratio exactly as for SAT. The propagation part of the inference problem should thus be relatively harder for HUGIN belief revision to handle at low C/V-ratios since repeated propagations are required to arrive at an MPE $\boldsymbol{x}^*$ [50]. As the C/V-ratio is increased, a question is how the resulting decrease in $E(\mathbb{U})$ — suggesting decreased inference time due to fewer propagations — will interact with the increase in inference time due to larger cliques in the clique tree. The experimental part of this article sheds light on this question.

6 Experiments with Hard and Easy Synthetic Networks

In this section we report on experiments performed with the HUGIN system using BNs generated by implementations of the BPART and MPART algorithms. Different parameters, as presented in Section 4.3.1, that control the nature of BNs generated by BPART and MPART have been varied in the experiments. Which parameters should be systematically varied? Our main concern in answering this question is to make sure that inference hardness, in terms of tree clustering's maximal clique size, can be varied in an interesting fashion. Some of the parameters, such as total number of BN nodes $N = V + C$ and number of states per BN node S, are obviously tied to the complexity of inference, and are somewhat less interesting. Other structural and distributional effects may not be as obvious and are therefore the main focus in the experiments below.

Specifically, we study the effect — on maximal clique size and inference times — of varying the C/V-ratio, the graph regularity parameter R, and the CPT value parameters F and Q.

In the following, Section 6.1 outlines the methodology used. In Section 6.2, we present results from computational experiments that show how maximal clique size and inference time vary with the C/V-ratio for SAT-like BNs generated using the BPART construction. Section 6.3 follows up on Section 6.2 and compares, using the same BPART instances, the MinimumCliqueWeight and MinimumFillInWeight triangulation heuristics. Section 6.4 presents experiments for BNs of varying regularity, again for the BPART construction. Section 6.5 provides experimental results for the MPART construction — unlike for the other experiments we here used `random` CPTs and also kept the total number of BN nodes constant.

6.1 Methodology

In the experiments reported here, we generated random samples of BN instances according to the BPART or MPART constructions, ran the Hugin system [24] on the samples, and recorded key clique tree characteristics as introduced in Definition 10. In particular, we focused on the closely related statistics for maximal clique size ℓ (5) and maximal number of nodes in a clique h (3). Given the random generation of instances, these characteristics may be considered random meta-variables, and we investigated maximal clique size $\mathbb{L}$ and number of nodes in a maximal clique $\mathbb{H}$, with corresponding sample statistics such as maximal clique size sample mean $\bar{x}_{\mathbb{L}}$ and maximal clique size sample standard deviation $s_{\mathbb{L}}$.[5] For tree clustering, the maximal clique size results obviously impact both belief updating and belief revision since both rely on clique trees.

While we focus primarily on maximal clique size results, we also present some inference time results for belief revision. Specifically, we present inference time statistics such as sample median m, sample mean $\bar{x}$, and sample standard deviation s for the time (in seconds) to compute an MPE $\boldsymbol{x}^* \in \{\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_u^*\}$ — all as a function of the C/V- ratio. These results are, we believe, interesting in their own right and also upper bound the inference time for belief updating. For the timing experiments, a Dell 410 700MHz Pentium III CPU with 1GB of RAM and using up to 9GB of swap space was used.

The number of parents of a non-root node was $P = 3$ in all experiments except for those reported in Section 6.5, where non-root nodes have two parents, so $P = 2$. Note that state space sizes of cliques including ℓ (6), "clique sizes" for short, are given as numbers indicating memory requirements, where the amount of primary memory (RAM) required for storage is implementation-dependent. For instance, an implementation of tree clustering might use a `double` data type requiring 8 bytes, in which case the amount of RAM r needed to store a 24-node clique consisting of binary ($S = 2$) nodes is $r = \ell \times 8$ bytes $= S^h \times 8$ bytes $= 2^{24} \times 8$ bytes $= 128$MB.

We do not report on experiments with as large SAT-like networks as the propositional formulas used in earlier experimental research on SAT — see, for example, [55] or repositories such as SATLIB at `http://www.satlib.org`. Hugin was not able to process these large networks for non-trivial C/V-ratios. For the same reason, many of our results do not approach or go beyond the region where $C/V \approx 4.25$, even though this is the phase-transition region for SAT formulas [55]. This is in line with our goal of constructing BNs for benchmarking and understanding BN computational hardness rather than focusing on the $C/V \approx 4.25$ phase transition. We should also note that Hugin's default settings were generally used in these experiments. The MinimumFillInWeight triangulation heuristic was used, except in Section 6.3. Hugin's default "off" settings were used for compression and approximation. As far as evidence for leaf nodes, the default was to clamp binary nodes to 1 (or **true**).

6.2 BPART Class B Networks: Hardness and the C/V Phenomenon

What is the empirical impact, on maximal clique sizes and inference times, of varying the C/V-ratio when generating irregular ($R =$ **false**) BPART BNs? In order to investigate this question, 800 SAT-like BNs were generated using the signature BPART(`uniform`, `or`, *30, C, 2,* **false**, *3*), varying the number of leaf nodes from $C = 60$ to $C = 102$. In other words, the C/V-ratio was varied from $C/V = 2.0$ to $C/V = 3.4$. The leaf nodes were clamped to 1 during Hugin inference. In Table 4, Figure 7, and parts of Table 5, maximal clique size and inference time results from these experiments are reported.

[5] The term "meta-variable" is used to distinguish these random variables from the random variables or nodes making up the BNs themselves.

Maximal clique size		C/V-ratio of BPART Class B (irregular) BNs								
Nodes $\mathbb{H}$	**State space $\mathbb{L}$**	**2.0**	**2.2**	**2.4**	**2.6**	**2.8**	**3.0**	**3.2**	**3.4**	**Total**
14	16,384	5	1							6
15	32,768	26	7							33
16	65,536	48	28	10	1					87
17	131,072	19	37	35	23	10	2			126
18	262,144	2	23	41	50	33	10	7	1	167
19	524,288		4	13	20	41	46	30	16	170
20	1,048,576			1	6	14	41	42	39	143
21	2,097,152					2	1	20	37	60
22	4,194,304							1	7	8
Number of instances		100	100	100	100	100	100	100	100	800
Mean number of nodes $\bar{x}_{\mathbb{H}}$		15.87	16.86	17.60	18.07	18.65	19.29	19.78	20.33	18.31
Mean size (in 1000s) $\bar{x}_{\mathbb{L}}$		70.94	150.6	238.6	329.7	503.3	720.9	1,077	1,565	582.0

Table 4: Experimental results showing statistics for BNs generated by the BPART algorithm with $R =$ **false** (the irregular case), with $V = 30$ and C ranging from 60 to 102. The number of nodes in and sizes of the maximal cliques are shown for instances with varying C/V ratios. The smallest maximal clique contained $h = 14$ nodes (for 5 instances with $C/V = 2.0$), while the largest maximal clique contained $h = 22$ nodes (for 7 instances with $C/V = 3.4$).

BN parameters			BPART Class A (regular)				BPART Class B (irregular)				Ratios	
V	**C**	**C/V**	$m^r_{\mathbb{T}}$	$\bar{x}^r_{\mathbb{T}}$	$s^r_{\mathbb{T}}$	$\bar{x}^r_{\mathbb{L}}$	$m^i_{\mathbb{T}}$	$\bar{x}^i_{\mathbb{T}}$	$s^i_{\mathbb{T}}$	$\bar{x}^i_{\mathbb{L}}$	$\bar{x}^r_{\mathbb{T}}/\bar{x}^i_{\mathbb{T}}$	$\bar{x}^r_{\mathbb{L}}/\bar{x}^i_{\mathbb{L}}$
30	60	2.0	16.70	19.4	11.5	382.1	3.67	3.10	2.10	70.94	3.13	5.39
30	66	2.2	29.00	33.1	16.4	655.4	7.11	8.21	5.01	150.6	4.04	4.35
30	72	2.4	42.84	45.6	19.8	923.0	9.94	11.61	6.75	238.6	3.93	3.87
30	78	2.6	58.41	65.8	31.8	1,376	14.60	18.38	12.52	329.7	3.58	4.17
30	84	2.8	101.45	109.8	59.7	2,081	21.43	26.31	17.98	503.3	4.17	4.13
30	90	3.0	121.05	131.0	66.3	2,642	29.41	35.32	21.36	720.9	3.71	3.66
30	96	3.2	156.45	158.4	65.4	3,324	45.64	53.78	33.90	1,077	2.95	3.09
30	102	3.4	168.30	192.0	97.4	4,708	58.74	67.36	42.79	1,565	2.85	3.01

Table 5: The effect of regular and irregular BPART BNs on HUGIN maximal clique sizes and computation times (in seconds) is shown. The following sample statistics are presented for the regular case: median computation time $m^r_{\mathbb{T}}$, mean computation time $\bar{x}^r_{\mathbb{T}}$, standard deviation for computation time $s^r_{\mathbb{T}}$, and mean maximal clique size $\bar{x}^r_{\mathbb{L}}$. Similar statistics are presented, with "i" superscripts, for the irregular case.

From the results and analysis provided in Table 4, we conclude that for BPART the maximal clique size increases, on average, with the C/V-ratio. This is in correspondence with the theoretical results earlier in the article. Figure 7 shows, using linear regression, how the number of nodes in the maximal clique varies with the C/V-ratio. This linear regression result shows that for BPART, given the parameters used here, the sample mean number of nodes in the maximal clique, $\bar{x}_{\mathbb{H}}$, grows linearly with C/V-ratio. We obtain the following empirical expressions for the maximal number of nodes in a clique h and maximal clique size ℓ: $h = 3.06 \times C/V + 10.0$ and $\ell = 2^h = 2^{3.06 \times C/V + 10.0}$. The regression is statistically significant, with an $R^2 = 0.716$, an F-ratio of 2013, and a p-value of 2.10×10^{-220}. The 95% confidence interval for the slope of the regression line is $(2.93, 3.20)$.

Let us now discuss the inference time results presented in Figure 7 and Table 5. The linear growth of $\bar{x}_{\mathbb{H}}$ and h with C/V translates into exponential growth of $\bar{x}_{\mathbb{L}}$ and ℓ with C/V. This exponential growth of ℓ with C/V in turn explains the approximately exponential growth in inference times reported in Figure 7. This result provides an empirical answer to the question raised in Section 5.6: On the one hand, we recall that fewer propagations due to a decrease in $E(\mathbb{U})$ should cause a *decrease* in the total inference time. On the other hand, an increase in expected maximal clique size $E(\mathbb{H})$ should cause an *increase* in inference time. Clearly, the latter effect outweighs the former here.

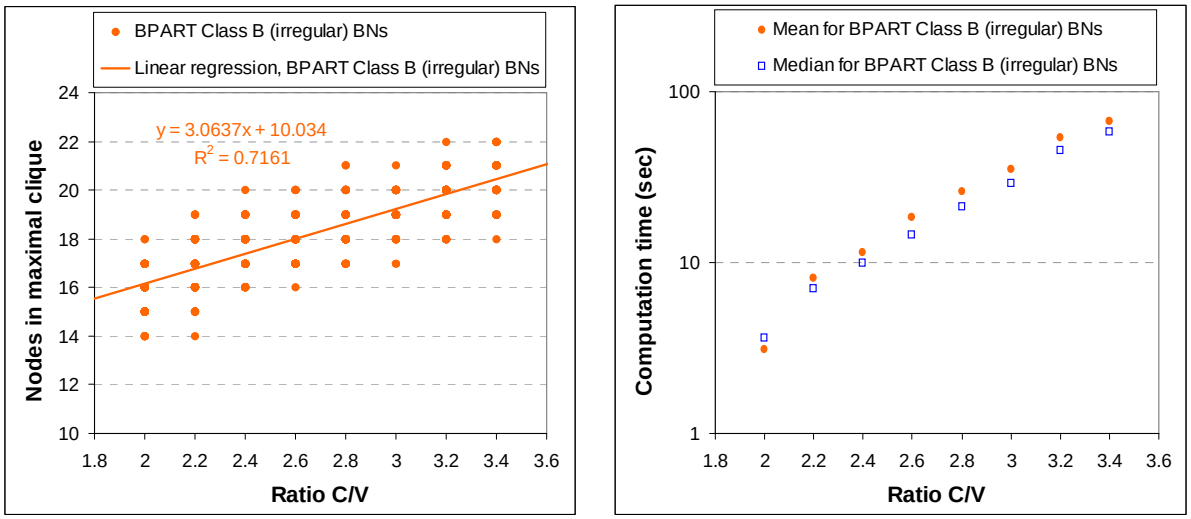

Figure 7: These results are for Class B (irregular) BPART BNs with $V = 30$ root nodes and a C/V-ratio ranging from $C/V = 2.0$ to $C/V = 3.4$. Left: The number of nodes in the clique tree's maximal clique is plotted as a function of the C/V-ratio. In this scatter plot, points representing BN instances as well as linear regression results are displayed. Right: In this graph of averages and medians, the MPE computation time sample means $\bar{x}_{\mathbb{T}}$ and sample medians $m_{\mathbb{T}}$ are shown.

We note, in Table 5, that the standard deviations $s^i_{\mathbb{T}}$ for the inference times are substantial. The high standard deviation is typical for this area of research and has been observed also for SAT [55, 62, 63]. Both Table 4 and Figure 7 provide evidence for why the inference time standard deviation $s^i_{\mathbb{T}}$ is substantial: Inference time clearly depends on the maximal clique size $\mathbb{L}$, which is exponential in the number of nodes in the maximal clique $\mathbb{H}$. A change in $\mathbb{H}$ has an exponential effect on $\mathbb{L}$, which again has an approximately linear effect on inference time. In the column $C/V = 2.0$ in Table 4, $\mathbb{H}$ ranges from 14 to 18 giving a 28.6% increase from the smallest ($\mathbb{H} = 14$) to the largest value ($\mathbb{H} = 18$). This causes $\mathbb{L}$ to range from 16,384 to 262,144, a 1500% increase from the smallest ($\mathbb{L}$ =16,384) to the largest ($\mathbb{L}$ = 262,144) value.

Table 6 contains further details for the seven extreme instances listed in the column $C/V = 2.0$ in Table 4. Table 6 displays the five instances with smallest maximal clique size $\ell(\beta_0) = \ell(\beta_{12}) = \ell(\beta_{73}) = \ell(\beta_{89}) = \ell(\beta_{92}) = 16,384$ and the two instances with largest maximal clique size $\ell(\beta_6) = \ell(\beta_{80}) = 262,144$. The following example illustrates how total clique tree size k were distributed among k_M and k_R (Definition 33) for one of these example BPART instances.

Example 49 *Table 6 contains a* BPART *(**uniform**, **or**, 30, 60, 2, **false**, 3) instance β_{12}. The total state space size for the clique tree β'''_{12} is given by $k\left(\beta'''_{12}\right) = k_M\left(\beta'''_{12}\right) + k_R\left(\beta'''_{12}\right) = 75,840$, where $k_M\left(\beta'''_{12}\right) = 60 \times 2^4 = 960$ and $k_R\left(\beta'''_{12}\right) = 2 \times 2^4 + 1 \times 2^5 + \cdots + 1 \times 2^{13} + 4 \times 2^{14} = 73,792$.*

In terms of number of cliques in the above example, the cliques of size $S^i = 2^4$ are clearly dominant, with $q_i = 60$ cliques contributing towards $k_M\left(\beta'''_{12}\right)$: $60 \times 2^4 = 960$. However, in terms of impact on $k\left(\beta'''_{12}\right)$, $k_R\left(\beta'''_{12}\right)$ and in particular the maximal cliques of size $S^i = 2^{14}$ are much more important, with contribution $4 \times 2^{14} = 65,536$.

Several points can be made regarding these seven extreme instances. First, while most cliques have size $|\Omega_H| = 16$ as indicated by the lower bound of Theorem 34, there is a strong heavy tail effect present, in particular in β_6 with total clique size $k = 439,776$ and in β_{80} with $k = 284,192$. The three largest cliques make up 93.1% and 96.6% of total clique size in BNs β_6 and β_{80} respectively. In fact, β_{80} is extreme; the size ℓ of the largest clique is 32 times that of the second largest clique. Clearly, whether cliques of size 2^{17} and 2^{18} are present (for β_6 and β_{80}) or absent (for the rest) has, in this example, a dramatic impact on total clique tree size and thereby on variation in total clique tree size between the different instances. This is a pattern that may in part explain the large standard deviations for inference times in Table 5.

Clique size		Clique trees for BPART Class B BNs, $C/V = 2.0$						
Nodes h	**State space** ℓ	β'''_0	β'''_{12}	β'''_{73}	β'''_{89}	β'''_{92}	β'''_6	β'''_{80}
4	16	60	62	61	60	60	60	60
5	32	2	1	3		3	1	1
6	64	2	1	2		1		1
7	128	2	4	1	3	4		2
8	256	1	2	2	2		2	1
9	512	2		1	2	3	2	2
10	1,024	1		1		1		1
11	2,048		2		1			1
12	4,096	2	1	1	1	2	2	2
13	8,192			1	3	1		1
14	16,384	2	4	4	4	2	2	
15	32,768							
16	65,536							
17	131,072						1	
18	262, 144						1	1
Maximal clique size ℓ		16,384	16,384	16,384	16,384	16,384	262,144	262,144
Total clique tree size k		61,056	75,840	81,200	99,138	53,344	439,776	284,192
Maximal number of nodes h		14	14	14	14	14	18	18
Maximal clique $\frac{\ell}{k} \times 100\%$		26.8%	21.6%	20.2%	16.5%	30.7%	59.6%	92.2%
Top three cliques		60.4%	64.8%	60.5%	49.6%	76.8%	93.1%	96.6%

Table 6: Randomly generated BPART Class B instances, for $V = 30$ and $C = 60$, showing the seven instances with the smallest and largest maximal clique sizes. A column contains, for the clique tree β'''_i of a BN instance β_i, the number of cliques it contains for different clique sizes. From a sample of 100 BNs, the five columns β_0, β_{12}, β_{73}, β_{89}, and β_{92} present the BN instances with smallest maximal clique size $\ell = 16,384$, while the two columns β_6 and β_{80} present the BN instances with largest maximal clique size $\ell = 262,144$. Of these instances, β_6 has the largest total clique size $k = 439,776$, while β_{92} has the smallest total clique size $k = 53,344$.

In related experiments, details of which are omitted due to space restrictions, the randomly generated BNs consisted of $V \in \{20, 30, 35, 40\}$ root nodes, the number of leaf nodes C was varied from 40 to 140, and the C/V-ratio was ranging from 2.0 to 4.0. The results were similar to what was reported above: As the C/V-ratio increased, the mean inference time increased at an approximately exponential rate.

Stated qualitatively, our experiments confirm the hypothesis that a "high" C/V-ratio implies a "large" maximal clique size which again implies a "long" inference time. There is a strong causal relationship from a high C/V-ratio to a large maximal clique size. Of course, this is for the BPART topology, using certain ranges for the input parameters of BPART, and for a particular tree clustering triangulation heuristic MinimumFillInWeight. In the next section we turn to a more detailed study of triangulation heuristics.

6.3 BPART Class B Networks: Triangulation Heuristics

Here, we report on experiments with the MinimumCliqueWeight triangulation heuristic. A first question is whether varying the C/V-ratio has a similar impact when using the MinimumCliqueWeight heuristic as was observed for the MinimumFillInWeight heuristic in Section 6.2. In order to answer this question, additional experiments were performed using the BPART Class B instances from Section 6.2, but this time compiled by tree clustering using the MinimumCliqueWeight heuristic. The maximal clique sizes resulting from this set of experiments were analyzed using linear regression. The regression results for MinimumCliqueWeight were statistically significant, with an $R^2 = 0.695$, a standard error of 0.906, and an F-ratio of 1822. This regression gives the following empirical results for the maximal number of nodes in a clique h and maximal clique size ℓ: $h = 2.98 \times C/V + 10.5$ and $\ell = 2^h = 2^{2.98 \times C/V + 10.5}$. The 95% confidence interval for the slope of h is $(2.85, 3.12)$ with a p-value of 3.3×10^{-208}. The 95% confidence

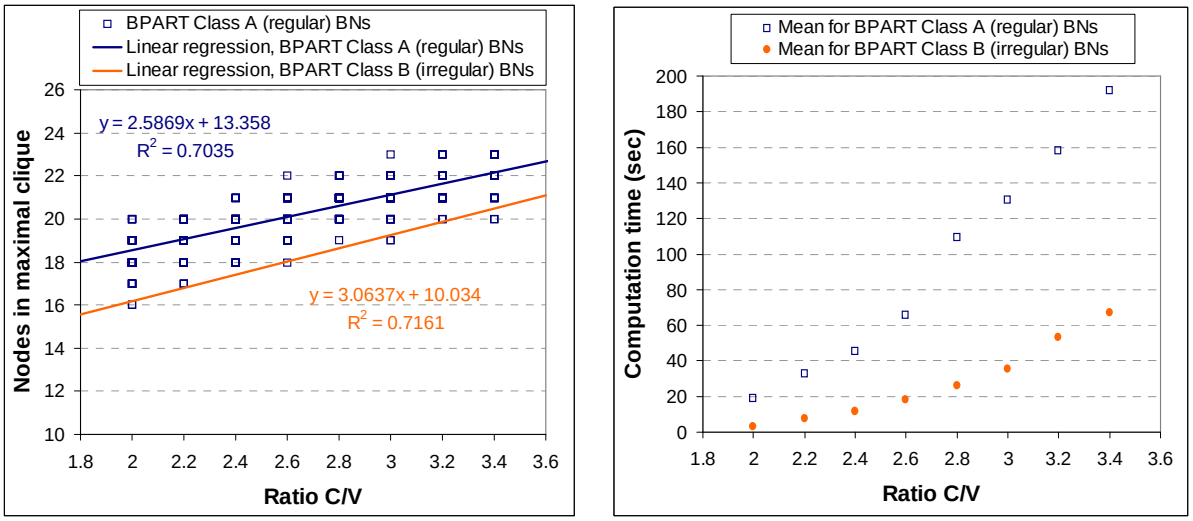

Figure 8: Tree clustering results for BPART BNs with $V = 30$ root nodes and C/V-ratios from $C/V = 2.0$ to $C/V = 3.4$ are shown. The plots reflect both Class B irregular and Class A regular BNs. Left: The number of nodes in the maximal clique is shown as a function of the C/V-ratio. Data points representing Class A regular BPART instances as well as regression results are displayed. The regression line for Class B irregular BPART instances is also included. Right: The mean computation times (in seconds) is plotted as a function of C/V-ratio. Specifically, the sample mean $\bar{x}^r_{\mathbb{T}}$ for the Class A regular BNs and the sample mean $\bar{x}^i_{\mathbb{T}}$ for the Class B irregular BNs are shown for varying levels of the C/V-ratio.

interval for the intercept of the regression line is $(10.13, 10.89)$ with a p-value of 2.9×10^{-273}. These results show that the number of nodes in the maximal clique h grows linearly with C/V-ratio also when the MinimumCliqueWeight heuristic is used. This gives an exponential growth in ℓ with implications similar to what was discussed in Section 6.2 for MinimumFillInWeight.

A second question is whether MinimumCliqueWeight and MinimumFillInWeight produce very different maximal clique sizes. Comparing the respective regression lines, it is clear that MinimumFillInWeight is, on average, slightly superior to MinimumCliqueWeight for the C/V range considered here. One might ask whether the difference is statistically significant. Out of the 800 BN samples, there were 525 instances in which the maximal clique sizes were the same for the two heuristics, 231 instances in which the MinimumFillInWeight was better, and 44 instance in which MinimumCliqueWeight was better. Doing a paired comparison using the sign test, we found a difference between the two heuristics at the 1% significance level; MinimumFillInWeight was indeed better according to this statistical test.

6.4 BPART Class A Networks: Hardness and Graph Regularity

What is the impact, on maximal clique sizes and inference times, of varying the C/V-ratio when generating regular ($R =$ **true**) BPART BNs? To answer this question and to complement our analysis of regularity, as presented in Section 5.4, we created BNs using the BPART construction while varying the C/V-ratio, as earlier, but generating Class A (regular) BNs. Specifically, the signature used for BN construction was BPART(`uniform`, `or`, *V, C, 2,* **true**, *3*). A similar signature BPART(`uniform`, `or`, *V, C, 2,* **false**, *3*) was used to construct Class B (irregular) BNs in Section 6.2. The essential difference between the two signatures is that in the relaxed Class A regular case, each root node has essentially the same number of children, while in the Class B irregular case, the number of children is exactly distributed as $b(C, P/V)$ (see Theorem 20) and approximately distributed as $b(CP, 1/V)$ (see Theorem 21).

Maximal clique size and inference time results for these experiments, where leaf nodes again were clamped to 1 during Hugin inference, are presented in Table 5 and in Figure 8. Table 5's columns $\bar{x}^r_{\mathbb{L}}$ and $\bar{x}^i_{\mathbb{L}}$ summarize the bottom rows of Table 7 and Table 4 respectively. Table 7 presents in detail how the maximal

Maximal clique size		C/V-ratio of BPART Class A (regular) BNs								
Nodes $\mathbb{H}$	**State space $\mathbb{L}$**	**2.0**	**2.2**	**2.4**	**2.6**	**2.8**	**3.0**	**3.2**	**3.4**	**Total**
16	65,536	3								3
17	131,072	14	4							18
18	262,144	44	16	10	1					71
19	524,288	31	44	31	14	1	2			123
20	1,048,576	8	36	48	48	28	13	3	3	187
21	2,097,152			11	36	57	53	45	17	219
22	4,194,304				1	14	31	48	57	151
23	8,388,608						1	4	23	28
Number of BN instances		100	100	100	100	100	100	100	100	800
Mean number of nodes $\bar{x}_{\mathbb{H}}$		18.27	19.12	19.60	20.22	20.84	21.16	21.53	22.00	20.34
Mean size (in 1000s) $\bar{x}_{\mathbb{L}}$		382.1	655.4	923	1,376	2,081	2,642	3,324	4,708	2011

Table 7: Experimental tree clustering results showing statistics for BNs generated by BPART with input parameters $R =$ **true** (Class A regular case), $V = 30$, and C ranging from 60 to 102. The number of nodes and size of the maximal clique is shown for varying C/V ratios. The smallest maximal clique contains $h = 16$ nodes (for $C/V = 2.0$), while the largest maximal clique contains $h = 23$ nodes (for $C/V = 3.4$).

clique size and number of nodes in the maximal clique vary with the C/V-ratio.

Figure 8 summarizes linear regression results, showing how the number of nodes in the maximal clique, $\bar{x}_{\mathbb{H}}$, varies linearly with the C/V-ratio. The regression gives these results for the maximal number of nodes in a clique h and maximal clique size ℓ: $h = 2.59 \times C/V + 13.4$ and $\ell = 2^h = 2^{2.59 \times C/V + 13.4}$. The regression, which is based on 800 observations, is statistically significant, with $R^2 = 0.703$, an F-ratio of 1893, and a p-value of 7.5×10^{-213}. The 95% confidence interval for the slope of the regression line is $(2.47, 2.70)$. For the parameters used here, the result for h shows that for the BPART Class A construction, the number of nodes in the maximal clique grows linearly with C/V-ratio. This linear growth of h explains, similar to what was discussed in Section 6.2, the approximately exponential growth of ℓ and increasing inference times.

Can we conclude that the maximal clique sizes for BPART Class A and Class B BNs are significantly different? Comparing with the corresponding results for Class B BNs in Table 4, we note that the regular Class A instances in Table 7 are skewed towards higher maximal clique sizes. A statistical t-test for two population means, assuming unknown population means and variances, was performed for the maximal clique sizes of the Class A and Class B samples. With $t = 26.5$, and critical values for t of 1.65 (one-tail) and 1.96 (two-tail), the null-hypothesis that the two population means are equal, $\mu^i_{\mathbb{L}} = \mu^r_{\mathbb{L}}$, is rejected. This result, along with Table 7 and Figure 8, sheds additional light on the difference in computation times between Class A and Class B BNs. This result is also consistent with the theoretical analysis in Section 5.4.

The regular BPART BNs on average required three to four times more time for computation compared to irregular BPART BNs. More formally, consider the sample mean computation times for the regular case, $\bar{x}^r_{\mathbb{T}}$, versus the irregular case, $\bar{x}^i_{\mathbb{T}}$. From the ratios $\bar{x}^r_{\mathbb{T}}/\bar{x}^i_{\mathbb{T}}$ one can easily determine that HUGIN consistently was faster on irregular than on regular BNs: $2.85 \leq \bar{x}^r_{\mathbb{T}}/\bar{x}^i_{\mathbb{T}} \leq 4.17$ in Table 5. The results for maximal clique size sample means, $\bar{x}^r_{\mathbb{L}}$ and $\bar{x}^i_{\mathbb{L}}$, were in line with the results for $\bar{x}^r_{\mathbb{T}}$ and $\bar{x}^i_{\mathbb{T}}$. From the ratios between these sample means, $\bar{x}^r_{\mathbb{L}}/\bar{x}^i_{\mathbb{L}}$, we see that there was a corresponding larger sample mean for Class A regular BNs than for Class B irregular BNs: $3.01 \leq \bar{x}^r_{\mathbb{L}}/\bar{x}^i_{\mathbb{L}} \leq 5.39$ in Table 5.

In summary, we have shown that regularity is a factor not only for iterated belief propagation when used on information theory BNs [28,48], but also for tree clustering on closely related BNs. Confirming and adding detail to the analytical results, we have shown empirically that regular Class A BNs are harder, on average, than irregular Class B BNs due to their larger maximal cliques.

6.5 MPART Class B Networks: Hardness and Conditional Probabilities

In Section 6.2, Section 6.3, and Section 6.4 we reported results for BPART BNs. What happens if several of the parameters as well as the topology are changed such that quite different BNs are generated? Is the C/V-ratio still important when BNs are generated using the MPART construction? These questions are

BN parameters				Statistics for MPART BNs							
V	C	V+C	C/V	$m_{\mathbb{T}}$	$\bar{x}_{\mathbb{C}}$	$s_{\mathbb{C}}$	$\bar{x}_{\mathbb{P}}$	$s_{\mathbb{P}}$	$\bar{x}_{\mathbb{T}}$	$s_{\mathbb{T}}$	$\bar{x}_{\mathbb{L}}$
146	110	256	0.75	0.156	0.148	0.013	0.010	0.008	0.158	0.014	0.21
136	120	256	0.88	0.203	0.195	0.047	0.014	0.009	0.209	0.050	1.36
126	130	256	1.03	0.281	0.268	0.071	0.047	0.046	0.315	0.109	13.40
116	140	256	1.21	0.688	0.566	0.359	0.300	0.352	0.867	0.710	97.08
106	150	256	1.42	4.204	3.439	3.594	3.174	3.617	6.614	7.208	1,186
104	152	256	1.46	8.039	5.565	5.121	5.365	5.163	10.929	10.281	1,951
102	154	256	1.51	12.080	10.030	10.935	9.852	11.103	19.882	22.035	3,509
100	156	256	1.56	20.420	17.811	22.477	17.585	22.568	35.396	45.044	7,229

Table 8: Inference times (in seconds) for a total of 800 BNs generated using the MPART construction are shown. Median computation time $m_{\mathbb{T}}$, mean compilation time $\bar{x}_{\mathbb{C}}$ with standard deviation $s_{\mathbb{C}}$, mean propagation time $\bar{x}_{\mathbb{P}}$ with standard deviation $s_{\mathbb{P}}$, mean computation time $\bar{x}_{\mathbb{T}}$ with standard deviation $s_{\mathbb{T}}$, and the mean size of the maximal clique $\bar{x}_{\mathbb{L}}$ are also presented.

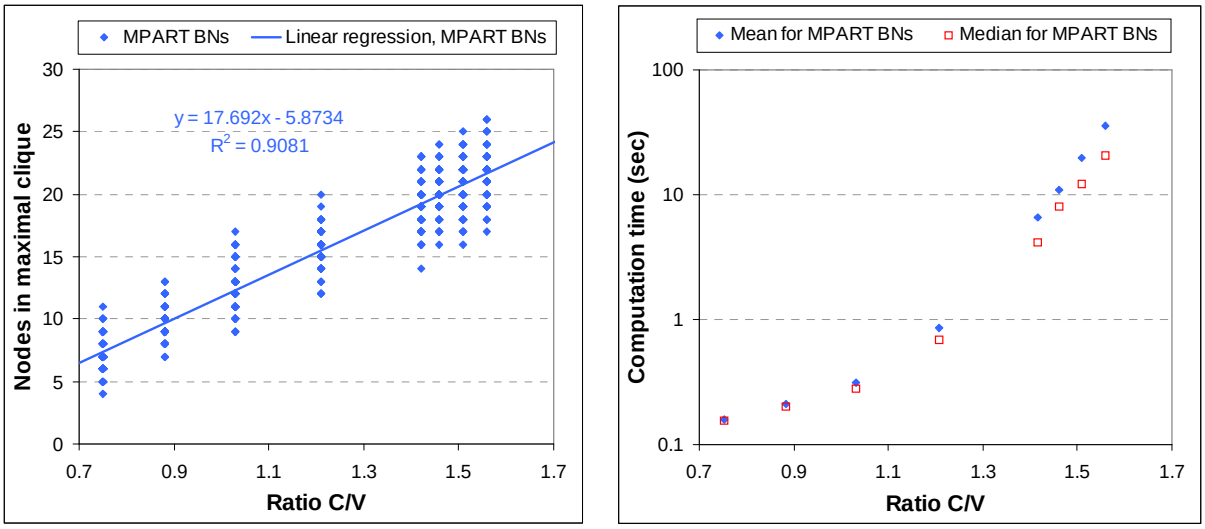

Figure 9: For BNs generated using the MPART construction, these plots display tree clustering results as function of C/V-ratio. Here, the C/V-ratio ranges from $C/V = 0.75$ to $C/V = 1.56$. These results are for non-root nodes C ranging from $C = 110$ to $C = 156$, and for root nodes V ranging from $V = 146$ to $V = 100$. Left: The number of nodes in the maximal clique is plotted as a function of the C/V-ratio. This scatter plot shows BN instances as well as linear regression results. Right: The sample mean computation times $\bar{x}_{\mathbb{T}}$ and sample median computation times $m_{\mathbb{T}}$ are displayed in this log-plot.

investigated empirically in this section, using a data set containing 800 MPART samples. The signature used to generate BNs was MPART(`random`, `random`, *V, C, 2,* **false**, *2*) with $C \in \{110, \ldots, 156\}$ and $V = 256 - C$, giving a C/V-ratio ranging from 0.75 to 1.56. Following Kask and Dechter [41], we kept N constant, varied V, and determined C by setting $C = N - V$. The effect of this is a non-linear change in the C/V-ratio as a function of change in C. Also note that there are $P = 2$ parents per non-root node rather than $P = 3$ as used in the experiments above; other differences are that `random` CPTs were used and nodes were not clamped during inference.

Figure 9 plots results in the form of individual data points — number of nodes in the maximal clique of a sample BN as function of C/V-ratio — and also displays linear regression results. The following empirical results for h and ℓ were obtained: $h = 17.7 \times C/V - 5.87$ and $\ell = 2^h = 2^{17.7 \times C/V - 5.87}$ The regression is statistically significant, with an $R^2 = 0.908$, an F-ratio of 7882, and a p-value of 0. The 95% confidence interval for the slope of the regression line for h is $(17.3, 18.1)$. As before, the regression results show that h

Maximal clique size		C/V-ratio of MPART BNs							
Nodes $\mathbb{H}$	**State space $\mathbb{L}$**	**0.75**	**0.88**	**1.03**	**1.21**	**1.42**	**1.46**	**1.51**	**1.56**
4	16	1							
5	32	9							
6	64	25							
7	128	32	3						
8	256	17	17						
9	512	12	23	3					
10	1,024	3	29	7					
11	2,048	1	18	18					
12	4,096		7	18	4				
13	8,192		3	31	3				
14	16,384			7	10	1			
15	32,768			8	26				
16	65,536			7	22	4	1	1	
17	131,072			1	23	11	4	3	1
18	262, 144				10	22	10	9	3
19	524,288				1	18	16	10	7
20	1,048,576				1	25	33	21	17
21	2,097,152					10	21	21	22
22	4,194,304					5	9	20	20
23	8,388,608					4	4	9	18
24	16,777,216						2	5	5
25	33,554,432							1	4
26	67,108,864								3
Number of BN instances		100	100	100	100	100	100	100	100
Mean number of nodes $\bar{x}_{\mathbb{H}}$		7.08	9.75	12.61	15.79	19.15	20.07	20.72	21.58
Mean size (in 1000s) $\bar{x}_{\mathbb{L}}$		0.21	1.36	13.40	97.08	1,186	1,951	3,509	7,229

Table 9: Experimental results for tree clustering on BNs generated by the MPART construction. Here, $C + V = 256$ for all C/V ratios, with C ranging from 110 to 156 and V ranging from 100 to 146. The number of nodes in and sizes of the maximal cliques are shown for a number of instances for varying C/V ratios. The smallest maximal clique contains $h = 4$ nodes (for $C/V = 0.75$), while the largest maximal clique contains $h = 26$ nodes (for $C/V = 1.56$).

increases linearly with C/V-ratio, giving an approximately exponential growth in ℓ with the C/V-ratio. An approximately exponential growth in maximal clique size ℓ explains, to a large extent, the inference times reported in Table 8 and to the right in Figure 9. Table 9 provides further details on how statistics of the maximal clique size, including $\bar{x}_{\mathbb{H}}$ and $\bar{x}_{\mathbb{L}}$, increase with the C/V-ratio.

The results for MPART were in many respects similar to the results for BPART. However, there was slightly greater variation in the MPART case, as can be seen by comparing the sample standard deviations or by comparing Table 4 and Table 9. Also, the growth in maximal clique size and thus in computation time as a function of C/V-ratio was in fact stronger for MPART than for BPART. This is clearly in part due to the different CPTs used, following the discussion in Section 5.6. For MPART, CPT values were picked from a `random` distribution, giving just one propagation, while for BPART multiple propagations were in general needed (see Section 5.6).

We also note that these MPART BNs are similar to the multipartite BNs investigated by Kask and Dechter [41]. They also randomly generated BNs with $C/V \leq 0.75$ and $N = V + C = 256$ nodes. For $C/V \leq 0.75$, our MPART samples were relatively easy for HUGIN to solve on the average. For example, the highest C/V-ratio used by Kask and Dechter, $C/V = 0.75$, had for MPART a sample mean inference time of $\bar{x}_{\mathbb{T}} = 0.1579$ seconds (see Table 8, row $C/V = 0.75$) and a maximal clique size sample mean $\bar{x}_{\mathbb{L}} = 210$ (see Table 9, column $C/V = 0.75$). For MPART, the largest maximal clique size in the $C/V = 0.75$ subsample is $\ell = 2,048$. When using MPART BNs with $C/V \leq 0.75$ in experiments with other algorithms, one should

keep in mind that these BNs are in fact relatively easy for HUGIN to solve on average. On the other hand, with $C/V \geq 1.50$ these MPART BNs are typically quite challenging.

7 Conclusion and Future Work

The performance of Bayesian network (BN) inference algorithms has in previous research been empirically evaluated using BN instances from applications. To complement such experiments, randomly generated BNs have also been used [7, 17, 34, 41, 56, 69, 70]. We believe that a certain amount of care is required when randomly generating BNs. The generation algorithms need to provide "knobs" for controlling the difficulty of the generated instances. The synthetic BNs need to be idealized, to support analysis, but at the same time they also need to be somewhat realistic and relevant to applications.

We have developed, based on previous research, a paradigm for systematically generating increasingly hard random instances for BN inference. One of the classes of BNs, the bipartite BPART networks generated by the BPART algorithm, extends research on generating hard instances for satisfiability problems [55]. Here, we have exploited the relationship between computing an MPE and finding a satisfying assignment of a corresponding CNF formula to construct hard instances for the MPE problem. These BPART networks are also similar in structure to application BNs from medicine [67] and information theory [27, 28, 49], and our results should be relevant to these two areas of research. The other class of BNs, the MPART networks, is closely related to an approach of Kask and Dechter [41]. For these MPART networks we have analyzed the relationship to the BPART BNs.

Different algorithms for BN inference in general and MPE computation in particular have been investigated using our experimental paradigm [51]. In this article, we have focused on the HUGIN tree clustering algorithm [2, 37, 39, 46]. HUGIN is one of the best probabilistic inference algorithms available, and using this algorithm, and systematically varying some structural and distributional parameters of the synthetic Bayesian networks, we have shown how HUGIN inference is impacted. In particular, we randomly generated BNs by using the BPART and MPART algorithms and varying these parameters:

- the ratio C/V of the number of non-root nodes, C, to the number of root nodes, V, in the BN,
- the regularity structure of the BN's underlying graph, and
- the conditional distribution tables (CPTs) of the nodes in the BN.

We have carefully studied how these parameters impact inference hardness, expressed in terms of maximal clique size or inference time, in tree clustering. We identified, for HUGIN, an easy-hard-harder pattern for both the BPART and MPART networks. For both classes, generating random networks can result in very easy instances. On the other hand, by carefully varying parameters along certain dimensions, one can construct BNs that existing tree clustering algorithms cannot handle. Specifically, we have found that the C/V-ratio, through its impact on maximal clique size, is an indication of the inferential hardness of the network under suitable structural and parametric assumptions. As the C/V-ratio grows, even when fixing the number $N = C + V$ of nodes in the network, the inference problem becomes harder, on average, due to an increasing maximal clique size. We now summarize our results in more detail.

Four structurally distinct classes of BNs were identified in this article — Class A, Class B, Class C, and Class D BNs — of which we more closely investigated Class A (regular) and Class B (irregular) BNs. For Class B irregular BPART BNs, our analysis showed an easy-hard-harder pattern with increasing C/V-ratio. Through regression analysis, a linear relationship was established between the C/V-ratio and the mean number of nodes in the maximal clique, giving an approximately exponential growth in maximal clique size which was also reflected in HUGIN tree clustering inference time. Results were similar for tree clustering's MINIMUMFILLINWEIGHT and MINIMUMCLIQUEWEIGHT triangulation heuristics, with the former being slightly but significantly better in terms of optimizing maximal clique size.

A second structural parameter we considered, again using the BPART construction, was the regularity of the underlying graph of the BN. Our analysis showed that regular Class A BPART BNs should be harder than irregular Class B BPART BNs, and this expectation was confirmed in experiments. A regression analysis exhibited exponential growth in maximal clique size as a function of C/V-ratio, similar to the irregular case. We also showed experimentally, while keeping other parameters of the generated networks

fixed, that maximal clique size sample means were from 3.0 to 5.4 times greater for Class A BNs compared to Class B BNs. These results also shed new light on the computational benefit of irregularity in information theory BNs [27, 28, 48].

Our studies with MPART used quite different input parameters for BN sample generation. Still, the regression results bear resemblance to BPART's regression results. There turned out to be approximately linear growth in the mean number of nodes in the maximal clique as a function of C/V, giving an approximately exponential growth in maximal state space size, from which one can expect an approximately exponential growth in inference times. In fact, the graph of averages showed slightly stronger than exponential growth. It also turned out that previous work with a similar class of BNs, using $C/V \leq 0.75$ [41], correspond to a relatively easy region of the MPART distribution.

Since the complexity of most exact Bayesian network inference algorithms — including tree clustering algorithms, conditioning algorithms, and elimination algorithms — depend on treewidth or optimal (minimal) maximal clique size [5, 17, 20, 21], we believe that our results are of interest to researchers investigating exact Bayesian network inference algorithms. Our empirical results are limited by the fact that we employed the suboptimal MinimumFillInWeight and MinimumCliqueWeight triangulation heuristics; however due to the hardness of BN inference problems [1, 14, 60, 66] and the widespread adoption of these and similar heuristics we believe that our results are of significant interest.

In addition to showing some interesting aspects of Hugin and the tree clustering approach, we believe this line of research to be essential so that valid experimental evaluation of other algorithms can be performed. For instance, in related work we have used BPART and MPART networks to benchmark stochastic local search and have found strong dependence of inference time on the C/V-ratio, qualitatively similar to the results reported here [51, 53]. The approach of constructing synthetic BNs has guided us in developing a stochastic local search approach to the point where it outperforms Hugin on certain synthetic instances as well as on certain application BNs [51, 53]. Similar research would help in developing a better understanding of different algorithms under varying conditions. In particular, our results can be used by other researchers to focus their work on areas in the space of Bayesian networks where we found time- or space-consumption to be relatively high for tree clustering.

This research can be extended in several other directions. It is important to develop a better understanding of other dimensions (beyond C/V, regularity and different CPT types), intermediate cases, and other inference algorithms. While we have studied extreme cases in a few dimensions, it is essential to perform similar studies for other dimensions. Other important areas include improved analytical models for clique tree cluster formation and growth, loop formation, interactions between loops, and the placement of fill-in edges in the moral graphs induced by BNs. A better understanding of the relationship between synthetically generated networks and networks from applications would also be a natural extension of this research.

Acknowledgments

The research reported here was largely conducted while Ole J. Mengshoel was at the University of Illinois, Urbana-Champaign. Ole J. Mengshoel and David C. Wilkins gratefully acknowledge support in part by ONR Grant N00014-95-1-0749, ARL Grant DAAL01-96-2-0003, and NRL Grant N00014-97-C-2061. Dan Roth gratefully acknowledges the support of NSF grants IIS-9801638 and SBR-987345. Vadim Bulitko, David Fried, Song Han, William Hsu, Brent Spillner, and anonymous reviewers are acknowledged for comments related to this work. David Fried, Song Han, and Misha Voloshin are acknowledged for their co-development of the software used in the experiments.

References

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

[2] S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen. HUGIN—a shell for building Bayesian belief universes for expert systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 2, pages 1080–1085, Detroit, MI, August 1989.

[3] S. Andreassen, M. Woldbye, B. Falck, and S.K. Andersen. MUNIN – A causal probabilistic network for interpretation of electromyographic findings. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 366–372, Milan, Italy, August 1987.

[4] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18(2):155–193, April 1979.

[5] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey. *BIT*, 25:2–23, 1985.

[6] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Meththods*, 8:277–284, 1987.

[7] A. Becker and D. Geiger. Approximation algorithms for the loop cutset problem. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 60–68, San Francisco, CA, 1994.

[8] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, New York, 1972.

[9] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla. Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Engineering and System Safety*, 71(3):249–260, 2001.

[10] N. Chater and C. D. Manning. Probabilistic models of language processing and acquisition. *TRENDS in Cognitive Sciences*, 10(7):335–344, 2006.

[11] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 331–337, Sidney, Australia, 1991.

[12] M. Ciaramita and M. Johnson. Explaining away ambiguity: Learning verb selectional preference with Bayesian networks. In *18th International Conference on Computational Linguistics (COLING-00)*, pages 187–193, Saarbrücken, Germany, 2000.

[13] D. Clark, J. Frank, I. Gent, E. MacIntyre, N. Tomov, and T. Walsh. Local search and the number of solutions. In *Proceedings of the Second International Conference on Principles and Practices of Constraint Programming*, volume 1118 of *LNCS*, pages 119–133, 1996.

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

[15] P. Crescenzi and V. Kann. A compendium of NP optimization problems. Technical Report SI/RR-95/02, Dipartimento di Scienze dell'Informazione, Universita di Roma "La Sapienza", Roma, Italy, 1995.

[16] A. Darwiche. Conditioning methods for exact and approximate inference in causal networks. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 99–107, Montreal, Canada, 1995.

[17] A. Darwiche. Recursive conditioning. *Artificial Intelligence*, 126(1-2):5–41, 2001.

[18] A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

[19] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[20] R. Dechter and Y. El Fattah. Topological parameters for time-space tradeoff. *Artificial Intelligence*, 125(1-2):93–118, 2001.

[21] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34(1):1–38, 1987.

[22] F. J. Diez. Local conditioning in Bayesian networks. *Artificial Intelligence*, 87(1-2):1–20, 1996.

[23] G. A. Dirac. Some theorems on abstract graphs. In *Proc. London Math. Soc.*, volume 2, pages 69–81, 1952.

[24] Hugin Expert. *Hugin API: Reference Manual*. Hugin Expert, 2004.

[25] J. Franco and M. Paull. Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5:77–87, 1983.

[26] J. Frank, P. Cheeseman, and J. Stutz. When gravity fails: Local search topology. *Journal of Artificial Intelligence Research*, 7:249–281, 1997.

[27] B. J. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambridge, MA, 1998.

[28] R. G. Gallager. Low density parity check codes. *IRE Transactions on Information Theory*, 8:21–28, Jan 1962.

[29] X. Ge, D. Eppstein, and P. Smyth. The distribution of loop lengths in graphical models for turbo decoding. *IEEE Trans. Information Theory*, 47(6):2549–2553, September 2001.

[30] P. W. Gu, J. Purdom, J. Franco, and B. W. Wah. *Satisfiability Problem: Theory and Applications*, chapter Algorithms for the Satisfiability SAT Problem: A Survey, pages 19–152. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1997.

[31] M. Henrion. Search-based methods to bound diagnostic probabilities in very large belief networks. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-91)*, pages 142–150, University of California at Los Angeles, CA, 1991.

[32] E. J. Horvitz, H. J. Suermondt, and G. F. Cooper. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence (UAI-89)*, pages 182–193, Windsor, Ontario, 1989. Morgan Kaufmann.

[33] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15:225–263, 1996.

[34] J. S. Ide and F. G. Cozman. Generating random Bayesian networks. In *Proceedings on 16th Brazilian Symposium on Artificial Intelligence*, pages 366–375, Porto de Galinhas, Brazil, November 2002.

[35] J. S. Ide, F. G. Cozman, and F. T. Ramos. Generating random Bayesian networks with constraints on induced width. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 323–327, 2004.

[36] W. Jackson. Hamilton cycles in regular 2-connected graphs. *Journal Comb. Theory Ser. B*, 29:27–46, 1980.

[37] F. V. Jensen. *An Introduction to Bayesian Networks*. Springer-Verlag, New York, 1996.

[38] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *SIAM Journal on Computing*, 4:269–282, 1990.

[39] F. V. Jensen, K. G. Olesen, and S. K. Andersen. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20(5):637–659, August 1990.

[40] P. Jones, C. Hayes, D. Wilkins, R. Bargar, J. Sniezek, P. Asaro, O. J. Mengshoel, D. Kessler, M. Lucenti, I. Choi, N. Tu, and J. Schlabach. CoRAVEN: Modeling and design of a multimedia intelligent infrastructure for collaborative intelligence analysis. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, pages 914–919, San Diego, CA, October 1998.

[41] K. Kask and R. Dechter. Stochastic local search for Bayesian networks. In *Proceedings Seventh International Workshop on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, Jan 1999. Morgan Kaufmann.

[42] U. Kjaerulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 2:7–17, 1992.

[43] I. Kononenko. Inductive and Bayesian learning in medical diagnosis. *Applied Artificial Intelligence*, 7:317–337, 1993.

[44] A. M. C. A. Koster, H. L. Bodlaender, and S. P. M. van Hoesel. Treewidth: Computational experiments. In H. Broersma, U. Faigle, J. Hurink, and S. Pickl, editors, *Electronic Notes in Discrete Mathematics*, volume 8. Elsevier Science Publishers, 2001.

[45] H. Langseth and L. Portinale. Bayesian networks in reliability. *Reliability Engineering and System Safety*, 92(1):92–108, 2007.

[46] S. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society series B*, 50(2):157–224, 1988.

[47] Z. Li and B. D'Ambrosio. Efficient inference in Bayes nets as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11(1):55–81, 1994.

[48] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Improved low-density parity-check codes using irregular graphs and belief propagation. In *International Symposium on Information Theory*, Cambridge, MA, Aug 1998.

[49] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2002.

[50] A. Madsen. Computing MPEs in Hugin. Personal communication, April 2003.

[51] O. J. Mengshoel. *Efficient Bayesian Network Inference: Genetic Algorithms, Stochastic Local Search, and Abstraction*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, April 1999.

[52] O. J. Mengshoel, D. Roth, and D. C. Wilkins. Hard and easy Bayesian networks for computing the most probable explanation. Technical Report UIUCDCS-R-2000-2147, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, January 2000.

[53] O. J. Mengshoel, D. Roth, and D. C. Wilkins. Stochastic greedy search: Computing the most probable explanation in Bayesian networks. Technical Report UIUCDCS-R-2000-2150, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, February 2000.

[54] O. J. Mengshoel and D. C. Wilkins. Raven: Bayesian networks for human-computer intelligent interaction. In M. S. Vassiliou and T. S. Huang, editors, *Computer Science Handbook for Displays*, pages 209–219. Rockwell Scientific Company, 2001.

[55] D. Mitchell, B. Selman, and H. J. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465, San Jose, CA, 1992.

[56] J. D. Park and A. Darwiche. Approximating MAP using local search. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 403–410, Seattle, WA, 2001.

[57] J. Pearl. A constraint - propagation approach to probabilistic reasoning. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 357–369. Elsevier, Amsterdam, Netherlands, 1986.

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

[59] N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of treewidth. *J. Algorithms*, 7:309–322, 1986.

[60] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996.

[61] C. C. Ruokangas and O. J. Mengshoel. Information filtering using Bayesian networks: effective user interfaces for aviation weather data. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces*, pages 280–283, Miami, FL, 2003.

[62] B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 337–343, Seatttle, WA, 1994.

[63] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446, San Jose, CA, July 1992.

[64] R. D. Shachter, S. K. Andersen, and P. Szolovits. Global conditioning for probabilistic inference in belief networks. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 514–522, Seattle, WA, 1994.

[65] P. P. Shenoy. A valuation-based language for expert systems. *International Journal of Approximate Reasoning*, 5(3):383–411, 1989.

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

[67] M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30(4):241–255, 1991.

[68] C. Skaaning Jensen and A. Kong. Blocking Gibbs sampling for linkage analysis in large pedigrees with many loops. Research Report R-96-2048, Department of Computer Science, Aalborg University, Denmark, 1996.

[69] H. J. Suermondt and G. F. Cooper. Probabilistic inference in multiply connected belief networks using loop cutsets. *International Journal of Approximate Reasoning*, 4:283–306, 1990.

[70] R. L. Welch. Real time estimation of Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, pages 533–544, Portland, Oregon, 1996.

[71] M. Yokoo. Why adding more constraints makes a problem easier for hill-climbing algorithms: Analyzing landscapes of CSPs. In *Proceedings of the Third International Conference on Principles and Practice of Constraint Programming*, volume 1330 of *LNCS*, pages 357–370. Springer Verlag, 1997.

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