CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Helsingin yliopiston digitaalinen arkisto

Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)

Bayesian Network Structure Learning with Integer Programming: Polytopes, Facets and Complexity (Extended Abstract)*

James Cussens
University of York
United Kingdom

Matti Järvisalo
University of Helsinki
Finland

Janne H. Korhonen
Aalto University
Finland

Mark Bartlett
University of York
United Kingdom

Abstract

Developing accurate algorithms for learning structures of probabilistic graphical models is an important problem within modern AI research. Here we focus on score-based structure learning for Bayesian networks as arguably the most central class of graphical models. A successful generic approach to optimal Bayesian network structure learning (BNSL), based on integer programming (IP), is implemented in the GOBNILP system. Despite the recent algorithmic advances, current understanding of foundational aspects underlying the IP based approach to BNSL is still somewhat lacking. In this paper, we provide theoretical contributions towards understanding fundamental aspects of *cutting planes* and the related *separation problem* in this context, ranging from NP-hardness results to analysis of polytopes and the related facets in connection to BNSL.

1 Introduction

The study of probabilistic graphical models is a central topic in modern artificial intelligence research. Bayesian networks [Koller and Friedman, 2009] form a central class of probabilistic graphical models. A central problem related to Bayesian networks (BNs) is that of learning them from data. An essential part of this learning problem is aimed at learning the *structure* of a Bayesian network—represented as a directed acyclic graph—that accurately represents the (hypothetical) joint probability distribution underlying the data.

There are two principle approaches to Bayesian network learning: *constraint-based* and *score-based*. In the constraint-based approach [Spirtes *et al.*, 1993; Colombo *et al.*, 2012] the goal is to learn a network which is consistent with conditional independence relations which have been inferred from the data. The *score-based* approach to Bayesian network structure learning (BNSL)—focused on here—treats the BNSL problem as a combinatorial optimization problem of finding a BN structure that optimises a score function for given data.

Learning an optimal BN structure is a computationally challenging problem: even the restriction of the BNSL problem where only *BDe* scores [Heckerman *et al.*, 1995] are allowed is known to be NP-hard [Chickering, 1996]. Due to NP-hardness, much work on BNSL has focused on developing approximate, local search style algorithms [Tsamardinos *et al.*, 2006] that in general cannot guarantee that optimal structures in terms of the objective function are found. Recently, despite its complexity, several advances in *exact* approaches to BNSL have surfaced [Koivisto and Sood, 2004; Silander and Myllymäki, 2006; Cussens, 2011; de Campos and Ji, 2011; Yuan and Malone, 2013; van Beek and Hoffmann, 2015], ranging from problem-specific dynamic programming branch-and-bound algorithms to approaches based on A*-style state-space search, constraint programming, and integer linear programming (IP), which can, with certain restrictions, learn provably-optimal BN structures with tens to hundreds of nodes.

As shown in a recent study [Malone *et al.*, 2014], perhaps the most successful exact approach to BNSL is provided by the GOBNILP system [Cussens, 2011]. GOBNILP implements a *branch-and-cut* approach to BNSL, using state-of-the-art IP solving techniques together with specialised BNSL cutting planes. The focus of this work is on providing further understanding of the IP approach to BNSL from the theoretical perspective.

Viewed as a constrained optimisation problem, a central source of intractability of BNSL is the *acyclicity* constraint imposed on BN structures. In the IP approach to BNSL—as implemented by GOBNILP—the acyclicity constraint is handled in the branch-and-cut framework via deriving specialised cutting planes called *cluster constraints* which were originally introduced by Jaakkola *et al.* [2010]. These cutting planes are found by solving a sequence of so-called *sub-IPs* arising from solutions to linear relaxations of the underlying IP formulation of BNSL without the acyclicity con-

*This paper is an extended abstract of an article in the Journal of Artificial Intelligence Research [Cussens *et al.*, 2017]. The authors gratefully acknowledge financial support from: UK Medical Research Council Grant G1002312 (JC, MB); Senior Postdoctoral Fellowship SF/14/008 from KU Leuven (JC); UK NC3RS Grant NC/K001264/1 (JC); Academy of Finland under grants 251170 COIN Centre of Excellence in Computational Inference Research, 276412, and 284591 (MJ); Research Funds of the University of Helsinki (MJ); and Icelandic Research Fund grant 152679-051 (JHK). Part of the work was done while JHK was at University of Helsinki and at Reykjavik University.

straint. Finding these cutting planes is an example of a *separation problem* for a linear relaxation solution, so called since the cutting plane will separate that solution from the set of feasible solutions to the original (unrelaxed) problem. Understanding fundamental aspects of these cutting planes and the sub-IPs used to find them is important not only from a purely theoretical perspective, but also since it holds out the promise of further improving the efficiency of state-of-the-art approaches to solving BNSL exactly. This is the focus of and underlying motivation for this work.

In this extended abstract we summarise the main results presented in [Cussens *et al.*, 2017]. First, we study the computational complexity of the separation problem solved via sub-IPs. We establish that the sub-IPs are themselves NP-hard to solve. From the practical perspective, this both gives a theoretical justification for applying an exact IP solver to solve the sub-IPs within GOBNILP, and motivates further work on improving the efficiency of the sub-IP solving via either improved exact techniques and/or further approximate algorithms.

Second, we study the *facets* of the convex hull of acyclic digraphs. As a key theoretical result, we show that cluster constraints are in fact facet-defining inequalities of this convex hull. From the more practical perspective, achieved via exhaustive computation, we provide a complete enumeration of facets for low-dimensional convex hulls. Mapping to practice, explicit knowledge of such facets has the potential to further speed up state-of-the-art BNSL solving by integrating (some of) these facets explicitly into the search.

2 BN Structure Learning

We focus on the Integer Programming (IP) approach to exact score-based Bayesian network structure learning. The essence of this technique is to encode the BNSL problem as an equivalent IP formulation. IP solving techniques can then be used to find an optimal solution to the problem and hence an optimal BN.

We restrict attention to *decomposable* score functions, where the score is defined locally by the parent set choices for each random variable, i, in the vertex set V. Specifically, for $i \in V$ and $J \subseteq V \setminus \{i\}$, let $i \leftarrow J$ denote the the pair (i, J), called a *family*. In our framework, we assume that the score function gives a *local score* $c_{i \leftarrow J}$ for each family $i \leftarrow J$. A global score $c(B)$ for each candidate structure (V, B) is then defined as

$$c(B) = \sum_{i \in V} c_{i \leftarrow \mathrm{Pa}(i,B)}, \tag{1}$$

and the task is to find an acyclic digraph (V, B) maximising $c(B)$ over all acyclic digraphs over V.

We allow for the possibility that not all $J \subseteq V \setminus \{i\}$ are allowed as candidate parent sets for i, writing $\mathcal{P}(i)$ to denote the set of parent sets which are allowed. For example, we may only wish to consider parent sets of small cardinality either to reflect prior knowledge or to make solving easier. In fact, restricting parent set cardinality does not make BNSL easier in this sense: *any* BNSL problem can be converted into one where no parent set contains more than two parents, at the expense of creating a polynomial number of auxiliary vertices [Cussens *et al.*, 2017].

MAXIMISE

$$\sum_{i \in V, J \in \mathcal{P}(i)} c_{i \leftarrow J} x_{i \leftarrow J} \tag{2}$$

SUBJECT TO

$$\sum_{J \in \mathcal{P}(i)} x_{i \leftarrow J} = 1 \qquad \forall i \in V \tag{3}$$

$$\sum_{i \in C} \sum_{J \in \mathcal{P}(i): J \cap C = \emptyset} x_{i \leftarrow J} \geq 1 \qquad \forall C \subseteq V, |C| > 1 \tag{4}$$

$$x_{i \leftarrow J} \in \{0, 1\} \qquad \forall i \in V, J \in \mathcal{P}(i) \tag{5}$$

Figure 1: The IP formulation of the BNSL problem.

3 The GOBNILP System

GOBNILP [Cussens, 2011] is a program that has been developed to find optimal Bayesian networks by solving an IP formulation of the BNSL problem using the SCIP IP system [Achterberg, 2007]. The search is implemented as a branch-and-cut approach, the essentials of which are outlined next.

The IP formulation [Bartlett and Cussens, 2017] with which we work is shown in Figure 1. The binary IP variables used, $x_{i \leftarrow J}$, correspond to node i having the set of nodes J as parents in the network. If set to 1, it means that J is the parent set of i in the network; if set to 0, J is not the parent set of i.

The objective function follows from the global score. For any particular valid assignment of values to variables, the objective function sums the local scores for the chosen families (i.e. those where $x_{i \leftarrow J} = 1$) and ignores the others. This corresponds directly to the definition of the global score for that network, as given in Equation 1.

Equation 3 enforces the constraint that each node has exactly one parent set, while inequalities (4) prevent any solution in which there would exist a cycle in the resulting graph.

Inequalities (4) are just one way that cycles can be prevented [Cussens, 2010; Peharz and Pernkopf, 2012; Cussens *et al.*, 2013]. These particular constraints are known as *cluster constraints* [Jaakkola *et al.*, 2010] as each corresponds to the constraint that any cluster (set) of nodes must have at least one node with no parents in that cluster. An illustration of how acyclic graphs satisfy all cluster constraints but cyclic graphs do not is given in Figure 2.

GOBNILP begins with a relaxed version of the IP problem in which (i) the integrality constraints on all variables are removed, and (ii) all cluster constraints are removed. This relaxed problem is first solved, giving an optimal relaxed solution. In almost all cases, this relaxed solution is not a valid solution to the original IP problem due to violating cluster constraints

One or more cluster constraints that are violated by the relaxed solution are identified and added to the relaxed problem, and this modified relaxed problem is solved again.

The process of solving the relaxed problem and then adding violated cluster constraints repeats until the relaxed

Figure 2: An acyclic and a cyclic graph for vertex set $\{a, b, c, d\}$. For each cluster of vertices C where $|C| > 1$ let $f(C)$ be the number of vertices in C who have no parents in C. Abbreviating e.g. $\{a, b\}$ to ab, for the left-hand graph we have: $f(ab) = 1$, $f(ac) = 1$, $f(ad) = 2$, $f(bc) = 1$, $f(bd) = 2$, $f(cd) = 1$, $f(abc) = 1$, $f(abd) = 2$, $f(acd) = 1$, $f(bcd) = 1$ and $f(abcd) = 1$. For the right-hand graph we have: $f(ab) = 1$, $f(ac) = 1$, $f(ad) = 2$, $f(bc) = 2$, $f(bd) = 1$, $f(cd) = 1$, $f(abc) = 1$, $f(abd) = 1$, $f(acd) = 1$, $f(bcd) = 1$ and $f(abcd) = 0$. The cluster constraint for cluster $\{a, b, c, d\}$ is violated by the right-hand graph since these vertices form a cycle.

solution no longer breaks any cluster constraints. If the solution thus obtained assigns integer values (i.e. 0 or 1) to all variables, then an optimal solution to the problem has been found. Alternatively (and more commonly), if the relaxed solution contains one or more variables with fractional values then the solver branches on one of the variables with a fractional value, forming one subproblem in which this variable is fixed to 0 and one in which it is fixed to 1. The algorithm continues in this manner (cutting and then branching) for each of these problem instances.

4 Complexity of the GOBNILP Sub-IPs

As GOBNILP adds cluster constraints as cutting planes, it is necessary to detect which of these constraints are violated by the solution to the relaxed problem, or alternatively to determine that the solution respects all cluster constraints. In GOBNILP one of the techniques used to do this is through a *sub-IP* — a second IP that yields a solution that corresponds to a violated cluster constraint if one exists or is infeasible if the current solution obeys all cluster constraints.

Suppose that the solution to the current relaxed problem is x^*. For each variable $x^*_{i \leftarrow J} > 0$, the sub-IP will contain an associated binary variable $y_{i \leftarrow J}$. If the solution to the sub-IP sets one of these variables to 1, it indicates that the corresponding variable in the original IP is included in the cluster constraint found. Additionally, for each $i \in V$, a binary variable y_i is created. Variables of this type which are set to 1 in the solution indicate that the associated node is a member of the cluster which defines the cut.

As shown in [Cussens, 2011; Bartlett and Cussens, 2017], the sub-IP in Figure 3 either detects a cluster constraint to add as a cut if this is possible, or is infeasible if no such cut exists.

GOBNILP spends a considerable proportion of its run time solving this problem. It is therefore an important issue to understand the complexity of this problem. In particular, is there an algorithm with polynomial time complexity that can either detect a cluster constraint to add as a cutting plane or determine with certainty that none exists? A polynomial-time algorithm could potentially be used to significantly speed up

MAXIMISE

$$\sum_{i,J:x^*_{i \leftarrow J}>0} x^*_{i \leftarrow J} \cdot y_{i \leftarrow J} - \sum_{i \in V} y_i \tag{6}$$

SUBJECT TO

$$y_{i \leftarrow J} \Rightarrow y_i \qquad \forall y_{i \leftarrow J} \tag{7}$$

$$y_{i \leftarrow J} \Rightarrow \bigvee_{j \in J} y_j \qquad \forall y_{i \leftarrow J} \tag{8}$$

$$\sum_{i,J:x^*_{i \leftarrow J}>0} x^*_{i \leftarrow J} \cdot y_{i \leftarrow J} - \sum_{i \in V} y_i > -1 \tag{9}$$

$$y_{i \leftarrow J}, y_i \in \{0, 1\} \tag{10}$$

Figure 3: The GOBNILP Sub-IP.

GOBNILP. While certain polynomial-time solvable cases can be identified — specifically, when x^* is integer-valued and so represents a (possibly cyclic) directed graph — the complexity of solving the sub-IPs in the general case has been an open question.

We show that the task solved by the sub-IPs, which we call the *weak separation problem* for BNSL, is in fact NP-hard. Formally, we define the weak separation problem for BNSL as follows: given a solution x^* to a relaxed problem, find a *separating cluster* $C \subseteq V$, $|C| > 1$, for which

$$\sum_{i \in C} \sum_{J \in \mathcal{P}(i): J \cap C \neq \emptyset} x^*_{i \leftarrow J} > |C| - 1, \tag{11}$$

or establish that no such C exists.

Theorem 1. *The weak separation problem for BNSL is NP-hard, even when restricted to instances where each parent set has size at most* 2*, that is,* $J \in \mathcal{P}(i)$ *for all* $i \in V$ *only if* $|J| \leq 2$.

Theorem 1 is proven by a reduction from vertex cover. Informally, the main idea of the proof is to represent each edge $\{u, v\} \in E$ in the original vertex cover instance $G = (V, E)$ by assigning weight $x^*_{s \leftarrow \{u,v\}} = 1/|E|$ in the new instance, where s is a node not appearing in V, as illustrated in Figure 4. Clearly, $U \subseteq V$ is a vertex cover in G if and only

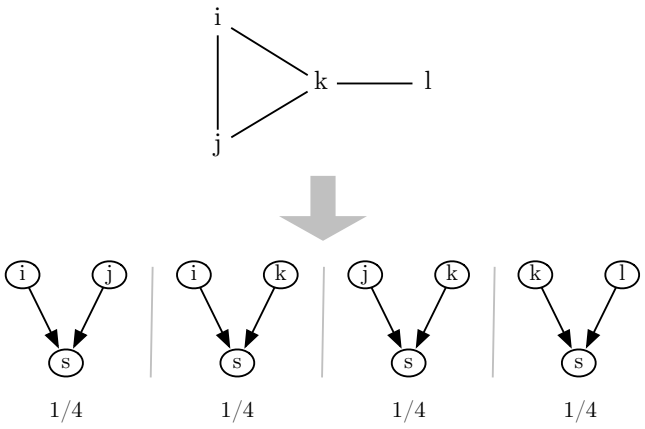

Figure 4: The basic gadget of the reduction in Theorem 1.

if the total weight of terms $x^*_{s\leftarrow\{u,v\}}$ such that U intersects the parent set is 1. In other words, the summation constraint $J \cap C \neq \emptyset$ in inequality (11) is used to model the fact that a vertex cover is a hitting set for edges. Finally, we duplicate this construction for multiple new nodes s, and use terms $x^*_{v\leftarrow\{s\}}$ to control the size of the vertex cover; for full details, see [Cussens *et al.*, 2017].

5 BNSL Facets

Let $\mathcal{G}$ be the set of (family-variable encoded) acyclic digraphs for some vertex set V. $P_{\mathrm{F}}(V, \mathcal{P}_V)$ denotes the *convex hull* of $\mathcal{G}$: the set of all convex combinations of elements of $\mathcal{G}$, or equivalently the smallest convex set containing $\mathcal{G}$. We call $P_{\mathrm{F}}(V, \mathcal{P}_V)$ the *family-variable polytope*[1]. It is a polytope with $\mathcal{G}$ as its vertices.

There is a set of linear inequalities which provide a minimal representation of the family-variable polytope. Such inequalities are known as *facet-defining*. To get an intuition of what a facet is, consider the cube with vertices

$$\begin{aligned}&(x=0, y=0, z=0)\,,\\ &(x=0, y=0, z=1)\,,\\ &\ldots,\\ &(x=1, y=1, z=1)\,.\end{aligned}$$

This is a 3-d polytope with 6 vertices and 6 *facets* defined by the 6 facet-defining inequalities

$$\begin{aligned}&x \geq 0\,, && x \leq 1\,, && y \geq 0\,,\\ &y \leq 1\,, && z \geq 0\,, && z \leq 1\,.\end{aligned}$$

As we discuss in Section 6, when $|V| \leq 4$ it is possible to compute the set of all facet-defining inequalities of the family-variable polytope. For bigger vertex sets this is infeasible, however it remains important to identify facet-defining inequalities since they make the best cuts. In [Cussens *et al.*, 2017], we show that *cluster constraints are facet-defining*. In fact, the stronger result that all k*-cluster constraints*, as defined in [Cussens, 2011], are facet-defining is given.

Theorem 2. *For any $C \subseteq V$ and any positive integer $\kappa < |C|$, the valid inequality*

$$\sum_{i\in C} \sum_{J\subseteq V\setminus\{i\}:|J\cap C|\geq\kappa} x_{i\leftarrow J} \leq |C| - \kappa \qquad (12)$$

is facet-defining for the family variable polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$.

We also show in [Cussens *et al.*, 2017] that any facet-defining inequality for a vertex set V can be 'lifted' to generate a facet-defining inequality for a bigger vertex set V':

Theorem 3. *Let $\mathcal{P}_V(i) := 2^{V\setminus\{i\}}$ for all $i \in V$. Let*

$$\sum_{i\in V} \sum_{J\in\mathcal{P}_V(i), J\neq\emptyset} \alpha_{i\leftarrow J} x_{i\leftarrow J} \leq \beta \qquad (13)$$

be a facet-defining inequality of the family variable polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$ which is not a lower bound on a variable. Let V' be a node set such that $V \subseteq V'$. Then

$$\sum_{i\in V} \sum_{J\in\mathcal{P}_V(i), J\neq\emptyset} \alpha_{i\leftarrow J} \left(\sum_{J':J\subseteq J'\subseteq V'\setminus\{i\}} x_{i\leftarrow J'} \right) \leq \beta \quad (14)$$

is facet-defining for $P_{\mathrm{F}}(V', \mathcal{P}_{V'})$ and is not a lower bound on a variable.

Theorems 2 and 3 concern family-variable polytopes when all parent sets are allowed for each vertex. Since in practical BNSL it is often necessary (or desirable) to limit the choice of parent sets in some way, it is important to be able to determine when a facet-defining inequality remains facet-defining when some parent sets are disallowed. The following theorem of [Cussens *et al.*, 2017] helps us do this.

Theorem 4. *Let $\pi x \leq \pi_0$ define a facet for the family-variable polytope $P_{\mathrm{F}}(V, \mathcal{P})$. Suppose that $\pi_{i\leftarrow J} = \pi_{i\leftarrow J'}$ for some $i \in V$, $J, J' \in \mathcal{P}(i)$ with $J \subsetneq J'$, $J \neq \emptyset$. Let $\breve{\pi}$ be π with the component $\pi_{i\leftarrow J'}$ removed. Let $\breve{\mathcal{P}}$ be identical to $\mathcal{P}$ except that J' is removed from $\mathcal{P}(i)$. Then $\breve{\pi}x \leq \pi_0$ defines a facet for the polytope $P_{\mathrm{F}}(V, \breve{\mathcal{P}})$.*

Given a facet-defining inequality of an all-parent-sets-allowed polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$ and a parent set cardinality limit κ, Theorem 4 establishes that if the coefficients for all family variables $x_{i\leftarrow J'}$ with $|J'| > \kappa$ are not strictly larger than the coefficient for some family variable $x_{i\leftarrow J}$ with $J \subsetneq J'$ so that $|J| \leq \kappa$, then the inequality also defines a facet for the polytope with family variables restricted by κ. In [Cussens *et al.*, 2017] this is confirmed computationally for the case where $|V| = 4$ and $\kappa = 2$. It follows that a normal ($k = 1$) cluster constraint is facet-defining for *any* limit κ on the size of parent sets.

6 Enumeration of Low-Dimensional Facets

Complementing the theoretical results, we provide a complete enumeration of facets for low-dimensional family-variable polytopes. Mapping to practice, explicit knowledge of such facets has the potential to further speed up state-of-the-art BNSL solving by integrating (some of) these facets explicitly into the search.

We provide a *complete* enumeration (see [Cussens *et al.*, 2017, Appendix A] for details) of the facet-defining inequalities over 2–4 nodes and confirm the enumeration is consistent with the theoretical results. The enumeration was done using the `cdd` polyhedral computation software [Fukuda, 2016].

- For node set of size 2, i.e., $V = \{a, b\}$, there are 3 acyclic digraphs and three facet-defining inequalities: the two lower bounds $x_{a\leftarrow b} \geq 0$ and $x_{a\leftarrow b} \geq o$ and the 1-cluster constraint $x_{a\leftarrow b} + x_{b\leftarrow a} \leq 1$.
- For $|V| = 3$, there are 25 acyclic digraphs and and the convex hull of these 25 acyclic digraphs (i.e. the family-variable polytope) has 17 facet-defining inequalities
- For $|V| = 4$, there are 543 acyclic digraphs, and 135 facet-defining inequalities of the family variable polytope.

[1] In the full length article [Cussens *et al.*, 2017] we also consider two other related polytopes: the *digraph* and the *cluster* polytopes.

References

[Achterberg, 2007] Tobias Achterberg. *Constraint Integer Programming*. PhD thesis, TU Berlin, July 2007.

[Bartlett and Cussens, 2017] Mark Bartlett and James Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258 – 271, 2017.

[Chickering, 1996] David Maxwell Chickering. Learning Bayesian networks is NP-Complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: AI & Statistics V*, chapter 12, pages 121–130. Springer, 1996.

[Colombo *et al.*, 2012] Diego Colombo, Marloes H. Maathuis, Markus Kalisch, and Thomas S. Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *Annals of Statistics*, 40:294–321, 2012.

[Cussens *et al.*, 2013] James Cussens, Mark Bartlett, Elinor M. Jones, and Nuala A. Sheehan. Maximum likelihood pedigree reconstruction using integer linear programming. *Genetic Epidemiology*, 37(1):69–83, Janary 2013.

[Cussens *et al.*, 2017] James Cussens, Matti Järvisalo, Janne H. Korhonen, and Mark Bartlett. Bayesian network structure learning with integer programming: Polytopes, facets and complexity. *Journal of Artificial Intelligence Research*, 58:185–229, 2017.

[Cussens, 2010] James Cussens. Maximum likelihood pedigree reconstruction using integer programming. In *Proceedings of the Workshop on Constraint Based Methods for Bioinformatics (WCB-10)*, 2010.

[Cussens, 2011] James Cussens. Bayesian network learning with cutting planes. In Fabio G. Cozman and Avi Pfeffer, editors, *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 153–160. AUAI Press, 2011.

[de Campos and Ji, 2011] P. de Campos, Cassio and Qiang Ji. Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research*, 12:663–689, 2011.

[Fukuda, 2016] Komei Fukuda. cdd & ccdplus homepage. www.inf.ethz.ch/personal/fukudak/cdd_home, November 2016. Online; accessed 31 December 2016.

[Heckerman *et al.*, 1995] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning discrete Bayesian networks. *Machine Learning*, 20:197–243, 1995.

[Jaakkola *et al.*, 2010] Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In Yee Whye Teh and D. Mike Titterington, editors, *Proceedings of 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9 of *Journal of Machine Learning Research Workshop and Conference Proceedings*, pages 358–365. JMLR.org, 2010.

[Koivisto and Sood, 2004] Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

[Koller and Friedman, 2009] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[Malone *et al.*, 2014] Brandon Malone, Kustaa Kangas, Matti Järvisalo, Mikko Koivisto, and Petri Myllymäki. Predicting the hardness of learning Bayesian networks. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014)*, pages 2460–2466. AAAI Press, 2014.

[Peharz and Pernkopf, 2012] Robert Peharz and Franz Pernkopf. Exact maximum margin structure learning of Bayesian networks. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*. icml.cc / Omnipress, 2012.

[Silander and Myllymäki, 2006] Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI 2006)*, pages 445–452. AUAI Press, 2006.

[Spirtes *et al.*, 1993] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction and Search*. Springer, 1993.

[Tsamardinos *et al.*, 2006] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, October 2006.

[van Beek and Hoffmann, 2015] Peter van Beek and Hella-Franziska Hoffmann. Machine learning of Bayesian networks using constraint programming. In Gilles Pesant, editor, *Proceedings of the 21st International Conference on Principles and Practice of Constraint Programming (CP 2015)*, volume 9255 of *Lecture Notes in Computer Science*, pages 429–445. Springer, 2015.

[Yuan and Malone, 2013] Changhe Yuan and Brandon Malone. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, October 2013.