Bayesian network structure learning with integer programming: polytopes, facets and complexity.

CUSSENS, J., JÄRVISALO, M., KORHONEN, J.H., BARTLETT, M.

2017

Journal of Artificial Intelligence Research 58 (2017) 185-229 Submitted 05/16; published 01/17

Bayesian Network Structure Learning with Integer Programming: Polytopes, Facets and Complexity

James Cussens JAMES.CUSSENS@YORK.AC.UK
Department of Computer Science
& York Centre for Complex Systems Analysis
University of York, United Kingdom

Matti Järvisalo MATTI.JARVISALO@HELSINKI.FI
Helsinki Institute for Information Technology HIIT
Department of Computer Science
University of Helsinki, Finland

Janne H. Korhonen JANNE.H.KORHONEN@AALTO.FI
Department of Computer Science
Aalto University, Finland

Mark Bartlett MARK.BARTLETT@YORK.AC.UK
Department of Computer Science
University of York, United Kingdom

Abstract

The challenging task of learning structures of probabilistic graphical models is an important problem within modern AI research. Recent years have witnessed several major algorithmic advances in structure learning for Bayesian networks—arguably the most central class of graphical models—especially in what is known as the score-based setting. 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. Understanding fundamental aspects of *cutting planes* and the related *separation problem* 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. In this paper, we make several theoretical contributions towards these goals: (i) we study the computational complexity of the separation problem, proving that the problem is NP-hard; (ii) we formalise and analyse the relationship between three key polytopes underlying the IP-based approach to BNSL; (iii) we study the facets of the three polytopes both from the theoretical and practical perspective, providing, via exhaustive computation, a complete enumeration of facets for low-dimensional *family-variable* polytopes; and, furthermore, (iv) we establish a tight connection of the BNSL problem to the acyclic subgraph problem.

1. Introduction

The study of probabilistic graphical models is a central topic in modern artificial intelligence research. Bayesian networks (Koller & Friedman, 2009) form a central class of probabilistic graphical models that finds applications in various domains (Hugin Expert A/S, 2016; Sheehan, Bartlett, & Cussens, 2014). A central problem related to Bayesian networks (BNs) is that of learning them from data. An essential part of this learning problem is to aim 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 principal approaches to Bayesian network learning: *constraint-based* and *score-based*. In the constraint-based approach (Spirtes, Glymour, & Scheines, 1993; Colombo, Maathuis, Kalisch, & Richardson, 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) 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, Geiger, & Chickering, 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, Brown, & Aliferis, 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 & Sood, 2004; Silander & Myllymäki, 2006; Cussens, 2011; de Campos & Ji, 2011; Yuan & Malone, 2013; van Beek & 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, Kangas, Järvisalo, Koivisto, & Myllymäki, 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 optimization 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*. 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 constraint. 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 article.

The main contributions of this article are the following.

- We study the computational complexity of the separation problem solved via sub-IPs with connections to the general separation problem for integer programs. As a main result, in Section 5 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.

- We formalise and analyse the relationship between three key polytopes underlying the IP-based approach to BNSL in Section 4. Stated in generic abstract terms, starting from the *digraph polytope* defined by a linear relaxation of the IP formulation without the acyclicity constraint, the search progresses towards an optimal BN structure via refining the digraph polytope towards the *family-variable polytope*, i.e. the convex hull of acyclic digraphs over the set of nodes in question. The complete set of cluster constraints gives rise to the *cluster polytope* as an intermediate.

- We study the *facets* of the three polytopes both from the theoretical and practical perspective (Section 6). As a key theoretical result, we show that cluster constraints are in fact facet-defining inequalities of the family-variable polytope. From the more practical perspective, achieved via exhaustive computation, we provide a complete enumeration of facets for low-dimensional family-variable polytopes. Mapping to practice, explicit knowledge of such facets has the potential for providing further speed-ups in state-of-the-art BNSL solving by integrating (some of) these facets explicitly into search.

- In Section 7 we derive facets of polytopes corresponding to (i) BNs consistent with a given node ordering and (ii) BNs with specified sink nodes. We then use the results on sink nodes to show how a family-variable polytope for p nodes can be constructed from a family-variable polytope for $p-1$ nodes using the technique of *lift-and-project*.

- Finally, in Section 8 we provide a tight connection of the BNSL problem to the *acyclic subgraph problem*, as well as discussing the connection of the polytope underlying this problem to the three central polytopes underlying BNSL.

Before detailing the main contributions, we recall the BNSL problem in Section 2 and discuss the integer programming based approach to BNSL, central to this work, in Section 3.

2. Bayesian Network Structure Learning

In this section, we recall the problem of learning optimal Bayesian network structures in the central score-based setting.

2.1 Bayesian Networks

A Bayesian network represents a joint probability distribution over a set of random variables $Z = (Z_i)_{i \in V}$. A Bayesian network consists of a *structure* and *parameters*:

- The *structure* is an acyclic digraph (V, B) over the node set V. For edge $i \leftarrow j \in B$ we say that i is a *child* of j and j is a *parent* of i, and for a variable $i \in V$, we denote the set of parents of i by $\mathrm{Pa}(i, B)$.

- The *parameters* define a distribution for each of the random variables Z_i for $i \in V$ conditional on the values of the parents, that is, the values

$$\Pr\big(Z_i = z_i \mid Z_j = z_j \text{ for } j \in \mathrm{Pa}(i, B)\big)\,.$$

The joint probability distribution of the Bayesian network is defined in terms of the structure and the parameters as

$$\Pr(Z_i = z_i \text{ for } i \in V) = \prod_{i \in V} \Pr\big(Z_i = z_i \mid Z_j = z_j \text{ for } j \in \mathrm{Pa}(i, B)\big)\,.$$

As mentioned before, our focus is on learning Bayesian networks from data. Specifically, we focus on the Bayesian network structure learning (BNSL) problem. Once a BN structure has been decided, its parameters can be learned from the data. See, for example, the book by Koller and Friedman (2009) on techniques for parameter estimation for a given BN structure.

2.2 Score-Based BNSL

In the integer programming based approach to BNSL which is the focus of this work, the learning problem is cast as a constrained optimisation problem: each candidate BN structure has a score measuring how well it 'explains' the given data and the task is to find a BN structure which maximises that score. This score function is defined in terms of the data, but for our purposes, it is sufficient to abstract away the details, which are given, for example, by Koller and Friedman (2009).

Specifically, in this paper we restrict attention to *decomposable* score functions, where the score is defined locally by the parent set choices for each $i \in V$. 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.

In practice, one may want to restrict the set of parent sets in some way, given the large number of possible parents sets and the NP-hardness of BNSL. Typically this is done by limiting the cardinality of each candidate parent set, although other restrictions, perhaps reflecting prior knowledge, can also be used. To facilitate this, we assume that a BNSL instance also defines a set of permissible parent sets $\mathcal{P}(i) \subseteq 2^{V \setminus \{i\}}$ for each node i. For simplicity we shall only consider BNSL problems where $\emptyset \in \mathcal{P}(i)$ for all nodes. This also ensures that the empty graph, at least, is a permitted BN structure. Thus, the full formulation of the BNSL problem is as follows.

Definition 1 (BNSL)**.** A *BNSL instance* is a tuple $(V, \mathcal{P}, c)$, where

1. V is a set of nodes;

2. $\mathcal{P}\colon V \to 2^{2^V}$ is a function where, for each vertex $i \in V$, $\mathcal{P}(i) \subseteq 2^{V\setminus\{i\}}$ is the set of permissible parent sets for that vertex, and $\emptyset \in \mathcal{P}(i)$; and

3. c is a function giving the local score $c_{i\leftarrow J}$ for each $i \in V$ and $J \in \mathcal{P}(i)$.

Given a BNSL instance $(V, \mathcal{P}, c)$, the *BNSL problem* is to find an edge set $B \subseteq V \times V$ which maximises (1) subject to the following two conditions.

1. $\mathrm{Pa}(i, B) \in \mathcal{P}(i)$ for all $i \in V$.

2. (V, B) is acyclic.

2.3 BNSL with Small Parent Sets

As mentioned, it is common to put an upper bound on the cardinality of permitted parent sets. More precisely, a common setting is that we have a constant κ and the BNSL instances we consider are restricted so that all $J \in \mathcal{P}(i)$ satisfy $|J| \leq \kappa$. For the rest of the paper we use the convention that κ denotes this upper bound on parent set size.

In practice, BNSL instances with large node set size can often be solved to optimality fairly quickly when κ is small. For example, with $\kappa = 2$, Sheehan et al. (2014) were able to solve BNSL instances with $|V| = 1614$ in between 3 and 42 minutes. Even though BNSL remains NP-hard unless $\kappa = 1$ (Chickering, 1996), such results suggest that *in practice* the value of κ is an important determining factor of the hardness of a BNSL instance.

However, we will show in the following that the situation is somewhat more subtle: we show that any BNSL instance can be converted to a BNSL instance with $\kappa = 2$ and the same set of optimal solutions without significantly increasing the total size $|V|+\sum_{i\in V}|\mathcal{P}(i)|$ of the instance. This suggests, to a degree, that this total instance size is an important control parameter for the hardness of BNSL instances; naturally, with larger κ, a smaller number of nodes is required for a large total size.[1]

We first introduce some useful notation identifying the set of families in a BNSL instance. For a given set V of nodes and permitted parent sets $\mathcal{P}(i)$, let

$$\mathcal{F}(V, \mathcal{P}) := \{i \leftarrow J \mid i \in V, J \in \mathcal{P}(i)\},$$

so that $\sum_{i\in V}|\mathcal{P}(i)| = |\mathcal{F}(V, \mathcal{P})|$ and total instance size is $|V| + |\mathcal{F}(V, \mathcal{P})|$.

Theorem 2. *Given a BNSL instance $(V, \mathcal{P}, c)$ with the property that for each $i \in V$, $\mathcal{P}(i)$ is downwards-closed, that is, $I \subseteq J \in \mathcal{P}(i)$ implies $I \in \mathcal{P}(i)$, we can construct another BNSL instance $(V', \mathcal{P}', c')$ in time $\mathrm{poly}\big(|V| + |\mathcal{F}(V, \mathcal{P})|\big)$ such that*

1. *$|V'| = O\big(|V| + |\mathcal{F}(V, \mathcal{P})|\big)$ and $|\mathcal{F}(V', \mathcal{P}')| = O\big(|\mathcal{F}(V, \mathcal{P})|\big)$,*

2. *$|J| \leq 2$ for all $J \in \mathcal{P}'(i)$ and $i \in V'$, and*

3. *there is one-to-one correspondence between the optimal solutions of $(V, \mathcal{P}, c)$ and $(V', \mathcal{P}', c')$.*

1. The conversion to a BNSL instance with $\kappa = 2$ presented here may influence the runtime performance of BNSL solvers in practice. For example, we have observed through experimentation that the runtime performance of the GOBNILP system often degrades if the conversion is applied before search.

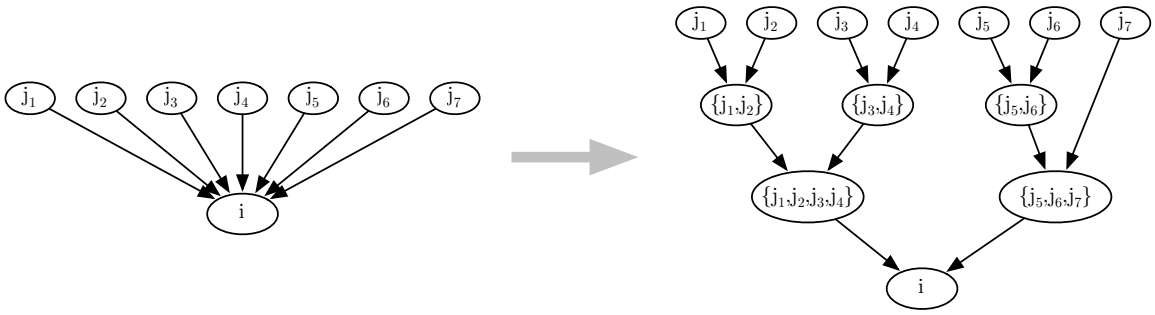

Figure 1: The basic idea of the reduction in Theorem 2. Selecting the parent set $\{j_1, j_2, j_3, j_4, j_5, j_6, j_7\}$ for node i in the original instance corresponds to selecting the parent set $\big\{\{j_1, j_2, j_3, j_4\}, \{j_5, j_6, j_7\}\big\}$ in the transformed instance. Note that the parent sets for the nodes labelled with sets are fixed.

Moreover, the claim holds even when $(V, \mathcal{P}, c)$ does not satisfy the downwards-closed property, with bounds $|V'| = O\big(|V| + \kappa\,|\mathcal{F}(V, \mathcal{P})|\big)$ and $|\mathcal{F}(V', \mathcal{P}')| = O\big(\kappa\,|\mathcal{F}(V, \mathcal{P})|\big)$, where κ is the size of the largest parent set permitted by $\mathcal{P}$.

Proof. Given $(V, \mathcal{P}, c)$, we construct a new instance $(V', \mathcal{P}', c')$ as follows. As a first step, we iteratively go through the permissible parent sets $J \in \mathcal{P}(i)$ for each $i \in V$ and add the corresponding new parent set to $\mathcal{P}'(i)$ using the following rules; Figure 1 illustrates the basic idea.

- If $|J| \leq 2$, we add J to $\mathcal{P}'(i)$ with score $c'_{i \leftarrow J} = c_{i \leftarrow J}$.
- If $J = \{j, k, l\}$, then we create a new node $I \in V'$ corresponding to the subset $I = \{k, l\}$, and add the set $J' = \{j, I\}$ to $\mathcal{P}'(i)$ with score $c'_{i \leftarrow J'} = c_{i \leftarrow J}$.
- If $|J| \geq 4$, we partition J into two sets J_1 and J_2 with $||J_1| - |J_2|| \leq 1$ and create new corresponding nodes $J_1, J_2 \in V'$. We then add $J' = \{J_1, J_2\}$ to $\mathcal{P}'(i)$ with score $c'_{i \leftarrow J'} = c_{i \leftarrow J}$.

In the above steps, new nodes corresponding to subsets of V will be created only once, re-using the same node if it is required multiple times.

Unless all original parent sets have size at most two, this process will create new nodes $J \in V'$ corresponding to subsets $J \subseteq V$ with $|J| \geq 2$. For each such new node J, we allow exactly one permissible parent set (of size 2) besides the empty set, as follows.

- If $J = \{j, k\}$, then set $\mathcal{P}'(J) = \{\emptyset, \{j, k\}\}$.
- If $J = \{j, k, l\}$, then set $\mathcal{P}'(J) = \big\{\emptyset, \{j, \{k, l\}\}\big\}$, choosing j arbitrarily and creating a new node $\{k, l\}$ if necessary.
- If $|J| \geq 4$, then we partition J into some J_1 and J_2 where $||J_1| - |J_2|| \leq 1$ and set $\mathcal{P}'(J) = \{\emptyset, \{J_1, J_2\}\}$, again creating new nodes J_1 and J_2 if necessary.

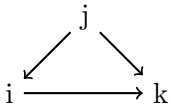

Figure 2: A digraph with 3 nodes.

However, we want to disallow the choice of $\emptyset$ for all new nodes in all optimal solutions, so we will set $c'_{J \leftarrow \emptyset} = \min\big(-|V|, M|V|\big)$, where M is the minimum score given to any family by c, and set the local score for the other parent set choices to 0.

The creation of these parent sets may require the creation of yet further new nodes. If so, we create the permissible parent sets for each of them in the same way, iterating the process as long as necessary. This will clearly terminate, and if $(V, \mathcal{P}, c)$ satisfies the downwards-closed property, this will create exactly one new node in V' for each original permissible parent set, implying the bounds for $|V'|$ and $|\mathcal{F}(V', \mathcal{P}')|$. If the original instance does not have the downwards-closed property, the process may create up to $|J|$ new nodes for each original $J \in \mathcal{P}(i)$, which in turn implies the weaker bound.

Finally, note that any optimal solution to $(V', \mathcal{P}', c')$ cannot pick the empty set as a parent set for a node corresponding to a subset of V. It is now not difficult to see that, from any optimal solution to our newly created BNSL instance, we can 'read off' an optimal solution to the original instance. □

3. An Integer Programming Approach to Bayesian Network Structure Learning

In this section we discuss integer programming based approaches to BNSL, focusing on the branch-and-cut approach implemented by the GOBNILP system for BNSL which motivates the theoretical results presented in this article.

3.1 An Integer Programming Formulation of BNSL

Recall, from Section 2, that we refer to a node i together with its parent set J as a *family*. In the IP formulation of BNSL we create a *family variable* $x_{i \leftarrow J}$ for each potential family. A family variable is a binary indicator variable: $x_{i \leftarrow J} = 1$ if J is the parent set for i and $x_{i \leftarrow J} = 0$ otherwise. It is not difficult to see that any digraph (acyclic or otherwise) with $|V|$ nodes can be encoded by a zero-one vector whose components are family variables and where exactly $|V|$ family variables are set to 1. Figure 2 and Table 1 show an example graph and its family variable encoding, respectively.

Although every digraph can thus be encoded as a zero-one vector, it is clearly not the case that each zero-one vector encodes a digraph. The key to the IP approach to BNSL is to add appropriate linear constraints so that all and only zero-one vectors representing acyclic digraphs satisfy all the constraints.

The most basic constraints are illustrated by the arrangement of the example vector in Table 1 into three rows, one for each node. It is clear that exactly one family variable for

$i \leftarrow \{\}$	$i \leftarrow \{j\}$	$i \leftarrow \{k\}$	$i \leftarrow \{j,k\}$
0	1	0	0
$j \leftarrow \{\}$	$j \leftarrow \{i\}$	$j \leftarrow \{k\}$	$j \leftarrow \{i,k\}$
1	0	0	0
$k \leftarrow \{\}$	$k \leftarrow \{i\}$	$k \leftarrow \{j\}$	$k \leftarrow \{i,j\}$
0	0	0	1

Table 1: A vector in $\mathbb{R}^{12}$ which is the family variable encoding of the digraph in Figure 2 where all possible parent sets are permitted. Here each of the 12 components is labelled with the appropriate family and the vector is displayed in three rows.

each child node must equal one. So we have $|V|$ *convexity constraints*

$$\sum_{J \in \mathcal{P}(i)} x_{i \leftarrow J} = 1 \quad \forall i \in V, \tag{2}$$

each of which may have an exponential number of terms. It is not difficult to see that any vector x that satisfies all convexity constraints encodes a digraph. However, without further constraints, the digraph need not be acyclic. There are a number of ways of ruling out cycles (Cussens, 2010; Peharz & Pernkopf, 2012; Cussens, Bartlett, Jones, & Sheehan, 2013). In this paper we focus on *cluster constraints* first introduced by Jaakkola, Sontag, Globerson, and Meila (2010). A *cluster* is simply a subset of nodes with at least 2 elements. For each cluster $C \subseteq V$ ($|C| > 1$) the associated cluster inequality is

$$\sum_{i \in C} \sum_{J \in \mathcal{P}(i): J \cap C = \emptyset} x_{i \leftarrow J} \geq 1. \tag{3}$$

An alternative formulation, which exploits the convexity constraints, is

$$\sum_{i \in C} \sum_{J \in \mathcal{P}(i): J \cap C \neq \emptyset} x_{i \leftarrow J} \leq |C| - 1. \tag{4}$$

To see that cluster inequalities suffice to rule out cycles, note that, for any cluster C and digraph x, the left-hand side (LHS) of (3) is a count of the number of vertices in C that in x have no parents in C. Now suppose that the nodes in some cluster C formed a cycle; it is clear that in that case the LHS of (3) would be 0, violating the cluster constraint. On the other hand, suppose that x encodes an acyclic digraph. Since the digraph is acyclic, there is an associated total ordering in which parents precede their children. Let $C \subseteq V$ be an arbitrary cluster. Then the earliest element of C in this ordering will have no parents in C and so the LHS of (3) is at least 1 and the cluster constraint is satisfied. An illustration of how acyclic graphs satisfy all cluster constraints and cyclic graphs do not is given in Figure 3.

It follows that any zero-one vector x that satisfies the convexity constraints (2) and cluster constraints (3) encodes an acyclic digraph. The final ingredient in the IP approach to BNSL is to specify objective coefficients for each family variable. These are simply the

Figure 3: 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 (i.e. the LHS of (3)). 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.

local scores $c_{i \leftarrow J}$ introduced in Section 2. Collecting these elements together, we can define the IP formulation of the BNSL as follows.

$$\text{MAXIMISE} \quad \sum_{i \in V, J \in \mathcal{P}(i)} c_{i \leftarrow J} x_{i \leftarrow J} \tag{5}$$

$$\text{SUBJECT TO} \quad \sum_{J \in \mathcal{P}(i)} x_{i \leftarrow J} = 1 \qquad \forall i \in V \tag{6}$$

$$\sum_{i \in C} \sum_{J \in \mathcal{P}(i) : J \cap C = \emptyset} x_{i \leftarrow J} \geq 1 \qquad \forall C \subseteq V,\ |C| > 1 \tag{7}$$

$$x_{i \leftarrow J} \in \{0, 1\} \qquad \forall i \in V,\ J \in \mathcal{P}(i) \tag{8}$$

3.2 The GOBNILP System

The GOBNILP system solves the IP problem defined by (5–8) for a given set of objective coefficients $c_{i \leftarrow J}$. These coefficients are either given as input to GOBNILP or computed by GOBNILP from a discrete dataset with no missing values. The GOBNILP approach to solving this IP is fully detailed by Bartlett and Cussens (2015); here we overview the essential ideas.

Since there are only $|V|$ convexity constraints (6), these are added as initial constraints to the IP. Initially, no cluster constraints (7) are in the IP, so we have a relaxed version of the original problem. Moreover, in its initial phase GOBNILP relaxes the integrality condition (8) on the family variables into $x_{i \leftarrow J} \in [0, 1]\ \forall i \in V,\ J \in \mathcal{P}(i)$, so that only linear relaxations of IPs are solved. So GOBNILP starts with a 'doubly' relaxed problem: the constraints ruling out cycles are missing and the integrality condition is also dropped.

A linear relaxation of an IP is a linear program (LP). GOBNILP uses an external LP solver such as SoPlex or CPLEX to solve linear relaxations. The solution (call it x^*) to the initial LP will be a digraph where a highest scoring parent set for each node is chosen, a digraph which will almost certainly contain cycles. Note that this initial solution happens to be integral, even though it is the solution to an LP not an IP. GOBNILP then attempts

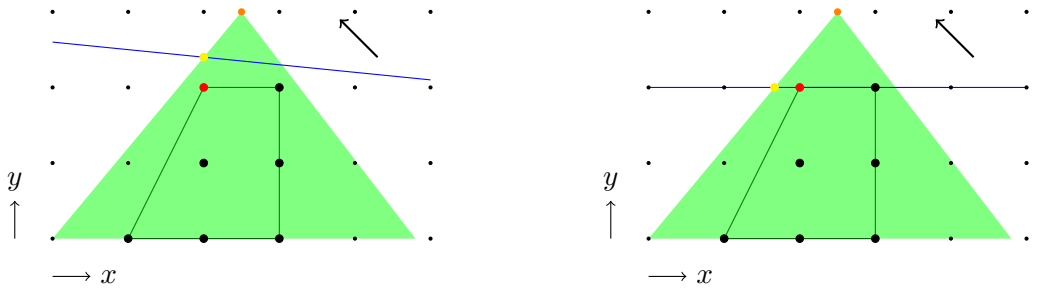

Figure 4: Illustration of the cutting plane technique for a problem with 2 integer-valued variables x and y. **In both figures**: The 7 large dots indicate the 7 feasible solutions $(x = 1, y = 0)$, $(x = 2, y = 0)$, $(x = 3, y = 0)$, $(x = 2, y = 1)$, $(x = 3, y = 1)$, $(x = 2, y = 2)$ and $(x = 3, y = 2)$. The red dot indicates the optimal solution $(x = 2, y = 2)$. The objective function is $-x + y$ and is indicated by the arrow. The boundary of the convex hull of feasible solutions is shown. A relaxation of the problem is indicated by the green region with the orange dot indicating the optimal solution to the relaxed problem. **In the left-hand figure**: The blue line represents a cutting plane—a linear inequality—which separates the solution to the relaxed problem from the convex hull of feasible solutions. The yellow dot indicates the optimal solution to the relaxed problem once this cut is added. **In the right-hand figure**: Similar to the left-hand figure except that a better cut has been added. The right-hand yellow dot has a lower objective value than that on the left and thus provides a better upper bound.

to find clusters C such that the associated cluster constraint is violated by x^*. Since the cluster constraints are added in this way they are called *cutting planes*: each one *cuts* off an (infeasible) solution x^* and since they are linear each one defines a (high-dimensional) *plane*. These cluster constraints are added to the LP, producing a new LP which is then solved, generating a new solution x^*. This process is illustrated in Figure 4; since the cutting planes found by GOBNILP are rather hard to visualise, we use a (non-BNSL) IP problem with only two variables to illustrate the basic ideas behind the cutting plane approach. Note that the relaxation in Figure 4 contains (infeasible) integer solutions. Many of the relaxations solved by GOBNILP (notably the initial one) also allow infeasible integer solutions—which correspond to cyclic digraphs.

The process of LP solving and adding cluster constraint cutting planes is continued until either (i) an LP solution is produced which corresponds to an acyclic digraph, or (ii) this is not the case, but no further cluster constraint cutting planes can be found. In the first (rare) case, the BNSL instance has been solved. The objective value of each x^* that is produced is an upper bound on the objective value of an optimal digraph (since it is an exact solution to a relaxed version of the original BNSL instance), so if x^* corresponds to an acyclic digraph it must be optimal.

The second (typical) case can occur since even if we were to add all (exponentially-many) cluster constraints to the LP there is no guarantee that the solution to that LP would be integral. (This hypothetical LP including all cluster constraints defines what we call the

cluster polytope which will be discussed in Section 4.3.) However, since we only add those cluster constraints which are *cutting planes* (i.e. which cut off the solution x^* to some linear relaxation) in practice only a small fraction of cluster constraints are actually added.[2]

Once no further cluster constraint cutting planes can be found GOBNILP stops ignoring the integrality constraint (8) on family variables and exploits it to make progress. If no cluster constraint cutting planes can be found, and the problem has not been solved, then x^*, the solution to the current linear relaxation, must be fractional, i.e. there must be at least one family variable $x_{i \leftarrow J}$ such that $0 < x^*_{i \leftarrow J} < 1$. One option is then to *branch* on such a variable to create two sub-problems: one where $x_{i \leftarrow J}$ is fixed to 0 and one where it is fixed to 1. Note that x^* is infeasible in the linear relaxations of both sub-problems but there is an optimal solution in at least one of the sub-problems. GOBNILP also has the option of branching on sums of mutually exclusive family variables. For example, given nodes i, j, and k, GOBNILP has the option of branching on $x_{i \leftarrow \{j\}} + x_{i \leftarrow \{j,k\}} + x_{j \leftarrow \{i\}} + x_{j \leftarrow \{i,k\}}$, a quantity which is either 0 or 1 in an acyclic digraph. GOBNILP then recursively applies the cutting plane approach to both sub-problems. GOBNILP is thus a *branch-and-cut* approach to IP solving.

These are the essentials of the GOBNILP system, although the current implementation has many other aspects. In particular, under default parameter values, GOBNILP switches to branching on a fractional variable if the search for cluster constraint cutting planes is taking too long. GOBNILP is implemented with the help of the SCIP system (Achterberg, 2007) and it uses SCIP to generate many other cutting planes in addition to cluster constraints. GOBNILP also adds in other initial inequalities in addition to the convexity constraints. For example, if we had three nodes i, j, and k, the inequality $x_{i \leftarrow \{j,k\}} + x_{j \leftarrow \{i,k\}} + x_{k \leftarrow \{i,j\}} \leq 1$ would be added. All these extra constraints are redundant in the sense that they do not alter the set of optimal solutions to the IP (5–8). They do, however, have a great effect in the time taken to identify a provably optimal solution.

3.3 BNSL Cutting Planes via Sub-IPs

The *separation problem* for an IP is the problem of finding a cutting plane which is violated by the current linear relaxation of the IP, or to show that none exists. In this paper we focus on the special case of finding a *cluster constraint* cutting plane for an LP solution x^*, or showing none exists. We call this the *weak separation problem*. We call it the 'weak' separation problem since cluster constraints are not the only possible cutting planes.

In GOBNILP this problem is solved via a sub-IP, as described previously by Bartlett and Cussens (2015). Given an LP solution x^* to separate, the variables of the sub-IP include binary variables $y_{i \leftarrow J}$ for each family such that $x^*_{i \leftarrow J} > 0$. In addition, binary variables y_i for each $i \in V$ are created. The constraints of the sub-IP are such that $y_i = 1$ indicates that i is a member of some cluster whose associated cluster constraint is a cutting plane for x^*. $y_{i \leftarrow J} = 1$ indicates that the family variable $x_{i \leftarrow J}$ appears in the cluster constraint. The sub-IP is given by

2. We have yet to explore the interesting question of how large this fraction might be.

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

$$\text{subject to} \quad y_{i\leftarrow J} \Rightarrow y_i \qquad \forall y_{i\leftarrow J} \tag{10}$$

$$y_{i\leftarrow J} \Rightarrow \bigvee_{j \in J} y_j \qquad \forall y_{i\leftarrow J} \tag{11}$$

$$\sum_{i,J \,:\, x^*_{i\leftarrow J}>0} x^*_{i\leftarrow J} \cdot y_{i\leftarrow J} - \sum_{i \in V} y_i > -1 \tag{12}$$

$$y_{i\leftarrow J}, y_i \in \{0,1\} \tag{13}$$

The sub-IP constraints (10–11) are displayed as propositional clauses for brevity, but note that these are linear constraints. They can be written as $(1 - y_{i\leftarrow J}) + y_i \geq 1$ and $(1 - y_{i\leftarrow J}) + \sum_{j \in J} y_j \geq 1$, respectively. The constraint (12) dictates that only solutions with objective value strictly greater than -1 are allowed. In the GOBNILP implementation this constraint is effected by directly placing a lower bound on the objective rather than posting the linear constraint (12), since the former is more efficient.

It is not difficult to show—Bartlett and Cussens (2015) provide the detail—that any feasible solution to sub-IP (9–13) determines a cutting plane for x^* and that a proof of the sub-IP's infeasibility establishes that there is no such cutting plane. Since GOBNILP spends much of its time solving sub-IPs in the hunt for cluster constraint cutting planes, the issue of whether there is a better approach is important. Is it really a good idea to set up a sub-IP each time a cutting plane is sought? Is there some algorithm (perhaps a polynomial-time one) that can be directly implemented to provide a faster search for cutting planes? In Section 5 we make progress towards answering these questions. We show that the weak separation problem is *NP-hard* and so (assuming P $\neq$ NP) there is no polynomial-time algorithm for weak separation.

4. Three Polytopes Related to the BNSL IP

As explained in Section 3.2, in the basic GOBNILP algorithm one first (i) uses only the convexity constraints, then (ii) adds cluster constraints, and, if necessary, (iii) branches on variables to solve the IP. These three stages correspond to three different *polytopes* which will be defined and analyzed in Sections 4.2–4.4. Before providing this analysis we first give essential background on linear inequalities, polytopes and polyhedra (Conforti, Cornuéjols, & Zambelli, 2014). We follow the notation of Conforti et al. (2014), which is standard throughout the mathematical programming literature: for $x, y \in \mathbb{R}^n$, (1) "$x \leq y$" means that $x_i \leq y_i$ for all $i = 1, \ldots, n$ and (2) "xy" where $x, y \in \mathbb{R}^n$ is the scalar or 'dot' product (i.e. $x^T y$).

4.1 Linear Inequalities, Polytopes and Polyhedra

Definition 3. A point $x \in \mathbb{R}^n$ is a *convex combination* of points in $S \subseteq \mathbb{R}^n$ if there exists a finite set of points $x^1, \ldots, x^p \in S$ and scalars $\lambda_1, \ldots, \lambda_p$ such that

$$x = \sum_{j=1}^{p} \lambda_j x^j, \qquad \sum_{j=1}^{p} \lambda_j = 1, \qquad \lambda_1, \ldots, \lambda_p \geq 0.$$

Definition 4. The *convex hull* $\text{conv}(S)$ of a set $S \subseteq \mathbb{R}^n$ is the inclusion-wise minimal convex set containing S, i.e. $\text{conv}(S) = \{x \in \mathbb{R}^n \mid \ x \text{ is a convex combination of points in } S\}$.

Definition 5. A subset P of $\mathbb{R}^n$ is a *polyhedron* if there exists a positive integer m, an $m \times n$ matrix A, and a vector $b \in \mathbb{R}^m$ such that

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

Definition 6. A subset Q of $\mathbb{R}^n$ is a *polytope* if Q is the convex hull of a finite set of vectors in $\mathbb{R}^n$.

Theorem 7 (Minkowski-Weyl Theorem for Polytopes)**.** *A subset Q of $\mathbb{R}^n$ is a* polytope *if and only if Q is a bounded polyhedron.*

What the Minkowski-Weyl Theorem for Polytopes states is that a polytope can either be described as the convex hull of a finite set of points or as the set of feasible solutions to some linear program. It follows that, for a given linear objective, an optimal point can be found by solving the linear program. This is a superficially attractive prospect since linear programs can be solved in polynomial time.

Unfortunately, for NP-hard problems (such as BNSL) it is impractical to create, let alone solve, the linear program due to the size of A and b. Fully characterising the inequalities $Ax \leq b$ is also typically difficult. However, it is useful to identify at least some of these inequalities. These inequalities define *facets* of the polytope. A facet is a special kind of *face* defined as follows.

Definition 8. A *face* of a polyhedron $P \subseteq \mathbb{R}^n$ is a set of the form

$$F := P \cap \{x \in \mathbb{R}^n \mid cx = \delta\},$$

where $cx \leq \delta$ ($c \in \mathbb{R}^n, \delta \in \mathbb{R}$) is a *valid inequality* for P, i.e. all points in P satisfy it. We say the inequality $cx \leq \delta$ *defines* the face. A face is *proper* if it is non-empty and properly contained in P. An inclusion-wise maximal proper face of P is called a *facet*.

So, for example, a cube is a 3-dimensional polytope (it is also a polyhedron) with 6 2-dimensional faces, 12 1-dimensional faces and 6 0-dimensional faces (the vertices). The 2-dimensional faces are facets since each of them is proper and not contained in any other face. The convex hull of the 7 points $(x = 1, y = 0)$, $(x = 2, y = 0)$, $(x = 3, y = 0)$, $(x = 2, y = 1)$, $(x = 3, y = 1)$, $(x = 2, y = 2)$ and $(x = 3, y = 2)$, whose boundary is represented in Figure 4, is 2-dimensional and has 4 1-dimensional facets (shown in Figure 4) and 4 0-dimensional faces. Note that the 'good' cut in the right-hand figure of Figure 4 is a facet-defining inequality.

Facets are important since they are given by the 'strongest' inequalities defining a polyhedron. The set of all facet-defining inequalities of a polyhedron provides a minimal representation $Ax \leq b$ of that polyhedron, so any cutting plane which is not facet-defining is thus 'redundant'. The formal definition of redundancy is provided by Wolsey (1998, p.141). Practically, facet-defining inequalities are good inequalities to add as cutting planes since they, and they alone, are guaranteed not to be dominated by any other valid inequality and also not by any linear combination of other valid inequalities. Identifying facets is thus an important step in improving the computational efficiency of an IP approach.

A face of an n-dimensional polytope is a facet if and only if it has dimension $n-1$. (Note that the 6 facets of a 3-dimensional cube are indeed 2-dimensional.) To prove that a face F has dimension $n-1$ it is enough to find n *affinely independent* points in F. Affine independence is defined as follows (Wolsey, 1998).

Definition 9. The points $x^1, \dots x^k \in \mathbb{R}^n$ are *affinely independent* if the $k-1$ directions $x^2 - x^1, \dots, x^k - x^1$ are linearly independent, or alternatively the k vectors $(x^1, 1), \dots (x^k, 1) \in \mathbb{R}^{n+1}$ are linearly independent.

Note that if $x^1, \dots x^k \in \mathbb{R}^n$ are linearly independent they are also affinely independent.

Having provided these basic definitions we now move on to consider three polytopes of increasing complexity: the *digraph polytope* (Section 4.2), the *cluster polytope* (Section 4.3) and finally, our main object of interest, the *family variable polytope* (Section 4.4).

4.2 The Digraph Polytope

The digraph polytope is simply the convex hull of all digraphs permitted by $\mathcal{P}$. Before providing a formal account of this polytope we define some notation. For a given set of nodes V and permitted parent sets $\mathcal{P}(i)$, recall from Section 2.3 that the set of families is defined as

$$\mathcal{F}(V, \mathcal{P}) := \{i \leftarrow J \mid i \in V, J \in \mathcal{P}(i)\}.$$

Furthermore, we denote the set of families that remain once the empty parent set for each vertex is removed by

$$\mathbf{F}(V, \mathcal{P}) := \mathcal{F}(V, \mathcal{P}) \setminus \{i \leftarrow \emptyset \mid i \in V\}.$$

In this and subsequent sections $\mathbf{F}(V, \mathcal{P})$ will serve as an index set. We will abbreviate $\mathcal{F}(V, \mathcal{P})$ and $\mathbf{F}(V, \mathcal{P})$ to $\mathcal{F}$ and $\mathbf{F}$ unless it is necessary or useful to identify the node set V and permitted parent sets $\mathcal{P}(i)$.

For any edge set $A \subseteq V \times V$, it is clear that any 0-1 vector in $\mathbb{R}^A$ corresponds to a (possibly cyclic) subgraph of $D = (V, A)$. However, there are many 0-1 vectors in $\mathbb{R}^{\mathcal{F}}$ (or $\mathbb{R}^{\mathbf{F}}$) which do not correspond to digraphs, namely those where $x_{i \leftarrow J} = x_{i \leftarrow J'} = 1$ for some $i \leftarrow J, i \leftarrow J' \in \mathcal{F}$ with $J \neq J'$. So clearly inequalities other than simple variable bounds are required to define the digraph polytope.

Since any digraph (cyclic or acyclic) satisfies the $|V|$ convexity constraints (2), the digraph polytope if expressed using the variables in $\mathcal{F}$ will not be *full-dimensional*—the dimension of the polytope will be less than the number of variables. This is inconvenient since only full-dimensional polytopes have a unique minimal description in terms of their facets.

To arrive at a full-dimensional polytope we remove the $|V|$ family variables with empty parent sets and define the digraph polytope using index set $\mathbf{F}(V, \mathcal{P})$. Let $P_{\mathrm{G}}(V, \mathcal{P})$ be the *digraph polytope* which is the convex hull of all points in $\mathbb{R}^{\mathbf{F}(V,\mathcal{P})}$ that correspond to digraphs (cyclic and acyclic).

$$P_{\mathrm{G}}(V, \mathcal{P}) := \mathrm{conv}\Big\{ x \in \mathbb{R}^{\mathbf{F}(V,\mathcal{P})} \;\Big|\; \exists B \subseteq V \times V \text{ s.t.} \quad \mathrm{Pa}(i, B) \in \mathcal{P}(i)\ \forall i \in V \text{ and } x_{i \leftarrow J} = \mathbb{1}(J = \mathrm{Pa}(i, B))\ \forall J \in \mathcal{P}(i)\ \setminus \emptyset \Big\}. \tag{14}$$

We will abbreviate $P_{\mathrm{G}}(V, \mathcal{P})$ to P_{G} where this will not cause confusion.

Proposition 10. *P_{G} is full-dimensional.*

Proof. The digraph with no edges is represented by the zero vector in $\mathbb{R}^{\mathbf{F}}$. Each vector in $\mathbb{R}^{\mathbf{F}}$ with only one component $x_{i \leftarrow J}$ set to 1 and all others set to 0 represents an acyclic digraph (denoted $e^{i \leftarrow J}$) and so is in P_{G}. These vectors together with the zero vector are clearly a set of $|\mathbf{F}| + 1$ affinely independent vectors from which it follows that P_{G} is full-dimensional in $\mathbb{R}^{\mathbf{F}}$. □

P_{G} is a simple polytope: it is easy to identify all its facets.

Proposition 11. *The facet-defining inequalities of P_{G} are*

1. $\forall i \leftarrow J \in \mathbf{F} : x_{i \leftarrow J} \geq 0$ *(variable lower bounds), and*
2. $\forall i \in V : \sum_{i \leftarrow J \in \mathbf{F}(V,\mathcal{P})} x_{i \leftarrow J} \leq 1$ *('modified' convexity constraints).*

Proof. We use Wolsey's third approach to establishing that a set of linear inequalities define a convex hull (Wolsey, 1998, p.145). Let $c \in \mathbb{R}^{\mathbf{F}}$ be an arbitrary objective coefficient vector. It is clear that the linear program maximising cx subject to the given linear inequalities has an optimal solution which is an integer vector representing a digraph: simply choose a 'best' parent set for each $i \in V$. (If all coefficients are non-positive choose the empty parent set.) Moreover for any digraph x, it easy to see that there is a c such that x is an optimal solution to the LP. It is also easy to see that each of the given linear inequalities is *necessary*—removing any one of them results in a different polytope. The result follows. □

Proposition 11 establishes the unsurprising fact that the polytope defined by GOBNILP's initial constraints is $P_{\mathrm{G}}(V, \mathcal{P})$, the convex hull of all digraphs permitted by $\mathcal{P}$. It follows that we will have $x^* \in P_{\mathrm{G}}$ for any LP solution x^* produced by GOBNILP after adding cutting planes.

4.3 The Cluster Polytope

Although GOBNILP only adds those cluster constraints which are needed to separate LP solutions x^*, it is useful to consider the polytope which would be produced if all were added. The cluster polytope $P_{\mathrm{CLUSTER}}(V, \mathcal{P})$ is defined by adding all cluster constraints

to the facet-defining inequalities of the digraph polytope $P_{\mathrm{G}}(V, \mathcal{P})$, thus ruling out (family variable encodings of) cyclic digraphs.

$$P_{\mathrm{CLUSTER}}(V,\mathcal{P}) := \Big\{ x \in \mathbb{R}^{\mathbf{F}(V,\mathcal{P})} \;\Big|\; x_{i \leftarrow J} \geq 0 \quad \forall i \leftarrow J \in \mathbf{F}(V,\mathcal{P}), \text{ and} \\ \sum_{i \leftarrow J \in \mathbf{F}(V,\mathcal{P})} x_{i \leftarrow J} \leq 1 \quad \forall i, \text{ and} \\ \sum_{i \in C} \sum_{J \in \mathcal{P}(i) : J \cap C \neq \emptyset} x_{i \leftarrow J} \leq |C| - 1 \quad \forall C \subseteq V,\ |C| > 1 \Big\}.$$

We will abbreviate $P_{\mathrm{CLUSTER}}(V, \mathcal{P})$ to P_{CLUSTER} where this will not cause confusion.

Proposition 12. *P_{CLUSTER} is full-dimensional.*

Proof. Proof is essentially the same as that for Proposition 10. □

As with the digraph polytope, we use the index set $\mathbf{F}$ to ensure full-dimensionality, and consequently have to use formulation (4) for cluster constraints. Clearly $P_{\mathrm{CLUSTER}} \subseteq P_{\mathrm{G}}$ (and the inclusion is proper if $|V| > 1$). Since GOBNILP only adds some cluster constraints, the feasible set for each LP that is solved during its cutting plane phase is a polytope P where $P_{\mathrm{CLUSTER}} \subseteq P \subseteq P_{\mathrm{G}}$. More important is the connection between P_{CLUSTER} and the family variable polytope which we now introduce.

4.4 The Family Variable Polytope

The *family variable polytope* $P_{\mathrm{F}}(V, \mathcal{P})$ is the convex hull of acyclic digraphs with node set V which are permitted by $\mathcal{P}$. To define $P_{\mathrm{F}}(V, \mathcal{P})$ it is first useful to introduce notation for the set of acyclic subgraphs of some digraph. Let $D = (V, A)$ be a digraph, and

$$\mathcal{A}(D) := \{B \subseteq A \mid B \text{ is acyclic in } D\}. \tag{15}$$

Now consider the case where $D = (V, V \times V)$. The *family variable polytope* $P_{\mathrm{F}}(V, \mathcal{P})$ is

$$P_{\mathrm{F}}(V,\mathcal{P}) := \mathrm{conv}\Big\{ x \in \mathbb{R}^{\mathbf{F}(V,\mathcal{P})} \;\Big|\; \exists B \in \mathcal{A}(D) \text{ s.t. } \mathrm{Pa}(i,B) \in \mathcal{P}(i)\ \forall i \in V \text{ and} \\ x_{i \leftarrow J} = \mathbb{1}(J = \mathrm{Pa}(i,B))\ \forall J \in \mathcal{P}(i) \setminus \emptyset \Big\}. \tag{16}$$

We will abbreviate $P_{\mathrm{F}}(V, \mathcal{P})$ to P_{F} where this will not cause confusion.

Proposition 13. *P_{F} is full-dimensional.*

Proof. Proof is essentially the same as that for Proposition 10. □

It is clear that $P_{\mathrm{F}} \subseteq P_{\mathrm{CLUSTER}} \subseteq P_{\mathrm{G}}$. We will see in Section 6 that although cluster constraints turn out to be facet-defining inequalities of P_{F}, they are not the only facet-defining inequalities, and so (if $|V| > 2$) $P_{\mathrm{F}} \subsetneq P_{\mathrm{CLUSTER}}$. We do, however, have that $\mathbb{Z}^{|\mathbf{F}|} \cap P_{\mathrm{F}} = \mathbb{Z}^{|\mathbf{F}|} \cap P_{\mathrm{CLUSTER}}$, since acyclic digraphs are the only zero-one vectors to satisfy all cluster and modified convexity constraints. These facts have important consequences for the IP approach to BNSL. They show that (i) cluster constraints are a good way of ruling

out cycles (since they are facet-defining inequalities of P_{F}) and that (ii) one can solve a BNSL by just using cluster constraints and branching on variables (to enforce an integral solution). That $P_{\mathrm{F}} \subsetneq P_{\mathrm{CLUSTER}}$ also implies that it may be worth searching for facet-defining cuts which are not cluster inequalities, for example those discovered by Studený (2015).

5. Computational Complexity of the BNSL Sub-IPs

In this section we focus on the computational complexity of the BNSL sub-IPs, formalized as the weak separation problem for BNSL. As the main result of this section, we show that this problem is NP-hard.

The *weak separation problem* for BNSL is as follows: given a $x^* \in P_{\mathrm{G}}$, find a *separating cluster* $C \subseteq V$, $|C| > 1$, for which

$$\sum_{i \in C} \sum_{J \in \mathcal{P}(i): J \cap C \neq \emptyset} x^*_{i \leftarrow J} > |C| - 1, \tag{17}$$

or establish that no such C exists. We first give a simple necessary condition on separating clusters.

Definition 14. Given $x^* \in P_{\mathrm{G}}$ define $\lceil D \rceil(x^*)$, the *rounding-up digraph* for x^*, as follows: $i \leftarrow j$ is an edge in $\lceil D \rceil(x^*)$ iff there is a family $i \leftarrow J$ such that $j \in J$ and $x^*_{i \leftarrow J} > 0$.

Proposition 15. *If C is a separating cluster for x^*, then $\lceil D \rceil(x^*)_C$, the subgraph of the rounding-up digraph restricted to the nodes C, is cyclic.*

Proof. Since $x^* \in P_{\mathrm{G}}$, x^* is a convex combination of extreme points of P_{G}. So we can write $x^* = \sum_{k=1}^{K} \alpha_k x^k$ where each x^k represents a graph and $\sum_{k=1}^{K} \alpha_k = 1$. For each graph x^k, let x^k_C be the subgraph restricted to the nodes C. It is easy to see that if x^k_C is acyclic, then $\sum_{i \in C} \sum_{J \in \mathcal{P}(i): J \cap C \neq \emptyset} x^k_{C_{i \leftarrow J}} \leq |C| - 1$. So if x^k_C is acyclic for all $k = 1, \ldots, K$, then $\sum_{i \in C} \sum_{J \in \mathcal{P}(i): J \cap C \neq \emptyset} x^*_{C_{i \leftarrow J}} \leq |C| - 1$. But if $\lceil D \rceil(x^*)_C$ is acyclic, then so are all the x^k_C. The result follows. □

Proposition 15 leads to a heuristic algorithm for the weak separation problem (which is available as an option in GOBNILP). Given an LP solution x^*, the rounding up digraph $\lceil D \rceil(x^*)$ is constructed and cycles in that digraph are searched for using standard techniques. For each cycle found, the corresponding cluster is checked to see whether it is a separating cluster for x^*. We now consider the central result on weak separation.

Theorem 16. *The weak separation problem for BNSL is NP-hard, even when restricted to instances $(V, \mathcal{P}, c)$ where $J \in \mathcal{P}(i)$ for all $i \in V$ only if $|J| \leq 2$.*

Proof. We prove the claim by reduction from vertex cover; that is, given a graph $G = (V, E)$ and an integer k, we construct $x^* \in P_{\mathrm{G}}(V', \mathcal{P}')$ over a vertex set V' and permitted parent sets $\mathcal{P}'$ such that there is a cluster $C \subseteq V'$ with $|C| > 1$ and

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

if and only if there is a vertex cover of size at most k for G.

Specifically, let us denote $n = |V|$ and $m = |E|$. We construct $x^* \in P_{\mathrm{G}}(V', \mathcal{P}')$ as follows; Figure 5 illustrates the basic idea.

1. The vertex set is $V' = V \cup S$, where S is disjoint from V and $|S| = m$.
2. For $s \in S$ and $\{u, v\} \in E$, we set $x^*_{s \leftarrow \{u,v\}} = 1/m$; in particular, $\sum_{\{u,v\} \in E} x^*_{s \leftarrow \{u,v\}} = 1$ for all $s \in S$.
3. For $s \in S$ and $v \in V$, we set
$$x^*_{v \leftarrow \{s\}} = \frac{k}{m(k+1)}\,.$$
4. $x^*_{i \leftarrow \emptyset} = 0$ for all $i \in V'$.
5. For all other choices of $i \in V'$ and $J \subseteq V' \setminus \{i\}$: $J \notin \mathcal{P}'(i)$.

Finally, for a cluster $C \subseteq V'$, we define the score $w(C)$ as

$$w(C) = \sum_{i \in C} \sum_{J \in \mathcal{P}'(i) : J \cap C \neq \emptyset} x^*_{i \leftarrow J} - |C|\,.$$

Now we claim that there is a set $C \subseteq V'$ with $w(C) > -1$ if and only if G has a vertex cover of size at most k; this suffices to prove the claim.

First, we observe that if $U \subseteq V$ is a vertex cover in G, then

$$\begin{aligned} w(U \cup S) &= -|U| + \sum_{v \in U} \sum_{s \in S} x_{v \leftarrow \{s\}} - |S| + \sum_{s \in S} \sum_{e \in E} \frac{1}{m} \\ &= -|U| + |U| \frac{mk}{m(k+1)} - |S| + |S| \frac{m}{m} \\ &= -|U| \Big(1 - \frac{k}{k+1}\Big) = -\frac{|U|}{k+1}\,, \end{aligned}$$

which implies that $w(U \cup S) > -1$ if $|U| \leq k$.

Now let $C \subseteq V'$, and let us denote $C_V = C \cap V$ and $C_S = C \cap S$. If $|C_V| \geq k+1$, then we have

$$\begin{aligned} w(C) &\leq -|C_V| + |C_V| \frac{|C_S| k}{m(k+1)} \\ &\leq -|C_V| + |C_V| \frac{k}{k+1} \\ &= -|C_V| \Big(1 - \frac{k}{k+1}\Big) = \frac{|C_V|}{k+1} \leq -1\,. \end{aligned}$$

On the other hand, let us consider the case where $|C_V| \leq k$ but C_V is not a vertex cover for G; we may assume that $C_V \neq \emptyset$, as otherwise we would have $w(C) = -|C| \leq -1$. Let

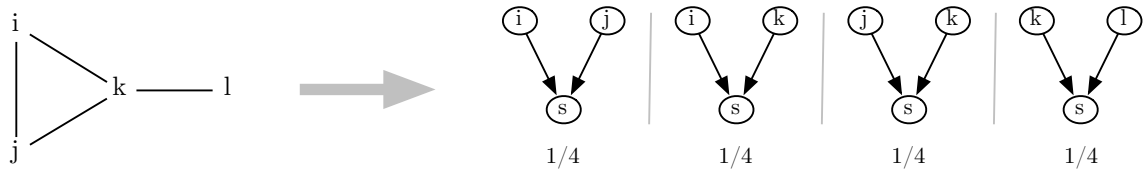

Figure 5: The basic gadget of the reduction in Theorem 16. Each edge $u, v \in V$ in the original instance $G = (V, E)$ is represented by assigning weight $x^*_{s \leftarrow \{u,v\}} = 1/|E|$ in the new instance, where s is a new node. Clearly, $U \subseteq V$ is a vertex cover in G if and only if total weight of terms $x^*_{s \leftarrow \{u,v\}}$ such that U intersects the parent set is 1.

us write $H = \{e \in E \mid C_V \cap e \neq \emptyset\}$ for the set of edges covered by C_V. Since we assume that C_V is not a vertex cover, we have $|H| \leq m - 1$. Thus, it holds that

$$\begin{aligned}
w(C) &= -|C_V| + |C_V|\frac{|C_S|k}{m(k+1)} - |C_S| + |C_S|\frac{|H|}{m} \\
&\leq -|C_V| + |C_V|\frac{|C_S|k}{m(k+1)} - |C_S| + |C_S|\frac{m-1}{m} \\
&= -|C_V|\Big(1 - \frac{|C_S|k}{m(k+1)}\Big) - \frac{|C_S|}{m} \\
&\leq -\Big(1 - \frac{|C_S|k}{m(k+1)}\Big) - \frac{|C_S|}{m} \\
&= -1 - |C_S|\Big(\frac{1}{m} - \frac{k}{m(k+1)}\Big) = -1 - \frac{|C_S|}{m(k+1)} < -1\,.
\end{aligned}$$

Thus, if C_V is not a vertex cover of size at most k, then $w(C) \leq -1$.

□

6. Facets of the Family Variable Polytope

In this section a number of facets of the family variable polytope are identified and certain properties of facets are given. Section 6.1 provides simpler results, and Sections 6.2–6.4 more substantial ones, including a tight connection between facets and cluster constraints, liftings of facets, and the influence of restricting parent sets on facets. In Appendix A we provide a complete enumeration of the facet-defining inequalities over 2–4 nodes and confirm the enumeration is consistent with the theoretical results presented here.

6.1 Simple Results on Facets

We start by showing that the full-dimensional family variable polytope P_{F} is *monotone* via a series of lemmas. Once we have proved this result, we will use it to establish elementary properties of facets of P_{F} and find the simple facets of the polytope.

Definition 17. A nonempty polyhedron $P \subseteq \mathbb{R}^n_{\geq 0}$ is *monotone* if $x \in P$ and $0 \leq y \leq x$ imply $y \in P$.

Lemma 18. *Let $x \in P_{\mathrm{F}}$ and let the vector y be such that $y_{i' \leftarrow J'} = 0$ for some $i' \leftarrow J'$ and $y_{i \leftarrow J} = x_{i \leftarrow J}$ if $i \leftarrow J \neq i' \leftarrow J'$. Then $y \in P_{\mathrm{F}}$.*

Proof. Since $x \in P_{\mathrm{F}}$, $x = \sum \alpha_k x^k$ where each x^k is an extreme point of P_{F} corresponding to an acyclic digraph. For each x^k define the vector y^k where $y^k_{i' \leftarrow J'} = 0$ and all other components of y^k are equal to those of x^k. Each y^k is also an extreme point corresponding to an acyclic digraph (a subgraph of x^k). We clearly have that $y = \sum \alpha_k y^k$ and so $y \in P_{\mathrm{F}}$. □

Lemma 19. *Let $x \in P_{\mathrm{F}}$ and let y be any vector such that $0 \leq y_{i' \leftarrow J'} \leq x_{i' \leftarrow J'}$ for some $i' \leftarrow J'$ and $y_{i \leftarrow J} = x_{i \leftarrow J}$ if $i \leftarrow J \neq i' \leftarrow J'$. Then $y \in P_{\mathrm{F}}$.*

Proof. If $x_{i' \leftarrow J'} = 0$ then $y = x$ and the result is immediate, so assume that $x_{i' \leftarrow J'} > 0$. Consider z which is identical to y except that $z_{i' \leftarrow J'} = 0$. We have $y = \frac{y_{i' \leftarrow J'}}{x_{i' \leftarrow J'}} x + \left(1 - \frac{y_{i' \leftarrow J'}}{x_{i' \leftarrow J'}}\right) z$. By Lemma 18 $z \in P_{\mathrm{F}}$. Since x is also in P_{F} and y is a convex combination of x and z it follows that $y \in P_{\mathrm{F}}$. □

Proposition 20. *$P_{\mathrm{F}}(V)$ is monotone.*

Proof. Suppose $x \in P_{\mathrm{F}}$ and $0 \leq y \leq x$. Construct a sequence of vectors $x = y^0, y^1, \ldots, y^k, \ldots, y^{|\mathbf{F}|} = y$ by replacing each component $x_{i \leftarrow J}$ by $y_{i \leftarrow J}$ one at a time (in any order). By Lemma 19 each $y^k \in P_{\mathrm{F}}$, so $y \in P_{\mathrm{F}}$. □

Hammer, Johnson, and Peled (1975) showed that a polytope is monotone if and only if it can be described by a system $x \geq 0$, $Ax \leq b$ with $A, b \geq 0$. This gives the following result for P_{F}.

Theorem 21. *Each facet-defining inequality of $P_{\mathrm{F}}(V)$ is either (i) a lower bound (of zero) on a family variable, or (ii) an inequality of the form $\pi x \leq \pi_0$, where $\pi \geq 0$ and $\pi_0 > 0$.*

Proof. From Proposition 20 and the result of Hammer et al. (1975) we have the result but with $\pi_0 \geq 0$. That $\pi_0 > 0$ follows directly by full-dimensionality. □

Proposition 22. *The following hold.*

1. *$x_{i \leftarrow J} \geq 0$ defines a facet of $P_{\mathrm{F}}(V, \mathcal{P})$ for all families $i \leftarrow J \in \mathbf{F}(V, \mathcal{P})$.*
2. *For all $i \in V$, if $J' \in \mathcal{P}(i')$ implies $\exists J \neq \emptyset \in \mathcal{P}(i)$ for all other $i' \in V$, where $i \notin J'$ or $i' \notin J$, then $\sum_{J \neq \emptyset, J \in \mathcal{P}(i)} x_{i \leftarrow J} \leq 1$ defines a facet of $P_{\mathrm{F}}(V, \mathcal{P})$.*

Proof. (1) follows from the monotonicity of $P_{\mathrm{F}}(V, \mathcal{P})$ (Hammer et al., 1975, Proposition 2). For (2) first define, for any $i \leftarrow J \in \mathbf{F}(V, \mathcal{P})$ the unit vector $e^{i \leftarrow J} \in \mathbb{R}^{\mathbf{F}(V, \mathcal{P})}$, where $e^{i \leftarrow J}_{i \leftarrow J} = 1$ and all other components of $e^{i \leftarrow J}$ are 0. For each $i \in V$ define $S_i = \{e^{i \leftarrow J} \mid J \neq \emptyset, J \in \mathcal{P}(i)\} \cup \{e^{i' \leftarrow J'} + e^{i \leftarrow J} \mid i' \neq i, J' \neq \emptyset, J' \in \mathcal{P}(i'), J \neq \emptyset,$ and either $i \notin J'$ or $i' \notin J\}$.

There is an obvious bijection between family variables and the elements of S_i so $|S_i| = |\mathbf{F}(V, \mathcal{P})|$. It is easy to see that the vectors in S_i are linearly independent (and thus affinely independent) and that each is an acyclic digraph satisfying $\sum_{J \neq \emptyset, J \in \mathcal{P}(i)} x_{i \leftarrow J} = 1$. The result follows. □

Recall that we use the name *modified convexity constraints* to describe inequalities of the form $\sum_{J \neq \emptyset, J \in \mathcal{P}(i)} x_{i \leftarrow J} \leq 1$. That each node can have exactly one parent set in any digraph is a convexity constraint. If we remove the empty parent set, this convexity constraint becomes an inequality, and is thus *modified*. We have now shown that each modified convexity constraint defines a facet of $P_{\mathrm{F}}(V, \mathcal{P})$ as long as a weak condition is met. In fact, we have found this weak condition to be almost always met in practice. Note also that it is always met when all parent sets are allowed (as long as $|V| > 2$).

We now show that if $\pi x \leq \pi_0$ defines a facet of the family-variable polytope, then, for each family, there is an acyclic digraph 'containing' that family for which $\pi x \leq \pi_0$ is 'tight'.

Proposition 23. *If $\pi x \leq \pi_0$ defines a facet of P_{F} which is not a lower bound on a family variable, then for all families $i \leftarrow J \in \mathbf{F}$, there exists an extreme point x of P_{F} such that $x_{i \leftarrow J} = 1$ and $\pi x = \pi_0$.*

Proof. Recall that by definition each extreme point of P_{F} is a zero-one vector (representing an acyclic digraph). Now suppose that there were some $i \leftarrow J \in \mathbf{F}$ such that $x_{i \leftarrow J} = 0$ for any extreme point x of P_{F} such that $\pi x = \pi_0$. Since $\pi x \leq \pi_0$ defines a facet, there is a set of $|\mathbf{F}|$ affinely independent extreme points satisfying $\pi x = \pi_0$. By our assumption, each such extreme point will also satisfy $x_{i \leftarrow J} = 0$. $x_{i \leftarrow J} \geq 0$ defines a facet. However, it is not possible for a set of $|\mathbf{F}|$ affinely independent points to lie on two distinct facets. The result follows. □

Proposition 23 helps us prove an important property of facet-defining inequalities of P_{F}: coefficients are non-decreasing as parent sets increase. The proof of the following proposition rests on the simple fact that removing edges from an acyclic digraph always results in another acyclic digraph.

Proposition 24. *Let $\pi x \leq \pi_0$ be a facet-defining inequality of P_{F}. Then $J \subseteq J'$ implies $\pi_{i \leftarrow J} \leq \pi_{i \leftarrow J'}$.*

Proof. Since $\pi x \leq \pi_0$ defines a facet, there exists an extreme point x' such that $x'_{i \leftarrow J'} = 1$ and $\pi x' = \pi_0$. Note that $x'_{i \leftarrow J} = 0$. Since x' is an extreme point, it encodes an acyclic digraph. Let y be identical to x' except that $y_{i \leftarrow J} = 1$ and $y_{i \leftarrow J'} = 0$. Since $J \subseteq J'$, y also encodes an acyclic digraph and so is in P_{F} so $\pi y \leq \pi_0 = \pi x'$. Thus $\pi y - \pi x' \leq 0$. However, $\pi y - \pi x' = \pi_{i \leftarrow J} - \pi_{i \leftarrow J'}$, and the result follows. □

6.2 Cluster Constraints are Facets of the Family Variable Polytope

In this section we show that each *κ-cluster inequality* is facet-defining for the family variable polytope in the special case where the cluster C is the entire node set V and where all parent sets are allowed for each vertex. The κ-cluster inequalities (Cussens, 2011) are a generalisation of cluster inequalities (3). The cluster inequalities (3) are κ-cluster inequalities for the special case of $\kappa = 1$.

In the next section (Section 6.3) we will show how to 'lift' facet-defining inequalities. This provides an easy generalisation (Theorem 29) of the result of this section which shows that, when all parent sets are allowed, *all* κ-cluster inequalities are facets, not just those for which $C = V$. As a special case, this implies that the cluster inequalities devised by

Jaakkola et al. (2010) are facets of the family variable polytope when all parent sets are allowed.

An alternative proof for the fact that κ-cluster inequalities are facet-defining was recently provided by Cussens et al. (2016, Corollary 4) The proof establishes not only that κ-cluster inequalities are facet-defining, but also that they are *score-equivalent*. A face of the family variable polytope is said to be score-equivalent if it is the optimal face for some *score equivalent objective*, where the *optimal face* of an objective is the face containing all optimal solutions. An objective function is score equivalent if it gives the same value to any two acyclic digraphs which are Markov equivalent (encode the same conditional independence relations). In later work, Studený (2015) went further and showed that κ-cluster inequalities form just part of a more general class of facet-defining inequalities which can be defined in terms of *connected matroids*. However, we believe that our proof, as presented in the following, is valuable since it relies only on a direct application of a standard technique for proving that an inequality is facet-defining, and does not require any connection to be made to score-equivalence, let alone matroid theory. In addition, the general result (our Theorem 29) further shows how our results on 'lifting' can be usefully applied.

First we define *κ-cluster inequalities*. There is a κ-cluster inequality for each cluster $C \subseteq V$, $|C| > 1$, and each $\kappa < |C|$ which states that there can be at most $|C| - \kappa$ nodes in C with at least κ parents in C. It is clear that such inequalities are at least valid, since all acyclic digraphs clearly satisfy them. We begin by considering the special case of $C = V$ where the κ-cluster inequality states that there can be at most $|V| - \kappa$ nodes with at least κ parents. We first introduce some helpful notation.

Definition 25. $\mathcal{P}_V$ is defined as follows: $\mathcal{P}_V(i) := 2^{V\setminus\{i\}}$, for all $i \in V$.

We will now show that κ-cluster inequalities are facet-defining.

Theorem 26. *For any positive integer $\kappa < |V|$, the following valid inequality defines a facet of the family variable polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$:*

$$\sum_{i \in V} \sum_{J \subseteq V\setminus\{i\}, |J| \geq \kappa} x_{i \leftarrow J} \leq |V| - \kappa. \tag{18}$$

Proof. An indirect method of establishing affine independence is used. It is given, for example, by Wolsey (1998, p.144). Let $x^1, \ldots, x^t$ be the set of all acyclic digraphs in $P_{\mathrm{F}}(V, \mathcal{P}_V)$ satisfying

$$\sum_{i \in V} \sum_{J \subseteq V\setminus\{i\}, |J| \geq \kappa} x_{i \leftarrow J} = |V| - \kappa. \tag{19}$$

Suppose that all these points lie on some generic hyperplane $\mu x = \mu_0$. Now consider the system of linear equations

$$\sum_{i \in V} \sum_{J \neq \emptyset, J \subseteq V\setminus\{i\}} \mu_{i \leftarrow J} x^{\iota}_{i \leftarrow J} = \mu_0 \text{ for } \iota = 1, \ldots, t. \tag{20}$$

Note that $\dim P_{\mathrm{F}}(V, \mathcal{P}_V) = |\mathbf{F}(V, \mathcal{P}_V)| = |V|(2^{|V|-1} - 1)$ and so there are the same number of $\mu_{i \leftarrow J}$ variables. The system (20), in the $|V|(2^{|V|-1} - 1) + 1$ unknowns (μ, μ_0), is now solved. This is done in three stages. First we show that $\mu_{i \leftarrow J}$ must be zero if $|J| < \kappa$. Then

we show that the remaining $\mu_{i\leftarrow J}$ must all have the same value. Finally, we show that this common value is 1 whenever μ_0 is $|V| - \kappa$.

To do this it is useful to consider acyclic tournaments on V. These are acyclic digraphs where there is a directed edge between each pair of distinct nodes. It is easy to see that

1. for any $\kappa < |V|$, every acyclic tournament on V satisfies (19), and that

2. for any $x_{i\leftarrow J}$ there is an acyclic tournament, where $x_{i\leftarrow J} = 1$.

Let x be an acyclic tournament on V with $x_{i\leftarrow J} = 1$ for some $i \in V$, $|J| < \kappa$, i.e. J is the non-empty parent set for i in x. Now consider x' which is identical to x except that i has no parents, so that $x - x' = e^{i\leftarrow J}$. Since x is an acyclic tournament it satisfies (19). But it is also easy to see that x' satisfies (19), since no parent set of size at least κ has been removed. So $\mu_{i\leftarrow J} = \mu e^{i\leftarrow J} = \mu(x - x') = \mu x - \mu x' = \mu_0 - \mu_0 = 0$. $\mu_{i\leftarrow J} = 0$ whenever $|J| < \kappa$. Call this Result 1.

Consider now two distinct parent sets J and J' for some $i \in V$ where $J \geq \kappa$ and $J' \geq \kappa$. Let g be an acyclic tournament on the node set $V \setminus \{i\}$. Let x be the acyclic digraph on node set V obtained by adding $\{i\}$ to g and drawing edges from each member of J to i. Similarly, let x' be the acyclic digraph obtained by drawing edges from J' to i instead, so that $x - x' = e^{i\leftarrow J} - e^{i\leftarrow J'}$. It is not difficult to see that both x and x' satisfy (19). So $\mu_{i\leftarrow J} - \mu_{i\leftarrow J'} = \mu(e^{i\leftarrow J} - e^{i\leftarrow J'}) = \mu(x - x') = \mu x - \mu x' = \mu_0 - \mu_0 = 0$. So $\mu_{i\leftarrow J} = \mu_{i\leftarrow J'}$. Call this Result 2.

Now consider variables $x_{i\leftarrow J}$ and $x_{i'\leftarrow J'}$ where $i \neq i'$, $J \cup \{i\} = J' \cup \{i'\}$ and $|J| = |J'| = \kappa$. First note that in an acyclic tournament, (i) there is exactly one parent set of each size $0, \ldots, \kappa, \ldots |V| - 1$ and so (ii) the nodes of an acyclic tournament can be totally ordered according to parent set size, and thus (iii) any total ordering of nodes determines a unique acyclic tournament. Let x be any acyclic tournament where $x_{i\leftarrow J} = 1$ and $x_{i'\leftarrow J^{(<\kappa)}} = 1$ for some parent set $J^{(<\kappa)}$ where $|J^{(<\kappa)}| < \kappa$. Clearly there are many such acyclic tournaments. Note that since x is an acyclic tournament, $J^{(<\kappa)} \subseteq J \setminus \{i, i'\}$. Now consider the acyclic tournament x' produced by swapping i and i' in the total order associated with x. This generates an acyclic tournament x' where $x'_{i'\leftarrow J'} = 1$ and $x'_{i\leftarrow J^{(<\kappa)}} = 1$. Note that components of x and x' corresponding to family variables with parent set size strictly above κ are equal. Components of μ corresponding to family variables with parent set size strictly below κ all equal zero. From this we have that $\mu x - \mu x' = \mu_{i\leftarrow J} - \mu_{i\leftarrow J'}$. Since $\mu x - \mu x' = \mu_0 - \mu_0 = 0$, this shows that $\mu_{i\leftarrow J} = \mu_{i'\leftarrow J'}$ Call this Result 3.

Now consider a pair of variables $\mu_{i\leftarrow J''}$ and $\mu_{i'\leftarrow J'''}$ where $i \neq i'$, and the only restriction is that $|J''|, |J'''| \geq \kappa$. If some other pair of variables $\mu_{i\leftarrow J}$ and $\mu_{i'\leftarrow J'}$ meet the conditions of Result 3, then $\mu_{i\leftarrow J} = \mu_{i'\leftarrow J'}$. However, by Result 2 $\mu_{i\leftarrow J''} = \mu_{i\leftarrow J}$ and $\mu_{i'\leftarrow J'''} = \mu_{i'\leftarrow J'}$. Thus $\mu_{i\leftarrow J''} = \mu_{i'\leftarrow J'''}$.

So by the transitivity of equality $\mu_{i\leftarrow J} = \mu_{i'\leftarrow J'}$ for any i, i', J, J' where $|J| \geq \kappa$, $|J'| \geq \kappa$. Recall that we also have that $\mu_{i\leftarrow J} = 0$ whenever $|J| < \kappa$.

Suppose that $\mu_0 = 0$. Since all non-zero $\mu_{i\leftarrow J}$ are equal and thus have the same sign, the only possible solution is for all $\mu_{i\leftarrow J} = 0$. Suppose then instead that $\mu_0 \neq 0$. Then wlog we can set $\mu_0 = |V| - \kappa$. In each of the t equations (20), after substituting $\mu_{i\leftarrow J} = 0$ for $|J| < \kappa$, we have $|V| - \kappa$ terms on the left hand side (LHS) which are known to be equal. On the right hand side (RHS) the value is $|V| - \kappa$, so all terms on the LHS must equal one.

Each term $\mu_{i\leftarrow J}$ where $|J| \geq \kappa$, occurs in at least one of t equations (20), so this is enough to establish that $\mu_{i\leftarrow J} = 1$ whenever $|J| \geq \kappa$. Thus, unless all $\mu_{i\leftarrow J} = 0$, the only possible solution to the system of linear equations (20) with RHS $|V| - \kappa$ is

- $\mu_{i\leftarrow J} = 0$ if $|J| < \kappa$, and
- $\mu_{i\leftarrow J} = 1$ if $|J| \geq \kappa$.

These values match those in (19) and so (18) is facet-defining. □

6.3 Lifting Facets of the Family Variable Polytope

In this section we show that if all parent sets are allowed, then facet-defining inequalities for the family variable polytope for some node set V can be 'lifted' to provide facets for any family variable polytope for an enlarged node set $V' \supsetneq V$.

Lemma 27. *Recall that $\mathcal{P}_V(i) := 2^{V\setminus\{i\}}$ for all $i \in V$. Let*

$$\sum_{i\in V} \sum_{J\in\mathcal{P}_V(i), J\neq\emptyset} \alpha_{i\leftarrow J} x_{i\leftarrow J} \leq \beta \tag{21}$$

be a facet-defining inequality for the family variable polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$ which is not a lower bound on a variable. Let $V' = V \cup \{i'\}$ where $i' \notin V$. Then

$$\sum_{i\in V} \sum_{J\in\mathcal{P}_V(i), J\neq\emptyset} \alpha_{i\leftarrow J}(x_{i\leftarrow J} + x_{i\leftarrow J\cup\{i'\}}) \leq \beta \tag{22}$$

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

Proof. Since (21) is facet-defining, there is a set $S_0 \subseteq \mathbb{R}^{\mathbf{F}(V,\mathcal{P}_V)}$ of affinely independent acyclic digraphs, with node set V, lying on its hyperplane. For each acyclic digraph in S_0, create an acyclic digraph with node set $V \cup \{i'\}$ by adding i' as an isolated node. Let $S_1 \subseteq \mathbb{R}^{\mathbf{F}(V',\mathcal{P}_{V'})}$ be the set of acyclic digraphs so created. Note that all members of S_1 lie on the hyperplane for (22). Each vector in S_1 corresponds to a vector in S_0 with a zero vector of length $|\mathbf{F}(V', \mathcal{P}_{V'})| - |\mathbf{F}(V, \mathcal{P}_V)|$ concatenated. Since S_0 is an affinely independent set, so is S_1.

For each non-empty subset $J \subseteq V$, construct an acyclic digraph by adding $e^{i'\leftarrow J}$ to an arbitrary member of S_1. Clearly the end result is an acyclic digraph lying on the hyperplane for (22). Let S_2 be the set of all such acyclic digraphs.

For each $J \subseteq V$, $i \in V$, construct an acyclic digraph by finding an acyclic digraph $x \in S_1$ such that $x_{i\leftarrow J} = 1$ and adding an arrow from i' to i. Note that it is always possible to find an acyclic digraph with $x_{i\leftarrow J} = 1$. If this were not the case, then (21) would be a lower bound on $x_{i\leftarrow J}$. It is not difficult to see that any such acyclic digraph lies on the hyperplane defined by (22). Let S_3 be the set of all such acyclic digraphs.

Let $S = S_1 \cup S_2 \cup S_3$. S_2 and S_3 have exactly one acyclic digraph for each component $x_{i\leftarrow J}$ involving the node i' (either $i = i'$ or $i' \in J$). S_1 has an acyclic digraph for each component $x_{i\leftarrow J}$ not involving i'. So $|S| = \dim P_{\mathrm{F}}(\mathbf{F}(V', \mathcal{P}_{V'})) = |\mathbf{F}(V', \mathcal{P}_{V'})|$. It remains to be established that the S is a set of affinely independent vectors.

Suppose $\sum_{x^i \in S} \alpha_i x^i = 0$ and $\sum_{x^i \in S} \alpha_i = 0$. Each component $x_{i \leftarrow J}$ involving i' is set to 1 in exactly one acyclic digraph in $S_2 \cup S_3$. Thus $\alpha_i = 0$ for $x^i \in S_2 \cup S_3$. So $\sum_{x^i \in S_1} \alpha_i x^i = 0$ and $\sum_{x^i \in S_1} \alpha_i = 0$. The result then follows from the affine independence of the set S_1. □

Theorem 28. *Recall that $\mathcal{P}_V(i) := 2^{V \setminus \{i\}}$ for all $i \in V$. Let*

$$\sum_{i \in V} \sum_{J \in \mathcal{P}_V(i), J \neq \emptyset} \alpha_{i \leftarrow J} x_{i \leftarrow J} \leq \beta \tag{23}$$

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

$$\sum_{i \in V} \sum_{J \in \mathcal{P}_V(i), J \neq \emptyset} \alpha_{i \leftarrow J} \left(\sum_{J' : J \subseteq J' \subseteq V' \setminus \{i\}} x_{i \leftarrow J'} \right) \leq \beta \tag{24}$$

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

Proof. Repeated application of Lemma 27. □

Using Theorem 28, Theorem 26 can now be 'lifted' to establish that all k-cluster inequalities are facet-defining.

Theorem 29. *Recall that $\mathcal{P}_V(i) := 2^{V \setminus \{i\}}$ for all $i \in V$. For any $C \subseteq V$ and any positive integer $\kappa < |C|$, the valid inequality*

$$\sum_{i \in C} \sum_{J \subseteq V \setminus \{i\} : |J \cap C| \geq \kappa} x_{i \leftarrow J} \leq |C| - \kappa \tag{25}$$

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

Proof. By Theorem 26, (25) is facet-defining for the family variable polytope for node set C. By applying Theorem 28 it follows that it also facet-defining for the family variable polytope for any node set $V \supseteq C$. □

6.4 Facets When Parent Sets Are Restricted

The results in the preceding sections have all been for the special case $\mathcal{P}_V$ when all possible parent sets are allowed for each node. If some parent sets are ruled out, for example by an upper bound κ on parent set cardinality, then some κ-cluster inequalities and some modified convexity constraints may not be facets.

To see this, suppose we had $V = \{a, b, c\}$. If all parent sets are allowed, then Theorem 29 shows that this 2-cluster inequality for $C = \{a, b, c\}$,

$$x_{a \leftarrow \{b,c\}} + x_{b \leftarrow \{a,c\}} + x_{c \leftarrow \{a,b\}} \leq 1, \tag{26}$$

is facet-defining. However, if $\{a, b\}$ is not allowed as a parent set for c, then the inequality becomes

$$x_{a \leftarrow \{b,c\}} + x_{b \leftarrow \{a,c\}} \leq 1, \tag{27}$$

which is not facet-defining since it is dominated by the 1-cluster inequality for $C = \{a, b\}$,

$$x_{a\leftarrow\{b\}} + x_{a\leftarrow\{b,c\}} + x_{b\leftarrow\{a\}} + x_{b\leftarrow\{a,c\}} \leq 1. \tag{28}$$

As another example, suppose $\{c\}$ were removed from $\mathcal{P}(a)$. Then condition 2 of Proposition 22 is no longer met, and the modified convexity constraint for a becomes

$$x_{a\leftarrow\{b\}} + x_{a\leftarrow\{b,c\}} \leq 1, \tag{29}$$

which cannot be facet-defining since it is dominated by the inequality (28).

For any $\mathcal{P}$ we have that the polytope $P_{\mathrm{F}}(V, \mathcal{P})$ is a *face* of the all-parent-sets-allowed polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$ defined by the valid inequality

$$\sum_{i\in V} \sum_{J\in\mathcal{P}_V(i)\setminus\mathcal{P}(i)} x_{i\leftarrow J} \geq 0. \tag{30}$$

The issue then is whether it is possible to determine when a facet of $P_{\mathrm{F}}(V, \mathcal{P}_V)$ is also a facet of this face. The issue of determining the facets of a face is of general interest. As Boyd and Pulleyblank (2009) note "As it is often technically much simpler to obtain results about facets for a full dimensional polyhedron than one of lower dimension, it would be nice to ... know under what conditions an inequality inducing a facet of P also induces a facet of a face F of P ." They go on to state that "... we know of no reasonable general result of this type".

However, in the case of the the family variable polytope, there is a strong result which shows that many facets of a family variable polytope $P_{\mathrm{F}}(V, \mathcal{P})$ induce facets of a lower-dimensional family variable polytope $P_{\mathrm{F}}(V, \breve{\mathcal{P}})$ where $\breve{\mathcal{P}}(i) \subseteq \mathcal{P}(i)$ for all $i \in V$. In particular, this result shows that some facets of the all-parent-sets-allowed polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$ are also facets of the polytope that results by limiting the cardinality of parent sets. To establish this result we first prove a lemma.

Lemma 30. *Let $x \in P_{\mathrm{F}}(V, \mathcal{P})$. Let $i \in V$ and let $J, J' \in \mathcal{P}(i)$ with $J \subsetneq J'$, $J \neq \emptyset$. Define $\breve{x}$ as follows: $\breve{x}_{i\leftarrow J} = x_{i\leftarrow J} + x_{i\leftarrow J'}$, $\breve{x}_{i\leftarrow J'} = 0$, and x and $\breve{x}$ are equal in all other components. Then $\breve{x}$ is also in the family-variable polytope $P_{\mathrm{F}}(V, \mathcal{P})$.*

Proof. Since $x \in P_{\mathrm{F}}(V, \mathcal{P})$, $x = \sum_{k=1}^{K} \alpha_k x^k$ where each x^k is an extreme point of $P_{\mathrm{F}}(V, \mathcal{P})$ corresponding to an acyclic digraph. For each x^k define $\breve{x}^k$ as follows: $\breve{x}^k_{i\leftarrow J} = x^k_{i\leftarrow J} + x^k_{i\leftarrow J'}$, $\breve{x}^k_{i\leftarrow J'} = 0$ and x^k and $\breve{x}_k$ are equal in all other components. It is clear that each $\breve{x}^k$ corresponds to an acyclic digraph which differs from x^k iff J' is the parent set for i in x^k, in which case J becomes the parent set for i in $\breve{x}^k$. The digraph remains acyclic since $J \subsetneq J'$. It is also clear that $\breve{x} = \sum_{k=1}^{K} \alpha_k \breve{x}^k$ and so $\breve{x} \in P_{\mathrm{F}}(V, \mathcal{P})$. □

The main result of this section now follows. Our proof makes use of the elementary but useful fact that the number of linearly independent rows in a matrix (row rank) and the number of linearly independent columns in a matrix (column rank) are equal.

Theorem 31. *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 $P_{\mathrm{F}}(V, \breve{\mathcal{P}})$.*

Proof. Since $\pi x \leq \pi_0$ is facet-defining for $P_{\mathrm{F}}(V, \mathcal{P})$ it is obvious by Theorem 21 that $\breve{\pi} x \leq \pi_0$ is at least a valid inequality for $P_{\mathrm{F}}(V, \breve{\mathcal{P}})$. We now show that this valid inequality defines a facet by proving the existence of $|\mathbf{F}(V, \breve{\mathcal{P}})|$ affinely independent points lying in the facet.

Recall that $\mathbf{F}(V, \mathcal{P})$ is the set of families determined by vertices V and allowed parent sets $\mathcal{P}$. Abbreviate $|\mathbf{F}(V, \mathcal{P})|$ to m and note that $|\mathbf{F}(V, \breve{\mathcal{P}})| = m - 1$. Since $\pi x \leq \pi_0$ defines a facet for the family-variable polytope $P_{\mathrm{F}}(V, \mathcal{P})$, there are m affinely independent points $x^1, \ldots, x^k, \ldots, x^m$ lying in this facet (i.e. $\pi x^k = \pi_0$, $x^k \in P_{\mathrm{F}}(V, \mathcal{P})$ for $k = 1, \ldots, m$). Since these points are affinely independent, the points $(x^1, 1), \ldots, (x^k, 1), \ldots, (x^m, 1)$ in $\mathbb{R}^{m+1}$ are linearly independent.

Let A_1 be the $m \times (m + 1)$ matrix whose rows are the $(x^k, 1)$. Since the rows are linearly independent, A_1 has rank m. Construct a new matrix A_2 by adding the column for family $i \leftarrow J'$ to that for $i \leftarrow J$. Since this is an elementary operation it does not change the rank of the matrix (Cohn, 1982), and so A_2 has rank m. Now construct an $m \times m$ matrix A_3 by removing the column for $i \leftarrow J'$ from A_2. Denote the rows of A_3 by $(\breve{x}^1, 1), \ldots, (\breve{x}^k, 1), \ldots, (\breve{x}^m, 1)$. From Lemma 30 it follows that each $\breve{x}^k$ is in $P_{\mathrm{F}}(V, \breve{\mathcal{P}})$. Since $\pi_{i \leftarrow J} = \pi_{i \leftarrow J'}$, it is not difficult to see that each $\breve{x}^k$ satisfies $\breve{\pi} x = \pi_0$. Since A_2 has rank m, there are m linearly independent columns in A_2 and, since A_3 is A_2 with one column removed, at least $m - 1$ linearly independent columns in A_3. So A_3 has rank of at least $m - 1$. But this means that there are $m - 1$ linearly independent rows in A_3, so there are $m - 1$ points among the $\breve{x}^k$ that are affinely independent. So there are $m - 1$ affinely independent points in $P_{\mathrm{F}}(V, \breve{\mathcal{P}})$ satisfying $\breve{\pi} x = \pi_0$ and thus $\breve{\pi} x \leq \pi_0$ defines a facet of $P_{\mathrm{F}}(V, \breve{\mathcal{P}})$. □

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 31 states 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 Appendix A this is confirmed for the case where $|V| = 4$ and $\kappa = 2$. It follows that a normal ($k = 1$) cluster constraint is a facet for *any* limit κ on the size of parent sets. This explains why normal cluster constraints are more useful to look for than k-cluster constraints for $k > 1$. In GOBNILP, although the user can ask the system to look for k-cluster constraints up to some defined limit $k \leq K$, the default is to only search for normal ($k = 1$) cluster constraints since this has been observed to lead to faster solving.

7. Faces of the Family Variable Polytope Defined by Orders and by Sinks

In this section we analyse faces of the all-parent-sets-allowed family variable polytope defined by total orders and sink nodes, respectively. Faces of a polytope are themselves polytopes, and in this section we establish a complete characterisation of the facets of both types of polytope. Moreover, the faces defined by sink nodes lead to a useful *extended representation* for the family variable polytope which can be used to relate family variable polytopes for different numbers of nodes.

7.1 Order-Defined Faces

Let $<$ be some total order on the node set V. An acyclic digraph (V, B) is *consistent with* $<$ if $i \leftarrow j \in B \Rightarrow j < i$, so that parents come before children in the ordering. The valid inequality $\sum_{i,J:(\exists j \in J \text{ s.t. } i<j)} x_{i \leftarrow J} \geq 0$ defines a face of the family variable polytope

$$P_{\mathrm{F}}(V, <) = \Big\{ x \in P_{\mathrm{F}}(V, \mathcal{P}_V) \;\Big|\; \sum_{i,J:(\exists j \in J: i<j)} x_{i \leftarrow J} = 0 \Big\}. \tag{31}$$

In $P_{\mathrm{F}}(V, <)$ each family variable inconsistent with $<$ is set to zero. This is the only restriction on x. So clearly all acyclic digraphs consistent with $<$ lie on the face $P_{\mathrm{F}}(V, <)$ and no digraphs inconsistent with $<$ do. It is also clear that any acyclic digraph lies on $P_{\mathrm{F}}(V, <)$ for at least one choice of $<$.

Remark 32. *Abbreviate $|V|$ to p. We have that $\dim(P_{\mathrm{F}}(V, <)) = 2^p - p - 1$. If the family variables clamped to zero in $P_{\mathrm{F}}(V, <)$ are removed, $P_{\mathrm{F}}(V, <)$ is full-dimensional in $\mathbb{R}^{2^p - p - 1}$. (Recall that $\dim(P_{\mathrm{F}}(V, \mathcal{P}_V)) = p(2^{p-1} - 1)$.)*

Remark 33. *If x is an extreme point of $P_{\mathrm{F}}(V, \mathcal{P}_V)$, then $x \in \bigcup_{<} P_{\mathrm{F}}(V, <)$.*

Note that exactly one acyclic tournament lies on $P_{\mathrm{F}}(V, <)$ for any choice of $<$.

Proposition 34. *The facet-defining inequalities of the full-dimensional polytope $P_{\mathrm{F}}(V, <) \subseteq \mathbb{R}^{2^p - p - 1}$ are*

1. *the variable lower bounds $x_{i \leftarrow J} \geq 0$, and*
2. *the modified convexity constraints $\sum_{J \subseteq V: J \neq \emptyset, j \in J \rightarrow j < i} x_{i \leftarrow J} \leq 1$,*

where variables $x_{i \leftarrow J}$ with $j \in J, i < j$ have been removed.

Proof. Let $c \in \mathbb{R}^{2^p - p - 1}$ be an arbitrary objective coefficient vector. Consider solving the LP with objective c subject to the linear inequalities given above. It is clear that an optimal solution to this LP is obtained by choosing a parent set J for each $i \in V$ such that $c_{i \leftarrow J}$ is maximal (or choosing none if all $c_{i \leftarrow J}$ are negative or there are no parent sets available). This is an integer solution. The result follows. □

7.2 Sink-Defined Faces

For some particular $j \in V$, consider the valid inequality $\sum_{i \neq j, j \in J} x_{i \leftarrow J} \geq 0$. This defines a face $P_{\mathrm{F}}(V, j)$ of the family variable polytope as

$$P_{\mathrm{F}}(V, j) := \Big\{ x \in P_{\mathrm{F}}(V, \mathcal{P}_V) \;\Big|\; \sum_{j \in J, i \neq j} x_{i \leftarrow J} = 0 \Big\}. \tag{32}$$

This face contains all acyclic digraphs for which j is a *sink*—it has no children. Since every acyclic digraph has at least one sink, each extreme point of the family variable polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$ lies on a face $P_{\mathrm{F}}(V, j)$ for at least one choice of j.

Remark 35. *Abbreviate $|V|$ to p and recall that $\dim(P_{\mathrm{F}}(V,\mathcal{P}_V)) = p(2^{p-1}-1)$. We have that $\dim(P_{\mathrm{F}}(V,j)) = \dim(P_{\mathrm{F}}(V\setminus\{j\},\mathcal{P}_{V\setminus\{j\}})) + 2^{p-1} - 1 = (p-1)(2^{p-2}-1) + 2^{p-1} - 1 = (p+1)2^{p-2} - p$. If the family variables clamped to zero in $P_{\mathrm{F}}(V,j)$ are removed, $P_{\mathrm{F}}(V,j)$ is full-dimensional in $\mathbb{R}^{(p+1)2^{p-2}-p}$.*

Remark 36. *Every acyclic digraph contains at least one sink. So if x is an extreme point of $P_{\mathrm{F}}(V,\mathcal{P}_V)$, then $x \in \bigcup_{j\in V} P_{\mathrm{F}}(V,j)$.*

Proposition 37. *The facet-defining inequalities of the full-dimensional polytope $P_{\mathrm{F}}(V,j) \subseteq \mathbb{R}^{(p+1)2^{p-2}-p}$ are*

1. *the facet-defining inequalities of the polytope $P_{\mathrm{F}}(V\setminus\{j\},\mathcal{P}_{V\setminus\{j\}})$, and*
2. *the modified convexity constraint for j, namely $\sum_{J\subseteq V\setminus\{j\},J\neq\emptyset} x_{j\leftarrow J} \leq 1$.*

Proof. Let $c \in \mathbb{R}^{(p+1)2^{p-2}-p}$ be an arbitrary objective coefficient vector and consider solving the LP with objective c subject to the linear inequalities given above. Since j is constrained to be a sink, an optimal solution in $P_{\mathrm{F}}(V,j)$ is obtained by choosing a maximally scoring parent set for j and then an optimal acyclic digraph for $V\setminus\{j\}$. Since we have all the facets of the polytope $P_{\mathrm{F}}(V\setminus\{j\})$, the optimal acyclic digraph for $V\setminus\{j\}$ is a maximal solution to the LP restricted to the relevant variables. So the full LP has an integer solution. The result follows. □

7.3 A Sink-Based Extended Representation for the Family Variable Polytope

Since $P_{\mathrm{F}}(V,j) \subseteq P_{\mathrm{F}}(V,\mathcal{P}_V)$, for each $j \in V$ we have $\bigcup_{j\in V} P_{\mathrm{F}}(V,j) \subseteq P_{\mathrm{F}}(V,\mathcal{P}_V)$ and so $\mathrm{conv}\left(\bigcup_{j\in V} P_{\mathrm{F}}(V,j)\right) \subseteq \mathrm{conv}\left(P_{\mathrm{F}}(V,\mathcal{P}_V)\right) = P_{\mathrm{F}}(V,\mathcal{P}_V)$. However, as noted in Remark 36, if x is an extreme point of $P_{\mathrm{F}}(V,\mathcal{P}_V)$, then $x \in \bigcup_{j\in V} P_{\mathrm{F}}(V,j)$, so $P_{\mathrm{F}}(V,\mathcal{P}_V) \subseteq \mathrm{conv}\left(\bigcup_{j\in V} P_{\mathrm{F}}(V,j)\right)$, and thus $P_{\mathrm{F}}(V,\mathcal{P}_V) = \mathrm{conv}\left(\bigcup_{j\in V} P_{\mathrm{F}}(V,j)\right)$. Since there are only $|V| = p$ sink-defined faces, this leads to a compact extended representation for the family variable polytope $P_{\mathrm{F}}(V,\mathcal{P}_V)$ in terms of the polytopes $P_{\mathrm{F}}(V,j)_{j\in V}$. Since by Proposition 37 each $P_{\mathrm{F}}(V,j)$ can be defined using $P_{\mathrm{F}}(V\setminus\{j\})$, this allows $P_{\mathrm{F}}(V,\mathcal{P}_V)$ to be defined by the $P_{\mathrm{F}}(V\setminus\{j\},\mathcal{P}_{V\setminus\{j\}})$. In Appendix B we detail how this is done for the specific case of $|V| = 4$; here we describe the method for the general case.

A union of polytopes can be modelled by introducing additional variables. We follow the (standard) approach described by Conforti et al. (2014, §2.11). For each $j \in V$, we introduce a binary variable x_j and add the constraint

$$\sum_{j\in V} x_j = 1, \tag{33}$$

where x_j indicates that node j is a distinguished sink. The constraint states that in each acyclic digraph we can choose exactly one sink as the distinguished sink for that digraph.

Next, for each $j \in V$, $i \leftarrow J \in \mathbf{F}(V,\mathcal{P}_V)$, we introduce a new variable $x_{j,i\leftarrow J}$ indicating that i has J as its (non-empty) parent set and that j is the distinguished sink. In other

words $x_{j,i\leftarrow J} = x_j x_{i\leftarrow J}$. We add the following constraints linking the $x_{j,i\leftarrow J}$ to the original $x_{i\leftarrow J}$:

$$x_{i\leftarrow J} = \sum_{j\in V} x_{j,i\leftarrow J}. \tag{34}$$

Denote the vector of $x_{j,i\leftarrow J}$ components for some j as x^j. Then for each $j \in V$ and each facet-defining inequality $\pi x \leq \pi_0$ of $P_{\mathrm{F}}(V, j)$ we add the constraint

$$\pi^j x^j \leq \pi_0 x_j, \tag{35}$$

where $\pi^j_{j,i\leftarrow J} = \pi_{i\leftarrow J}$, and also the variable bounds

$$0 \leq x_{j,i\leftarrow J} \leq x_j. \tag{36}$$

Equations and inequalities (33–36) define $\bigcup_{j\in V} P_{\mathrm{F}}(V, j)$. To formulate $P_{\mathrm{F}}(V, \mathcal{P}_V) = \mathrm{conv}\big(\bigcup_{j\in V} P_{\mathrm{F}}(V, j)\big)$, it suffices to merely drop the integrality condition on the x_j variables, thus allowing $P_{\mathrm{F}}(V, \mathcal{P}_V)$ to be defined in terms of the lower-dimensional $P_{\mathrm{F}}(V, j)$.

8. Relating BNSL and the Acyclic Subgraph Problem

As the final contribution of this article, we establish a tight connection between BNSL and the acyclic subgraph problem.

8.1 BNSL as the Acyclic Subgraph Problem

BNSL is closely related to the well-known *acyclic subgraph problem* (*ASP*) (Grötschel, Jünger, & Reinelt, 1985). An instance of ASP is defined by digraph $D = (V, A)$ with edge weights $c(i \leftarrow j) \in \mathbb{R}$ for every edge $i \leftarrow j \in A$, and the goal is to find an acyclic subdigraph $D' = (V, B)$ of D which maximises

$$\sum_{i\leftarrow j\in B} c(i \leftarrow j). \tag{37}$$

In ASP, the objective function is a linear function of (indicators for) the edges of some digraph; in BNSL, by contrast, the aim is to maximise an objective which is a linear function of (indicators for) *sets* of edges. As a Bayesian network structure learning instance can consist of up to $\Omega(2^n)$ input values, it is presumably in general not possible to encode a BNSL instance as a ASP instance over the same node set as the original BNSL instance, as this would require in the worst case encoding an exponential number of parent set scores into a quadratic number of edge weights. However, we will next show that we can construct a BNSL-to-ASP reduction by introducing new nodes to represent all possible parent sets of the original instances $(V, \mathcal{P}, c)$, similarly as in Theorem 2.

Theorem 38. *Given BNSL instance $(V_1, \mathcal{P}, c)$, we can construct an ASP instance $D = (V, A)$ such that*

1. *$|V| = O\big(|V_1| + |\mathcal{F}(V_1, \mathcal{P})|\big)$, and*
2. *there is one-to-one correspondence between the optimal solutions of D and $(V_1, \mathcal{P}, c)$.*

Moreover, given $(V_1, \mathcal{P}, c)$*, the instance* D *can be constructed in time* $\text{poly}\big(|V_1|+|\mathcal{F}(V_1,\mathcal{P})|\big)$.

Proof. Define the digraph $D = (V, A)$ where $V = V_1 \cup V_2 \cup V_3$ and

- $V_2 = \{J \subseteq V_1 \mid \ J \in \mathcal{P}(i) \text{ for some } i \in V_1\}$,
- $V_3 = \{i \leftarrow J \mid i \in V, J \in \mathcal{P}(i)\}$.

See Figure 6 for an example node set where V_1 is on the top row, V_2 the middle one and V_3 the bottom row.

The edge set for D is the disjoint union of four (colour-coded) edge sets $A = A_1 \cup A_2 \cup A_3 \cup A_4$ where

- $A_1 = \{(i, J) \mid i \in V_1, J \in V_2, i \in J\}$ (blue),
- $A_2 = \{(i \leftarrow J, i) \mid i \leftarrow J \in V_3, i \in V_1\}$ (black),
- $A_3 = \{(J, i \leftarrow J) \mid J \in V_2, i \leftarrow J \in V_3\}$ (red), and
- $A_4 = \{(i \leftarrow J, J') \mid i \leftarrow J \in V_3, J' \in V_2, i \notin J', J \neq J'\}$ (green).

These four edge sets are coloured correspondingly in the example of Figure 6.

Define an ASP instance for $D = (V, A)$ where each (red) edge in A_3 $(J, i \leftarrow J)$ has weight $c_{i \leftarrow J}$; we will assume that the scores $c(i \leftarrow J)$ are strictly positive for all feasible parent sets choices, as adding the same value to each score will not change the optimal structures. All other edges receive a weight sufficiently big to ensure that they are included in any optimal acyclic edge set. For example, giving each such edge a weight equal to a sum of all $c_{i \leftarrow J}$ weights plus 1 will suffice.

Note that $(V, A \setminus A_3)$ is acyclic. Recall also the objective coefficients of the ASP instance have been chosen to ensure that $A \setminus A_3 \subseteq B$ for any optimal edge set B in D. Intuitively, we will thus only care about how the optimal solution looks on the edge set A_3, and use this information to recover a solution to the original BNSL instance.

Let (V, B) be an optimal solution to ASP instance D and define a digraph (V_1, B') as follows: $B' = \{i \leftarrow j \mid j \in J \text{ and } (J, i \leftarrow J) \in B\}$. We will show (i) there is exactly one edge of form $(J, i \leftarrow J) \in A_3$ for each $i \in V_1$, (ii) the graph (V_1, B') graph is acyclic and (iii) that it is an optimal solution to the given BNSL instance $(V_1, \mathcal{P}, c)$.

(i) Suppose that $(J, i \leftarrow J)$ and $(J', i \leftarrow J')$ were both in B for some $i \in V_1$ and $J, J' \in V_2, J \neq J'$. This is not possible because the edges $(i \leftarrow J, J')$ and $(i \leftarrow J', J)$ are both in A_4 and thus in B. Having $(J, i \leftarrow J)$ and $(J', i \leftarrow J')$ both in B would cause a cycle $(i \leftarrow J) \to J \to (i \leftarrow J') \to J \to (i \leftarrow J)$ in B, and so is impossible.

(ii) For any i, j, J with $i \neq j, j \in J$, there exist the following edges: the blue edge $(j, J) \in A_1$ and the black edge $(i \leftarrow J, i) \in A_2$. Note that both of these edges will be in B. If the red edge $(J, i \leftarrow J) \in A_3$ is also in B then we have the following path in B: $j \to J \to (i \leftarrow J) \to i$. So if j is a parent of i in B', then there is a path from j to i in B. So if there were a cycle $i_1 \to i_2 \ldots i_n \to i_1$ in B' there would be a cycle from i_1 to i_1 in B. Since B is acyclic this is a contradiction and so B' must also be acyclic.

(iii) We first show that any feasible solution G to the BNSL instance $(V_1, \mathcal{P}, c)$ corresponds to a feasible solution to the ASP instance $D = (V, A)$. This feasible solution to

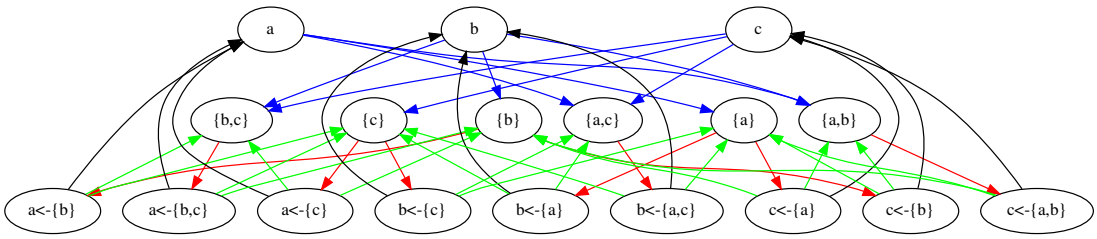

Figure 6: ASP digraph for the BNSL instance with node set $\{a, b, c\}$

$D = (V, A)$ consists of the edges $A_1 \cup A_2 \cup A_4$ together with those red edges in A_3 corresponding to the parent set choices for G. We need to show that this edge set—call it $B(G)$—is acyclic in $D = (V, A)$. Since G is acyclic there is a total order $<_{V_1}$ on the nodes V_1 such that parents always come before children in this order. We show that $<_{V_1}$ determines a total order $<_V$ on the nodes V such that parents always come before children in $B(G)$ which establishes that $B(G)$ is acyclic.

To aid understanding we first do this for the case where $V_1 = \{a, b, c\}$ and G is such that $a <_{V_1} b <_{V_1} c$. The general result is established later. In the special case all red edges (in A_3) which are inconsistent with $<_{V_1}$ will be absent from $B(G)$. In particular, since a is allowed no parents, the red edges going to the nodes $a \leftarrow \{b\}$, $a \leftarrow \{c\}$, and $a \leftarrow \{b, c\}$ will be absent. This means that these nodes are source nodes in $B(G)$, so put these as the first 3 elements of the order $<_V$. Since $a \in V_1$ has only these 3 nodes as parents, put a as the 4th element in $<_V$. Since the only parent for $\{a\}$ is a, put $\{a\}$ as the 5th element. Since c cannot be a parent of b, the red arrows going to $b \leftarrow \{c\}$ and $b \leftarrow \{a, c\}$ are absent from $B(G)$, so these nodes are sources in $B(G)$. Also the only arrow going to $b \leftarrow \{a\}$ is from $\{a\}$ which is already in the order. This allows us to put $b \leftarrow \{a\}$, $b \leftarrow \{c\}$ and $b \leftarrow \{a, c\}$ as the next elements in $<_V$. Having done this b can be placed next, and then $\{b\}$ and $\{a, b\}$. The final placements are $c \leftarrow \{a\}$, $c \leftarrow \{b\}$ and $c \leftarrow \{a, b\}$, then c and then the remaining nodes $\{c\}, \{a, c\}, \{b, c\}$ and $\{a, b, c\}$.

In the general case, suppose we have G with a consistent ordering of its nodes $i_1 <_{V_1} i_2 \cdots <_{V_n} i_n$. We construct a total ordering of the nodes of V consistent with $B(G)$ as follows. Start with the nodes $i_1 \leftarrow J$ (in any order), and then put i and after that $\{i\}$. Then for $k = 2, \ldots, n$ add nodes as follows: the $i_k \leftarrow J$ nodes, then i_k and then each J such that $i_k \in J$ and $J \subseteq \{i_1 \ldots i_k\}$. It is not difficult to see that this total order contains all nodes of V and is consistent with $B(G)$, so $B(G)$ is acyclic.

Now suppose B' were not an optimal solution to the BNSL instance $(V_1, \mathcal{P}, c)$. In that case there would be some strictly better solution corresponding to an acyclic graph G for which $B(G)$ would be a feasible solution to the ASP instance $D = (V, A)$ and this solution would be strictly better than B. This is a contradiction since B is an optimal solution and so it follows that B' is an optimal solution to $(V_1, \mathcal{P}, c)$. □

Note that, as Martí and Reinelt (2011) show, ASP is equivalent to the *linear ordering problem* (LOP). This means that pure LOP approaches can be used to solve ASP and thus BNSL.

8.2 Relating the BNSL and Acyclic Subgraph Problem Polytopes

There is a polytope naturally associated with any instance of ASP. Let $\mathbb{R}^A$ be a real vector space where every component of a vector $y \in \mathbb{R}^A$ is indexed by an edge $i \leftarrow j \in A$. For every edge set $B \subseteq A$, the incidence vector $y^B \in \mathbb{R}^A$ of B is defined by $y^B_{i \leftarrow j} = 1$ if $i \leftarrow j \in B$ and $y^B_{i \leftarrow j} = 0$ if $i \leftarrow j \notin B$. The *acyclic subgraph polytope* $P_{\mathrm{AC}}(D)$ is

$$P_{\mathrm{AC}}(D) := \operatorname{conv}\bigl\{y^B \in \mathbb{R}^A \bigm| B \in \mathcal{A}(D)\bigr\}. \tag{38}$$

It is not hard to see that the all-parent-sets-allowed family variable polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$ can be projected onto the ASP polytope where the ASP edgeset $A = V \times V$. Equivalently, BNSL is an *extended formulation* of such ASP instances. Since the ASP has been extensively studied it is important to investigate which results on ASP 'translate' to BNSL.

We can see that the ASP instance is a projection of the BNSL instance by introducing the edge indicator variables $y_{i \leftarrow j}$ into BNSL together with the 'linking' equations

$$y_{i \leftarrow j} = \sum_{J : j \in J} x_{i \leftarrow J}. \tag{39}$$

The introduction of these variables (dimensions) and equations leaves the family variable polytope unaltered except that it now 'lives in' a higher-dimensional space. 'Projecting away' the $x_{i \leftarrow J}$ variables from this higher-dimensional family variable polytope then produces the ASP polytope.

Using this relationship it is easy to map any ASP instance with edgeset $A = V \times V$ into a BNSL instance: simply set $c_{i \leftarrow J} = \sum_{j \in J} c(i \leftarrow j)$. A solution to the BNSL instance so produced will be a solution to the original ASP instance with the same objective value. A direct reverse mapping is only possible if there are edge weights such that the local score for each family is the sum of the weights of the edges corresponding to that family.

Proposition 39. *If $\pi y \leq \pi_0$ is a valid inequality for ASP, then $\pi' x \leq \pi_0$ is a valid inequality, where $\pi'_{i \leftarrow J} = \sum_{j \in J} \pi_{i \leftarrow j}$.*

Proof. Let $x^* \in \mathbb{R}^{\mathbf{F}}$ represent an acyclic digraph and let $y^* \in \mathbb{R}^A$ represent the same digraph. We have that $\pi y^* \leq \pi_0$. It is obvious that $\pi y^* = \pi' x^*$. So all acyclic digraphs represented by family variables satisfy $\pi' x \leq \pi_0$. The result follows. □

9. Conclusions

Integer programming, and specifically the IP-based GOBNILP system, offers a state-of-the-art practical approach to the NP-hard optimization problem of learning optimal Bayesian network structures, BNSL. Thus providing fundamental insights into the IP approach to BNSL is important both from the purely scientific perspective—dealing with a central class of probabilistic graphical models with various applications in AI—and for developing a

better understanding of the approach in the hope of further improving the current algorithmic approaches to BNSL. With these motivations, in this work we shed light on various fundamental computational and representational aspects of BNSL. From the practical perspective, many of our main contributions have tight connections to IP cutting planes derived in practice during search for optimal network structures. Specifically, our contributions include for example the following. We showed that the separation problem which in practice yields problem-specific BNSL cutting planes within GOBNILP is in fact NP-hard, a previously open problem. We studied the relationship between three key polytopes underlying BNSL. We analyzed the facets of the three polytopes, and established that the so-called cluster constraints giving rise to BNSL cutting planes are in fact facet-defining inequalities of the family-variable polytope central to BNSL. We also provide (in Appendix A) a complete enumeration of facets for low-dimensional family-variable polytopes, connecting with problem-specific cutting planes ruling out all network structures with short cyclic substructures. In summary, the theoretical results presented in this work deepen the current understanding of fundamental aspects of BNSL from various perspectives.

Acknowledgments

We thank three anonymous reviewers and the editor for useful criticism which has helped us improve the paper. 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.

Appendix A. Enumeration of Facets for Low-Dimensional Family Variable Polytopes

In Section 6 we provided general results on the facets of the family variable polytope. In this section, we provide a complete listing of all facet-defining inequalities (i.e. a minimal description of the convex hull by inequalities) of the family variable polytope $P_{\mathrm{F}}(V, \mathcal{P}_V)$ for $|V| = 2, 3, 4$. We will observe that all lower bounds on variables, modified convexity constraints and κ-cluster inequalities are indeed among the facets found, as predicted by our theoretical results. Proposition 24 and the lifting theorem (Theorem 28) are also consistent with the list of facets. In the case of $|V| = 4$, we also see that there are many facets *not* given in Section 6. In Section A.4 we enumerate all facet-defining inequalities for $|V| = 4$, where at most two parents are allowed and observe that the results are consistent with Theorem 31.

We use a, b, c and d to label the nodes. To simplify notation, we abbreviate variables such as $x_{b \leftarrow \{a,c\}}$ to $x_{b \leftarrow ac}$.

A.1 Node Set of Size 2

When $|V| = 2$, there are 3 acyclic digraphs and $\mathbf{F}(V, \mathcal{P}_V) = \{a \leftarrow \{b\}, b \leftarrow \{a\}\}$. There are three facets: the two lower bounds and the 1-cluster constraint $x_{a \leftarrow b} + x_{b \leftarrow a} \leq 1$.

A.2 Node Set of Size 3

When $|V| = 3$, there are 25 acyclic digraphs and

$$\begin{aligned}\mathbf{F}(V, \mathcal{P}_V) = \{&a \leftarrow \{b\}, a \leftarrow \{c\}, a \leftarrow \{b, c\},\\ &b \leftarrow \{a\}, b \leftarrow \{c\}, b \leftarrow \{a, c\},\\ &c \leftarrow \{a\}, c \leftarrow \{b\}, c \leftarrow \{a, b\}\}.\end{aligned}$$

Using the `cdd` computer program (Fukuda, 2016), we found all the facets of the convex hull of the 25 acyclic digraphs. There are 17 facet-defining inequalities:

- 9 lower bounds on the 9 $x_{i \leftarrow J}$ family variables;
- 3 modified convexity constraints, one for each of a, b and c;
- 4 1-cluster constraints, one for each of the clusters $\{a, b\}$, $\{a, c\}$, $\{b, c\}$ and $\{a, b, c\}$; and
- 1 2-cluster constraint for the cluster $\{a, b, c\}$.

A.3 Node Set of Size 4

When $|V| = 4$, there are 543 acyclic digraphs and

$$\begin{aligned}\mathbf{F}(V, \mathcal{P}_V) = \{&a \leftarrow \{b\}, a \leftarrow \{c\}, a \leftarrow \{d\}, a \leftarrow \{b, c\}, a \leftarrow \{b, d\}, a \leftarrow \{c, d\}, a \leftarrow \{b, c, d\},\\ &b \leftarrow \{a\}, b \leftarrow \{c\}, b \leftarrow \{d\}, b \leftarrow \{a, c\}, b \leftarrow \{a, d\}, b \leftarrow \{c, d\}, b \leftarrow \{a, c, d\},\\ &c \leftarrow \{a\}, c \leftarrow \{b\}, c \leftarrow \{d\}, c \leftarrow \{a, b\}, c \leftarrow \{a, d\}, c \leftarrow \{b, d\}, c \leftarrow \{a, b, d\},\\ &d \leftarrow \{a\}, d \leftarrow \{b\}, d \leftarrow \{c\}, d \leftarrow \{a, b\}, d \leftarrow \{a, c\}, d \leftarrow \{b, c\}, d \leftarrow \{a, b, c\}\}.\end{aligned}$$

Using `cdd` we discovered that there are 135 facet-defining inequalities of the family variable polytope:

- 28 lower bounds on the 28 $x_{i \leftarrow J}$ family variables;
- 4 modified convexity constraints, one for each of a, b, c and d;
- 6 1-cluster constraints for each of the $\binom{4}{2} = 6$ clusters of size 2;
- 4 1-cluster constraints for each of the $\binom{4}{3} = 4$ clusters of size 3;
- 1 1-cluster constraint for the $\binom{4}{4} = 1$ cluster of size 4;
- 4 2-cluster constraints for each of the $\binom{4}{3} = 4$ clusters of size 3;
- 1 2-cluster constraint for the $\binom{4}{4} = 1$ cluster of size 4;

- 1 3-cluster constraint for the $\binom{4}{4} = 1$ cluster of size 4; and
- 86 other facet-defining inequalities.

We now list these 86 other facet-defining inequalities. These 86 inequalities fall into 9 permutation classes, and we give just one member of each of these 9 classes. By symmetry, any permutation of the 4 nodes a, b, c and d in a facet-defining inequality will produce another facet-defining inequality. Some permutations do not change the inequality. We indicate this, for each permutation class, by showing which nodes can be permuted without changing the facet. For example, the expression $ab|cd$ indicates that either a and b, or c and d, can be swapped without altering the inequality, so that there are $4!/(2 \times 2) = 6$ distinct inequality in such a permutation class.

For each permutation class, we give the (arbitrarily chosen) name for that class that is used by the GOBNILP system. The names run from 4B to 4I—there is no permutation class called '4A', since, at one time in GOBNILP, this was used to designate κ-cluster inequalities. With the exception of '4F' and '4J' inequalities, if the user wants, GOBNILP can search for these facets as cutting planes for a given LP solution. By default only '4B' cutting planes are looked for, since these cutting planes have empirically been found to perform well. Interestingly, 4B facets can be defined in terms of connected matroids, as noted by Studený (2015).

4B facets $ab|cd$

$$\begin{aligned} & x_{a\leftarrow b} + x_{a\leftarrow bc} + x_{a\leftarrow bd} + x_{a\leftarrow cd} + x_{a\leftarrow bcd} \\ +& x_{b\leftarrow a} + x_{b\leftarrow ac} + x_{b\leftarrow ad} + x_{b\leftarrow cd} + x_{b\leftarrow acd} \\ +& x_{c\leftarrow ad} + x_{c\leftarrow bd} + x_{c\leftarrow abd} \\ +& x_{d\leftarrow ac} + x_{d\leftarrow bc} + x_{d\leftarrow abc} \qquad \leq 2 \end{aligned} \tag{40}$$

6 inequalities

4C facets $a|b|cd$

$$\begin{aligned} & x_{a\leftarrow c} + x_{a\leftarrow d} + x_{a\leftarrow bc} + x_{a\leftarrow bd} + x_{a\leftarrow cd} + x_{a\leftarrow bcd} \\ +& x_{b\leftarrow cd} + x_{b\leftarrow acd} \\ +& x_{c\leftarrow ab} + x_{c\leftarrow bd} + x_{c\leftarrow abd} \\ +& x_{d\leftarrow ab} + x_{d\leftarrow bc} + x_{d\leftarrow abc} \qquad \leq 2 \end{aligned}$$

12 inequalities

4D facets $a|b|cd$

$$\begin{aligned} & x_{a\leftarrow b} + x_{a\leftarrow c} + x_{a\leftarrow d} + x_{a\leftarrow bc} + x_{a\leftarrow bd} + 2x_{a\leftarrow cd} + 2x_{a\leftarrow bcd} \\ +& x_{b\leftarrow a} + x_{b\leftarrow c} + x_{b\leftarrow d} + x_{b\leftarrow ac} + x_{b\leftarrow ad} + x_{b\leftarrow cd} + x_{b\leftarrow acd} \\ +& x_{c\leftarrow a} + x_{c\leftarrow ab} + x_{c\leftarrow ad} + x_{c\leftarrow abd} \\ +& x_{d\leftarrow a} + x_{d\leftarrow ab} + x_{d\leftarrow ac} + x_{d\leftarrow abc} \qquad \leq 3 \end{aligned}$$

12 inequalities

4E facets $a|bcd$

$$\begin{aligned} & x_{a\leftarrow bc} + x_{a\leftarrow bd} + x_{a\leftarrow cd} + 2x_{a\leftarrow bcd} \\ +& x_{b\leftarrow ac} + x_{b\leftarrow ad} + x_{b\leftarrow acd} \\ +& x_{c\leftarrow ab} + x_{c\leftarrow ad} + x_{c\leftarrow abd} \\ +& x_{d\leftarrow ab} + x_{d\leftarrow ac} + x_{d\leftarrow abc} \qquad \leq 2 \end{aligned}$$

4 inequalities

4F facets $ab|cd$

$$\begin{aligned} & x_{a\leftarrow cd} + x_{a\leftarrow bcd} \\ +& x_{b\leftarrow cd} + x_{b\leftarrow acd} \\ +& x_{c\leftarrow a} + x_{c\leftarrow b} + x_{c\leftarrow d} + x_{c\leftarrow ab} + x_{c\leftarrow ad} + x_{c\leftarrow bd} + 2x_{c\leftarrow abd} \\ +& x_{d\leftarrow a} + x_{d\leftarrow b} + x_{d\leftarrow c} + x_{d\leftarrow ab} + x_{d\leftarrow ac} + x_{d\leftarrow bc} + 2x_{d\leftarrow abc} \leq 3 \end{aligned}$$

6 inequalities

4G facets $a|b|c|d$

$$\begin{aligned} & x_{a\leftarrow cd} + x_{a\leftarrow bcd} \\ +& x_{b\leftarrow c} + x_{b\leftarrow ac} + x_{b\leftarrow cd} + x_{b\leftarrow acd} \\ +& x_{c\leftarrow b} + x_{c\leftarrow d} + x_{c\leftarrow ab} + x_{c\leftarrow ad} + x_{c\leftarrow bd} + 2x_{c\leftarrow abd} \\ +& x_{d\leftarrow a} + x_{d\leftarrow b} + x_{d\leftarrow c} + x_{d\leftarrow ab} + 2x_{d\leftarrow ac} + x_{d\leftarrow bc} + 2x_{d\leftarrow abc} \leq 3 \end{aligned}$$

24 inequalities

4H facets $a|b|cd$

$$\begin{aligned} & x_{a\leftarrow c} + x_{a\leftarrow d} + x_{a\leftarrow bc} + x_{a\leftarrow bd} + x_{a\leftarrow cd} + 2x_{a\leftarrow bcd} \\ +& x_{b\leftarrow acd} \\ +& x_{c\leftarrow ab} + x_{c\leftarrow abd} \\ +& x_{d\leftarrow ab} + x_{d\leftarrow abc} \qquad \leq 2 \end{aligned}$$

12 inequalities

4I facets $ab|cd$

$$\begin{aligned} & x_{a\leftarrow c} + x_{a\leftarrow d} + x_{a\leftarrow bc} + x_{a\leftarrow bd} + x_{a\leftarrow cd} + 2x_{a\leftarrow bcd} \\ +& x_{b\leftarrow c} + x_{b\leftarrow d} + x_{b\leftarrow ac} + x_{b\leftarrow ad} + x_{b\leftarrow cd} + 2x_{b\leftarrow acd} \\ +& x_{c\leftarrow a} + x_{c\leftarrow b} + x_{c\leftarrow d} + 2x_{c\leftarrow ab} + x_{c\leftarrow ad} + x_{c\leftarrow bd} + 2x_{c\leftarrow abd} \\ +& x_{d\leftarrow a} + x_{d\leftarrow b} + x_{d\leftarrow c} + 2x_{d\leftarrow ab} + x_{d\leftarrow ac} + x_{d\leftarrow bc} + 2x_{d\leftarrow abc} \qquad \leq 4 \end{aligned}$$

6 inequalities

4J facets $a|bcd$

$$\begin{aligned} & x_{a\leftarrow b} + x_{a\leftarrow c} + x_{a\leftarrow d} + 2x_{a\leftarrow bc} + 2x_{a\leftarrow bd} + 2x_{a\leftarrow cd} + 2x_{a\leftarrow bcd} \\ +& x_{b\leftarrow a} + x_{b\leftarrow ac} + x_{b\leftarrow ad} + x_{b\leftarrow cd} + x_{b\leftarrow acd} \\ +& x_{c\leftarrow a} + x_{c\leftarrow ab} + x_{c\leftarrow ad} + x_{c\leftarrow bd} + x_{c\leftarrow abd} \\ +& x_{d\leftarrow a} + x_{d\leftarrow ab} + x_{d\leftarrow ac} + x_{d\leftarrow bc} + x_{d\leftarrow abc} \qquad \leq 3 \end{aligned}$$

4 inequalities

A.4 Node Set of Size 4, Parent Set Size At Most 2

By Theorem 31, if we have 4 nodes but only allow acyclic digraphs with at most two parents, then the following facet-defining inequalities from Section A.3 (with family variables $x_{a\leftarrow bcd}$, $x_{b\leftarrow acd}$, $x_{c\leftarrow abd}$ and $x_{d\leftarrow abc}$ removed) should be facet-defining inequalities of the resulting polytope.

- 24 lower bounds on the 24 $x_{i\leftarrow J}$ family variables;
- 4 modified convexity constraints, one for each of a, b, c and d;
- 6 1-cluster constraints for each of the $\binom{4}{2} = 6$ clusters of size 2;
- 4 1-cluster constraints for each of the $\binom{4}{3} = 4$ clusters of size 3;
- 1 1-cluster constraint for the $\binom{4}{4} = 1$ cluster of size 4;
- 4 2-cluster constraints for each of the $\binom{4}{3} = 4$ clusters of size 3; and
- 1 2-cluster constraint for the $\binom{4}{4} = 1$ cluster of size 4.

In addition all facet-defining inequalities of types 4B, 4C, 4D and 4J should remain facet-defining. There are 6, 12, 12 and 4 of these, respectively. This adds up to a total of 24+4+6+4+1+4+1+6+12+12+4=78 facet-defining inequalities. Using `cdd` we computed the facet-defining inequalities of the convex hull of the (family-variable encoded) 443 acyclic digraphs with 4 nodes and where each node has at most 2 parents. We found, as expected, that all of these 78 inequalities were included. Moreover, we found that these 78 constitute the *complete set* of facet-defining inequalities—there are no others.

Appendix B. Lift-and-Project for Family Variable Polytopes

In this appendix, we apply a 'lift-and-project' method based on the sink-based extended representation of Section 7.3 to derive a representation of $P_{\mathrm{F}}(\{a,b,c,d\}, \mathcal{P}_{\{a,b,c,d\}})$, whose facet-defining inequalities are given in Section A.3, in terms of the polytopes $P_{\mathrm{F}}(\{b,c,d\}, \mathcal{P}_{\{b,c,d\}})$, $P_{\mathrm{F}}(\{a,c,d\}, \mathcal{P}_{\{a,c,d\}})$, $P_{\mathrm{F}}(\{a,b,d\}, \mathcal{P}_{\{a,b,d\}})$ and $P_{\mathrm{F}}(\{a,b,c\}, \mathcal{P}_{\{a,b,c\}})$, whose facet-defining inequalities are given in Section A.2. First we have the relevant formulation of (33),

$$x_a + x_b + x_c + x_d = 1, \tag{41}$$

stating that exactly one of the four nodes is the distinguished sink in any acyclic digraph using those four nodes. Recall that $x_{j,i\leftarrow J}$ indicates that j is the distinguished sink and that J is the parent set for i so that $x_{j,i\leftarrow J} = 0$ if $j \in J$, so that, for example, $x_{b,a\leftarrow b} = 0$. With this observation we can write the linking equations (34) as follows.

$$x_{a\leftarrow b} = x_{a,a\leftarrow b} + x_{c,a\leftarrow b} + x_{d,a\leftarrow b} \tag{42}$$
$$x_{a\leftarrow c} = x_{a,a\leftarrow c} + x_{b,a\leftarrow c} + x_{d,a\leftarrow c} \tag{43}$$
$$x_{a\leftarrow d} = x_{a,a\leftarrow d} + x_{b,a\leftarrow d} + x_{c,a\leftarrow d} \tag{44}$$
$$x_{b\leftarrow a} = x_{b,b\leftarrow a} + x_{c,b\leftarrow a} + x_{d,b\leftarrow a} \tag{45}$$
$$x_{b\leftarrow c} = x_{a,b\leftarrow c} + x_{b,b\leftarrow c} + x_{d,b\leftarrow c} \tag{46}$$
$$x_{b\leftarrow d} = x_{a,b\leftarrow d} + x_{b,b\leftarrow d} + x_{c,b\leftarrow d} \tag{47}$$
$$x_{c\leftarrow a} = x_{b,c\leftarrow a} + x_{c,c\leftarrow a} + x_{d,c\leftarrow a} \tag{48}$$
$$x_{c\leftarrow b} = x_{a,c\leftarrow b} + x_{c,c\leftarrow b} + x_{d,c\leftarrow b} \tag{49}$$
$$x_{c\leftarrow d} = x_{a,c\leftarrow d} + x_{b,c\leftarrow d} + x_{c,c\leftarrow d} \tag{50}$$
$$x_{d\leftarrow a} = x_{b,d\leftarrow a} + x_{c,d\leftarrow a} + x_{d,d\leftarrow a} \tag{51}$$
$$x_{d\leftarrow b} = x_{a,d\leftarrow b} + x_{c,d\leftarrow b} + x_{d,d\leftarrow b} \tag{52}$$
$$x_{d\leftarrow c} = x_{a,d\leftarrow c} + x_{b,d\leftarrow c} + x_{d,d\leftarrow d} \tag{53}$$

$$x_{a\leftarrow bc} = x_{a,a\leftarrow bc} + x_{d,a\leftarrow bc} \tag{54}$$
$$x_{a\leftarrow bd} = x_{a,a\leftarrow bd} + x_{c,a\leftarrow bd} \tag{55}$$
$$x_{a\leftarrow cd} = x_{a,a\leftarrow cd} + x_{b,a\leftarrow cd} \tag{56}$$
$$x_{b\leftarrow ac} = x_{b,b\leftarrow ac} + x_{d,b\leftarrow ac} \tag{57}$$
$$x_{b\leftarrow ad} = x_{b,b\leftarrow ad} + x_{c,b\leftarrow ad} \tag{58}$$
$$x_{b\leftarrow cd} = x_{b,b\leftarrow cd} + x_{a,b\leftarrow cd} \tag{59}$$
$$x_{c\leftarrow ab} = x_{c,c\leftarrow ab} + x_{d,c\leftarrow ab} \tag{60}$$
$$x_{c\leftarrow ad} = x_{b,c\leftarrow ad} + x_{c,c\leftarrow ad} \tag{61}$$
$$x_{c\leftarrow bd} = x_{a,c\leftarrow bd} + x_{c,c\leftarrow bd} \tag{62}$$
$$x_{d\leftarrow ab} = x_{c,d\leftarrow ab} + x_{d,d\leftarrow ab} \tag{63}$$
$$x_{d\leftarrow ac} = x_{b,d\leftarrow ac} + x_{d,d\leftarrow ac} \tag{64}$$
$$x_{d\leftarrow bc} = x_{a,d\leftarrow bc} + x_{d,d\leftarrow bc} \tag{65}$$

$$x_{a\leftarrow bcd} = x_{a,a\leftarrow bcd} \tag{66}$$
$$x_{b\leftarrow acd} = x_{b,b\leftarrow acd} \tag{67}$$
$$x_{c\leftarrow abd} = x_{c,c\leftarrow abd} \tag{68}$$
$$x_{d\leftarrow abc} = x_{d,d\leftarrow abc} \tag{69}$$

Thirdly we have all equations of type (35). We label all inequalities with x_a on the RHS as follows. The modified convexity constraints for a, b, c and d are labelled a-a, a-b, a-c and a-d, respectively. All other constraints are cluster constraints which we label as a-C, where

C is the cluster and $\kappa = 1$ and a-2-C, where C is the cluster and $\kappa = 2$. Inequalities with x_b, x_c and x_d on the RHS are labelled analogously. The 36 inequalities of type (35) are now listed using this labelling convention.

$$x_{a,b\leftarrow c} + x_{a,b\leftarrow d} + x_{a,b\leftarrow cd} \leq x_a \quad \text{(a-b)}$$
$$x_{a,c\leftarrow b} + x_{a,c\leftarrow d} + x_{a,c\leftarrow bd} \leq x_a \quad \text{(a-c)}$$
$$x_{a,d\leftarrow b} + x_{a,d\leftarrow c} + x_{a,d\leftarrow cd} \leq x_a \quad \text{(a-d)}$$
$$x_{a,b\leftarrow c} + x_{a,b\leftarrow cd} + x_{a,c\leftarrow b} + x_{a,c\leftarrow bd} \leq x_a \quad \text{(a-bc)}$$
$$x_{a,b\leftarrow d} + x_{a,b\leftarrow cd} + x_{a,d\leftarrow b} + x_{a,d\leftarrow bd} \leq x_a \quad \text{(a-bd)}$$
$$x_{a,c\leftarrow d} + x_{a,c\leftarrow bd} + x_{a,d\leftarrow c} + x_{a,d\leftarrow cd} \leq x_a \quad \text{(a-cd)}$$
$$x_{a,b\leftarrow c} + x_{a,b\leftarrow d} + x_{a,b\leftarrow cd} + x_{a,c\leftarrow b} + x_{a,c\leftarrow d} + x_{a,c\leftarrow bd} + x_{a,d\leftarrow b} + x_{a,d\leftarrow c} + x_{a,d\leftarrow cb} \leq 2x_a \quad \text{(a-bcd)}$$
$$x_{a,b\leftarrow cd} + x_{a,c\leftarrow bd} + x_{a,d\leftarrow bc} \leq x_a \quad \text{(a-2-bcd)}$$
$$x_{a,a\leftarrow b} + x_{a,a\leftarrow c} + x_{a,a\leftarrow d} + x_{a,a\leftarrow bc} + x_{a,a\leftarrow bd} + x_{a,a\leftarrow cd} + x_{a,a\leftarrow bcd} \leq x_a \quad \text{(a-a)}$$

$$x_{b,a\leftarrow c} + x_{b,a\leftarrow c} + x_{b,a\leftarrow cd} \leq x_b \quad \text{(b-a)}$$
$$x_{b,c\leftarrow a} + x_{b,c\leftarrow d} + x_{b,c\leftarrow ad} \leq x_b \quad \text{(b-c)}$$
$$x_{b,d\leftarrow a} + x_{b,d\leftarrow c} + x_{b,d\leftarrow ac} \leq x_b \quad \text{(b-d)}$$
$$x_{b,a\leftarrow c} + x_{b,a\leftarrow cd} + x_{b,c\leftarrow a} + x_{b,c\leftarrow ad} \leq x_b \quad \text{(b-ac)}$$
$$x_{b,a\leftarrow d} + x_{b,a\leftarrow cd} + x_{b,d\leftarrow a} + x_{b,d\leftarrow ac} \leq x_b \quad \text{(b-ad)}$$
$$x_{b,c\leftarrow d} + x_{b,c\leftarrow bd} + x_{b,d\leftarrow c} + x_{b,d\leftarrow cd} \leq x_b \quad \text{(b-cd)}$$
$$x_{b,a\leftarrow c} + x_{b,a\leftarrow d} + x_{b,a\leftarrow cd} + x_{b,c\leftarrow a} + x_{b,c\leftarrow d} + x_{b,c\leftarrow ad} + x_{b,d\leftarrow a} + x_{b,d\leftarrow c} + x_{b,d\leftarrow ac} \leq 2x_b \quad \text{(b-acd)}$$
$$x_{b,a\leftarrow cd} + x_{b,c\leftarrow ad} + x_{b,d\leftarrow ac} \leq x_b \quad \text{(b-2-acd)}$$
$$x_{b,b\leftarrow a} + x_{b,b\leftarrow c} + x_{b,b\leftarrow d} + x_{b,b\leftarrow ac} + x_{b,b\leftarrow ad} + x_{b,b\leftarrow cd} + x_{b,b\leftarrow acd} \leq x_b \quad \text{(b-b)}$$

$$x_{c,a\leftarrow b} + x_{c,a\leftarrow d} + x_{c,a\leftarrow bd} \leq x_c \quad \text{(c-a)}$$
$$x_{c,b\leftarrow a} + x_{c,b\leftarrow d} + x_{c,b\leftarrow ad} \leq x_c \quad \text{(c-b)}$$
$$x_{c,d\leftarrow a} + x_{c,d\leftarrow b} + x_{c,d\leftarrow ab} \leq x_c \quad \text{(c-d)}$$
$$x_{c,a\leftarrow b} + x_{c,a\leftarrow bd} + x_{c,b\leftarrow a} + x_{c,b\leftarrow ad} \leq x_c \quad \text{(c-ab)}$$
$$x_{c,a\leftarrow d} + x_{c,a\leftarrow bd} + x_{c,d\leftarrow a} + x_{c,d\leftarrow ab} \leq x_c \quad \text{(c-ad)}$$
$$x_{c,b\leftarrow d} + x_{c,b\leftarrow ad} + x_{c,d\leftarrow b} + x_{c,d\leftarrow ab} \leq x_c \quad \text{(c-bd)}$$
$$x_{c,a\leftarrow b} + x_{c,a\leftarrow d} + x_{c,a\leftarrow bd} + x_{c,b\leftarrow a} + x_{c,b\leftarrow d} + x_{c,b\leftarrow ad} + x_{c,d\leftarrow a} + x_{c,d\leftarrow b} + x_{c,d\leftarrow ab} \leq 2x_c \quad \text{(c-abd)}$$
$$x_{c,a\leftarrow bd} + x_{c,b\leftarrow ad} + x_{c,d\leftarrow ab} \leq x_c \quad \text{(c-2-abd)}$$
$$x_{c,c\leftarrow a} + x_{c,c\leftarrow b} + x_{c,c\leftarrow d} + x_{c,c\leftarrow ab} + x_{c,c\leftarrow ad} + x_{c,c\leftarrow bd} + x_{c,c\leftarrow abd} \leq x_c \quad \text{(c-c)}$$

$$x_{d,a\leftarrow b} + x_{d,a\leftarrow c} + x_{d,a\leftarrow bc} \leq x_d \tag{d-a}$$
$$x_{d,b\leftarrow a} + x_{d,b\leftarrow c} + x_{d,b\leftarrow ac} \leq x_d \tag{d-b}$$
$$x_{d,c\leftarrow a} + x_{d,c\leftarrow a} + x_{d,c\leftarrow ab} \leq x_d \tag{d-c}$$
$$x_{d,a\leftarrow b} + x_{d,a\leftarrow bc} + x_{d,b\leftarrow a} + x_{d,b\leftarrow ac} \leq x_d \tag{d-ab}$$
$$x_{d,a\leftarrow c} + x_{d,a\leftarrow bc} + x_{d,c\leftarrow a} + x_{d,c\leftarrow ab} \leq x_d \tag{d-ac}$$
$$x_{d,b\leftarrow c} + x_{d,b\leftarrow ac} + x_{d,c\leftarrow b} + x_{d,c\leftarrow ab} \leq x_d \tag{d-cd}$$
$$\begin{aligned} x_{d,a\leftarrow b} + x_{d,a\leftarrow c} + x_{d,a\leftarrow bc} + x_{d,b\leftarrow a} + x_{d,b\leftarrow c} + x_{d,b\leftarrow ac} \\ + x_{d,c\leftarrow a} + x_{d,c\leftarrow b} + x_{d,c\leftarrow ab} \leq 2x_d \end{aligned} \tag{d-abc}$$
$$x_{d,a\leftarrow bc} + x_{d,b\leftarrow ac} + x_{d,c\leftarrow ab} \leq x_d \tag{d-2-abd}$$
$$\begin{aligned} x_{d,d\leftarrow a} + x_{d,d\leftarrow b} + x_{d,d\leftarrow c} + x_{d,d\leftarrow ab} \\ + x_{d,d\leftarrow ac} + x_{d,d\leftarrow bc} + x_{d,d\leftarrow abc} \leq x_d \end{aligned} \tag{d-d}$$

Using (66–69) it is possible to eliminate the variables $x_{a,a\leftarrow bcd}$, $x_{b,b\leftarrow acd}$, $x_{c,c\leftarrow abd}$ and $x_{d,d\leftarrow abc}$, and (66–69) from the representation. This leaves us with a representation of $P_{\mathrm{F}}(\{a,b,c,d\}, \mathcal{P}_{\{a,b,c,d\}})$ using $4+28+4\times(9+6) = 92$ variables, 25 equations, 36 inequalities of type (35), four lower bounds on the variables x_j, 56 lower bounds (of 0) on the variables $x_{i,j\leftarrow J}$ where $|J| < 3$ and four lower bounds (of 0) on the variables $x_{i\leftarrow J}$ where $|J| = 3$. In total we have 100 inequalities.

We have given an explicit extended representation of $P_{\mathrm{F}}(\{a,b,c,d\}, \mathcal{P}_{\{a,b,c,d\}})$. Here is that representation described more briefly.

- $x_a + x_b + x_c + x_d = 1$.
- $24/2 = 12$ unique permutations of $x_{a\leftarrow b} = x_{a,a\leftarrow b} + x_{c,a\leftarrow b} + x_{d,a\leftarrow b}$.
- $24/2 = 12$ unique permutations of $x_{a\leftarrow bc} = x_{a,a\leftarrow bc} + x_{d,a\leftarrow bc}$.
- $24/2 = 12$ unique permutations of $x_{a,b\leftarrow c} + x_{a,b\leftarrow d} + x_{a,b\leftarrow cd} \leq x_a$.
- $24/2 = 12$ unique permutations of $x_{a,b\leftarrow c} + x_{a,b\leftarrow cd} + x_{a,c\leftarrow b} + x_{a,c\leftarrow bd} \leq x_a$.
- $24/6 = 4$ unique permutations of $x_{a,b\leftarrow c} + x_{a,b\leftarrow d} + x_{a,b\leftarrow cd} + x_{a,c\leftarrow b} + x_{a,c\leftarrow d} + x_{a,c\leftarrow bd} + x_{a,d\leftarrow b} + x_{a,d\leftarrow c} + x_{a,d\leftarrow cb} \leq 2x_a$.
- $24/6 = 4$ unique permutations of $x_{a,b\leftarrow cd} + x_{a,c\leftarrow bd} + x_{a,d\leftarrow bc} \leq x_a$.
- $24/6 = 4$ unique permutations of $x_{a,a\leftarrow b} + x_{a,a\leftarrow c} + x_{a,a\leftarrow d} + x_{a,a\leftarrow bc} + x_{a,a\leftarrow bd} + x_{a,a\leftarrow cd} + x_{a\leftarrow bcd} \leq x_a$.
- 4 lower bounds on the variables x_j.
- 56 lower bounds on variables $x_{i,j\leftarrow J}$ where $|J| < 3$.
- 4 lower bounds on variables $x_{i\leftarrow J}$ where $|J| = 3$.

The crucial point is that the convex hull of solutions to our extended representation can be found by simply dropping the integrality restrictions on variables. Conforti et al. (2014,

p. 71) provide the relevant proof. If we 'project away' the additional variables from this convex hull we end up with $P_{\mathrm{F}}(V, \mathcal{P}_V) = \mathrm{conv}\left(\bigcup_{j \in V} P_{\mathrm{F}}(V, j)\right)$.

We now show explicitly that the facet-defining inequalities of $P_{\mathrm{F}}(\{a,b,c,d\}, \mathcal{P}_{\{a,b,c,d\}})$ can be derived by projection from our extended representation. This projection is done by forming linear combinations of extended representation facet-defining inequalities which only contain 'normal' family variables $x_{i \leftarrow J}$.

For example, consider adding the following inequalities: (a-a), (a-2-bcd), (b-b), (b-2-acd), (c-c), (c-ab), (d-d) and (d-ab). Note that the RHS of this inequality is $x_a + x_a + x_b + x_b + x_c + x_c + x_d + x_d = 2$. So the result is

$$\begin{aligned}
&x_{a,a\leftarrow b} + x_{a,a\leftarrow c} + x_{a,a\leftarrow d} + x_{a,a\leftarrow bc} + x_{a,a\leftarrow bd} + x_{a,a\leftarrow cd} + x_{a\leftarrow bcd} \\
&+x_{a,b\leftarrow cd} + x_{a,c\leftarrow bd} + x_{a,d\leftarrow bc} \\
&+x_{b,b\leftarrow a} + x_{b,b\leftarrow c} + x_{b,b\leftarrow d} + x_{b,b\leftarrow ac} + x_{b,b\leftarrow ad} + x_{b,b\leftarrow cd} + x_{b\leftarrow acd} \\
&+x_{b,a\leftarrow cd} + x_{b,c\leftarrow ad} + x_{b,d\leftarrow ac} \\
&+x_{c,c\leftarrow a} + x_{c,c\leftarrow b} + x_{c,c\leftarrow d} + x_{c,c\leftarrow ab} + x_{c,c\leftarrow ad} + x_{c,c\leftarrow bd} + x_{c\leftarrow abd} \\
&+x_{c,a\leftarrow b} + x_{c,a\leftarrow bd} + x_{c,b\leftarrow a} + x_{c,b\leftarrow ad} \\
&+x_{d,d\leftarrow a} + x_{d,d\leftarrow b} + x_{d,d\leftarrow c} + x_{d,d\leftarrow ab} + x_{d,d\leftarrow ac} + x_{d,d\leftarrow bc} + x_{d\leftarrow abc} \\
&+x_{d,a\leftarrow b} + x_{d,a\leftarrow bc} + x_{d,b\leftarrow a} + x_{d,b\leftarrow ac} \leq 2.
\end{aligned}$$

Using (42-65) we can simplify this to

$$\begin{aligned}
&x_{a\leftarrow b} + x_{a,a\leftarrow c} + x_{a,a\leftarrow d} + x_{a\leftarrow bc} + x_{a\leftarrow bd} + x_{a\leftarrow cd} + x_{a\leftarrow bcd} \\
&+x_{a,b\leftarrow cd} \\
&+x_{b\leftarrow a} + x_{b,b\leftarrow c} + x_{b,b\leftarrow d} + x_{b\leftarrow ac} + x_{b\leftarrow ad} + x_{b,b\leftarrow cd} + x_{b\leftarrow acd} \\
&+ \\
&+x_{c,c\leftarrow a} + x_{c,c\leftarrow b} + x_{c,c\leftarrow d} + x_{c,c\leftarrow ab} + x_{c\leftarrow ad} + x_{c\leftarrow bd} + x_{c\leftarrow abd} \\
&+ \\
&+x_{d,d\leftarrow a} + x_{d,d\leftarrow b} + x_{d,d\leftarrow c} + x_{d,d\leftarrow ab} + x_{d\leftarrow ac} + x_{d\leftarrow bc} + x_{d\leftarrow abc} \\
&+ \leq 2.
\end{aligned} \tag{70}$$

This inequality can then be weakened by adding the lower bounds for the 14 remaining extended variables which results in the 4B facet (40) of $P_{\mathrm{F}}(\{a,b,c,d\}, \mathcal{P}_{\{a,b,c,d\}})$.

We now show how each of the facet classes 4B-4J for $P_{\mathrm{F}}(\{a,b,c,d\}, \mathcal{P}_{\{a,b,c,d\}})$ listed in Section A.3 can be derived by projection from the extended representation. Projection is achieved by multiplying each facet-defining inequality in the extended representation by a non-negative scalar. Let the vector of these scalars be denoted $u \geq 0$. In the following list we only provide positive components of u and do not bother to list those components of u corresponding to variable lower bounds. Note that since these u vectors generate *facet-defining* inequalities of $P_{\mathrm{F}}(\{a,b,c,d\}, \mathcal{P}_{\{a,b,c,d\}})$, they must be *extreme* rays of the relevant projection cone (Balas, 2005).

4B facet

$u_{a-a} = 1, u_{a-2-bcd} = 1, u_{b-b} = 1, u_{b-2-acd} = 1, u_{c-c} = 1, u_{c-ab} = 1, u_{d-d} = 1, u_{d-ab} = 1$

4C facet

$u_{a-a} = 1, u_{a-2-bcd} = 1, u_{b-b} = 1, u_{b-a} = 1, u_{c-c} = 1, u_{c-ad} = 1, u_{d-d} = 1, u_{d-ac} = 1$

4D facet

$u_{a-a} = 2, u_{a-b} = 1, u_{b-b} = 1, u_{b-ac} = 1, u_{b-ad} = 1, u_{c-c} = 1, u_{c-abd} = 1, u_{d-d} = 1, u_{d-abc} = 1$

4E facet

$u_{a-a} = 2, u_{b-b} = 1, u_{b-2-acd} = 1, u_{c-c} = 1, u_{c-2-abd} = 1, u_{d-d} = 1, u_{d-2-abc} = 1$

4F facet

$u_{a-a} = 1, u_{a-bcd} = 1, u_{b-b} = 1, u_{b-acd} = 1, u_{c-c} = 2, u_{c-d} = 1, u_{d-d} = 2, u_{d-c} = 1$

4G facet

$u_{a-a} = 1, u_{a-bcd} = 1, u_{b-b} = 1, u_{b-ad} = 1, u_{b-cd} = 1, u_{c-c} = 2, u_{c-d} = 1, u_{d-d} = 2, u_{d-bc} = 1$

4H facet

$u_{a-a} = 2, u_{b-b} = 1, u_{b-a} = 1, u_{c-c} = 1, u_{c-ad} = 1, u_{d-d} = 1, u_{d-ac} = 1$

4I facet

$u_{a-a} = 2, u_{a-bcd} = 1, u_{b-b} = 2, u_{b-acd} = 1, u_{c-c} = 2, u_{c-ad} = 1, u_{c-bd} = 1, u_{d-d} = 2, u_{d-ac} = 1, u_{d-bc} = 1$

4J facet

$u_{a-a} = 2, u_{a-2-bcd} = 1, u_{b-b} = 1, u_{b-ac} = 1, u_{b-ad} = 1, u_{c-c} = 1, u_{c-ab} = 1, u_{c-ad} = 1, u_{d-d} = 1, u_{d-ab} = 1, u_{d-ac} = 1$

We have shown how to generate all facets of $P_{\mathrm{F}}(V, \mathcal{P}_V)$ for $|V| = 4$ from the $|V| = 3$ case. This was done by constructing the desired convex hull using an extended representation and then projecting away the extraneous variables. Although in this case we already had the convex hull for $|V| = 4$ (by direct computation using `cdd`) it is clear that the same technique could be used to construct the convex hull for $|V| = 5$ and above. The difficulty with this approach is identifying which projections $u \geq 0$ generate facets. It was noted above that we can restrict attention to u which are extreme rays of the relevant projection cone. However, in general, not all extreme rays generate facets, it also necessary that the number of dimensions 'lost' when projecting the entire polytope matches the number lost when projecting the face whose projection is the putative facet (Balas, 2005). We do not investigate this here, leaving this issue for future work.

References

Achterberg, T. (2007). *Constraint Integer Programming*. Ph.D. thesis, TU Berlin.

Balas, E. (2005). Projection, lifting and extended formulation in integer and combinatorial optimization. *Annals of Operation Research*, *140*, 125–161.

Bartlett, M., & Cussens, J. (2015). Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*. Available online, in Press.

Boyd, S., & Pulleyblank, W. R. (2009). Facet generating techniques. In Cook, W., Lovász, L., & Vygen, J. (Eds.), *Research Trends in Combinatorial Optimization, Bonn Workshop on Combinatorial Optimization 2008*, pp. 33–55. Springer.

Chickering, D. M. (1996). Learning Bayesian networks is NP-Complete. In Fisher, D., & Lenz, H.-J. (Eds.), *Learning from Data: AI & Statistics V*, chap. 12, pp. 121–130. Springer.

Cohn, P. M. (1982). *Algebra*, Vol. 1. Wiley.

Colombo, D., Maathuis, M. H., Kalisch, M., & Richardson, T. S. (2012). Learning high-dimensional directed acyclic graphs with latent and selection variables. *Annals of Statistics*, *40*, 294–321.

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Integer Programming*. Springer.

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

Cussens, J. (2011). Bayesian network learning with cutting planes. In Cozman, F. G., & Pfeffer, A. (Eds.), *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pp. 153–160. AUAI Press.

Cussens, J., Bartlett, M., Jones, E. M., & Sheehan, N. A. (2013). Maximum likelihood pedigree reconstruction using integer linear programming. *Genetic Epidemiology*, *37*(1), 69–83.

Cussens, J., Haws, D., & Studený, M. (2016). Polyhedral aspects of score equivalence in Bayesian network structure learning. *Mathematical Programming*. doi:10.1007/s10107-016-1087-2.

de Campos, Cassio, P., & Ji, Q. (2011). Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research*, *12*, 663–689.

Fukuda, K. (2016). cdd & ccdplus homepage. www.inf.ethz.ch/personal/fukudak/cdd_home. Online; accessed 31 December 2016.

Grötschel, M., Jünger, M., & Reinelt, G. (1985). On the acyclic subgraph polytope. *Mathematical Programming*, *33*(1), 28–42.

Hammer, P. L., Johnson, E., & Peled, U. N. (1975). Facets of regular 0-1 polytopes. *Mathematical Programming*, *8*, 179–206.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning discrete Bayesian networks. *Machine Learning*, *20*, 197–243.

Hugin Expert A/S (2016). Hugin case stories. www.hugin.com/index.php/resources/#cases. Online; accessed 31 December 2016.

Jaakkola, T., Sontag, D., Globerson, A., & Meila, M. (2010). Learning Bayesian network structure using LP relaxations. In Teh, Y. W., & Titterington, D. M. (Eds.), *Proceedings of 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, Vol. 9 of *Journal of Machine Learning Research Workshop and Conference Proceedings*, pp. 358–365. JMLR.org.

Koivisto, M., & Sood, K. (2004). Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, *5*, 549–573.

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Malone, B., Kangas, K., Järvisalo, M., Koivisto, M., & Myllymäki, P. (2014). Predicting the hardness of learning Bayesian networks. In Brodley, C. E., & Stone, P. (Eds.), *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014)*, pp. 2460–2466. AAAI Press.

Martí, R., & Reinelt, G. (2011). *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*. Springer.

Peharz, R., & Pernkopf, F. (2012). Exact maximum margin structure learning of Bayesian networks. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*. icml.cc / Omnipress.

Sheehan, N., Bartlett, M., & Cussens, J. (2014). Improved maximum likelihood reconstruction of complex multi-generational pedigrees. *Theoretical Population Biology*, *97*, 11–19.

Silander, T., & Myllymäki, P. (2006). A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI 2006)*, pp. 445–452. AUAI Press.

Spirtes, P., Glymour, C., & Scheines, R. (1993). *Causation, Prediction and Search*. Springer.

Studený, M. (2015). How matroids occur in the context of learning Bayesian network structure. In Meila, M., & Heskes, T. (Eds.), *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI 2015)*, pp. 832–841. AUAI Press.

Tsamardinos, I., Brown, L. E., & Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, *65*(1), 31–78.

van Beek, P., & Hoffmann, H. (2015). Machine learning of Bayesian networks using constraint programming. In Pesant, G. (Ed.), *Proceedings of the 21st International Conference on Principles and Practice of Constraint Programming (CP 2015)*, Vol. 9255 of *Lecture Notes in Computer Science*, pp. 429–445. Springer.

Wolsey, L. A. (1998). *Integer Programming*. John Wiley.

Yuan, C., & Malone, B. (2013). Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, *48*, 23–65.