International Journal of Approximate Reasoning 55 (2014) 1043–1071

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Learning Bayesian network structure: Towards the essential graph by integer linear programming tools

Milan Studený[a,*], David Haws[b]

[a] Institute of Information Theory and Automation of the ASCR, Prague, Czech Republic
[b] IBM T.J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598, USA

A R T I C L E I N F O

Article history:
Available online 8 October 2013

Keywords:
Learning Bayesian network structure
Integer linear programming
Characteristic imset
Essential graph

A B S T R A C T

The basic idea of the geometric approach to learning a Bayesian network (BN) structure is to represent every BN structure by a certain vector. If the vector representative is chosen properly, it allows one to re-formulate the task of finding the global maximum of a score over BN structures as an integer linear programming (ILP) problem. Such a suitable zero-one vector representative is the *characteristic imset*, introduced by Studený, Hemmecke and Lindner in 2010, in the proceedings of the 5th PGM workshop. In this paper, extensions of characteristic imsets are considered which additionally encode chain graphs without flags equivalent to acyclic directed graphs. The main contribution is a polyhedral description of the respective domain of the ILP problem, that is, by means of a set of linear inequalities. This theoretical result opens the way to the application of ILP software packages. The advantage of our approach is that, as a by-product of the ILP optimization procedure, one may get the *essential graph*, which is a traditional graphical BN representative. We also describe some computational experiments based on this idea.

1. Introduction

Learning a *Bayesian network* (BN) structure is the statistical task of model choice, where the candidate statistical structural models are ascribed to acyclic directed graphs. A score-and-search approach to this learning task consists in maximization of a *quality criterion* $\mathcal{Q}$, also called a *score* or a *scoring function*, which is a real function of the (acyclic directed) graph G and the observed database D. The value $\mathcal{Q}(G,D)$ says how much the BN structure defined by the graph G, that is, the statistical model ascribed to G, fits the database D. Note that some researchers in machine learning are accustomed to identify a BN structure with the respective equivalence class of graphs and prefer to talk about learning *an equivalence class* of Bayesian networks.

Two important technical assumptions on the criterion $\mathcal{Q}$ were pinpointed in the literature in connection with computational aspects of the above-mentioned maximization task. Since the goal is to learn a BN structure, $\mathcal{Q}$ should be *score equivalent* [3], which means it ascribes the same value to equivalent graphs, that is, to graphs defining the same BN structure. The other assumption is that $\mathcal{Q}$ is *decomposable*, which means $\mathcal{Q}(G,D)$ decomposes into contributions which correspond to factors in the factorization according to the graph G. Typically, it is required that $\mathcal{Q}$ is additively decomposable [5], that is, $\mathcal{Q}(G,D)$ is the sum of such local contributions, called *local scores*.

The geometric approach is to represent every BN structure by a certain vector so that such a criterion $\mathcal{Q}$ becomes an affine function of the vector representative, that is, a linear function plus a constant term. This idea was introduced by Studený already in 2005 [22] and then deepened by Studený, Vomlel and Hemmecke in 2010 [23]. A suitable (uniquely

* Corresponding author.
E-mail addresses: studeny@utia.cas.cz (M. Studený), dchaws@gmail.com (D. Haws).

0888-613X/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ijar.2013.09.016

determined) zero-one vector BN representative seems to be the *characteristic imset*, introduced recently by Studený, Hemmecke and Lindner [24,12].

Jaakkola et al. [13] and Cussens [6,7] came independently with an analogous geometric approach. The main difference is that they used certain special zero-one vector codes of (acyclic) directed graphs to represent (non-uniquely) BN structures. On the other hand, they made much more progress with the practical application of *integer linear programming* (ILP) tools. To overcome technical problems with the exponential length of their vectors they utilized the idea of the reduction of the search space developed by de Campos et al. [9,10], based on a particular form of databases and quality criteria occurring in practice.

We have compared [28] both methods of BN structure vector representation and found that the characteristic imset can be viewed as a (many-to-one) linear function of the above mentioned zero-one codes of directed graphs. We also found [28] that by transforming the polyhedral approximation used by Jaakkola et al. [13] and Cussens [7] one also gets a polyhedral approximation for the characteristic imset polytope. However, an unpleasant finding was that, while there is a one-to-one correspondence for their acyclicity-encoding inequalities, their basic non-negativity and equality constraints are transformed to much higher number of inequalities. Thus, direct transformation of their inequalities does not lead to a suitable ILP problem formulation in terms of characteristic imsets.

Lindner [17] in her thesis dealt with the problem of finding a workable LP relaxation of the characteristic imset polytope, that is, an outer polyhedral approximation such that the only vectors with integer components satisfying the considered inequalities are the characteristic imsets. To overcome/avoid this problem she used the idea of extending the characteristic imset with additional components, which is a useful standard trick in combinatorial optimization. Her additional components allowed her to encode acyclic directed graphs inducing the characteristic imset. She also reported on some computational experiments based her approach.

This paper is another step on the way to develop a consistent ILP approach based on characteristic imsets. Being inspired by Lindner [17], we introduce an extended vector BN structure representative which includes the characteristic imset and, moreover, encodes a certain special graph (equivalent to an acyclic directed graph). The main theoretical result is a *polyhedral characterization* of the domain of the respective ILP problem. Note that the objective in this ILP problem only depends on the characteristic imset and the additional components play an auxiliary role; this is different from the approach based on direct encoding of graphs [13,7].

More specifically, a set of linear inequalities is presented such that the only vectors with integer components in the polyhedron specified by those inequalities are the above mentioned extended characteristic imsets. The presented inequalities are classified in four groups. The number of inequalities in the first two groups is polynomial in the number of variables, that is, in the number of nodes of the graph, while the number of remaining inequalities is exponential. However, provided that the length of the vector representatives is limited/reduced to a quasi-polynomial number by the idea presented by de Campos and Ji [10], the number of inequalities in the third group can be reduced to a quasi-polynomial number as well. The inequalities in the fourth group correspond to acyclicity restrictions. In general, they cannot be reduced to a polynomial number, but the method of iterative constraint adding may be applied to solve the respective ILP problem. The overall number of our inequalities is lower than Lindner [17] provided.

The main advantage of our approach is that once an optimal solution is found one can use the obtained (extended) characteristic imset to get the corresponding *essential graph*, which is known as a standard (unique) graphical BN representative [2]. This graph can be obtained as the solution of a secondary ILP problem: it has another objective but shares with the main ILP problem the first two groups of inequalities. Thus, the second ILP problem has both polynomial length of vectors and polynomial number of inequalities.

We have also performed some computational experiments whose aim was to verify the feasibility of the approach. This was confirmed but the experiments also indicated that the number of our inequalities is too high to be able to compete with the running times presented at GOBNILP web page [32]. On the other hand, the experiments confirmed the assumption that the second ILP problem, used to get the essential graph, is easily solvable.

In Section 2 we recall basic concepts and some special results on chain graphs we use later in the paper. In Section 3 more details on the ILP approach to learning BN structure are given; specifically, we describe how to compute the objective in the characteristic imsets case (from local scores) and how to utilize the search space reduction [10] in the context of characteristic imsets. Section 4 describes the theoretical basis of our specific ILP approach; we say which graphs are encoded in our extended characteristic imsets, list/comment the inequalities and formulate the main results. In Section 5 we describe both phases to learn the optimal essential graph. Section 6 deals with two specific methods to solve the main ILP problem that we tested in our computational experiments, described then in Section 7. In Conclusions we comment the results and discuss further perspectives. Appendix A contains the proofs.

2. Basic concepts

Let N be a finite non-empty set of *variables*; to avoid the trivial case, assume $|N| \geqslant 2$. In statistical context, the elements of N correspond to random variables in consideration; in graphical context, they correspond to nodes.

2.1. Graphical concepts

Graphs considered in this paper have N as the set of nodes and they are *hybrid* in the sense that two types of edges between (distinct) nodes $i, j \in N$ are allowed, namely directed edges, called *arrows*, and denoted like $i \to j$ or $j \leftarrow i$, and undirected edges, called *lines*, and denoted like $i — j$ or $j — i$. Multiple edges are not allowed between two nodes. If there is an edge between nodes i and j, we say they are *adjacent*. A graph is *undirected* if it only has lines; it is *directed* if it only has arrows.

A *cycle* of length $m \geqslant 3$ in a graph H is the sequence ρ: $i_0, i_1, \ldots, i_m = i_0$, where $i_1, \ldots, i_m$ are distinct nodes, and, for each $r = 0, \ldots, m-1$, (i_r, i_{r+1}) is an edge in H. The cycle ρ is *chordless*, or minimal, if there is no other edge in H between nodes in $\{i_1, \ldots, i_m = i_0\}$ besides those which form the cycle ρ. An undirected graph is called *chordal* if it has no chordless cycle of length $m \geqslant 4$.

The cycle ρ is *directed* if $i_r \to i_{r+1}$ in H for each $r = 0, \ldots, m-1$. A directed graph is *acyclic* if it has no directed cycle. An equivalent definition of an acyclic directed graph G is that there exists an ordering $b_1, \ldots, b_{|N|}$ of all nodes of G which is consistent with the direction of arrows: $b_i \to b_j$ in G implies $i < j$. The set of *parents* of a node $i \in N$ in a (directed) graph G is the set $pa_G(i) \equiv \{j \in N\colon\ j \to i \text{ in } G\}$.

The cycle ρ is *semi-directed* if $i_0 \to i_1$ in H and, for each $r = 1, \ldots, m-1$ one has either $i_r \to i_{r+1}$ in H or $i_r — i_{r+1}$ in H. We say that a hybrid graph H is *3-acyclic* if it has no semi-directed cycle of length 3.

A set of nodes $C \subseteq N$ in a hybrid graph H is *connected* if, for every pair $i, j \in C$, there exists an undirected path between i and j, that is, a sequence of (distinct) nodes $i = i_1, \ldots, i_s = j$, $s \geqslant 1$ such that $i_r — i_{r+1}$ in H for $r = 1, \ldots, s-1$. The maximal connected sets (with respect to inclusion) are called the *connected components* of H.

A hybrid graph H is called a *chain graph* if (all) its components can be ordered into a sequence $C_1, \ldots, C_m$, $m \geqslant 1$, called a *chain*, such that if $i \to j$ in H, then $i \in C_r$ and $j \in C_s$ with $r < s$. An equivalent definition of a chain graph is that it is a hybrid graph which has no semi-directed cycle, or alternatively, which has no semi-directed chordless cycle; see [20, Lemma 2.1]. Thus, chain graphs are, in fact, acyclic hybrid graphs, they involve acyclic directed graphs. Also, clearly, every chain graph is a 3-acyclic hybrid graph.

An *immorality* in H is an induced subgraph (over three nodes) of the form $i \to k \leftarrow j$, that is, i and j are not adjacent in H. A *flag* in a hybrid graph H is an induced subgraph (over three nodes) of the form $i \to j — k$, which means that i and k are not adjacent.

2.2. Bayesian network structure

In statistical context, each variable $i \in N$ is assigned a finite (individual) *sample space* X_i (of possible values); assume $|\mathsf{X}_i| \geqslant 2$ for each $i \in N$ to avoid technical problems. The *joint sample space* is the Cartesian product $\mathsf{X}_N \equiv \prod_{i \in N} \mathsf{X}_i$. For $A \subseteq N$, the *projection* of an element $x = [x_i]_{i \in N}$ of X_N to A is the configuration/list of values $x_A = [x_i]_{i \in A}$. The symbol X_A will denote the respective *projection space*. Thus, for $A = \emptyset$, the projection space $\mathsf{X}_\emptyset$ is a singleton, consisting of the empty list; for $\emptyset \neq A$ one has $\mathsf{X}_A = \prod_{i \in A} \mathsf{X}_i$.

A *Bayesian network* is a pair (G, P), where G is an acyclic directed graph having N as the set of nodes and P a *Markovian* probability distribution (with respect to G) on the *joint sample space* X_N. This means that P satisfies conditional independence restrictions determined by G; see [15, §3.2.2] for details. The *BN structure* defined by an acyclic directed graph G is the class of Markovian probability distributions with respect to G on (fixed) $\mathsf{X}_N = \prod_{i \in N} \mathsf{X}_i$.

However, different graphs with the same set of nodes can be *Markov equivalent*, which means they define the same BN structure. The classic graphical characterization of equivalent graphs by Verma and Pearl [30] states that two graphs are Markov equivalent if and only if they have the same adjacencies and immoralities.

2.3. Essential graphs

A standard (unique) graphical representative of a BN structure is the following one.

Definition 1. Let $\mathcal{G}$ be a Markov equivalence class of acyclic directed graphs over N. The *essential graph* G^* of $\mathcal{G}$ is a hybrid graph over N defined as follows:

- $i \to j$ in G^* if $i \to j$ in every G from $\mathcal{G}$,
- $i — j$ in G^* if there are graphs G_1 and G_2 in $\mathcal{G}$ with $i \to j$ in G_1 and $i \leftarrow j$ in G_2.

An illustrative example of this concept is in Fig. 1. This particular terminology and the first graphical characterization of essential graphs was given by Anderson, Madigan and Perlman [2]. The characterization implies that every essential graph is a chain graph and has no flags. Actually, the class of *chain graphs without flags* appears to be important in this context. One can introduce a graphical concept of equivalence for such graphs.

Definition 2. Two chain graphs without flags G and H are *equivalent* if they share nodes, adjacencies and immoralities. Given two such graphs, we say that H is *larger* than G if, for any $i, j \in N$, $i \to j$ in H implies $i \to j$ in G.

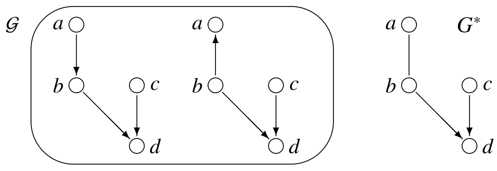


Fig. 1. An equivalence class $\mathcal{G}$ of acyclic directed graphs and the essential graph G^* of $\mathcal{G}$.

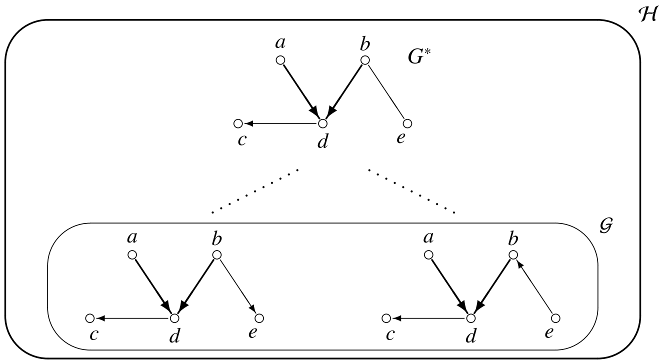


Fig. 2. An equivalence class $\mathcal{H}$ of chain graphs without flags, containing a Markov equivalence class of acyclic directed graphs $\mathcal{G}$.

These concepts are illustrated by Fig. 2, where the equivalence class $\mathcal{H}$ of three chain graphs without flags is shown. The upper graph is larger then the two lower graphs, which are incomparable from the point of view of this partial order. The meaning of this equivalence of G and H is that they define the same statistical model; cf. [21, Lemma 2]. The following characterization of the essential graphs [21, Corollary 4] will be utilized later.

Lemma 1. *Let $\mathcal{G}$ be a Markov equivalence class of acyclic directed graphs over N and $\mathcal{H}$ the equivalence class of chain graphs without flags such that $\mathcal{G} \subseteq \mathcal{H}$. Then the essential graph G^* of $\mathcal{G}$ is the largest graph in $\mathcal{H}$.*

Also Lemma 1 is illustrated by Fig. 2: the internal oval embraces Markov equivalence class of two acyclic directed graphs $\mathcal{G}$. Observe that $\mathcal{G}$ consists of the minimal graphs within $\mathcal{H}$ with respect to the considered ordering while G^* is the largest graph in $\mathcal{H}$. Further, an important observation is that a chain graph H without flags is equivalent to an acyclic directed graph G if and only if the induced subgraphs of H for its components are chordal undirected graphs; see [21, Lemma 3].

2.4. Characteristic imset

The concept of a characteristic imset was proposed by Studený, Hemmecke and Lindner as an algebraic representative of a BN structure in 2010 [24]. Note that the term *imset* is an acronym for **i**nteger-valued **m**ulti**set** and was introduced by Studený [22] to name vectors whose components are integers indexed by subsets of a fixed basic set.

For the purpose of this paper, the following equivalent definition of the characteristic imset is suitable.

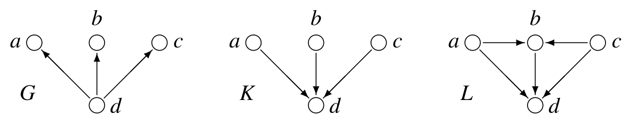
Definition 3. Let G be an acyclic directed graph over N. The *characteristic imset* for G can be introduced as the zero-one vector c_G with components $\mathsf{c}_G(S)$ where $S \subseteq N$, $|S| \geqslant 2$, and

$$\mathsf{c}_G(S) = 1 \quad \Longleftrightarrow \quad \exists i \in S \quad \text{such that} \quad S \setminus \{i\} \subseteq pa_G(i). \tag{1}$$

The characteristic imset concept is illustrated in Fig. 3. Note that in the figure we use an abbreviated notation ij for $\{i, j\}$ and ijk for $\{i, j, k\}$; the same notational principle is also applied later to sets (of cardinality two and three) indexing the components of our vectors. Thus, in the graph G from Fig. 3, there is no subset S of cardinality 3 with $S \setminus \{i\} \subseteq pa_G(i)$. However, for the graph K one has $pa_K(d) = \{a, b, c\}$, which implies that any subset of cardinality 3 except $\{a, b, c\}$ has $c_K(S) = 1$. In the graph L one even has $\mathsf{c}_L(\{a, b, c\}) = 1$.

The point is that two acyclic directed graphs G and H over N are Markov equivalent if and only if $\mathsf{c}_G = \mathsf{c}_H$; see [12, §3]. Moreover, [12, Corollary 2] implies that, for different $i, j, k \in N$,

(i) i and j are adjacent in G iff $\mathsf{c}_G(ij) = 1$,
(ii) $i \to k \leftarrow j$ is an immorality in G iff $\mathsf{c}_G(ijk) = 1$ and $\mathsf{c}_G(ij) = 0$.



	ab	ac	ad	bc	bd	cd	abc	abd	acd	bcd	$abcd$
c_G	0	0	1	0	1	1	0	0	0	0	0
c_K	0	0	1	0	1	1	0	1	1	1	1
c_L	1	0	1	1	1	1	1	1	1	1	1

Fig. 3. Three acyclic directed graphs and corresponding characteristic imsets.

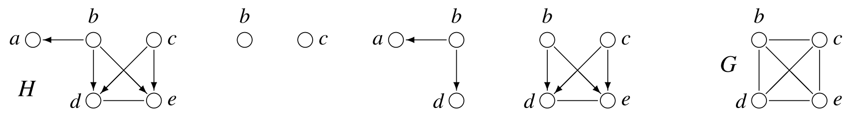


Fig. 4. Several examples of chain graphs without flags equivalent to acyclic directed graphs.

Thus, it follows from facts (i)–(ii) and the graphical characterization of Markov equivalence of acyclic directed graphs that the characteristic imset c_G is uniquely determined by its values $c_G(S)$ for $S \subseteq N$, $2 \leqslant |S| \leqslant 3$.

However, an important warning for the reader is that the components for sets of cardinality four and more do not depend linearly on the components of the cardinality two and three. The relation is non-linear: specifically, it was shown [26, Lemma 4.1] that, $c_G(S) = 1$ for a set $S \subseteq N$ with $|S| \geqslant 4$ if and only if there are at least 3 subsets T of S of cardinality $|T| = |S| - 1$ such that $c_G(T) = 1$. Another equivalent condition is that there are at least $|S| - 1$ subsets T of S of cardinality $|T| = |S| - 1$ such that $c_G(T) = 1$. Note that these two equivalent conditions gave us the inspiration to the below-mentioned extension inequalities (e.1)–(e.2) from Section 4.2. The fact that the relation is non-linear means that the restricted characteristic imset for sets of cardinality two and three is not a suitable vector representative of BN structure provided one plans to apply the ILP approach; see Section 3.1 for an explanation.

It appears to be appropriate to have a formula for the characteristic imset on the basis of any graph H in the class $\mathcal{H}$ from Lemma 1. For this purpose one needs the following auxiliary concept.

Definition 4. We say that a hybrid graph H over $S \subseteq N$ has a *super-terminal component* if there exists non-empty set $K \subseteq S$ such that

- K is a *complete set* in H, which means, for each pair of distinct nodes $i, k \in K$, one has $i — k$ in H,
- $\forall j \in S \setminus K\ \forall i \in K$ one has $j \to i$ in H.

It is easy to see that a super-terminal component K, if exists, is uniquely determined.

The concept is illustrated in Fig. 4. The graph H there is a chain graph without flags equivalent to an acyclic directed graph, which can be obtained from H by directing the line $d — e$. However, H is not the corresponding essential graph: the essential graph can be obtained from H by replacing the arrow $b \to a$ by a line. The induced subgraph of H for $S = bc$ has no adjacency and, therefore, does not have a super-terminal component. Neither the induced subgraph for $S = abd$ has such a component. However, the induced subgraph for $S = bcde$ already has the super-terminal component, namely $K = de$. Another example of such a graph is the graph G from Fig. 4: the whole node set $bcde$ is complete in G, and, therefore, a super-terminal component of G.

The following result follows directly from a former result [12, Theorem 2].

Lemma 2. *Let H be a chain graph without flags equivalent to an acyclic directed graph G. For any $S \subseteq N$, $|S| \geqslant 2$ one has $c_G(S) = 1$ if and only if the induced subgraph of H for S (denoted by H_S) has a super-terminal component.*

Lemma 2 is illustrated in later Table 1, where the characteristic imset corresponding to the graph H from Fig. 4 is included.

2.5. Straightforward codes of graphs

Jaakkola et al. [13] and Cussens [6,7] used a special method for vector encoding (acyclic) directed graphs over N. The vector η_G encoding G has components indexed by pairs $(i|B)$, where $i \in N$ and $B \subseteq N \setminus \{i\}$. Specifically, it is defined as follows:

$$\eta_G(i|B) = \begin{cases} 1 & B = pa_G(i), \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In their paper [13], Jaakkola et al. mentioned special inequalities for η_G, which they called the *cluster inequalities*. These inequalities correspond to sets $S \subseteq N$, $|S| \geqslant 2$:

$$1 \leqslant \sum_{i \in S} \sum_{B \subseteq N \setminus S} \eta_G(i|B). \tag{3}$$

The meaning of the inequality (3) is that there exists at least one node i in the set S with $pa_G(i) \cap S = \emptyset$. In other words, the induced subgraph G_S has at least one *initial node*, that is, a node $i \in S$ with no $j \in S$ such that $j \to i$ in G_S. Because G_S is an acyclic directed graph over S, the existence of such a node $i \in S$ is obvious. That is why (3) holds for every acyclic directed graph G.

Recently, we have established the relation of the characteristic imset to this straightforward code of G [28]. Actually, c_G is a linear function of η_G given by

$$\mathsf{c}_G(S) = \sum_{i \in S} \sum_{B,\, S \setminus \{i\} \subseteq B \subseteq N \setminus \{i\}} \eta_G(i|B) \quad \text{for } S \subseteq N,\ |S| \geqslant 2. \tag{4}$$

Indeed, (4) follows directly from Definition 3: clearly, at most one node $i \in S$ with $S \setminus \{i\} \subseteq pa_G(i)$ exists in an acyclic directed graph G_S.

2.6. Learning a BN structure

The task of *learning a BN structure* is to determine the structure on the basis of an observed (complete) *database* D, which is a sequence $x_1, \ldots, x_d$ of elements of the joint sample space $\mathsf{X}_N = \prod_{i \in N} \mathsf{X}_i$ ($d \geqslant 1$ is the length of the database). The *projection* of D to $A \subseteq N$ is the sequence of respective projections $x_A^1, \ldots, x_A^d$, denoted by the symbol D_A. Given a database D of length $d \geqslant 1$ and $y \in \mathsf{X}_A$, $A \subseteq N$ we use a special notation $d_{[y]} \equiv |\{l\colon\ 1 \leqslant l \leqslant d,\ x_A^l = y\}|$ for the number of occurrences of y in the database projection D_A. In particular, for the empty list $y \in \mathsf{X}_\emptyset$, one always has $d_{[y]} = d$. The concatenation of two configurations $y \in \mathsf{X}_A$ and $z \in \mathsf{X}_B$ for disjoint A and B will be denoted by $[y, z]$; it belongs to $\mathsf{X}_{A \cup B}$.

Learning a BN structure is often done by maximizing some *quality criterion*, also called a *score*, which is a bivariate real function $(G, D) \mapsto \mathcal{Q}(G, D)$, where G is an acyclic directed graph over N and D a database. The value $\mathcal{Q}(G, D)$ should say how the statistical model defined by G is good to explain the occurrence of D. For a formal definition of the relevant concept of *statistical consistency* we refer to Neapolitan's book [18].

Given the observed database D, the goal is to maximize $G \mapsto \mathcal{Q}(G, D)$. Since the aim is learn a BN structure, a natural assumption is that the criterion $\mathcal{Q}$ we are going to maximize should be *score equivalent*, which means, for every database D and acyclic directed graphs G and H,

$$\mathcal{Q}(G, D) = \mathcal{Q}(H, D) \quad \text{whenever } G \text{ and } H \text{ are Markov equivalent.}$$

The crucial technical assumption is that $\mathcal{Q}$ should be additively *decomposable*, which means, it has the form

$$\mathcal{Q}(G, D) = \sum_{i \in N} q_D\big(i \big| pa_G(i)\big), \quad \text{where the summands } q_D(*|*) \text{ are called } \textit{local scores.} \tag{5}$$

Moreover, it is required by Chickering [5] in his definition that each local score term $q_D(i|B)$, for $i \in N$ and $B \subseteq N \setminus \{i\}$, only depends on the database projection $D_{\{i\} \cup B}$. This is a natural requirement from the computational point of view, implying that the score for sparse graphs can be computed from low-dimensional projections of the database. One can re-write (5) in terms of η_G:

$$\mathcal{Q}(G, D) = \sum_{i \in N} \sum_{B \subseteq N \setminus \{i\}} q_D(i|B) \cdot \eta_G(i|B), \tag{6}$$

which allows one to interpret $\mathcal{Q}$ as (the restriction of) a linear function of η_G.

A well-known example of such a score is Schwarz's [19] *Bayesian information criterion* (BIC), whose local scores are given as follows:

$$\mathsf{bic}_D(i|B) = \underbrace{\sum_{y \in \mathsf{X}_B} \sum_{z \in \mathsf{X}_i} d_{[y,z]} \cdot \ln \frac{d_{[y,z]}}{d_{[y]}}}_{\mathsf{mll}_D(i|B)} - \frac{\ln d}{2} \cdot \underbrace{\Big\{ r(i) - 1 \Big\} \cdot \prod_{j \in B} r(j)}_{\mathsf{dim}(i|B)} \quad \text{for } i \in N,\ B \subseteq N \setminus \{i\}, \tag{7}$$

where $r(i) = |\mathsf{X}_i|$, $i \in N$ are the cardinalities of the individual sample spaces. In that formula, the conventions $0 \cdot \ln \frac{0}{*} \equiv 0$ and $\prod_{j \in \emptyset} r(j) \equiv 1$ are applied. The first term here is the local score of the *maximized log-likelihood* (MLL) criterion; what is subtracted is a penalty term reflecting the length of the database d and the local contribution to the *dimension* (DIM). The penalty term is to quantify the complexity of the statistical model given by G.

Another common example is the *Bayesian Dirichlet equivalence* (BDE) score [11], whose local scores are given by the following formula:

$$\mathsf{bde}_D(i|B) = \sum_{y\in \mathsf{X}_B} \left\{ \ln \frac{\Gamma(\alpha_{[y]})}{\Gamma(\alpha_{[y]}+d_{[y]})} - \sum_{z\in \mathsf{X}_i} \ln \frac{\Gamma(\alpha_{[y,z]})}{\Gamma(\alpha_{[y,z]}+d_{[y,z]})} \right\} \quad \text{for } i\in N,\ B\subseteq N\setminus\{i\}, \tag{8}$$

where Γ denotes the Gamma function and the α-terms are so-called *hyper-parameters*. The usual assumption is that the hyper-parameters are given by the formula $\alpha_{[y]} = \alpha^* \cdot (|\mathsf{X}_B|)^{-1} = \alpha^* \cdot (\prod_{j\in B} r(j))^{-1}$ for $y \in \mathsf{X}_B$, with some fixed $\alpha^* > 0$, called the *equivalent sample size*. In this special case, the criterion is called *Bayesian Dirichlet equivalence uniform* score and abbreviated as BDEU or BDeu [11]. Setting of hyper-parameters is often a hard problem, however, a modification of the Bayesian Dirichlet score has recently been proposed [4], which prevents a wrong setting of hyper-parameters.

3. Integer linear programming approach

The task to maximize a quality criterion $\mathcal{Q}$ can be re-formulated as an *integer linear programming* (ILP) problem. Indeed, by (6), every decomposable criterion $\mathcal{Q}$ can be interpreted as a linear function of η_G, where G falls within the class of acyclic directed graphs over N.

Jaakkola et al. [13] gave a finite list of valid linear constraints on η_G's, namely the evident non-negativity inequalities, the basic equality constraints $\sum_{B\subseteq N\setminus\{i\}} \eta_G(i|B) = 1$ for $i \in N$ and the cluster inequalities (3). These constraints characterize the vectors η_G in the sense that the only vectors with integer components satisfying them are the codes of acyclic directed graphs. Such a domain description, in terms of polyhedral geometry called an *LP relaxation* (of the respective polytope), allows one to turn the learning task into an ILP problem: to optimize a linear function over vectors with integer components within a polyhedron.

To overcome the technical problem with exponential length (in $|N|$) of vectors η_G the idea of *pruning* of their components was applied. The idea taken from de Campos et al. [9] is that a particular form of scoring criteria used in practice, namely BIC and BDeu, allows one to conclude (on the basis of an observed database) that the optimal graph G has no node $i \in N$ with large $|pa_G(i)|$. Therefore, one can exclude from consideration the respective components of the vector η_G because they vanish for optimal G. This pruning procedure is time demanding, but useful: as reported by de Campos et al. [10, §6], in practical cases it typically results in the reduction of the parent set cardinality to at most 5, only in a few cases the maximal cardinality was 7 or 8.

To overcome the problem with the exponential number of cluster inequalities (3) Jaakkola et al. [13] used the method of iterative constraint adding, where they employed the dual formulation (of their approximate LP problems) to guide the choice of a newly added cluster constraint.

Cussens [6] was interested in pedigree learning, in which case the parent set cardinality is bounded by 2. However, to ensure the acyclicity of the graph G he used another trick: the idea of *extending* the vector BN representatives. He added some additional components to the (shortened) vector η_G which allowed him to encode the total order of nodes consistent with the direction of arrows in the graph G. Then he introduced easily an LP relaxation for these extended vector representatives. Actually, the number of the added components and the number of inequalities ensuring acyclicity were both polynomial in $|N|$.

The later paper by Cussens [7] was partly inspired by Jaakkola et al. [13]: he considered non-extended η_G-codes and the same collection of inequalities as Jaakkola et al. The goal was unrestricted BN structure learning and to overcome the problem with the exponential number of these inequalities Cussens used the *cutting plane* approach.

3.1. ILP with characteristic imsets

A basic observation by Hemmecke, Lindner and Studený [12, Lemma 1] is that every score equivalent and additively decomposable criterion $\mathcal{Q}$ has the form

$$\mathcal{Q}(G, D) = \mathcal{Q}\big(G^{\emptyset}, D\big) + \sum_{S\subseteq N,\ |S|\geqslant 2} r^{\mathcal{Q}}_D(S)\cdot \mathsf{c}_G(S), \tag{9}$$

where $G^{\emptyset}$ is the empty graph over N, that is, a graph without adjacencies, and $r^{\mathcal{Q}}_D$ a uniquely determined vector, depending on the database D only, called the *revised data vector* (relative to $\mathcal{Q}$). Note that, while $\mathcal{Q}$ is an affine function of the characteristic imset, this is not true for the restriction of the characteristic imset to sets of cardinality two and three. This can be observed as the consequence of the non-linear relation between components of characteristic imsets mentioned in Section 2.4. In particular, such shortened characteristic imset cannot be utilized in the context of linear programming approach.

Given $\mathcal{Q}$, one is usually able to derive a mathematical formula for the components of the data vector $r^{\mathcal{Q}}_D$. An alternative way is to compute $r^{\mathcal{Q}}_D(S)$ for $S \subseteq N$, $|S| \geqslant 2$ directly from local scores.

Lemma 3. *Let $\mathcal{Q}$ be a score equivalent and additively decomposable criterion with local scores $q_D(*|*)$. Then, for any $S \subseteq N$, $|S| \geqslant 2$, one has*

$$r_D^{\mathcal{Q}}(S) = \sum_{K \subseteq R} (-1)^{|R \setminus K|} \cdot q_D(j|K) \quad \textit{where } j \in S \textit{ and } R = S \setminus \{j\}. \tag{10}$$

In particular, the right-hand side of (10) does not depend on the choice of $j \in S$.

Proof. The idea is to prove $r_D^{\mathcal{Q}}(S) = \sum_{K \subseteq R} (-1)^{|R|} \cdot (-1)^{|K|} \cdot q_D(j|K)$, which is a different form of (10), by induction on $|S|$. Given $S \subseteq N$, $|S| \geqslant 2$ and $j \in S$, put $R \equiv S \setminus \{j\}$ and consider the graph G with $pa_G(j) = R$ and $pa_G(i) = \emptyset$ for any $i \in N \setminus \{j\}$. Observe

$$\sum_{T \subseteq S,\ |T| \geqslant 2,\ j \in T} r_D^{\mathcal{Q}}(T) \overset{(1)}{=} \sum_{T \subseteq N,\ |T| \geqslant 2} r_D^{\mathcal{Q}}(T) \cdot \mathsf{c}_G(T) \overset{(9)}{=} \mathcal{Q}(G, D) - \mathcal{Q}\big(G^{\emptyset}, D\big) \overset{(5)}{=} q_D(j|R) - q_D(j|\emptyset),$$

which implies

$$r_D^{\mathcal{Q}}(S) = q_D(j|R) - q_D(j|\emptyset) - \sum_{T \subset S,\ |T| \geqslant 2,\ j \in T} r_D^{\mathcal{Q}}(T). \tag{11}$$

If $|S| = 2$ then (11) directly gives (10). If $|S| \geqslant 3$ then by (11) and the induction premise

$$\begin{aligned}
r_D^{\mathcal{Q}}(S) - q_D(j|R) &= -q_D(j|\emptyset) - \sum_{T \subset S,\ |T| \geqslant 2,\ j \in T} \ \sum_{L \subseteq R \cap T} (-1)^{|R \cap T|} \cdot (-1)^{|L|} \cdot q_D(j|L) \\
&= -q_D(j|\emptyset) + \sum_{T \subset S,\ |T| \geqslant 2,\ j \in T} \ \sum_{L \subseteq R \cap T} (-1)^{|T|} \cdot (-1)^{|L|} \cdot q_D(j|L) \\
&= -q_D(j|\emptyset) + \sum_{L \subset R} (-1)^{|L|} \cdot q_D(j|L) \cdot \sum_{T,|T| \geqslant 2, L \cup \{j\} \subseteq T \subset S} (-1)^{|T|} \\
&= -q_D(j|\emptyset) + q_D(j|\emptyset) \cdot \big[1 + (-1)^{|R|}\big] + \sum_{\emptyset \neq L \subset R} (-1)^{|L|} \cdot q_D(j|L) \cdot (-1)^{|R|},
\end{aligned}$$

where we used (for $L = \emptyset$)

$$\sum_{T,\ |T| \geqslant 2,\ \{j\} \subseteq T \subset S} (-1)^{|T|} = \underbrace{\sum_{T,\ \{j\} \subseteq T \subseteq S} (-1)^{|T|}}_{=0} - (-1)^{|\{j\}|} - (-1)^{|S|} = 1 + (-1)^{|R|}$$

and, analogously, for $\emptyset \neq L \subset R$, $\sum_{T, L \cup \{j\} \subseteq T \subset S} (-1)^{|T|} = (-1)^{|R|}$. Hence,

$$r_D^{\mathcal{Q}}(S) = \sum_{L \subseteq R} (-1)^{|L|} \cdot q_D(j|L) \cdot (-1)^{|R|},$$

which was desired. □

The formula (9) means that the criterion $\mathcal{Q}$ can be interpreted as an affine function of the characteristic imset c_G, which is a unique BN representative. Thus, to employ the methods of ILP, one has to come with an LP relaxation for characteristic imsets. We transformed [28] the above-mentioned LP relaxation by Jaakkola et al. [13] through (4) into the framework of characteristic imsets. A pleasant finding was that the cluster inequality (3) takes a neat form

$$\sum_{T \subseteq S,\ |T| \geqslant 2} \mathsf{c}(T) \cdot (-1)^{|T|} \leqslant |S| - 1. \tag{12}$$

Another non-trivial observation was that the transformed linear inequalities define an LP relaxation of the characteristic imset polytope. However, since the number of resulting inequalities is super-exponential in $|N|$, which is an unpleasant consequence of the many-to-one transformation, this LP relaxation does not seem to be suitable for practical purposes.

3.2. Search space reduction for characteristic imsets

Search space reduction described by de Campos et al. [10] is based on the following simple idea, mentioned already by Teyssier and Koller [29, §3.2]: provided that, for some $i \in N$ and $C \subset B \subseteq N \setminus \{i\}$, the observed database D is such that $q_D(i|C) > q_D(i|B)$, then (5) implies that no acyclic directed graph G maximizing $G \mapsto \mathcal{Q}(G, D)$ has $pa_G(i) = B$. In other words, there is no optimal graph G with $\eta_G(i|B) = 1$ and the optimality search can be limited to the graphs G with $\eta_G(i|B) = 0$.

In particular, if $\mathcal{Q}$ is a decomposable criterion and $S \subseteq N$, $|S| \geqslant 2$ such that

$$\forall i \in S\ \forall B,\ S \setminus \{i\} \subseteq B \subseteq N \setminus \{i\} \qquad \exists C \subset B\ q_D(i|C) > q_D(i|B), \tag{13}$$

then (4) implies that there is no optimal (acyclic directed) graph G with $\mathsf{c}_G(S) = 1$. In this case, the search for the optimal graph can be limited to graphs G with $\mathsf{c}_G(S) = 0$.

We have implemented a certain heuristic procedure for pruning components of the characteristic imset based on the above principle. The result of the procedure is the class of sets $\mathcal{T} \subseteq \{S \subseteq N\colon\ |S| \geqslant 2\}$ such that

- $\mathsf{c}_G(R) = 0$ for any $R \subseteq N$, $R \notin \mathcal{T}$ and optimal graph G,
- $\mathcal{T}$ is closed under subsets: $S \in \mathcal{T}$, $T \subseteq S$, $|T| \geqslant 2$ implies $T \in \mathcal{T}$.

Note that the second technical condition is not necessary, but simplifies things. It is also partially justified by (1). Indeed, realize that $\mathcal{T}$ has the interpretation of the class of sets S such that it might be the case that $\mathsf{c}_G(S) = 1$ for an optimal graph G. Provided $\mathsf{c}_G(S) = 1$, the condition (1) implies that $\mathsf{c}_G(T) = 1$ for quite many subsets T of S with $|T| \geqslant 2$. Therefore, if $S \in \mathcal{T}$, it makes sense to allow $T \in \mathcal{T}$ for any $T \subseteq S$, $|T| \geqslant 2$.

The aim of the procedure is, of course, to get $\mathcal{T}$ consisting of sets of small cardinality, that is,

- there exists a low upper bound $t \in \mathbb{N}$ such that $|S| \leqslant t$ for any $S \in \mathcal{T}$.

Note that in practical situations one can expect massive pruning of components of η_G. Thus, by (4), if the components of η_G were pruned up to the maximal parent set cardinality $t - 1$, for some $t \in \mathbb{N}$, then one can assume $\mathsf{c}_G(S) = 0$ for $S \subseteq N$ with $|S| > t$ in any optimal graph G. This upper cardinality bound t, or more precisely $|\mathcal{T}|$, is crucial for efficiency of the subsequent ILP procedure (see Section 5.1). Nevertheless (many technical) details of this pruning procedure are omitted in this paper; we plan another paper devoted particularly to this topic.

An important note is that one can utilize a previously performed search space reduction in local scores and easily transform it to a search space reduction in characteristic imsets. As reported by de Campos et al. [10], they applied their software to several databases and obtained caches of values of already pruned local scores. If we are in such a situation, then we can obtain $\mathcal{T}$ as the class of sets $S \subseteq N$, $|S| \geqslant 2$ such that there exists a local score $q_D(i|B)$ in the considered cache of pruned local scores with $S \subseteq \{i\} \cup B$. Then we only need the values $r_D^{\mathcal{Q}}(S)$ for $S \in \mathcal{T}$ and these can be computed by means of Lemma 3. However, if the cache of pruned local scores has "holes" in the sense that it contains $q_D(i|B)$ but not $q_D(i|C)$ for some $C \subset B$, then one has to compute $q_D(i|C)$ directly so that (10) can be applied.

4. Two phases to get the essential graph

In this section we describe the theoretical basis of our specific ILP approach. Note we describe here the general case, that is, without considering the search space reduction mentioned in Section 3.2. The pure version of the obtained ILP problem is still too complex to be solved by standard ILP software packages: this is because of both exponential length of vector codes and of the exponential number of inequalities (in $|N|$). We describe later, in Section 5, how things are simplified as the result of pruning (components of the characteristic imset).

The final aim of the learning procedure proposed in this paper is to get a zero-one vector, which is an extension of the characteristic imset and, simultaneously, encodes the essential graph. Therefore, the considered vectors have two kinds of components:

- $\mathsf{c}(S)$ for sets $S \subseteq N$, $|S| \geqslant 2$, intended to encode the values of the characteristic imset,
- $\mathsf{a}(i \to j)$ for ordered pairs $i, j \in N$, $i \neq j$, intended to encode the presence of an arrow $i \to j$ in a graph; in other words, these are the codes of *arrowheads*.

However, it seems to be convenient and advantageous, *in the first phase*, to allow a wider class of graphs. The idea is to consider a certain class of graphs which includes both the essential graphs and acyclic directed graphs; see below Section 4.1. What is presented in this section is the LP relaxation of the respective polytope, that is, of the convex hull of the set of considered vector codes; see Section 4.2. This allows one to specify properly the domain of the respective ILP problem; see Section 5.1.

Then, *in the second phase*, when the solution of the first ILP problem is already found and the characteristic imset c determined, another ILP problem is formulated. The second ILP problem is based on the same LP relaxation and its solution

Table 1
The extended characteristic imset for the graph H from Fig. 4.

	$a\to b$	$b\to a$	$a\to c$	$c\to a$	$a\to d$	$d\to a$	$a\to e$	$e\to a$	$b\to c$	$c\to b$	$b\to d$	$d\to b$	$b\to e$	$e\to b$	$c\to d$	$d\to c$	$c\to e$	$e\to c$	$d\to e$	$e\to d$
a_H	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0

	ab	ac	ad	ae	bc	bd	be	cd	ce	de	abc	abd	abe	acd	ace	ade	bcd	bce	bde	cde
c_H	1	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1

	$abcd$	$abce$	$abde$	$acde$	$bcde$	$abcde$
c_H	0	0	0	0	1	0

is the desired simultaneous code of the essential graph and the characteristic imset c. The second ILP procedure is much simpler than the first one (clearly polynomial in $|N|$); it can be viewed as an LP version of the reconstruction algorithm for the essential graph on the basis of the characteristic imset. Morerover, the second ILP procedure can be modified in such a way that the solution will be one of the respective acyclic directed graphs; see Section 5.2.

4.1. *The class of graphs and encoding*

The output of the first phase should be a graph over N which falls within the class of

chain graphs without flags that are equivalent to acyclic directed graphs.

Now, we introduce special zero-one vector codes for even more general graphs, because this appears to be convenient.

Definition 5. Let H be a hybrid graph over N which is 3-acyclic and has no flags. Then we ascribe to H a zero-one vector $(\mathsf{a}_H, \mathsf{c}_H)$ with components determined as follows:

$$\mathsf{a}_H(i\to j) = \begin{cases} 1 & i\to j \text{ in } H, \\ 0 & \text{otherwise,} \end{cases} \quad \text{for distinct } i, j\in N, \tag{14}$$

$$\mathsf{c}_H(S) = \begin{cases} 1 & H_S \text{ has a super-terminal component,} \\ 0 & \text{otherwise,} \end{cases} \quad \text{where } S\subseteq N,\ |S|\geqslant 2. \tag{15}$$

The a-part of the vector encodes the presence of the arrows $i\to j$ in the graph H, while the c-part is, by Lemma 2, the respective characteristic imset. Note that the number of added components $|N|\cdot(|N|-1)$ is polynomial in $|N|$. The above concept is illustrated by the example from Table 1.

Observe, that in case $|S|=2$, H_S has a super-terminal component if and only if the nodes in S are adjacent in H. Thus, the relation (15) means that, for distinct $i, j\in N$,

$$\mathsf{c}_H(ij)=1 \quad \Leftrightarrow \quad (i,j) \quad \text{is an edge in } H. \tag{16}$$

In case $|S|=3$, as H is 3-acyclic, H_S has a super-terminal component if and only if either H_S is *quasi-complete*, that is, every pair of distinct nodes in S is adjacent in H, or H_S is an immorality. In particular, for distinct $i, j, k\in N$,

$$\mathsf{c}_H(ijk)=1 \quad \Leftrightarrow \quad \begin{cases} \text{either } (i,j),\ (i,k),\ (j,k) \text{ are all edges in } H, \\ \text{or, up to a permutation of } i,\ j,\ k,\ \text{one has } i\to k\leftarrow j \text{ in } H. \end{cases} \tag{17}$$

4.2. *The list of inequalities*

In this section we formulate our inequalities and show that none of them is superfluous. Note that one can give graphical *interpretation* to (most of) the inequalities; for details see A.3. The inequalities are classified in four groups. The *basic non-negativity inequalities* are as follows:

(b.1) $\forall i, j\in N$ distinct, $0\leqslant \mathsf{a}(i\to j)$;
(b.2) $\forall S\subseteq N$, $|S|=3,4$, $0\leqslant \mathsf{c}(S)$.

The *consistency inequalities* mainly relate the a-part of the vector with its c-part:

(c.1) $\forall i, j\in N$ distinct, $\mathsf{a}(i\to j)+\mathsf{a}(j\to i)\leqslant \mathsf{c}(ij)$;
(c.2) $\forall i, j\in N$ distinct, $\mathsf{c}(ij)\leqslant 1$;

(c.3) $\forall i,j,k \in N$ distinct, $2\cdot \mathsf{c}(ijk) \leqslant 2\cdot \mathsf{c}(ij) + \mathsf{a}(i\to k) + \mathsf{a}(j\to k)$;
(c.4) $\forall i,j,k \in N$ distinct, $\mathsf{a}(i\to j) + \mathsf{c}(jk) \leqslant 1 + \mathsf{c}(ijk) + \mathsf{a}(j\to k)$;
(c.5) $\forall i,j,k \in N$ distinct, $\mathsf{a}(i\to j) + \mathsf{c}(jk) + \mathsf{c}(ik) \leqslant 2 + \mathsf{a}(i\to k) + \mathsf{a}(k\to j)$.

The *extension inequalities* ensure the c-part to be determined by $\mathsf{c}(S)$ for $2 \leqslant |S| \leqslant 3$:

(e.1) $\forall S\subseteq N,\ |S|\geqslant 3,\ \sum_{i\in S} \mathsf{c}(S\setminus\{i\}) \leqslant 2 + (|S|-2)\cdot \mathsf{c}(S)$;
(e.2) $\forall S\subseteq N,\ |S|\geqslant 4,\ (|S|-1)\cdot \mathsf{c}(S) \leqslant \sum_{i\in S} \mathsf{c}(S\setminus\{i\})$.

Finally, the *acyclicity inequalities* only concern the c-part and ensure that the solution is the graph in the considered class. In fact, these are just the transformed cluster inequalities (12):

(a.1) $\forall S\subseteq N,\ |S|\geqslant 4,\ \sum_{T\subseteq S,\,|T|\geqslant 2} \mathsf{c}(T)\cdot(-1)^{|T|} \leqslant |S|-1$.

As concerns the number of inequalities:

- the number of the basic non-negativity constraints is polynomial in $n\equiv |N|$, namely $2\cdot\binom{n}{2}+\binom{n}{3}+\binom{n}{4}$;
- the number of the consistency inequalities is also polynomial, namely $2\cdot\binom{n}{2}+15\cdot\binom{n}{3}$;
- the number of the extension inequalities is exponential, namely $2\cdot 2^n-\binom{n}{3}-2\cdot\binom{n}{2}-2\cdot n-2$, but there is a fair hope that one can in practice reduce their number substantially (see Section 5.1);
- the number of the acyclicity inequalities is also exponential, namely $2^n-\binom{n}{3}-\binom{n}{2}-n-1$, and one cannot reduce it to a polynomial amount in general, but can possibly design a reasonable way of avoiding the use of all of them; see Section 5.1.

Now, we show that none of the above inequalities is a consequence of the others. For each inequality an example of a vector is given which does not fulfill that particular inequality but satisfies all the other inequalities from the list:

(b.1) take $N=\{i,j\}$ and $\mathsf{a}(i\to j)=-1$ while $\mathsf{a}(j\to i)=\mathsf{c}(ij)=0$;
(b.2) take $N=S$ and put $\mathsf{c}(S)=-1$ while $\mathsf{a}\equiv 0$ and $\mathsf{c}(T)=0$ for $T\subset S$;
(c.1) take $N=\{i,j\}$ and $\mathsf{c}(ij)=\mathsf{a}(i\to j)=\mathsf{a}(j\to i)=1$;
(c.2) take $N=\{i,j\}$ and $\mathsf{c}(ij)=2$ while $\mathsf{a}(i\to j)=\mathsf{a}(j\to i)=1$;
(c.3) take $N=\{i,j,k\}$ and $\mathsf{c}(ijk)=\mathsf{c}(ik)=\mathsf{c}(jk)=1$ while $\mathsf{c}(ij)=0$ and $\mathsf{a}\equiv 0$;
(c.4) take $N=\{i,j,k\}$ and $\mathsf{a}(i\to j)=\mathsf{c}(jk)=\mathsf{c}(ij)=1$ but $\mathsf{c}(ik)=\mathsf{c}(ijk)=0$ and a vanishes in other components;
(c.5) take $N=\{i,j,k\}$ and $\mathsf{a}(i\to j)=1$, $\mathsf{c}\equiv 1$ while a vanishes in other components;
(e.1) in case $|S|=3$ take $N=\{i,j,k\}$ and $\mathsf{c}(ij)=\mathsf{c}(ik)=\mathsf{c}(jk)=1$, while $\mathsf{c}(ijk)=0$ and $\mathsf{a}\equiv 0$; if $|S|\geqslant 4$ take $N=S$, choose distinct $i,j\in S$ and put $\mathsf{a}(i\to k)=\mathsf{a}(j\to k)=1$ for every $k\in S\setminus\{i,j\}$, a vanishing for other components, $\mathsf{c}(ij)=\mathsf{c}(S)=0$, while $\mathsf{c}(T)=1$ for $T\subset S$, $T\neq\{i,j\}$;
(e.2) take $N=S$ and $\mathsf{c}(S)=1$, while $\mathsf{c}(T)=0$ for $T\subset S$, $\mathsf{a}\equiv 0$;
(a.1) take $N=S$, consider a total order $i_1,\ldots,i_s$, $s=|S|$ of elements of S and put $\mathsf{c}(\{i_1,i_s\})=1$, $\mathsf{c}(\{i_r,i_{r+1}\})=1$ for $r=1,\ldots,s-1$, while c vanishes for remaining components and $\mathsf{a}\equiv 0$.

4.3. Main results

We show (in Appendices A–B) the following theoretical result.

Theorem 1. *Given a finite non-empty set N, $|N|\geqslant 2$, a vector* (a,c) *with integer components satisfies the inequalities* (b.1)–(b.2), (c.1)–(c.5), (e.1)–(e.2) *and* (a.1) *if and only if there exists a* (*uniquely determined*) *chain graph H over N without flags equivalent to an acyclic directed graph such that* $(\mathsf{a},\mathsf{c})=(\mathsf{a}_H,\mathsf{c}_H)$.

In fact, a strengthened result is derived in Appendices A–B stating that, if (a.1) is omitted, one still gets a code of a certain graph H.

Theorem 2. *Given a finite non-empty set N, $|N|\geqslant 2$, a vector* (a,c) *with integer components satisfies the inequalities* (b.1)–(b.2), (c.1)–(c.5) *and* (e.1)–(e.2) *if and only if there exists a* (*uniquely determined*) *3-acyclic hybrid graph H without flags such that* $(\mathsf{a},\mathsf{c})=(\mathsf{a}_H,\mathsf{c}_H)$.

Another comment is that the acyclicity inequalities (a.1) can be modified. Specifically, they can be replaced by a simplified version

(a.1*) $\forall S\subseteq N,\ |S|\geqslant 4,\ \sum_{T\subseteq S,\,|T|=2} \mathsf{c}(T) - \sum_{T\subseteq S,\,|T|=3} \mathsf{c}(T) \leqslant |S|-1$, which may appear to be easier to implement. However, this change formally leads to a different LP relaxation of the same polytope, which is the convex hull of the considered set of codes. Note that Lindner [17] called (a.1*) the *DAG inequalities* and included them in the LP relaxation of her polytope, which was different (from the polytope considered here). The proof of the claim that (a.1) can be replaced by (a.1*) can be found in a research report [27].

5. The whole procedure and complexity considerations

Recall that a pre-processing step should be the search space reduction procedure mentioned in Section 3.2. Its result is a cache of values $r^{\mathcal{Q}}_D(S)$ for $S \in \mathcal{T}$, where $\mathcal{T} \subseteq \{S \subseteq N\colon\ |S| \geqslant 2\}$ is a class of sets closed under subsets such that

$$\mathsf{c}_G(R) = 0 \quad \text{for any } R \notin \mathcal{T} \text{ and optimal graph } G.$$

The hope is there is a small upper bound $t \in \mathbb{N}$ on the cardinality of sets in $\mathcal{T}$. This is also the situation when we deal with the problem of *restricted learning* with a prescribed upper bound $t \in \mathbb{N}$ for the size of cliques in the moral graph of G.

5.1. First phase: getting the characteristic imset

Therefore, having such a class $\mathcal{T}$, by (9), the maximization of $\mathcal{Q}$ is equivalent to the task to maximize the function

$$\mathsf{c}_G \mapsto \sum_{S\subseteq N,\ |S|\geqslant 2} r^{\mathcal{Q}}_D(S)\cdot \mathsf{c}_G(S),$$

over the class of characteristic imsets $\mathsf{c} = \mathsf{c}_G$ of acyclic directed graphs G over N satisfying

$$\mathsf{c}(R) = 0 \quad \text{for } R \subseteq N,\ |R| \geqslant 2,\ R \notin \mathcal{T}. \tag{18}$$

Clearly, on the linear subspace specified by (18) this linear objective coincides with the function

$$\mathsf{c}_G \mapsto \sum_{S\in\mathcal{T}} r^{\mathcal{Q}}_D(S)\cdot \mathsf{c}_G(S),$$

which is fully determined by the values $r^{\mathcal{Q}}_D(S)$ for $S \in \mathcal{T}$. Thus, the first ILP problem to be solved is to maximize the function

$$(\mathsf{a},\mathsf{c}) \mapsto \sum_{S\in\mathcal{T}} r^{\mathcal{Q}}_D(S)\cdot \mathsf{c}(S) \tag{19}$$

over the domain of vectors with integral components specified by the inequalities from Section 4.2 and the equality constraint (18). By Theorem 1, this is equivalent to maximization of (19) over the codes $(\mathsf{a}_H, \mathsf{c}_H)$ of chain graphs without flags equivalent to acyclic directed graphs. The function (19) only depends on the characteristic imset c_H, because $\mathsf{c}_H = \mathsf{c}_G$ for some (equivalent) acyclic directed graph G. Therefore, the solution of (19) maximizes $\mathcal{Q}$ in the domain of acyclic directed graphs over N. A possible modification would be to replace (a.1) by (a.1*) in the list of inequalities from Section 4.2; see the comment concluding Section 4.3.

The effective length of considered vectors (a,c) is $|\mathcal{T}| + 2\cdot\binom{n}{2}$, which is $\mathcal{O}(|N|^t)$, where $t \geqslant 2$ is the cardinality limit for sets in $\mathcal{T}$. The number of non-negativity and consistency inequalities is polynomial in $|N|$. As concerns the extension inequalities (e.1)–(e.2), provided $|\mathcal{T}|$ is small, most of them can be omitted, as they involve zero values and are trivially valid. Indeed, it is enough to consider only the following inequalities:

(e.1⋆) $\forall S \subseteq N,\ |S| \geqslant 3,\ S \in \mathcal{T}^\star,\ \ \sum_{i\in S} \mathsf{c}(S\setminus\{i\}) \leqslant 2 + (|S|-2)\cdot \mathsf{c}(S)$,
(e.2⋆) $\forall S \subseteq N,\ |S| \geqslant 4,\ S \in \mathcal{T},\ \ (|S|-1)\cdot \mathsf{c}(S) \leqslant \sum_{i\in S} \mathsf{c}(S\setminus\{i\})$,

with $\mathcal{T}^\star = \{S \subseteq N\colon\ \exists i \in S\ \ S\setminus\{i\} \in \mathcal{T}\}$. Thus, the number of inequalities (e.1)–(e.2) is, in fact, reduced to at most $\mathcal{O}(|N|^{t+1})$ size.

The number of acyclicity inequalities (a.1), respectively (a.1*), cannot be reduced to a polynomial amount, even if $t = 2$. Realize that (a.1) for $S \subseteq N$, $|S| \geqslant 4$ has the meaning of a forbidden chordless cycle (either undirected or semi-directed) composed just of the nodes in S in the respective graph H (see Appendix A.3). One can always have a chordless cycle of arbitrary high length in a sparse graph, for which reason it is not possible, in general, to reduce the number of inequalities in (a.1) to a polynomial number in $|N|$.

However, one can apply the idea of *iterative constraint adding*. In the first iteration, the acyclicity inequalities are omitted. The solution of the respective ILP problem corresponds, by Theorem 2, to a graph H with possible semi-directed/undirected chordless cycles. One can identify minimal sets S of nodes in H forming those cycles. The corresponding acyclicity inequalities are incorporated in the current list of inequalities and a revised ILP problem is solved. This procedure is repeated until either a solution without those cycles is found or one runs out of memory.

Note that if there is $R \notin \mathcal{T}$ with $|R| = 2$ then one can prune even more, namely in the a-part of the vector. Since the goal is to force the validity of (c.1), provided that $R \equiv \{i,j\} \notin \mathcal{T}$, we can also omit the components $\mathsf{a}(i \to j)$ and

$\mathsf{a}(j \to i)$, assuming they vanish. Then the respective inequalities (b.1) and (c.4)–(c.5) can be either omitted or simplified – see Appendix C for details.

5.2. Second phase: reconstruction of the essential graph

Assume that the first ILP problem has been successfully solved and a solution (a, c) found, which is a vector code for a chain graph H without flags equivalent to an acyclic directed graph over N. The graph H need not be an essential graph, but we are already sure that the vector $\mathsf{c} = \mathsf{c}_G$ is the characteristic imset for an optimal acyclic directed graph G over N. The task is now to get the corresponding essential graph.

The idea is to fix c and search through all a' such that $(\mathsf{a}', \mathsf{c})$ satisfies the inequalities from Section 4.2. This, in fact, means to search through all codes of chain graphs without flags equivalent to the optimal acyclic directed graph G. By Lemma 1, the class $\mathcal{H}$ of these graphs has the largest graph, namely the essential graph G^* of the Markov equivalence class $\mathcal{G}$ of acyclic directed graphs containing G. The largest graph in $\mathcal{H}$ can be characterized as the graph with the minimal amount of arrowheads in $\mathcal{H}$. Thus, it minimizes the function

$$H \in \mathcal{H} \longmapsto \sum_{i,j \in N,\, i \neq j} \mathsf{a}_H(i \to j).$$

Therefore, the second ILP problem is to minimize

$$\mathsf{a} \longmapsto \sum_{i,j \in N,\, i \neq j} \mathsf{a}_H(i \to j) \tag{20}$$

within the domain specified by c and the inequalities from Section 4.2. Since c is now fixed, most of the inequalities can be omitted (as they are automatically valid) and the number of remaining ones is already polynomial in $|N|$. We only need the values $\mathsf{c}(S)$ for $2 \leqslant |S| \leqslant 3$ and use (b.1), (c.1) and (c.3)–(c.5).

Note that an alternative to this ILP reconstruction procedure is the graphical procedure for reconstructing the essential graph from the characteristic imset suggested by Hemmecke, Lindner and Studený [12, §4.2]. However, the advantage of the ILP reconstruction procedure is that we stay in the ILP frame and can utilize the ILP software for this purpose. We also believe that this ILP reconstruction is much more elegant from the mathematical point of view.

The ILP approach also offers some additional utility, which can be appreciated by users, in particular if BN structure learning is made for the purpose of application in some other area, for example, medicine or biology. In such scenarios the statistician co-operates with an expert from that area and offers him/her a suitable representative of the BN structure which has been learned (by the first ILP procedure). The expert may not be interested in the essential graph but would like to see an acyclic directed graph, which he/she may be able to interpret. There could be several such graphs and the task is to choose from the Markov equivalence class that graph which is preferred by the expert according to his/her taste. For example, the expert would like, if possible, to have an arrow $a \to b$ instead of the arrow $b \to a$ because he/she thinks the symptom a is more likely to influence the symptom b than conversely.

This can be solved easily in the ILP framework: one can offer a modified reconstruction procedure, which on basis of the optimal characteristic imset, finds a preferred acyclic directed graph from the respective Markov equivalence class $\mathcal{G}$. This is because, in the context of Lemma 1, the elements of $\mathcal{G}$ can be viewed as the elements in $\mathcal{H}$ with the maximal amount of arrowheads. Thus, the co-operating expert can ascribe different positive weights $w(i \to j)$ to possible arrowheads and, determine in that way his/her criterion for the choice of the graph from $\mathcal{G}$, which takes into account his/her preference of directions of arrows. Then our aim should be to maximize the function

$$\mathsf{a} \longmapsto \sum_{i,j \in N,\, i \neq j} \mathsf{a}_H(i \to j) \cdot w(i \to j).$$

5.3. Summary of the procedure

1. A pre-processing step is the pruning components of the characteristic imset mentioned in Section 3.2.
2. The *first ILP task* is to

$$\text{maximize the function} \quad (\mathsf{a}, \mathsf{c}) \mapsto \sum_{S \in \mathcal{T}} r_D^{\mathcal{Q}}(S) \cdot \mathsf{c}(S) \tag{21}$$

   over the integer vectors within the polyhedron specified by (18) and the inequalities from Section 4.2. By Theorem 1 we know that the c-parts of these vectors are nothing but the characteristic imsets for acyclic directed graphs over N. In Section 6 we describe two methods we used to solve this ILP task.
3. Provided that we succeed to solve the first ILP task we obtain as the solution a vector $(\mathsf{a}_H, \mathsf{c}_H)$, where H is a chain graph over N, which has no flag and is equivalent to an acyclic directed graph. To find the corresponding essential graph we establish the *second ILP task*, namely to

$$\text{minimize the function} \quad (\mathsf{a},\mathsf{c}) \mapsto \sum_{i,j\in N,\, i\neq j} \mathsf{a}(i\to j) \tag{22}$$

with fixed $\mathsf{c} = \mathsf{c}_H$, as described in Section 5.2. The solution should be the simultaneous code of the respective optimal essential graph G^* and the characteristic imset.

6. Two methods to solve the first ILP problem

In our computational experiments, whose aim was just to verify feasibility of our approach, we developed two methods to solve the first ILP task (21) and compared them. The methods are described in this section, after some preliminaries.

6.1. The original aim

Our original intention from the PGM'12 conference paper by Studený [25], which was the basis of the present paper, was as follows:

- As the first iteration, we relax (21) and consider only the inequalities (b.1⋆)–(b.2⋆), (c.1⋆)–(c.5⋆), (e.1⋆)–(e.2⋆) from Appendix C. The solution of the starting ILP problem should be the code $(\mathsf{a}_H, \mathsf{c}_H)$ of a hybrid 3-acyclic graph H without flags, by Theorem 2.
- We check whether H is a chain graph equivalent to an acyclic directed graph by searching for chordless semi-directed/undirected cycles in H.
  - If there is no such a cycle, the graph H encodes the solution.
  - If not, we find all such chordless cycles in H, take the respective acyclicity inequalities (a.1), incorporate them in the considered system of inequalities and run a new ILP problem, that is, come to the previous step with an extended class of inequalities.
- Provided we do not get stuck with memory overload, the solution will be a vector $(\mathsf{a}_H, \mathsf{c}_H)$, where H is a chain graph over N, which has no flag and is equivalent to an acyclic directed graph.

However, this particular idea was not implemented, but led to the current approach.

6.2. The idea of a sub-ILP problem to find violated constraints

To avoid the graphical procedure of searching for chordless semi-directed/undirected cycles and remain in the ILP frame, we came to the idea to find violated cluster inequalities by solving a sub-ILP problem. This idea was inspired by a similar sub-ILP approach applied by Cussens [7].

Assume we have found a solution z to the inequalities (b.1⋆)–(b.2⋆), (c.1⋆)–(c.5⋆) and (e.1⋆)–(e.2⋆). Thus, z may be fractional and may violate the cluster inequalities (a.1). The goal here is to form an ILP problem to find the most violated cluster inequalities (a.1) with respect to z. For the sub-ILP task we consider variables $\mathsf{d}(i)$ for $i \in N$ and $\mathsf{d}(T)$ for all $T \in \mathcal{T}$. We include the following inequalities:

(f.1) $\mathsf{d}(i) \geqslant 0$ for $i \in N$,
(f.2) $\mathsf{d}(i) \leqslant 1$ for $i \in N$,
(f.3) $\mathsf{d}(T) \geqslant 0$ for $T \in \mathcal{T}$,
(g.1) $\mathsf{d}(T) \leqslant \mathsf{d}(i)$ for all $T \in \mathcal{T}$ and $i \in T$,
(g.2) $1 - |T| + \sum_{i\in T} \mathsf{d}(i) \leqslant \mathsf{d}(T)$ for all $T \in \mathcal{T}$,
(g.3) $\sum_{i\in N} \mathsf{d}(i) \geqslant 4$.

Now, consider the following ILP task, namely to

$$\text{maximize the function} \quad \mathsf{d} \mapsto \sum_{T\in\mathcal{T}} \mathsf{z}(T)\cdot \mathsf{d}(T)\cdot (-1)^{|T|} - \sum_{i\in N} \mathsf{d}(i) \tag{23}$$

subject to the constraints (f.1)–(f.3) and (g.1)–(g.3). The point is that any integer solution satisfying those constraints is a zero-one code of a subset $S \subseteq N$, such that $|S| \geqslant 4$, $\mathsf{d}(i) = 1$ if and only if $i \in S$, and $\mathsf{d}(T) = 1$ if and only if $T \subseteq S$.

Lemma 4. *Given a real vector z with components $\mathsf{z}(T)$ for $T \in \mathcal{T}$, all sets $S \subseteq N$, $|S| \geqslant 4$ such that the inequality* (a.1) *for S is violated by the zero extension c of z correspond to integral vectors in the polyhedron given by* (f.1)–(f.3) *and* (g.1)–(g.3) *for which the objective in* (23) *strictly exceeds -1.*

Proof. The inequalities (g.1) and (f.2) imply $\mathsf{d}(T) \leqslant 1$ for any $T \in \mathcal{T}$. Thus, any integral vector d in the considered polyhedron is a zero-one vector. Given such d put $S = \{i \in N\colon\ \mathsf{d}(i) = 1\}$. The inequality (g.3) implies $|S| \geqslant 4$. The condition (g.1) implies that if $T \in \mathcal{T}$ satisfies $\mathsf{d}(T) = 1$ then $T \subseteq S$ and the converse implication follows from (g.2). Hence, we write

$$\sum_{T\in\mathcal{T}} \mathsf{z}(T)\cdot\mathsf{d}(T)\cdot(-1)^{|T|} - \sum_{i\in N}\mathsf{d}(i) = \sum_{T\in\mathcal{T},\,T\subseteq S} \mathsf{z}(T)\cdot(-1)^{|T|} - |S| \overset{(18)}{=} \sum_{T\subseteq S,\,|T|\geqslant 2} \mathsf{c}(T)\cdot(-1)^{|T|} - |S|,$$

which exceeds -1 if and only if $\sum_{T\subseteq S,\,|T|\geqslant 2}\mathsf{c}(T)\cdot(-1)^{|T|} > |S|-1$; cf. (a.1) on p. 1053. □

Thus, by solving the ILP task (23) one can find all violated (a.1) inequalities for the zero extension c of z. In case z is an integer vector, it finds the desired chordless semi-directed/undirected cycles in the graph H with $\mathsf{c}=\mathsf{c}_H$. Provided the aim is to find at least one cut, for example the maximally violated (a.1) inequality, the size of the sub-ILP task (23) can be reduced by only considering variables $\mathsf{d}(T)$ corresponding to non-zero values in the current solution z. That is, let $Q := \bigcup_{T\in\mathcal{T},\mathsf{z}(T)>0} T$, and consider only creating variables $\mathsf{d}(i)$ for $i\in Q$, and $\mathsf{d}(T)$ for $T\in\mathcal{T}$ with $\mathsf{z}(T)>0$. This can considerably speed up the computation when $|Q|$ small.

6.3. *The first method*

The first method performs row-generation, also called lazy constraints, on the acyclic inequalities (a.1).

1. As the first iteration, we solve the LP problem (21) under constraints (b.1⋆)–(b.2⋆), (c.1⋆)–(c.5⋆) and (e.1⋆)–(e.2⋆).
2. The solution z of the starting LP problem is checked for violated (a.1) inequalities by the sub-ILP method described in Section 6.2. If there are violated cluster inequalities then they are incorporated to the current list of constraints and a new LP problem is solved.
3. Provided we find a solution z satisfying all cluster inequalities two cases may occur:
   - z is an integer vector and we stop as z is the final solution,
   - if z is fractional we solve an ILP problem (21) with the current list of inequalities.
4. If we find the solution z' of that ILP problem, it is checked for violated (a.1) inequalities by the sub-ILP method described in Section 6.2. Then two cases may occur:
   - if z' satisfies all (a.1) inequalities, it is the final solution,
   - if there are violated cluster inequalities for z' then they are incorporated to the current list of constraints and a new ILP problem is solved.

6.4. *The second method*

The second approach is based on augmenting the branch-and-cut ILP method [1,14,16] in the ILOG CPLEX software [31]. The idea is to call the CPLEX ILP solver only once to solve the ILP problem (21) under constraints (b.1⋆)–(b.2⋆), (c.1⋆)–(c.5⋆) and (e.1⋆)–(e.2⋆), but add the (a.1) inequalities as user cuts (cut sub-LP solutions during branch-and-bound), and respectively to add (a.1) inequalities as lazy constraints. The ILP solver is augmented in two ways:

- The ILP solver is augmented such that during the branch-and-cut process whenever a sub-LP is considered at a branch node, violated cluster inequalities (a.1) are added as global cuts.
- The ILP solver is augmented such that when a feasible integer solution is found, violated cluster inequalities (a.1) are added as lazy constraints.

7. Software and experiments

In this section we give some details on software and experiments which explored the validity of the approach described in the paper and the two methods mentioned in Section 6.3 and Section 6.4, with some sub-variants.

7.1. *Software developed*

In order to explore the feasibility of the approach, software was developed in C++ which used the IBM ILOG CPLEX Optimizer via the C API. Firstly, although the inequalities (c.1)–(c.5) are polynomial in size we can exploit the non-pruned sets in $\mathcal{T}$ to further reduce the size of the inequalities; see Appendix C. Moreover, in (e.1⋆) we need only add the constraint if on the left-hand side there are more than two terms in $\mathcal{T}$.

7.2. *The first series of experiments*

The first series of computational experiments was performed shortly before the original submission. Selected data sets were taken from GOBNILP web page [32], which most appeared in Cussen's 2011 paper [7]. These data sets had an artificial limit of two or three set on the size of parent sets of nodes. Note, that the time to compute the class of sets $\mathcal{T}$ and the revised data vector $r_D^{\mathcal{Q}}$ from a set of non-pruned local scores $q_D(i|B)$ for all these data sets (see Section 3.1) took a matter of seconds and is not reported.

Table 2
Shown are the number of variables in the problem (Num. vars.), the number of observations in the database (Num. obs.), the limit on parent set size for local scoring (Parent size), the number non-pruned parent sets (|*PaSet*|), the number of non-pruned components of characteristic imsets (|*cimset*|), the number of component of the a-vector (|*a*|) and the remaining columns are the numbers of inequalities of the type listed and the total number of inequalities.

Name	Num. vars.	Num. obs.	Parent size	$\lvert PaSet\rvert$	$\lvert cimset\rvert$	$\lvert a\rvert$	$\lvert c.1\star\rvert$	$\lvert c.3\star\rvert$	$\lvert c.4\star\rvert$	$\lvert c.5\star\rvert$	$\lvert e.1\star\rvert$	$\lvert e.2\star\rvert$	*Total # Ineq.*
Mildew	35	100	3	3520	4917	1064	532	6273	32 294	30 576	12 032	2294	84001
Mildew	35	1000	3	161	85	130	65	60	776	342	58	0	1301
Mildew	35	10 000	3	463	303	230	165	390	4366	3006	586	8	8521
Pigs	441	100	2	2692	1728	2526	1263	1395	24 790	6288	1059	0	34 795
Pigs	441	1000	2	15 847	14 095	13 506	6753	22 026	777 792	420 834	–	0	–
Pigs	441	10 000	2	304 219	249 257	68 970	34 485	644 316	15 168 416	–	–	–	–
Water	32	100	3	482	344	290	145	390	3514	2580	557	69	7255
Water	32	1000	3	573	281	352	176	252	4486	3108	557	21	8600
Water	32	10 000	3	961	676	546	273	927	9340	6744	1547	94	18 925
alarm	37	100	3	907	1462	830	415	2271	18 678	13 026	3419	290	38 099
alarm	37	1000	3	1928	1865	756	378	2796	16 226	12 150	4603	555	36 708
alarm	37	10 000	3	6473	5797	1010	505	7116	28 414	24 714	20 242	2920	83 911
asia	8	100	8	41	29	36	18	33	146	114	23	0	334
asia	8	1000	8	112	85	54	27	138	312	300	105	30	912
asia	8	10 000	8	169	99	54	27	132	312	300	111	39	921
hailfinder	56	100	3	244	183	194	97	201	1020	708	141	19	2186
hailfinder	56	1000	3	761	513	430	215	684	4022	2760	788	70	8539
hailfinder	56	10 000	3	3768	3182	1078	539	4029	23 352	15 912	9253	1300	54 385
insurance	27	100	3	279	265	298	149	315	3398	2010	372	11	6255
insurance	27	1000	3	774	687	470	235	1155	8350	6570	1790	67	18 167
insurance	27	10 000	3	3652	3282	688	344	4740	16 876	16 578	11 249	1358	51 145

Table 3
Shown are the running times in seconds for Method 1 and Method 2 including the number of LP loops and ILP loops done by Method 1 and the number of inequalities added for Method 2. Data sets with † symbol did not complete after 4 h. The third column reports on the time to find the essential graph then. Lastly the running times for GOBNILP taken from http://www.cs.york.ac.uk/aig/sw/gobnilp/.

Name	Method 1 time	Number LP loops	Number ILP loops	Method 2 time	Number cycle cuts	Time essential graph	GOBNILP time
Mildew100	14 400†	19	1	14 400†	–	–	2
Mildew1000	1	15	1	1	18	5	1
Mildew10 000	9	15	3	9	59	1	3
Pigs100	333	81	1	308	272	8	100
Pigs1000	14 400†	36	–	14 400†	–	–	2226
Pigs10 000	14 400†	–	–	14 400†	–	–	1025 (3.35%)
Water100	5694	40	1	14 400†	–	1	5
Water1000	35	48	1	27	530	1	6
Water10 000	14 400†	137	–	14 400†	–	–	26
alarm100	14400†	65	–	14 400†	–	–	3
alarm1000	14 400†	81	1	14 400†	–	–	5
alarm10 000	14 400†	52	–	14 400†	–	–	836
asia100	1	2	1	1	1	1	1
asia1000	2	8	1	2	20	1	1
asia10 000	3	8	1	93	68	1	1
hailfinder100	8	12	3	28	104	1	1
hailfinder1000	162	49	2	1214	789	1	9
hailfinder10 000	14 400†	166	1	14 400†	–	–	92
insurance100	18	31	1	7	182	1	1
insurance1000	61	38	1	49	278	1	4
insurance10 000	14 400†	121	1	14 400†	–	–	10

Table 2 contains 21 ILP problems with their node set cardinality, the limit on parent set size, number of non-pruned parent set variables $\eta(i|B)$, the number of non-pruned components of characteristic imsets ($|\mathcal{T}|$) and the number of inequalities produced for (c.1⋆)–(c.5⋆) and (e.1⋆)–(e.2⋆). Notice that often the number of components of c-vector is smaller than the number of non-pruned parent sets but, combined with the a-components, the length of the (a, c)-vector exceeds the length of the corresponding η-vector for these examples. We conjectured that this is mostly due to the low parent set size limit of three and with a higher parent set size limit the (a, c)-vector should become shorter than the η-vector. However, our later simulation experiments reported in Section 7.3 indicate that this phenomenon seems to be a paradoxical consequence of the pruning procedure from Section 3.2. Also the number of inequalities we generated appears to be quite high. In some cases the full set of inequalities could not be constructed (see dashes in Table 2).

The software implementation of the sub-ILP task from Section 6.2 attempts to solve the problem by only branching on the variables $\mathsf{d}(i)$ for $i \in N$, and always trying the $\mathsf{d}(i) = 1$ branch first. Moreover, an option was added to disable internal CPLEX cuts when solving this sub-ILP, which for many cases led to a significant improvement. When we solved the sub-ILP we also asked the CPLEX solver to return all known integral solutions to the problem. In this way we can add multiple violated cluster inequalities. An additional option was explored where the cluster sub-ILP returns the first solution which violates the cluster inequality (and not the most violated cluster inequality). In most cases this improves the sub-ILP speed, but overall degrades to performance of learning the BN structure as more cuts are needed. Additionally a constraint (with a small $\epsilon > 0$) can be added to sub-ILP such that the only solutions are indeed violated cluster inequalities. In practice the time to learn a BN structure using our approach is dominated by:

1. The time to solve linear relaxations of the problem.
2. The time to find violated cluster inequalities.

The former is greatly effected by the number of inequalities contained in the system. The latter is dominated by the number of variables in the original learning problem.

We also explored adding some or all of constraints (c.1⋆)–(c.5⋆) and (e.1⋆)–(e.2⋆) as lazy constraints to CPLEX. Although this did speed up the time to solve any sub-LP problem, it almost always lead to poorer overall performance. We also explored using inequalities (a.1⋆) (see Section 4.3) instead of (a.1), which also led to poorer performance, usually requiring many more cuts and many more iterations of Method 1, or nodes explored while performing branch-and-cut in Method 2. For some small problems, it was possible to initially add all cluster inequalities (a.1) for all $T \in \mathcal{T}$ at the beginning. In these cases, it was just as fast as applying the sub-ILP method to add cluster inequalities.

In Table 3 we give running times using Method 1 and Method 2. Included in the table is the number of seconds to solve the main ILP optimization problem, the number of LP loops, number of ILP loops, the number of cycle cuts added in Method 2 and lastly the time to compute the essential graph and the running times of GOBNILP. In most cases Method 1 outperformed Method 2. What is interesting is the few number of ILP loops required for many of the problems. However, the comparison with GOBNILP times shows that our 1st ILP solving method (Section 5.1) is not competitive from the point of view of running times. We pondered on the reasons for the computational speed differences and came to the conclusion

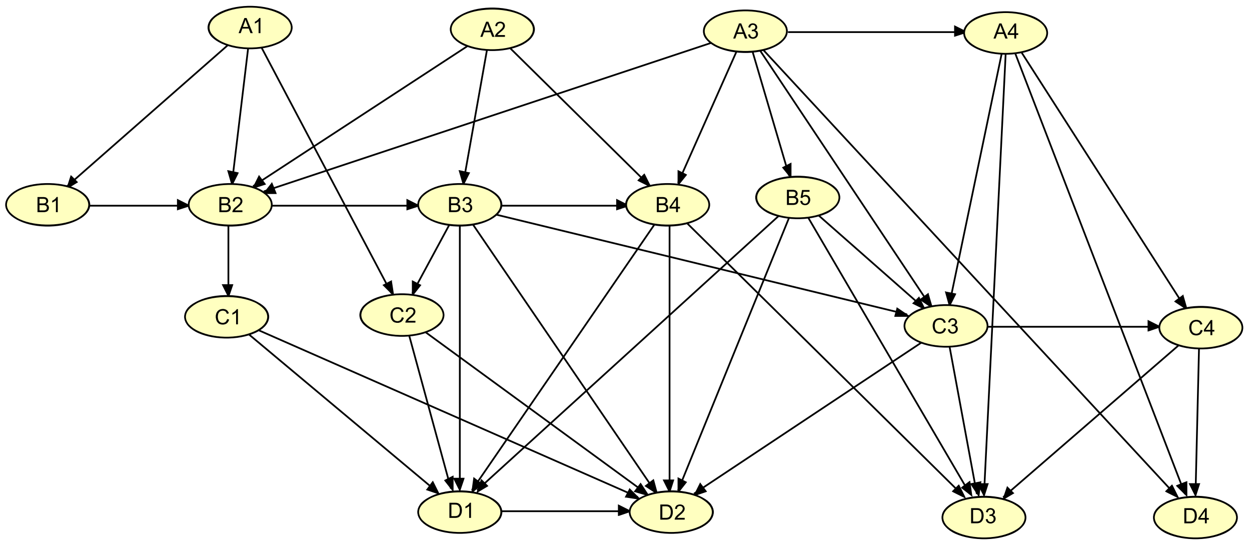


Fig. 5. An acyclic directed graph used for our simulation experiments.

that this is partly influenced by our longer vectors but the main reason is probably that the number of non-acyclicity inequalities we have to deal with is too high. Indeed, a detailed comparison of Tables 2 and 3 reveals that we were not able to finish some computations when the number of inequalities exceeds 9000. On the other hand, our computational experiments confirmed the hope that the 2nd ILP method (Section 5.2) is fast at finding the essential graph. Note that to choose some acyclic directed graph from the equivalence class takes essentially the same time.

7.3. The second series of experiments: simulated network

Before the revised version submission we performed the second series of computational experiments. One of the goals was to explore the influence of (the limit on) the parent set size, the second goal was to confirm/refuse our opinion that the number of inequalities practically handled by the (I)LP solver is the most influential factor to the computational efficiency.

We decided to test it by means of a simulated Bayesian network. More precisely, we generated an acyclic directed graph over 17 nodes shown in Fig. 5, which has the maximal parent size 7. Three (joint binary) Markovian probability distributions with respect to the graph in Fig. 5 were created and databases with 10 000 data points were simulated from those distributions. The database denoted by `bn17` came from a distribution whose conditional probability tables were made manually, the database `bn17b` was based on random sampling of conditional probability tables, as well as `bn17-noisyor`, where, the conditional probability tables were additionally required to satisfy certain special functional dependencies. To test the influence of the database length we considered shortening to 100 and 1000 data points, as well as considering the full database of 10 000 points.

We then applied our software to compute and prune the local scores both for BDeu criterion with $\alpha^* = 1$ and the BIC criterion. We were interested in whether the characteristic imset c generated from the pruned η-vector by the procedure mentioned in the end of Section 3.2 becomes shorter in these cases. This appeared not to be the case. Recall that the components of c are indexed by the class of sets $\mathcal{T} = \{S \subseteq N\colon\ |S| \geqslant 2,\ \exists$ non-pruned $q_D(i|B)$ with $S \subseteq \{i\} \cup B\}$, which is closed under subsets. We realized that the requirement on $\mathcal{T}$ being closed under subsets is unnecessarily restrictive, causing that one cannot utilize fully the result of the search space reduction procedure in our framework. Therefore, we explored what is the maximal possible reduction of the length of the c-vector in the considered cases, that is, we considered the class of sets $\mathcal{T}' = \{S \subseteq N\colon\ |S| \geqslant 2,\ \exists$ non-pruned $q_D(i|B)$ with $S \subseteq \{i\} \cup B$ and $i \in S\}$, which need not be closed under subsets. In this instance, we finally got slightly shorter vectors than the η-vector.

The results are reporter in Table 4, where we give the length of the pruned η-vector (η), the length of the maximally pruned characteristic imset (mset) and the length of the c-vectors for the class $\mathcal{T}$ closed under subsets (cimset). In the BDE case, when the sample size is small we can see a large increase in the length of characteristic imsets, however when the sample size is large the length of characteristic imsets can be smaller than the length of η. The length of the vectors nicely reflects what is the artificial parent size limit. We also observe the stabilization of the length of the vectors for longest databases when the maximal parent set size 7 is reached.

Tables 5, 6 and 7 in Appendix D gather the results of the simulation experiments. Each table is devoted to one of the tested databases and gives both the numbers of inequalities required for our ILP approach, the number added cluster inequalities, the running times of Method 2 and the running times of GOBNILP. The subcases in each table correspond to different database length and quality criterion, that is, BDE with $\alpha^* = 1$ and BIC. A maximum running time of 1 h was

Table 4
For both scoring criteria (namely BDE with $\alpha^* = 1$ and BIC), each considered data sets, the numbers of samples and the parent set size limit (plim) up to 10 given are the number of non-pruned pairs $(i|B)$ (η), number of minimally required characteristic imsets (mset), and the number of characteristic imsets.

`bn17` (BDE)

plim	100			1000			10 000		
	η	mset	cimset	η	mset	cimset	η	mset	cimset
2	48	32	34	141	87	99	371	216	229
3	51	45	56	164	113	137	784	578	682
4	52	55	74	166	119	147	1209	1081	1341
5	52	55	74	166	119	147	1400	1347	1722
6	53	113	183	166	119	147	1447	1423	1831
7	56	398	725	166	119	147	1454	1434	1847
8	89	4432	7515	166	119	147	1454	1434	1847
9	133	11 688	18 767	166	119	147	1454	1434	1847
10	176	25 300	36 779	166	119	147	1454	1434	1847

`bn17` (BIC)

plim	100			1000			10 000		
	η	mset	cimset	η	mset	cimset	η	mset	cimset
2	58	43	45	170	112	130	388	224	236
3	58	43	45	214	164	204	871	641	742
4	58	43	45	220	179	226	1438	1275	1595
5	58	43	45	220	179	226	1779	1750	2256
6	58	43	45	220	179	226	1879	1927	2529
7	58	43	45	220	179	226	1894	1956	2574
8	58	43	45	220	179	226	1894	1956	2574
9	58	43	45	220	179	226	1894	1956	2574
10	58	43	45	220	179	226	1894	1956	2574

`bn17b` (BDE)

plim	100			1000			10 000		
	η	mset	cimset	η	mset	cimset	η	mset	cimset
2	58	29	31	151	87	107	425	255	288
3	62	36	43	185	132	180	893	653	777
4	63	44	58	188	137	188	1270	1062	1347
5	63	44	58	188	137	188	1387	1212	1560
6	63	44	58	188	137	188	1405	1245	1611
7	67	475	862	188	137	188	1407	1257	1619
8	78	2035	3685	188	137	188	1407	1257	1619
9	131	13 320	20 488	188	137	188	1407	1257	1619
10	476	46 559	58 033	188	137	188	1407	1257	1619

`bn17b` (BIC)

plim	100			1000			10 000		
	η	mset	cimset	η	mset	cimset	η	mset	cimset
2	65	35	38	167	98	119	456	276	305
3	67	38	44	219	165	217	1004	727	846
4	67	38	44	231	199	276	1551	1308	1631
5	67	38	44	231	199	276	1806	1678	2155
6	67	38	44	231	199	276	1860	1766	2286
7	67	38	44	231	199	276	1862	1767	2287
8	67	38	44	231	199	276	1862	1767	2287
9	67	38	44	231	199	276	1862	1767	2287
10	67	38	44	231	199	276	1862	1767	2287

`bn17-noisyor` (BDE)

plim	100			1000			10 000		
	η	mset	cimset	η	mset	cimset	η	mset	cimset
2	427	292	329	1020	585	605	1593	728	732
3	599	501	628	2141	1429	1555	5535	2644	2664
4	642	635	833	2569	1918	2175	11 040	6183	6354
5	647	693	928	2601	1974	2256	14 120	9204	9813
6	654	914	1284	2602	1982	2266	14 609	10 009	10 838
7	668	1427	2215	2603	2014	2318	14 625	10 057	10 903
8	748	2099	3492	2603	2014	2318	14 625	10 057	10 903
9	748	2099	3492	2603	2014	2318	14 625	10 057	10 903
10	748	2099	3492	2603	2014	2318	14 625	10 057	10 903

`bn17-noisyor` (BIC)

plim	100			1000			10 000		
	η	mset	cimset	η	mset	cimset	η	mset	cimset
2	379	256	290	997	585	604	1601	731	734
3	398	287	332	1872	1277	1380	5593	2692	2705
4	398	287	332	1984	1398	1552	10 778	6118	6262
5	398	287	332	1984	1398	1552	12 734	8219	8638
6	398	287	332	1984	1398	1552	12 783	8301	8733
7	398	287	332	1984	1398	1552	12 783	8301	8733
8	398	287	332	1984	1398	1552	12 783	8301	8733
9	398	287	332	1984	1398	1552	12 783	8301	8733
10	398	287	332	1984	1398	1552	12 783	8301	8733

set and when available the gap to the optimal is given. The simulation experiments again confirmed that this particular ILP approach of ours is not competitive from the point of view of computational times when compared with GOBNILP [32].

However, the simulation certainly led to several useful observations, significant for future research. We believe that the main reason for inefficiency was revealed. We observe that the influence of the parent set size limit is mediated, through the number of inequalities. The simulation experiments confirmed our expectation from the first series of experiments that the number of inequalities handled is a crucial factor for computational efficiency. We were not able to finish the computation when this number was too high. This happened for the most of databases with 10 000 points but also for BDE and short databases with 100 points. The exception was the network `bn17` with BDE score and database length 10 000 and the parent size limit above 7 (see Table 5). The solution found indeed corresponds to the "generating" graph from Fig. 5. An interesting point here is that for smaller parent size limit 6, we failed to finish the computation.

Another interesting observation is that the database `bn17-noisyor`, the one with special functional dependency in generating conditional probability tables, was difficult to handle in comparison with the other two databases (see Table 7 in Appendix D). In this case we solved our ILP problems only for short databases and BIC criterion. Of course, due to the short database length, the solution was not the "generating" graph from Fig. 5. In the case of the `bn17-noisyor` instances, even GOBNILP failed – as did our method – on the sub-instance of 10 000 observations and BDE scoring for a parent set limit greater than three.

8. Conclusions

The theoretical advantage of the presented approach (in comparison with the other ILP approaches) is that the solution is a unique BN representative and the vector representatives are more compressed. The augmentation does not kill this comparative advantage because the extended characteristic imsets are still more than $\frac{|N|}{3}$-times shorter than the straightforward graph codes. However, as the computational experiments showed, the situation is reversed as the result of usual pre-processing procedure, namely the search space reduction described in Section 3.2. This procedure seems to be particularly advantageous for the straightforward graph codes because it allows one to better exploit the results of the pruning. Our experiments confirmed that typically our pruned c-vector (as suggested in Section 3.2) is longer the respective pruned η-vector. On the other hand, additional experiments (see Section 7.3) suggested that if one considers maximally shortened pruned characteristic imset then one can obtain a shorter vector than the pruned η-vector in practice, as well.

The contribution of the paper is mainly theoretical, motivated by the aim to develop a consistent ILP approach to learning BN structure based on characteristic imsets. We reformulated the BN structure learning task as a special ILP optimization problem and derived the formula for the respective objective (see Section 3.1). We have also extended the characteristic imset, our unique vector BN structure representative, by additional components with the aim to encode certain relevant graphs (Section 4.1). Finally, we came with polyhedral characterization of the domain of the respective ILP optimization problem, which includes as an option the case of graphs with possible cycles (see Section 4.2 and Section 4.3). Our approach is particularly suitable for getting the *essential graph*, the standard unique graphical BN structure representative, as the solution of a simple additional ILP optimization problem. Note that this second ILP problem is, in fact, independent of the main ILP problem, that is, of the task of finding optimal BN structure and be utilized as an extension to other ILP methods for BN structure learning. Thus, the main practical benefit of the presented approach is that it allows one to get directly the essential graph as the result of solving a secondary ILP problem, that is, to avoid in this way the need for additional graph-reconstruction procedure.

Motivated by the aim at practical solving of our main ILP optimization problem by the method of iterative constraint adding (see Section 6, specifically Section 6.2) we performed two series of computational experiments. The goal of the experiments was just to verify the ability to learn the BN structure, not to compete in running times with other, more developed, approaches. We fairly admit that, in our experiments, we have not succeeded to get better running times than reported by Cussens and Bartlett [32]. The experiments, mainly the second series, seem to confirm our opinion that the main reason for the comparative inefficiency of our approach is the large number of our non-acyclicity inequalities to handle. Although the number of these inequalities is theoretically polynomial in $|N|$, namely $\mathcal{O}(|N|^3)$, in the tested cases, it was much higher in comparison with simple linear number of non-negativity and equality constraints for the η-vector. Note that the exponential class of acyclicity cluster inequalities is the same for both approaches.

On the other hand, the experiments confirmed that the secondary ILP procedure to find the essential graph on basis of the characteristic imset (see Section 5.2) is fast. Thus, provided we find an optimal acyclic directed graph by some other method, the LP relaxation proposed in this paper can be used to get by a simple ILP procedure the respective essential graph or an equivalent acyclic directed graph with a prescribed preference of arrow directions.

One of the questions raised in Conclusions of the original PGM'12 conference paper [25] was whether, for the efficiency of learning, a smaller number of inequalities is better than a tighter LP relaxation. It seems that the experiments gave us a clear answer that to get an efficient ILP BN learning procedure one surely has to avoid large number of inequalities or come with a clever way to iteratively add them. Therefore, our future theoretical effort will be directed to finding an LP relaxation for characteristic imsets with a small number of inequalities. We hope there is a way of avoiding the exponential number of acyclicity inequalities, perhaps even by combining our approach with other ILP learning approaches. For example, Cussens et al. [8] recently came in the context of pedigree learning with a method of ruling out cycles based on $|N|$ additional integral components to the η-vector and $\mathcal{O}(|N|^2)$ inequalities.

An alternative research direction could be to apply the idea of iterative constraint adding to the collection of our non-acyclicity inequalities and find a way to handle the constraints by a specialized user cut detection algorithm.

Acknowledgements

We thank the reviewers for their comments and suggestions which helped to improve the quality and readability of the paper. We are grateful to Raymond Hemmecke for giving us some inspiration and to James Cussens for giving us useful advice concerning the formulation of the second method to solve our main ILP problem via cuts and lazy constraints, and the sub-ILP idea to find violated cluster inequalities. Moreover, we also thank James for providing us with working latest GOBNILP version we utilized for comparison in the second series of experiments and Jirka Vomlel for generating the simulated databases, on which these experiments were based. The research of Milan Studený has been supported by the grant GAČR n. 13-20012S.

Appendix A. Necessity of the inequalities

In this section, we show that the inequalities from Section 4.2 are necessarily valid for the vector codes from Definition 5. This allows us to interpret the inequalities in graphical terms.

A.1. Auxiliary observations

To verify the necessity of the extension inequalities we will need the next technical lemma.

Lemma 5. *Let H be a hybrid graph over N and $S \subseteq N$, $|S| \geqslant 3$.*

(a) *If H_S has a super-terminal component, then for at least $|S|-1$ subsets $T \subset S$ with $|T| = |S|-1$, the graph H_T has a super-terminal component.*
(b) *Provided H is 3-acyclic, if, for at least 3 subsets $T \subset S$ with $|T| = |S|-1$ the graph H_T has a super-terminal component, then H_S has a super-terminal component.*

Proof. (a) If H_S has a super-terminal component K (see Definition 4), then for each $j \in S \setminus K$, the graph $H_{S\setminus\{j\}}$ has K as a super-terminal component. If $|K| \geqslant 2$, then, for every $i \in K$, $H_{S\setminus\{i\}}$ has $K \setminus \{i\}$ as a super-terminal component.

(b) Assume that $k, m, \ell \in S$ are such that, for each $i \in \{k, m, \ell\}$, $H_{S\setminus\{i\}}$ has a super-terminal component. Since $H_{\{k,m,\ell\}}$ has no directed cycle, it has a node with no arrow directed towards it in $H_{\{k,m,\ell\}}$. Thus, without loss of generality assume that k has the property that $\neg(m \to k \text{ in } H)$ and $\neg(\ell \to k \text{ in } H)$. Now,

$$\text{let } K \text{ be a super-terminal component in } H_{S\setminus\{k\}}.$$

If $K \cap \{m, \ell\} \neq \emptyset$ then we can assume without loss of generality that $\ell \in K$. Thus, we distinguish three cases:

$\boxed{\text{A.}}$ $K \cap \{m, \ell\} = \emptyset$,

$\boxed{\text{B.}}$ $K = \{\ell\}$,

$\boxed{\text{C.}}$ $\ell \in K$ and $|K| \geqslant 2$.

In case $\boxed{\text{A.}}$ we have $K \subseteq S \setminus \{k, m, \ell\}$ and K is a super-terminal component in $H_{S\setminus\{k,m,\ell\}}$. Let L be a super-terminal component in $H_{S\setminus\{\ell\}}$. Then, because there exists $i \in K \subseteq S \setminus \{k, m, \ell\}$ with $m \to i$ in H, necessarily $m \notin L$. Also, $k \notin L$, as otherwise, for $m \notin L$, we observe $m \to k$ in H, which contradicts the choice of k. Hence, $L \subseteq S \setminus \{k, m, \ell\}$ and L is a super-terminal component in $H_{S\setminus\{k,m,\ell\}}$. By its uniqueness, we get $L = K$. This implies that K is a super-terminal component in H_S.

In case $\boxed{\text{B.}}$, that is $K = \{\ell\}$, let M be a super-terminal component in $H_{S\setminus\{m\}}$. Then, because for every $j \in S \setminus \{k, \ell\}$ one has $j \to \ell$ in H, necessarily $M \subseteq \{k, \ell\}$. Necessarily $\ell \in M$, for otherwise $M = \{k\}$ and $\ell \to k$ in H, which contradicts the choice of k. Then we have two options:

- $M = \{\ell\}$, in which case $k \to \ell$ in H and $K = M = \{\ell\}$ is a super-terminal component in H_S,
- or $M = \{k, \ell\}$, in which case $k — \ell$ in H. To show that $M = \{k, \ell\}$ is a super-terminal component in H_S, it remains to verify $m \to k$ in H. To this end, consider a super-terminal component L in $H_{S\setminus\{\ell\}}$. As $\forall j \in S \setminus \{k, m, \ell\}$ one has $j \to k$ in H, we have $L \subseteq \{k, m\}$. This implies that (k, m) is an edge in H. Because H is 3-acyclic and $m \to \ell — k$ in H, it implies $m \to k$ in H, which was desired.

Finally, in case $\boxed{\text{C.}}$ the assumption $|K| \geqslant 2$ implies that the set $K \setminus \{\ell\}$ is a super-terminal component in $H_{S\setminus\{k,\ell\}}$. Now, let L be a super-terminal component in $H_{S\setminus\{\ell\}}$. We distinguish two subcases:

$\boxed{\text{C.1.}}$ If $k \notin L$, then L is a super-terminal component in $H_{S\setminus\{k,\ell\}}$, and, by its uniqueness, $L = K \setminus \{\ell\}$. To show that K is a super-terminal component in H_S it remains to verify $k \to \ell$ in H. To this end, consider a super-terminal component M in $H_{S\setminus\{m\}}$. Because $\forall j \in S \setminus (\{k\} \cup K)$ one has $j \to \ell$ in H, we have $M \subseteq \{k\} \cup (K \setminus \{m\})$. As $K \setminus \{m\}$ is complete in H, there are three options for M, namely $M = \{k\}$, $M = K \setminus \{m\}$ and $M = \{k\} \cup (K \setminus \{m\})$. In each of these 3 cases, (k,ℓ) is an edge in H. There is at least one $i \in L \subseteq K$, and one has then $k \to i — \ell$ in H. Because H is 3-acyclic, it implies $k \to \ell$ in H, which was needed.

$\boxed{\text{C.2.}}$ If $k \in L$, then necessarily $m \in L$, for otherwise one has $m \to k$ in H, which contradicts the choice of k. Hence, $L \setminus \{k\}$ is a super-terminal component in $H_{S\setminus\{k,\ell\}}$, and, by its uniqueness, $K \setminus \{\ell\} = L \setminus \{k\} = K \cap L$. As $m \in K \cap L$, we have $k — m$ and $m — \ell$ in H. Let M be a super-terminal component in $H_{S\setminus\{m\}}$. Observe that $M \subseteq K \cup L$, because, $\forall j \in S \setminus (K \cup L)$, one has $j \to k$ in H. The next observation is that $k \in M$. Indeed, otherwise for some $i \in M \subseteq (K \cup L) \setminus \{m\}$ one has $k \to i$ in H, which means that $k \to i — m — k$ in H contradicting that H is 3-acyclic. Analogously, we derive $\ell \in M$ (exchange of k and ℓ). Because $k, \ell \in M$, we have $k — \ell$ in H, which implies that $K \cup L$ is complete in H_S. This allows us to derive that $K \cup L$ is a super-terminal component in H_S.

Thus, in all cases (and subcases) we showed that H_S has a super-terminal component. □

A.2. Necessity proof

We show now that the inequalities from Section 4.2 are valid for vector-codes introduced in Section 4.1.

Lemma 6. *Let H be a hybrid graph over N which is 3-acyclic and has no flags.*

(i) *Then the inequalities* (b.1)–(b.2), (c.1)–(c.5) *and* (e.1)–(e.2) *are valid for the vector* $(\mathrm{a}_H, \mathrm{c}_H)$ *defined in* (14)–(15).
(ii) *If H is a chain graph without flags equivalent to an acyclic directed acyclic graph, then, moreover, the inequalities* (a.1) *hold.*

Proof. The inequalities (b.1)–(b.2) and (c.2) are evident from (14)–(15).

(c.1) If both terms on the left-hand side (LHS) of $\mathrm{a}_H(i \to j) + \mathrm{a}_H(j \to i) \leqslant \mathrm{c}_H(ij)$ vanish, then one has LHS $= 0 \leqslant$ RHS, where RHS means the right-hand side. Assume that at least one term on LHS is non-zero, say $\mathrm{a}_H(i \to j) = 1$. Then $i \to j$ in H, and because H is a hybrid graph, $\neg(j \to i$ in $H)$. Thus, the other term on LHS vanishes: $\mathrm{a}_H(j \to i) = 0$. However, (i,j) is an edge in H then, which means, by (16), $\mathrm{c}_H(ij) = 1$. Hence, LHS $= 1 =$ RHS.

(c.3) If the LHS of $2\cdot\mathrm{c}_H(ijk) \leqslant 2\cdot\mathrm{c}_H(ij) + \mathrm{a}_H(i \to k) + \mathrm{a}_H(j \to k)$ vanishes, the inequality clearly holds. If $c_H(ijk) = 1 = \mathrm{c}_H(ij)$ then LHS $= 2 \leqslant$ RHS. Only in case $\mathrm{c}_H(ijk) = 1$ and $\mathrm{c}_H(ij) = 0$ we observe by (16) and (17) that $i \to k \leftarrow j$ is an immorality in H. Hence, LHS $= 2 =$ RHS.

(c.4) If at least one term on the LHS of $\mathrm{a}_H(i \to j) + \mathrm{c}_H(jk) \leqslant 1 + \mathrm{c}_H(ijk) + \mathrm{a}_H(j \to k)$ vanishes LHS $\leqslant 1 \leqslant$ RHS. Assume $\mathrm{a}_H(i \to j) = \mathrm{c}_H(jk) = 1$, that is, $i \to j$ in H and (j,k) is an edge in H. If $\mathrm{a}_H(j \to k) = 1$ then LHS $= 2 \leqslant$ RHS. If $\mathrm{a}_H(j \to k) = 0$, then, since H is a hybrid graph, either $j \leftarrow k$ in H or $j — k$ in H. Because H has no flag (of the form $i \to j — k$), in both these sub-cases one has $\mathrm{c}_H(ijk) = 1$ by (17). Thus, LHS $= 2 =$ RHS.

(c.5) If at least one term on the LHS of $\mathrm{a}_H(i \to j) + \mathrm{c}_H(jk) + \mathrm{c}_H(ik) \leqslant 2 + \mathrm{a}_H(i \to k) + \mathrm{a}_H(k \to j)$ vanishes then LHS $\leqslant 2 \leqslant$ RHS. If all three of them equal 1 then $i \to j$ in H and (j,k), (k,i) are both edges in H. As H is 3-acyclic one necessarily has either $i \to k$ in H or $k \to j$ in H. In particular, LHS $= 3 \leqslant$ RHS.

(e.1) If at most two terms on the LHS of $\sum_{i\in S} \mathrm{c}_H(S \setminus \{i\}) \leqslant 2 + (|S| - 2)\cdot\mathrm{c}_H(S)$ are non-zero then LHS $\leqslant 2 \leqslant$ RHS. If at least three of them are non-zero then, by (15), for at least 3 subsets $T \subset S$ with $|T| = |S| - 1$, the graph H_T has a super-terminal component. By Lemma 5(b), H_S has a super-terminal component, and, therefore $\mathrm{c}_H(S) = 1$. Hence, LHS $\leqslant |S| =$ RHS.

(e.2) If the LHS of $(|S| - 1)\cdot\mathrm{c}_H(S) \leqslant \sum_{i\in S} \mathrm{c}_H(S \setminus \{i\})$ vanishes then LHS $= 0 \leqslant$ RHS. However, if $\mathrm{c}_H(S) = 1$ then, by (15), H_S has a super-terminal component. By Lemma 5(a) observe that there exists at least $|S| - 1$ subsets $T \subset S$ with $|T| = |S| - 1$ with $\mathrm{c}_H(T) = 1$. Thus, by (15), LHS $= |S| - 1 \leqslant$ RHS.

Thus, the part (i) of Lemma 6 was proved. To verify the part (ii) assume that H is a chain graph without flags equivalent to an acyclic directed acyclic graph G. If follows from (15) and Lemma 2 that c_H equals to the characteristic imset c_G for G. Thus, we need to verify the inequality

(a.1) $\sum_{T\subseteq S, |T|\geqslant 2} \mathrm{c}_G(T)\cdot(-1)^{|T|} \leqslant |S| - 1$ for any acyclic directed graph G over N.

Let us fix $S \subseteq N$, $|S| \geqslant 4$. As G_S is acyclic, there exists a total order ρ of all nodes in S consistent with the direction of arrows in G_S. For every $j \in S$ let us put

$$\mathcal{T}(j) = \left\{T \subseteq \{j\} \cup B_j^{\rho}\colon\ |T| \geqslant 2,\ j \in T\right\},$$

where B^{ρ}_j is the set of predecessors of j in ρ. Clearly, the classes $\mathcal{T}(j)$, $j \in S$ define a partitioning of the class $\{T \subseteq S : |T| \geqslant 2\}$; one has $\mathcal{T}(i) = \emptyset$ for the first node i in ρ, that is, the one with $B^{\rho}_i = \emptyset$. The point is that, for every $j \in S \setminus \{i\}$, one has

$$\sum_{T \in \mathcal{T}(j)} \mathrm{c}_G(T) \cdot (-1)^{|T|} \leqslant 1. \tag{A.1}$$

Indeed, by (1), observe that, for $T \in \mathcal{T}(j)$, one has $\mathrm{c}_G(T) = 1$ if and only if $T = \{j\} \cup K$, where $K \subseteq pa_G(j) \cap S$. In particular, the sum on the LHS of (A.1) is

$$\sum_{\emptyset \neq K \subseteq pa_G(j) \cap S} (-1)^{|K|+1} = \begin{cases} 0 & \text{if } pa_G(j) \cap S = \emptyset, \\ 1 & \text{if } pa_G(j) \cap S \neq \emptyset. \end{cases}$$

Clearly, (a.1) can be obtained by summing (A.1) for $j \in S \setminus \{i\}$. □

A.3. Interpretation of the inequalities

It follows from the above proof of Lemma 6 what is the graphical interpretation of the inequalities from Section 4.2:

(c.1) $\mathrm{a}_H(i \to j) + \mathrm{a}_H(j \to i) \leqslant \mathrm{c}_H(ij)$ means that if $i \to j$ or $j \to i$ are (encoded in a_H as) arrows then (i, j) is (encoded in c_H as) an edge;
(c.2) $\mathrm{c}_H(ij) \leqslant 1$ means, together with (c.1), that there is no bi-directed edge between i and j; in other words, one cannot have simultaneously $i \to j$ in H and $j \to i$ in H;
(c.3) $2 \cdot \mathrm{c}_H(ijk) \leqslant 2 \cdot \mathrm{c}_H(ij) + \mathrm{a}_H(i \to k) + \mathrm{a}_H(j \to k)$ and its permutated versions allow one to conclude that if $\mathrm{c}_H(ijk) = 1$ then H_{ijk} has a super-terminal component;
(c.4) $\mathrm{a}_H(i \to j) + \mathrm{c}_H(jk) \leqslant 1 + \mathrm{c}_H(ijk) + \mathrm{a}_H(j \to k)$ prevents H to have a flag $i \to j — k$;
(c.5) $\mathrm{a}_H(i \to j) + \mathrm{c}_H(jk) + \mathrm{c}_H(ik) \leqslant 2 + \mathrm{a}_H(i \to k) + \mathrm{a}_H(k \to j)$ says that H has not a semi-directed 3-cycle of the form i, j, k, i with $i \to j$;
(e.1) $\sum_{i \in S} \mathrm{c}_H(S \setminus \{i\}) \leqslant 2 + (|S| - 2) \cdot \mathrm{c}_H(S)$ encodes the implication in Lemma 5(b);
(e.2) $(|S| - 1) \cdot \mathrm{c}_H(S) \leqslant \sum_{i \in S} \mathrm{c}_H(S \setminus \{i\})$ encodes the implication in Lemma 5(a);
(a.1) $\sum_{T \subseteq S, |T| \geqslant 2} \mathrm{c}_H(T) \cdot (-1)^{|T|} \leqslant |S| - 1$ means, loosely said, that the graph H_S has at least one initial node. The point is that all inequalities (a.1) together ensure the existence of an equivalent acyclic directed graph G, as shown in the following section (Lemma 9).

Another interpretation of (a.1) is that it excludes the existence of a chordless undirected or semi-directed cycle composed just of nodes of S. Indeed, in that case the LHS of (a.1) is $|S|$ because the summands are $|S|$-times 1 and 0 otherwise, which contradicts the validity of (a.1).

Appendix B. Sufficiency of the inequalities

In this section, we prove that the inequalities from Section 4.2 provide an LP relaxation for (the convex hull of) the set of codes of chain graphs without flags equivalent to acyclic directed graphs.

B.1. Auxiliary lemmas

We start with a couple of auxiliary observations.

Lemma 7. *Let* (a, c) *be a vector with integer components satisfying the inequalities* (b.1)–(b.2), (c.1)–(c.5) *and* (e.1)–(e.2). *Then* (e.2) *also holds for* $|S| = 3$, *that is,*

$$2 \cdot c(ijk) \leqslant \mathrm{c}(ij) + \mathrm{c}(ik) + \mathrm{c}(jk), \tag{B.1}$$

and (a, c) *is a zero-one vector.*

Proof. To show (B.1) let us fix the set $S = ijk$ and sum up all three corresponding inequalities (c.3) and then use three-times (c.1):

$$\begin{aligned} 6 \cdot c(ijk) &\leqslant 2 \cdot c(ij) + 2 \cdot c(ik) + 2 \cdot c(jk) + \mathrm{a}(i \to j) + \mathrm{a}(j \to i) \\ &\quad + \mathrm{a}(i \to k) + \mathrm{a}(k \to i) + \mathrm{a}(j \to k) + \mathrm{a}(k \to j) \leqslant 3 \cdot c(ij) + 3 \cdot c(ik) + 3 \cdot c(jk). \end{aligned}$$

Divide it by 3 and get (B.1). The next observation is that (a,c) is non-negative. The inequality $0 \leqslant \mathsf{c}(S)$ for $|S|=2$ follows from (b.1) and (c.1). The inequality $0 \leqslant \mathsf{c}(S)$ for $|S| \geqslant 5$ can be derived by induction on $|S|$ from (b.2) and (e.1): if $|S| \geqslant 5$ then by the induction premise and (e.1) one has

$$0 \leqslant 2 + \big(|S|-2\big)\cdot \mathsf{c}(S), \quad \text{which implies } -1 < \frac{-2}{|S|-2} \leqslant \mathsf{c}(S).$$

However, because $\mathsf{c}(S) \in \mathbb{Z}$, it implies $0 \leqslant \mathsf{c}(S)$.

Finally, we show that the components of (a,c) are at most 1. As concerns the a-part, by (b.1), (c.1) and (c.2) one has $\mathsf{a}(i \to j) \leqslant \mathsf{a}(i \to j) + \mathsf{a}(j \to i) \leqslant c(ij) \leqslant 1$. As concerns the c-part, we prove $\mathsf{c}(S) \leqslant 1$ by induction on $|S|$ from (c.2) using (e.2) for $|S| \geqslant 3$, which involves (B.1). Specifically, consider $|S| > 2$ and write by (e.2) and the induction premise

$$\big(|S|-1\big)\cdot \mathsf{c}(S) \leqslant \sum_{i \in S} \mathsf{c}\big(S \setminus \{i\}\big) \leqslant \sum_{i \in S} 1 = |S|.$$

Hence,

$$\mathsf{c}(S) \leqslant \frac{|S|}{|S|-1} < 2.$$

However, because $\mathsf{c}(S) \in \mathbb{Z}$, it implies $\mathsf{c}(S) \leqslant 1$. □

Lemma 8. *Let* c, c' *be zero-one vectors satisfying* (e.1)–(e.2)*. Provided* $\mathsf{c}(S) = \mathsf{c}'(S)$ *for* $S \subseteq N$ *with* $2 \leqslant |S| \leqslant 3$ *one has* $\mathsf{c} = \mathsf{c}'$.

Proof. We prove $\mathsf{c}(S) = \mathsf{c}'(S)$ by induction on $|S|$. Since they are both zero-one vectors and exchangeable, it is enough to show, for $|S| \geqslant 4$, the implication $\mathsf{c}(S) = 1 \ \Rightarrow \mathsf{c}'(S) = 1$. First, by (e.2) applied to c, we observe $3 \leqslant |S|-1 = (|S|-1)\cdot \mathsf{c}(S) \leqslant \sum_{i \in S} \mathsf{c}(S \setminus \{i\})$. By the induction premise the same holds for c' and one can write using (e.1) applied to c':

$$3 \leqslant \sum_{i \in S} \mathsf{c}'\big(S \setminus \{i\}\big) \leqslant 2 + \big(|S|-2\big)\cdot \mathsf{c}'(S).$$

That means

$$0 < \frac{1}{|S|-2} \leqslant \mathsf{c}'(S),$$

which, because $c'(S) \in \{0,1\}$, implies $\mathsf{c}'(S) = 1$. □

B.2. Sufficiency proof

Here is the desired result.

Lemma 9. *Given a finite non-empty set* N*, let* (a,c) *be a vector with integer components satisfying the inequalities* (b.1)–(b.2), (c.1)–(c.5) *and* (e.1)–(e.2).

(i) *Then there exists a (unique) hybrid graph* H *over* N *which is* 3*-acyclic and has no flags such that* $\mathsf{a} = \mathsf{a}_H$ *and* $\mathsf{c} = \mathsf{c}_H$.
(ii) *If, moreover, the inequalities* (a.1) *hold for* c *then* H *is a chain graph without flags equivalent to an acyclic directed graph over* N.

Proof. Note that the uniqueness of the graph H follows from (14) and (16). By Lemma 7, (a,c) is a zero-one vector. Let us define a hybrid graph H over N:

$$\begin{aligned} i \to j \text{ in } H \quad &\Leftrightarrow \quad \mathsf{a}(i \to j) = 1, \\ i - j \text{ in } H \quad &\Leftrightarrow \quad \mathsf{c}(ij) = 1 \;\&\; \mathsf{a}(i \to j) = 0 \;\&\; \mathsf{a}(j \to j) = 0. \end{aligned} \tag{B.2}$$

The definition is correct, since, by (c.1) and (c.2), $\mathsf{a}(i \to j) + \mathsf{a}(j \to i) \leqslant \mathsf{c}(ij) \leqslant 1$. Note

$$\mathsf{c}(ij) = 1 \quad \Leftrightarrow \quad (i,j) \text{ is an edge in } H \quad \Leftrightarrow \quad H_{ij} \text{ has a super-terminal component.} \tag{B.3}$$

Observe that H is 3-acyclic (see Section 2.1). Indeed, assume for a contradiction that H has a semi-directed cycle ρ: i,j,k,i with $i \to j$. Hence, by (B.2), (B.3) and (c.5) we have

$$3 = \mathsf{a}(i \to j) + \mathsf{c}(jk) + \mathsf{c}(ik) \leqslant 2 + \mathsf{a}(i \to k) + \mathsf{a}(k \to j),$$

saying $1 \leqslant \mathsf{a}(i \to k) + \mathsf{a}(k \to j)$, which implies, by (B.2), that either $i \to k$ in H or $k \to j$ in H. This contradict the assumption ρ is a semi-directed cycle.

The next step is to show that, for distinct $i, j, k \in N$,

$$\mathsf{c}(ijk) = 1 \quad \Leftrightarrow \quad H_{ijk} \text{ has a super-terminal component.} \tag{B.4}$$

For this purpose, realize that, because H is 3-acyclic, the graph H_{ijk} has a super-terminal component iff it is either quasi-complete or an immorality. For $\Rightarrow$ direction, if $\mathsf{c}(ijk) = 1$ and H_{ijk} is not quasi-complete, then at least one edge in H_{ijk} is missing. Assume without loss of generality that it is (i, j), which means, by (B.3), $\mathsf{c}(ij) = 0$. Write by (c.3)

$$2 = 2 \cdot \mathsf{c}(ijk) \leqslant 2 \cdot \mathsf{c}(ij) + \mathsf{a}(i \to k) + \mathsf{a}(j \to k) = \mathsf{a}(i \to k) + \mathsf{a}(j \to k).$$

This implies that both $\mathsf{a}(i \to k) = 1$ and $\mathsf{a}(j \to k) = 1$; by (B.2), $i \to k \leftarrow j$ is an immorality. For $\Leftarrow$ direction, in case H_{ijk} is quasi-complete, write by (B.3) and (e.1) for $S = ijk$:

$$3 = \mathsf{c}(ij) + \mathsf{c}(ik) + \mathsf{c}(jk) \leqslant 2 + \mathsf{c}(ijk),$$

which implies the desired conclusion $\mathsf{c}(ijk) = 1$. If H_{ijk} is an immorality, say $i \to j \leftarrow k$, then realize $\neg(j \to k \text{ in } H)$ and observe by (B.2) that $\mathsf{a}(i \to j) = 1$ and $\mathsf{a}(j \to k) = 0$; while $\mathsf{c}(jk) = 1$ by (B.3). This allows us to write by (c.4)

$$2 = \mathsf{a}(i \to j) + \mathsf{c}(jk) \leqslant 1 + \mathsf{c}(ijk) + \mathsf{a}(j \to k) = 1 + \mathsf{c}(ijk),$$

implying $\mathsf{c}(ijk) = 1$. Thus, (B.4) has been verified.

The next observation is that H has no flag. Assume for a contradiction it has a flag $i \to j — k$. As $\neg(j \to k \text{ in } H)$, by (B.2), $\mathsf{a}(i \to j) = 1$ and $\mathsf{a}(j \to k) = 0$. By (B.3) and (c.4)

$$2 = \mathsf{a}(i \to j) + \mathsf{c}(jk) \leqslant 1 + \mathsf{c}(ijk) + \mathsf{a}(j \to k) = 1 + \mathsf{c}(ijk).$$

Hence, $\mathsf{c}(ijk) = 1$, which, by (B.4), contradicts the assumption $i \to j — k$ is a flag in H.

Thus, consider the 3-acyclic hybrid graph H without flags defined in (B.2). By Lemma 6(i) the vector c_H given by (15) is a zero-one vector satisfying the conditions (e.1)–(e.2). By (B.3) and (B.4), we have $\mathsf{c}(S) = \mathsf{c}_H(S)$ for any $S \subseteq N$, $2 \leqslant |S| \leqslant 3$. As c also satisfies (e.1)–(e.2), by Lemma 8, we have $\mathsf{c} = \mathsf{c}_H$. By (14) and (B.2), we also have $\mathsf{a} = \mathsf{a}_H$, which concludes the proof of the (i)-part of Lemma 9.

As concerns the (ii)-part, assume c satisfies (a.1). As mentioned in Section 2.1, to show that H is a chain graph, it is enough to verify it has no semi-directed chordless cycle. Since H is 3-acyclic, we already know it has no semi-directed cycle of length 3. Thus, assume for a contradiction it has a semi-directed chordless cycle ρ: $i_0, i_1, \ldots, i_m = i_0$ of length $m \geqslant 4$ with $i_0 \to i_1$ in H. Because H has no flag, we observe that ρ has to be a directed cycle, that is, $i_r \to i_{r+1}$ in H for $r = 0, 1, \ldots, m-1$. Put $S = \{i_1, \ldots, i_m\}$. By the (i)-part, $\mathsf{c} = \mathsf{c}_H$; thus, using (15) observe that the only subsets $T \subseteq S$, $|T| \geqslant 2$ with $\mathsf{c}(T) = 1$ are the edges of the cycle ρ. In particular, if we substitute into (a.1) we get

$$|S| = \sum_{T \subseteq S,\, |T| \geqslant 2} \mathsf{c}(T) \cdot (-1)^{|T|} \leqslant |S| - 1,$$

which is a contradiction. Thus, H cannot have a semi-directed cycle of length $m \geqslant 4$.

As mentioned in (the end of) Section 2.3, to show that H is equivalent to an acyclic directed graph, it is enough to show it has no undirected chordless cycle of length $m \geqslant 4$. The proof by a contradiction is analogous to the case of a semi-directed cycle. If ρ: $i_0 — i_1 — \cdots — i_m = i_0$ is such a cycle, put $S = \{i_1, \ldots, i_m\}$ and get a contradiction with (a.1) in the same way. □

Thus, by combining Lemmas 6(i) and 9(i) we get Theorem 2, by using Lemmas 6(ii) and 9(ii) we get Theorem 1.

Appendix C. Reduced inequalities

Just as the inequalities (e.1)–(e.2) can be reduced to (e.1⋆) and (e.2⋆) (see Section 5.1), the set $\mathcal{T}$ can occasionally be used to further reduce the length of the vector (a, c) and the inequalities (b.1)–(b.2), (c.1)–(c.5). One needs components $\mathsf{a}(i \to j)$ and $\mathsf{c}(ij)$ only if $\{i, j\} \in \mathcal{T}$, which gives an obvious reduction of basic non-negativity inequalities:

(b.1⋆) $\forall \{i, j\} \in \mathcal{T}, \;\; 0 \leqslant \mathsf{a}(i \to j)$,
(b.2⋆) $\forall S \in \mathcal{T}, \;\; |S| = 3, 4, \;\; 0 \leqslant \mathsf{c}(S)$.

As concerns the consistency inequalities, since we assume $\mathsf{c}(R) = 0$ for all $R \notin \mathcal{T}$ and, also, $\mathsf{a}(i \to j) = 0$ for $\{i, j\} \notin \mathcal{T}$, they get the following form:

(c.1⋆) $\forall \{i, j\} \in \mathcal{T}, \;\; \mathsf{a}(i \to j) + \mathsf{a}(j \to i) \leqslant \mathsf{c}(ij)$,
(c.2⋆) $\forall \{i, j\} \in \mathcal{T}, \;\; \mathsf{c}(ij) \leqslant 1$,
(c.3⋆) $\forall \{i, j, k\} \in \mathcal{T}, \;\; 2 \cdot \mathsf{c}(ijk) \leqslant 2 \cdot \mathsf{c}(ij) + \mathsf{a}(i \to k) + \mathsf{a}(j \to k)$,
(c.4⋆) $\forall i, j, k \in N$ distinct with $\{i, j\}, \{j, k\} \in \mathcal{T}, \;\; \mathsf{a}(i \to j) + \mathsf{c}(jk) \leqslant 1 + \mathsf{c}(ijk) + \mathsf{a}(j \to k)$,
(c.5⋆) $\forall i, j, k \in N$ distinct with $\{i, j\}, \{j, k\}, \{i, k\} \in \mathcal{T}, \;\; \mathsf{a}(i \to j) + \mathsf{c}(jk) + \mathsf{c}(ik) \leqslant 2 + \mathsf{a}(i \to k) + \mathsf{a}(k \to j)$.

Appendix D. Results of simulation experiments

Table 5
Running times in seconds (cimset) of our ILP approach (Method 2), the number of acyclic inequalities generated (# (a.1)), and the running time of GOBNILP version 1.3 with SCIP version 3.0.0 using CPLEX Studio 12.5 for data sets `bn17` using BDeu ($\alpha = 1.0$) and BIC. Data sets are further partitioned by the number of samples used in the training data base $(100, 1000, 10\,000)$ and the parent set size limit (plim). Data sets were run for a maximum of 1 h (3600 s) and the percentage gap $(y - x)/x$ of the current best LP objective value y to the best current integer objective value x is given if terminated before a final solution was found and if available.

`bn17:100` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	26	24	150	60	10	0	2	1	1
3	35	54	298	156	29	3	2	1	1
4	39	78	362	204	43	9	2	1	1
5	39	78	362	204	43	9	2	1	1
6	51	177	666	420	145	73	31	33	1
7	85	495	1708	1254	689	475	–	3600 (199.20%)	1
8	133	1758	3902	3816	10 408	6796	–	3600 (NA)	1
9	136	2007	4080	4080	27 314	17 962	–	3600 (NA)	1
10	136	2040	4080	4080	54 345	35 963	–	3600 (NA)	1

`bn17:100` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	34	33	242	120	20	0	6	1	1
3	34	33	242	120	20	0	6	1	1
4	34	33	242	120	20	0	6	1	1
5	34	33	242	120	20	0	6	1	1
6	34	33	242	120	20	0	6	1	1
7	34	33	242	120	20	0	6	1	1
8	34	33	242	120	20	0	6	1	1
9	34	33	242	120	20	0	6	1	1
10	34	33	242	120	20	0	6	1	1

`bn17:1000` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	61	114	848	462	85	0	139	42	1
3	68	171	1060	648	129	12	60	16	1
4	68	180	1060	648	136	19	60	16	1
5	68	180	1060	648	136	19	60	16	1
6	68	180	1060	648	136	19	60	16	1
7	68	180	1060	648	136	19	60	17	1
8	68	180	1060	648	136	19	60	17	1
9	68	180	1060	648	136	19	60	17	1
10	68	180	1060	648	136	19	60	17	1

`bn17:1000` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	75	165	1264	798	154	0	346	161	1
3	83	282	1560	1086	258	27	218	151	1
4	83	303	1560	1086	272	42	38	11	1
5	83	303	1560	1086	272	42	38	11	1
6	83	303	1560	1086	272	42	38	11	1
7	83	303	1560	1086	272	42	38	11	1
8	83	303	1560	1086	272	42	38	11	1
9	83	303	1560	1086	272	42	38	10	1
10	83	303	1560	1086	272	42	38	11	1

`bn17:10000` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	97	396	2118	1680	440	0	–	3600 (5.10%)	9.69
3	118	975	3118	2808	1393	239	–	3600 (NA)	15.67
4	121	1221	3268	2988	2479	813	–	3600 (NA)	12.28
5	121	1251	3268	2988	2884	1184	–	3600 (3.98%)	2.74
6	121	1251	3268	2988	2950	1293	–	3600 (2.80%)	0.95
7	121	1251	3268	2988	2954	1309	487	2076	0.66
8	121	1251	3268	2988	2954	1309	487	2079	0.69
9	121	1251	3268	2988	2954	1309	487	2121	0.63
10	121	1251	3268	2988	2954	1309	487	2124	0.64

`bn17:10000` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	98	414	2164	1728	462	0	–	3600 (11.33%)	14.24
3	120	1029	3212	2916	1549	279	–	3600 (28.92%)	14.61
4	124	1332	3412	3156	3023	1027	–	3600 (NA)	21.20
5	125	1386	3464	3222	3857	1669	–	3600 (6.66%)	4.91
6	125	1389	3464	3222	4088	1941	–	3600 (3.31%)	4.67
7	125	1389	3464	3222	4126	1986	–	3600 (0.69%)	0.94
8	125	1389	3464	3222	4126	1986	–	3600 (0.69%)	0.99
9	125	1389	3464	3222	4126	1986	–	3600 (0.69%)	0.97
10	125	1389	3464	3222	4126	1986	–	3600 (0.69%)	0.98

Table 6
Running times in seconds (cimset) of our ILP approach (Method 2), the number of acyclic inequalities generated (# (a.1)), and the running time of GOBNILP version 1.3 with SCIP version 3.0.0 using CPLEX Studio 12.5 for data sets `bn17b` using BDeu ($\alpha = 1.0$) and BIC. Data sets are further partitioned by the number of samples used in the training data base (100, 1000, 10 000) and the parent set size limit (plim). Data sets were run for a maximum of 1 h (3600 s) and the percentage gap $(y - x)/x$ of the current best LP objective value y to the best current integer objective value x is given if terminated before a final solution was found and if available.

`bn17b:100` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	24	21	132	48	8	0	1	1	1
3	27	39	180	84	17	3	2	1	1
4	31	57	242	132	30	8	5	1	1
5	31	57	242	132	30	8	5	1	1
6	31	57	242	132	30	8	5	1	1
7	88	558	1868	1482	876	588	–	3600 (352.50%)	1
8	122	1317	3316	3042	4162	3124	–	3600 (NA)	1
9	136	2037	4080	4080	29 202	19 673	–	3600 (NA)	1
10	136	2040	4080	4080	–	–	–	3600 (NA)	1

`bn17b:100` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	28	30	186	72	12	0	3	1	1
3	29	39	206	84	16	2	3	1	1
4	29	39	206	84	16	2	3	1	1
5	29	39	206	84	16	2	3	1	1
6	29	39	206	84	16	2	3	1	1
7	29	39	206	84	16	2	3	1	1
8	29	39	206	84	16	2	3	1	1
9	29	39	206	84	16	2	3	1	1
10	29	39	206	84	16	2	3	1	1

`bn17b:1000` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	63	132	912	570	115	0	45	10	1
3	73	246	1234	846	210	25	41	7	1
4	73	252	1234	846	215	31	47	8	1
5	73	252	1234	846	215	31	47	8	1
6	73	252	1234	846	215	31	47	8	1
7	73	252	1234	846	215	31	47	7	1
8	73	252	1234	846	215	31	47	8	1
9	73	252	1234	846	215	31	47	7	1
10	73	252	1234	846	215	31	47	8	1

`bn17b:1000` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	67	156	1040	702	146	0	76	23	1
3	78	312	1412	1038	285	35	87	33	1
4	79	366	1440	1068	340	75	74	50	1
5	79	366	1440	1068	340	75	74	48	1
6	79	366	1440	1068	340	75	74	49	1
7	79	366	1440	1068	340	75	74	48	1
8	79	366	1440	1068	340	75	74	50	1
9	79	366	1440	1068	340	75	74	49	1
10	79	366	1440	1068	340	75	74	48	1

`bn17b:10000` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	113	525	2860	2490	696	0	–	3600 (4.77%)	9.07
3	124	1101	3416	3168	1773	286	–	3600 (27.10%)	15.97
4	127	1266	3576	3378	2747	798	–	3600 (4.11%)	19.27
5	127	1278	3576	3378	2954	1007	–	3600 (2.22%)	6.15
6	127	1278	3576	3378	2984	1058	–	3600 (0.75%)	5.21
7	127	1278	3576	3378	2984	1066	–	3600 (0.81%)	3.75
8	127	1278	3576	3378	2984	1066	–	3600 (0.83%)	3.72
9	127	1278	3576	3378	2984	1066	–	3600 (0.81%)	3.60
10	127	1278	3576	3378	2984	1066	–	3600 (0.81%)	3.65

`bn17b:10000` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	114	573	2912	2556	755	0	–	3600 (8.87%)	8.10
3	125	1173	3468	3234	1974	330	–	3600 (43.40%)	16.25
4	130	1380	3738	3594	3408	1041	–	3600 (7.18%)	38.27
5	130	1404	3738	3594	4092	1557	–	3600 (2.76%)	12.21
6	130	1404	3738	3594	4190	1688	–	3600 (1.49%)	2.83
7	130	1404	3738	3594	4190	1689	–	3600 (0.75%)	3.86
8	130	1404	3738	3594	4190	1689	–	3600 (0.75%)	3.95
9	130	1404	3738	3594	4190	1689	–	3600 (0.75%)	3.89
10	130	1404	3738	3594	4190	1689	–	3600 (0.76%)	3.96

Table 7
Running times in seconds (cimset) of our ILP approach (Method 2), the number of acyclic inequalities generated (# (a.1)), and the running time of GOBNILP version 1.3 with SCIP version 3.0.0 using CPLEX Studio 12.5 for data sets `bn17-noisyor` using BDeu ($\alpha = 1.0$) and BIC. Data sets are further partitioned by the number of samples used in the training data base (100, 1000, 10 000) and the parent set size limit (plim). Data sets were run for a maximum of 1 h (3600 s) and the percentage gap $(y-x)/x$ of the current best LP objective value y to the best current integer objective value x is given if terminated before a final solution was found and if available.

`bn17-noisyor:100` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	119	630	3162	2856	781	0	751	593	3.70
3	131	1017	3790	3660	1309	158	–	3600 (5.38%)	5.51
4	133	1164	3902	3816	1629	312	–	3600 (8.52%)	3.04
5	134	1233	3960	3900	1777	383	–	3600 (10.20%)	5.62
6	136	1425	4080	4080	2394	673	–	3600 (NA)	5.86
7	136	1584	4080	4080	3636	1551	–	3600 (NA)	6.23
8	136	1707	4080	4080	5514	2787	–	3600 (NA)	5.77
9	136	1707	4080	4080	5514	2787	–	3600 (NA)	5.74
10	136	1707	4080	4080	5514	2787	–	3600 (NA)	5.73

`bn17-noisyor:100` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	114	528	2920	2544	646	0	93	15	3.31
3	117	591	3062	2706	711	18	95	16	2.79
4	117	591	3062	2706	711	18	95	16	2.79
5	117	591	3062	2706	711	18	95	16	2.83
6	117	591	3062	2706	711	18	95	15	2.79
7	117	591	3062	2706	711	18	95	16	2.79
8	117	591	3062	2706	711	18	95	17	2.82
9	117	591	3062	2706	711	18	95	16	2.74
10	117	591	3062	2706	711	18	95	17	2.69

`bn17noisyor:1000` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	136	1407	4080	4080	1971	0	2902	3372	5.69
3	136	1839	4080	4080	3884	806	–	3600 (55.48%)	6.74
4	136	1851	4080	4080	4758	1422	–	3600 (NA)	4.76
5	136	1851	4080	4080	4811	1503	–	3600 (18.29%)	5.65
6	136	1851	4080	4080	4824	1513	–	3600 (NA)	5.05
7	136	1851	4080	4080	4878	1565	–	3600 (NA)	4.64
8	136	1851	4080	4080	4878	1565	–	3600 (NA)	4.65
9	136	1851	4080	4080	4878	1565	–	3600 (NA)	4.61
10	136	1851	4080	4080	4878	1565	–	3600 (NA)	4.47

`bn17-noisyor:1000` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	136	1404	4080	4080	1979	0	–	3600 (0.92%)	6.51
3	136	1749	4080	4080	3480	661	–	3600 (NA)	4.56
4	136	1749	4080	4080	3661	833	–	3600 (NA)	5.83
5	136	1749	4080	4080	3661	833	–	3600 (NA)	5.79
6	136	1749	4080	4080	3661	833	–	3600 (14.12%)	5.84
7	136	1749	4080	4080	3661	833	–	3600 (14.12%)	5.78
8	136	1749	4080	4080	3661	833	–	3600 (14.12%)	5.65
9	136	1749	4080	4080	3661	833	–	3600 (14.12%)	5.52
10	136	1749	4080	4080	3661	833	–	3600 (NA)	5.56

`bn17-noisyor:10000` (BDE)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	136	1788	4080	4080	2513	0	–	3600 (8.69%)	7.02
3	136	2040	4080	4080	7306	1848	–	3600 (NA)	291.07
4	136	2040	4080	4080	14 155	5538	–	3600 (NA)	3600 (0.22%)
5	136	2040	4080	4080	19 373	8997	–	3600 (NA)	3600 (0.12%)
6	136	2040	4080	4080	20 422	10 022	–	3600 (NA)	3600 (0.20%)
7	136	2040	4080	4080	20 473	10 087	–	3600 (NA)	3600 (0.19%)
8	136	2040	4080	4080	20 473	10 087	–	3600 (NA)	3600 (0.17%)
9	136	2040	4080	4080	20 473	10 087	–	3600 (NA)	3600 (0.17%)
10	136	2040	4080	4080	20 473	10 087	–	3600 (NA)	3600 (0.18%)

`bn17-noisyor:10000` (BCI)

plim	$c.1^*$	$c.3^*$	$c.4^*$	$c.5^*$	$e.1^*$	$e.2^*$	# (a.1)	cimset	GOBNILP
2	136	1794	4080	4080	2522	0	–	3600 (NA)	4.95
3	136	2040	4080	4080	7468	1889	–	3600 (NA)	311.74
4	136	2040	4080	4080	13 960	5446	–	3600 (NA)	3114.74
5	136	2040	4080	4080	17 562	7822	–	3600 (NA)	3367.72
6	136	2040	4080	4080	17 645	7917	–	3600 (NA)	2408.75
7	136	2040	4080	4080	17 645	7917	–	3600 (NA)	2401.67
8	136	2040	4080	4080	17 645	7917	–	3600 (NA)	2355.90
9	136	2040	4080	4080	17 645	7917	–	3600 (NA)	2216.08
10	136	2040	4080	4080	17 645	7917	–	3600 (NA)	2213.02

References

[1] T. Achterberg, T. Koch, A. Martin, Branching rules revisited, Oper. Res. Lett. 33 (2005) 42–54.
[2] S.A. Andersson, D. Madigan, M.D. Perlman, A characterization of Markov equivalence classes for acyclic digraphs, Ann. Stat. 25 (1997) 505–541.
[3] R.R. Bouckaert, Bayesian belief networks: from construction to evidence, PhD thesis, University of Utrecht, 1995.
[4] A. Cano, M. Gómez-Olmedo, A.R. Masegosa, S. Moral, Locally averaged Bayesian Dirichlet metrics for learning the structure and the parameters of Bayesian networks, Int. J. Approx. Reason. 54 (4) (2013) 526–540.
[5] D.M. Chickering, Optimal structure identification with greedy search, J. Mach. Learn. Res. 3 (2002) 507–554.
[6] J. Cussens, Maximum likelihood pedigree reconstruction using integer programming, in: Proceedings of the Workshop on Constraint Based Methods for Bioinformatics (WCBMB), 2010, pp. 9–19.
[7] J. Cussens, Bayesian network learning with cutting planes, in: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI), 2011, pp. 153–160.
[8] J. Cussens, M. Bartlett, E.M. Jones, N.A. Sheehan, Maximum likelihood pedigree reconstruction using integer programming, Genet. Epidemiol. 37 (1) (2013) 69–83.
[9] C.P. de Campos, Z. Zeng, Q. Ji, Structure learning Bayesian networks using constraints, in: Proceedings of the 26th International Conference on Machine Learning (ICML), 2009, pp. 113–120.
[10] C.P. de Campos, Q. Ji, Efficient structure learning Bayesian networks using constraints, J. Mach. Learn. Res. 12 (2011) 663–689.
[11] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Mach. Learn. 20 (1995) 194–243.
[12] R. Hemmecke, S. Lindner, M. Studený, Characteristic imsets for learning Bayesian network structure, Int. J. Approx. Reason. 53 (2012) 1336–1349.
[13] T. Jaakkola, D. Sontag, A. Globerson, M. Meila, Learning Bayesian network structure using LP relaxations, in: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), 2010, pp. 358–365.
[14] A. Land, S. Powell, Computer codes for problems of integer programming, Ann. Discrete Math. 5 (1979) 221–269.
[15] S.L. Lauritzen, Graphical Models, Clarendon Press, 1996.
[16] J.T. Linderoth, M.W.P. Savelsbergh, A computational study of search strategies for mixed integer programming, INFORMS J. Comput. 11 (1999) 173–187.
[17] S. Lindner, Discrete optimization in machine learning: learning Bayesian network structures and conditional independence implication, PhD thesis, TU Munich, 2012.
[18] R.E. Neapolitan, Learning Bayesian Networks, Pearson Prentice Hall, 2004.
[19] G.E. Schwarz, Estimation of the dimension of a model, Ann. Stat. 6 (1978) 461–464.
[20] M. Studený, A recovery algorithm for chain graphs, Int. J. Approx. Reason. 17 (1997) 265–293.
[21] M. Studený, Characterization of essential graphs by means of the operation of legal merging of components, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 12 (2004) 43–62.
[22] M. Studený, Probabilistic Conditional Independence Structures, Springer, 2005.
[23] M. Studený, J. Vomlel, R. Hemmecke, A geometric view on learning Bayesian network structures, Int. J. Approx. Reason. 51 (2010) 578–586.
[24] M. Studený, R. Hemmecke, S. Lindner, Characteristic imset: a simple algebraic representative of a Bayesian network structure, in: Proceedings of the 5th European Workshop on Probabilistic Graphical Models (PGM), 2010, pp. 257–264.
[25] M. Studený, Integer linear programming approach to learning Bayesian network structure: towards the essential graph, in: Proceedings of the 6th European Workshop on Probabilistic Graphical Models (PGM), 2012, pp. 307–314.
[26] M. Studený, D. Haws, R. Hemmecke, S. Lindner, Polyhedral approach to statistical learning graphical models, in: T. Hibi (Ed.), Harmony of Gröbner Bases and the Modern Industrial Society: the 2nd CREST-SBM International Conference, World Scientific, 2012, pp. 346–372.
[27] M. Studený, LP relaxations and pruning for characteristic imsets, research report n. 2323, Institute of Information Theory and Automation of the ASCR, Prague, June 2012, also available at http://staff.utia.cas.cz/studeny/f18.html.
[28] M. Studený, D. Haws, On polyhedral approximations of polytopes for learning Bayesian networks, J. Algebr. Stat. 4 (2013) 59–92.
[29] M. Teyssier, D. Koller, Ordering-based search: a simple and effective algorithm for learning Bayesian networks, in: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI), 2005, pp. 584–590.
[30] T. Verma, J. Pearl, Equivalence and synthesis of causal models, in: Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence (UAI), 1991, pp. 220–227.
[31] I.L.O.G. CPLEX, reference manual, 2013, http://www.ilog.com/products/cplex.
[32] GOBNILP, a web page, 2013, http://www.cs.york.ac.uk/aig/sw/gobnilp.