International Journal of Approximate Reasoning 50 (2009) 385–413

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

A reconstruction algorithm for the essential graph ☆

Milan Studený, Jiří Vomlel *

Instititute of Information Theory and Automation of the ASCR, Pod vodárenskou věží 4, Prague, CZ 182 08, Czech Republic

ARTICLE INFO

Article history:
Received 31 October 2006
Received in revised form 10 September 2008
Accepted 11 September 2008
Available online 1 October 2008

Keywords:
Bayesian network structure
Chain graph
Essential graph
Standard imset

ABSTRACT

A standard graphical representative of a Bayesian network structure is a special chain graph, known as an *essential graph*. An alternative algebraic approach to the mathematical description of this statistical model uses instead a certain integer-valued vector, known as a *standard imset*. We give a direct formula for the translation of any chain graph describing a Bayesian network structure into the standard imset. Moreover, we present a two-stage algorithm which makes it possible to reconstruct the essential graph on the basis of the standard imset. The core of this paper is the proof of the correctness of the algorithm.

1. Introduction

The general motivation for this paper is learning Bayesian network structures. Some learning procedures require a unique representative of each individual Bayesian network structure. The most popular graphical representative of a Bayesian network statistical model is the *essential graph* of the respective equivalence class of acyclic directed graphs. It is a chain graph describing shared features of acyclic directed graphs within the respective equivalence class. The term "essential" graph was proposed by Andersson, Madigan and Perlman, who also gave a graphical characterization of graphs that are essential graphs [1]. An alternative characterization of essential graphs, which is utilized in this paper, has recently been found in [18] and independently in [15].

Another possible representative of a Bayesian network statistical model is a certain integer-valued vector, named the *standard imset*. It is also a unique representative of a Bayesian network structure. To explain its advantage over graphical representatives we recall here some results from Chapter 8 of [19]. Some of the methods for learning Bayesian network structures are based on maximization of a *quality criterion* $\mathcal{Q}(G,D)$, which is a function of the structure, traditionally represented by a graph G, and of the database D. It is shown in Section 8.4 of [19] that if the criterion satisfies some commonly accepted assumptions, namely that it is decomposable and score-equivalent (cf. [4,6,7]), then it has the form of an affine function of the standard imset. More specifically, given a criterion $\mathcal{Q}$ of this type and provided that u_G denotes the standard imset for G, one has

$$\mathcal{Q}(G,D) = s^{\mathcal{Q}}_D - \langle t^{\mathcal{Q}}_D, u_G \rangle,$$

where $s^{\mathcal{Q}}_D$ is a real number (depending on the database and the criterion) and $t^{\mathcal{Q}}_D$ a vector (of the same dimension as the standard imset u_G), called *data vector* (relative to $\mathcal{Q}$), and $\langle *,* \rangle$ denotes the scalar product. That means the usual task to maximize

☆ This research has been supported by the Grants GAČR No. 201/04/0393, GAČR No. 201/08/0539, GAAVČR No. IAA100750603 and MŠMT No. 1M0572.
* Corresponding author.
E-mail address: vomlel@utia.cas.cz (J. Vomlel).

0888-613X/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2008.09.001

$\mathcal{L}(G,D)$ over G, if D is known, is equivalent to the task to minimize a linear function $u \mapsto \langle t_D^{\mathcal{L}}, u\rangle$ over the set of standard imsets! Thus, it is now evident that this algebraic point of view can simplify some existing methods for learning Bayesian network structures. Actually, we believe this paves the way for future use of efficient linear programming methods in this area.

There are further theoretical advantages of standard imsets. For example, they allow one to characterize (algebraically) the inclusion of Bayesian network statistical models. In particular, one can nearly immediately read from a pair of standard imsets whether two respective Bayesian network structures are neighbors in the sense of *inclusion neighborhood* [10] (see Section 8.4.1 of [19] for details). Another important fact is that standard imsets can be represented in the memory of a computer with polynomial complexity in the number of variables (see Sections 2.3 and 8 for details). Of course, if a standard imset is obtained as the result of a minimization procedure then one is naturally interested in determining the respective (essential) graph. That is why we think one needs some transition between the standard imset and the essential graph.

The topic of this paper is the transition between these two representatives of a Bayesian network statistical model. The transition from a graphical representative to the algebraic one is relatively simple. We give a formula for the standard imset on the basis of any chain graph which defines the respective statistical model. The formula is fairly simple for chain graphs without flags (= without certain special induced subgraphs). In particular, it can easily be applied to the essential graph as well as to any acyclic directed graph in the respective equivalence class.

The inverse procedure is more complex. First, we introduce a series of characteristics of a chain graph equivalent to an acyclic directed graph (= a chain graph defining the same statistical model as an acyclic directed graph). Some of these characteristics appear to be shared by equivalent chain graphs and, moreover, they can all be identified on the basis of the respective standard imset. These characteristics allow us to design a reconstruction algorithm for the essential graph. The final version of the algorithm has two parts. In the first stage, a certain sequence of sets of variables is obtained. The second stage is the proper reconstruction procedure for the essential graph. The algorithm can be implemented with polynomial complexity in the number of variables. A minor modification of the reconstruction algorithm gives a procedure for testing whether a given imset is standard.

The aim of this theoretical (mathematical) paper is to present the proof of the correctness of the reconstruction algorithm. Basic concepts and facts are recalled in Section 2 where we define the standard imset for an acyclic directed graph. A general formula for the standard imset, applicable to any chain graph equivalent to an acyclic directed graph, is given in Section 3. Then in Section 4, the above-mentioned characteristics of chain graphs are introduced. They are related to special algebraic characteristics of the standard imset in Section 5. In Section 6 we formulate lemmas on which the reconstruction algorithm is based. The algorithm is given in Section 7; the complexity issues are discussed in Section 8. In Conclusions we outline a possible future application of the algorithm. The proofs of crucial assertions are in Appendix.

All algorithms described in the paper were implemented as a suite of functions in R. R is a language for statistical computing and graphics. It is available as free software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. See http://www.r-project.org/. The suite of functions for imsets is available at http://staff.utia.cas.cz/vomlel/imset/.

2. Basic concepts

In this section we recall some of the concepts and facts commonly used in the paper.

2.1. Graphical notions

Graphs considered in this paper have a finite non-empty set of *nodes* N and two types of edges. An undirected edge, or a *line over* N, is an unordered pair $\{a,b\}$ where $a,b \in N$, $a \neq b$; we will use the notation $a - b$ then. A directed edge, or an *arrow over* N, is an ordered pair (a,b) where $a,b \in N$, $a \neq b$; we will use the notation $a \to b$ or $b \leftarrow a$. Note that our notation is in accordance with pictorial representations of edges.

In this paper, when we use (hybrid) *graph over* N we mean a graph H which has N as the set of nodes and has no multiple edges, that is,

$$\text{if } a \to b \text{ in } H \quad \text{then } \neg(a \leftarrow b \text{ in } H) \quad \text{and } \neg(a - b \text{ in } H).$$

We say that (an unordered) pair $[a,b]$, $a,b \in N$, $a \neq b$ is an edge in a hybrid graph H over N if either $a - b$ in H, $a \to b$ in H or $a \leftarrow b$ in H. If $a \to b$ in H, then a will be called a *parent* of b (in H); if $a - b$ in H, a will be called a *neighbor* of b (in H). Given a set of nodes A in H, the set of parents (in H) of nodes in A will be denoted by $pa_H(A)$, and the set of neighbors (in H) of nodes in A that are not in A by $ne_H(A)$.

An *undirected path* in H between nodes $a,b \in N$ is a sequence of distinct nodes $c_1,\ldots,c_n$, $n \geqslant 1$ such that $c_1 = a$, $c_n = b$ and $c_i - c_{i+1}$ in H for every $i = 1,\ldots,n-1$. A set of nodes C is *connected* in H if, for every $a,b \in C$, there exists an undirected path between a and b in H. Maximal connected subsets in H with respect to set inclusion are called *components* in H. Of course, the components form a partition of the set N. The class of components in a hybrid graph H will be denoted by $\mathscr{C}(H)$.

If $M \subseteq N$ is non-empty, then the *induced subgraph of* H *for* M is the graph H_M over M that has just those edges $[a,b]$ in H for which $\{a,b\} \subseteq M$; of course, the type of edges in H is kept in H_M.

2.1.1. Undirected graphs

An *undirected graph* is a hybrid graph having only undirected edges, that is, it has no arrows. Given a hybrid graph H over N, its *underlying graph* is an undirected graph G over N such that $a - b$ in G iff $[a,b]$ is an edge in H. A set of nodes K in an undirected graph H is named *complete*, if $a - b$ in H for every $a, b \in K$, $a \neq b$. A maximal complete set in an undirected graph H with respect to set inclusion will be called a *clique* of H. A *complete graph* is an undirected graph such that the set of its nodes N is complete, that is, it has just one clique.

An important concept is that of a *decomposable* (undirected) graph. There are several equivalent definitions of a decomposable graph (see Section 2.1.2 of [11]), one of them is that its cliques can be ordered into a sequence $K_1, \ldots, K_m$, $m \geqslant 1$ satisfying the *running intersection property* (cf. Proposition 2.17(ii) in [11]):

$$\forall i \geqslant 2 \ \exists k < i \quad S_i \equiv K_i \cap \left(\bigcup_{j<i} K_j \right) \subseteq K_k. \tag{1}$$

It is a well-known fact that the collection of sets S_i, $2 \leqslant i \leqslant m$ does not depend on the choice of an ordering that satisfies (1) – see Lemma 7.2 in [19]. We will call these sets *separators* of the graph. Moreover, the multiplicity $v(S)$ of a separator S, that is, the number of indices i for which $S = S_i$, also does not depend on the choice of an ordering that satisfies (1). Note that this definition implies that the class of cliques is disjoint with the class of separators. Another important fact is that every induced subgraph of a decomposable graph is decomposable – see Corollary 2.8 in [11].

Let $\mathscr{K}$ denote the collection of all cliques of a decomposable graph H. A clique K of H will be called a *leaf-clique* or, briefly, a *leaf* (of H) if either $\mathscr{K} = \{K\}$ or there exists $K' \in \mathscr{K} \setminus \{K\}$ such that $K \cap \bigcup(\mathscr{K} \setminus \{K\}) \subseteq K'$. Then the set $R \equiv K \setminus \bigcup(\mathscr{K} \setminus \{K\})$ will be called the *residual* of K; observe that it is necessarily non-empty. An equivalent definition of a leaf is that it is a clique which can occur as the last clique in an ordering that satisfies (1) – see Lemma 9.3 in Section A.3.[1] In particular, at least one leaf exists in $\mathscr{K}$.

2.1.2. Directed graphs

A *directed graph* is a hybrid graph having only directed edges, that is, it has no lines. A *directed cycle* in a hybrid graph G is a sequence $c_1, \ldots, c_n, c_{n+1} \equiv c_1$, $n \geqslant 3$ of nodes in G such that $c_1, \ldots, c_n$ are distinct and $c_i \to c_{i+1}$ in G for $i = 1, \ldots n$. An *acyclic directed graph* is a directed graph which has no directed cycle. A well-known equivalent definition is that it is a directed graph G such that all its nodes can be ordered into a sequence $a_1, \ldots, a_m$, $m \geqslant 1$ such that if $a_i \to a_j$ in G then $i < j$.

We understand a *Bayesian network* (structure) as a statistical model attributed to an acyclic directed graph. More specifically, given an acyclic directed graph G over N we interpret the elements of N as variables. Moreover, if a collection X_i, $i \in N$ of individual sample spaces (that is, non-empty finite sets) for variables is fixed, then one can introduce the corresponding class of probability distributions on $\prod_{i \in N} \mathsf{X}_i$, that is, a *statistical model*. The probability distributions in the class can either be introduced as those which factorize recursively according to G or, equivalently, as those which satisfy conditional independence restrictions given either by the moralization or by the d-separation criterion – see Section 3.2.2 of [11]. Detailed definitions of these concepts are not needed in this paper and are, therefore, omitted.

Terminological remark. Researchers in artificial intelligence usually introduce a Bayesian network, occasionally called a *Bayesian network model*, as a pair: an acyclic directed graph G and a (discrete) probability distribution P which factorizes according to G. That means, P belongs to the class of distributions attributed to G. To name shared features of distributions in the class, they often use the phrase a "Bayesian network structure", but they sometimes omit the word "structure". Some of them even use the phrase "Bayesian network" to name the graph G. On the other hand, researchers in statistics usually use the word "model" to name a parameterized class of distributions. Thus, from a statistical point of view, it would be more appropriate to name the class of distributions attributed to G, which can be nicely parameterized, by a *Bayesian network model*. But, as explained above, this phrase may have another meaning in computer science. In this paper, we make a compromise. Since IJAR journal is read mainly by computer scientists we have decided to use the phrase *Bayesian network structure*, but occasionally we also use the phrase *Bayesian network* ***statistical*** *model*.

2.1.3. Chain graphs

A *semi-directed cycle* in a hybrid graph H is a sequence $c_1, \ldots, c_n, c_{n+1} \equiv c_1$, $n \geqslant 3$ of nodes in H such that $c_1, \ldots, c_n$ are distinct, $c_1 \to c_2$ in H and either $c_i \to c_{i+1}$ in H or $c_i - c_{i+1}$ in H for $i = 2, \ldots n$. A *chain graph* is a hybrid graph without semi-directed cycles. An important consequence of this definition is that every induced subgraph of a chain graph is again a chain graph. Another useful observation is that if C is a connected set in a chain graph H then there is no arrow in H between nodes in C. In particular, H_C is an undirected graph and $pa_H(C)$ is disjoint with C for any $C \in \mathscr{C}(H)$.

The original definition of a chain graph is that it is a hybrid graph whose components can be ordered into a *chain* (see Lemma 2.1 in [17]), that is, into a sequence $C_1, \ldots, C_m$, $m \geqslant 1$ such that

- if $a \to b$ in H then $a \in C_i$ and $b \in C_j$ with $i < j$.

[1] Our terminology is motivated by another equivalent definition, namely a clique which can occur as a leaf in a junction tree of cliques – see Section 4.3 in [8] for the concept of a junction tree.

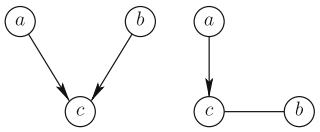

Fig. 1. An immorality and a flag.

It follows from this equivalent definition that every chain graph H has at least one *terminal component*, that is, a component $C \in \mathscr{C}(H)$ such that there is no arrow $a \to b$ in H with $a \in C$. Clearly, every undirected graph and every acyclic directed graph is a chain graph.

2.1.4. Special graphical concepts

Two types of configurations of three nodes in a graph will play an important role in the paper. An *immorality* in a chain graph G is an induced subgraph for a set $T = \{a, b, c\}$, in which $a \to c \leftarrow b$ and $[a, b]$ is not an edge in G. A *flag* in G is an induced subgraph for T, in which $a \to c$, $c - b$ and $[a, b]$ is not an edge. Fig. 1 illustrates these two concepts.

A set of nodes $A \subseteq N$ in a chain graph will be named an *ancestral set*[2] if it is closed under parents and neighbors:

$$\forall a \in A,\ b \in N,\ \text{if } [\ b \to a \text{ in } H \text{ or } b - a \text{ in } H\ ] \text{ then } b \in A.$$

The *moral graph* of a chain graph H over N is an undirected graph H^{mor} over N obtained as follows: $a - b$ in H^{mor} iff either $[a, b]$ is an edge in H or a, b are distinct elements of $pa_H(C)$ for a component $C \in \mathscr{C}(H)$. Note that like every acyclic directed graph every chain graph can be interpreted as a statistical model. The concept of the moral graph (and the concept of an ancestral set) plays a crucial role in the corresponding graphical criterion for determining the respective conditional independence restrictions – see Section 3.2.3 in [11]. Again, the criterion is omitted in this paper because it is not needed here.

Nevertheless, we will utilize the following special concept. If C is a component in a chain graph H then by the *closure graph* for C we will understand the moral graph of the induced subgraph for the set $D = C \cup pa_H(C)$. The closure graph for C will be denoted by $\overline{H}(C)$. Obviously, given $a, b \in D$, one has $a - b$ in $\overline{H}(C)$ if either $[a, b]$ is an edge in H_D or a, b are distinct elements of $pa_H(C)$.

When we say a *super-terminal component* in a chain graph H over N we mean a component C which is a complete set in H_C and $a \to b$ in H for every $a \in N \setminus C$, $b \in C$. Evidently, every component of this kind is terminal and if a chain graph has a super-terminal component then this component is uniquely determined. Actually, if a chain graph H has a super-terminal component then it is a unique terminal component in H.

2.2. Equivalence and essential graphs

Two chain graphs are called *Markov equivalent* if they represent the same statistical model. In standard situations,[3] this requirement is equivalent to the condition that the graphs are *independence equivalent*, that is, they define the same collection of conditional independence restrictions. Verma and Pearl [25] gave a direct graphical characterization of equivalent acyclic directed graphs: they are independence equivalent iff they have the same underlying graph and the same collection of immoralities. Note that Frydenberg [9] achieved an analogous result for chain graphs, but his characterization is more complex since chain graphs define a wider class of structural models. On the other hand, the same equivalence characterization as for acyclic directed graphs holds for chain graphs without flags – see Lemma 2 in [18].

An equivalence class $\mathscr{G}$ of acyclic directed graphs over N can be described by the *essential graph* of $\mathscr{G}$ which is a hybrid graph G^* over N defined as follows:

- $a \to b$ in G^* iff $a \to b$ in G for every $G \in \mathscr{G}$,
- $a - b$ in G^* iff there are $G_1, G_2 \in \mathscr{G}$ such that $a \to b$ in G_1 and $a \leftarrow b$ in G_2.

We say that a graph H over N is *an essential graph over N* if there exists an equivalence class $\mathscr{G}$ of acyclic directed graphs over N such that $H = G^*$.

Example 1. An example of the essential graph of an equivalence class of acyclic directed graphs is given in Fig. 2.

The original graphical characterization of essential graphs was given by Andersson, Madigan and Perlman – see Theorem 4.1 in [1].

[2] Note that it is sometimes called an *anterior set* by other authors [9].

[3] We mean those situations when, for every chain graph, the respective statistical model contains at least one perfectly Markovian distribution. In our case, this can be ensured by the assumption that every individual state space (cf. Section 2.1.2) has at least two distinct states. For further explanation see Section 6.1 of [19].

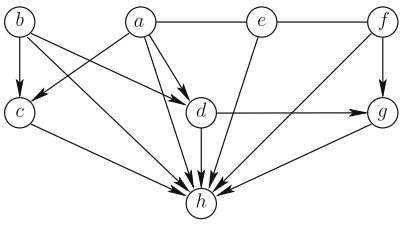

Fig. 2. An example of an essential graph.

Lemma 2.1. *A graph H over N is an essential graph over N iff it is a chain graph without flags, every induced subgraph H_C for $C \in \mathscr{C}(H)$ is decomposable and every arrow $a \to b$ in H belongs to (at least) one of four induced subgraphs from Fig. 3.*

The lemma has the following simple consequence.

Corollary 2.1. *Let H be an essential graph over N and $M \subseteq N$ is a set closed under parents in H. Then the induced subgraph H_M is an essential graph over M.*

Proof 1. We know that every induced subgraph of a chain graph without flags is a chain graph without flags. As an induced subgraph of a decomposable graph is decomposable, the induced subgraphs for components in H_M are decomposable as well. The last condition is also fulfilled for H_M: if $a \to b$ in H_M then $a \to b$ in H belongs to one of the four configurations from Fig. 3. As $\{a, b\} \subseteq M$ the assumption on M implies that $c \in M$, respectively, $c_1, c_2 \in M$. Thus, one of those configurations occurs in H_M as well. □

In this paper, we will utilize an alternative characterization of essential graphs based on a special operation of component merging. Let H be a chain graph without flags and C_u, C_ℓ be two components in H. We say that these two components can be *legally merged* if the following two conditions hold:

{i} $pa_H(C_\ell) \cap C_u$ is a non-empty complete set in H,
{ii} $pa_H(C_\ell) \setminus C_u = pa_H(C_u)$.

The situation is informally illustrated in Fig. 4. The following result was proven as Theorem 2 in [18], and also as Theorem 13 in [15].

Lemma 2.2. *A graph H over N is an essential graph over N iff it is a chain graph without flags such that H_C is decomposable for every component C in H and no pair of components in H can be legally merged.*

Another useful fact is the following one – see Lemma 3 in [18].

Lemma 2.3. *A chain graph H without flags is equivalent to an acyclic directed graph iff, for every component C of H, the induced subgraph H_C is decomposable.*

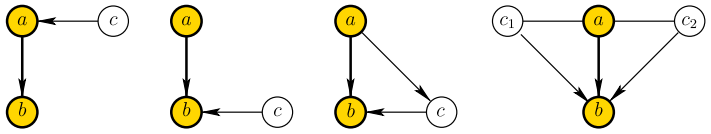

Fig. 3. Four configurations.

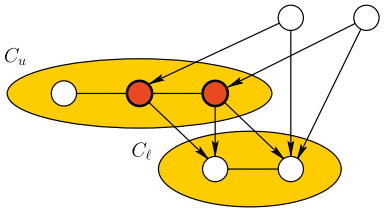

Fig. 4. Legal merging of components.

2.3. The concept of a standard imset

An integer-valued function on the power set $\{A; A \subseteq N\}$ of N will be called an *imset over N*. It can be viewed as a vector whose components are integers indexed by subsets of N. Given $A \subseteq N$, the symbol δ_A will denote an imset identifying this set:

$$\delta_A(B) = \begin{cases} 1 & \text{if } B = A, \\ 0 & \text{otherwise.} \end{cases}$$

However, to describe Bayesian network structures, it is suitable to consider only a certain subclass of the class of imsets. Given an acyclic directed graph G over N, the *standard imset for G* is given by the formula

$$u_G = \delta_N - \delta_\emptyset + \sum_{a \in N} \{\delta_{pa_G(a)} - \delta_{\{a\} \cup pa_G(a)}\}. \tag{2}$$

The point is that the standard imset is a uniquely determined representative of a Bayesian network statistical model. The following result is proven as Corollary 7.1 in [19].

Lemma 2.4. *Two acyclic directed graphs G and H are independence equivalent iff* $u_G = u_H$.

The reader may have some doubts whether the standard imset, which is of an exponential length in the number of variables $n = |N|$, can be represented in the memory of a computer. However, it is evident from (2) that any standard imset has at most $2n$ non-zero values.[4] One can keep in the memory only its non-zero values. The consequence is that the memory demands for representing the standard imset are polynomial in the number of variables, cf. Section 8 for further details.

The phrase "a standard imset" (over N) will mean an imset which can be obtained as the standard imset u_G for an acyclic directed graph G over N. To avoid potential misunderstanding let us emphasize that throughout this paper standard imsets always correspond to Bayesian network structures. The following remark is to explain the motive for our terminology.

Remark 1. In [19] a much wider class of *structural imsets* was introduced. These imsets allow one to describe quite a big class of conditional independence structures, including all (discrete) probabilistic structures. However, it may happen that two different structural imsets are *independence equivalent*, that is, they describe the same conditional independence structure. Then a natural question arises: is there a suitable unique representative of the independence equivalence class of structural imsets? We are obviously interested in a representative which exhibits some elegant properties and can, therefore, serve as a "standard" representative of the equivalence class. The question has the desired positive answer in the framework of Bayesian network structures – see Section 7.2.1 of [19]. The corresponding "standard" representative is the imset given by (2). Note that one can also establish the concept of a standard imset for a decomposable (undirected) graph, but this appears to be a special case of the standard imset representing a Bayesian network structure – see Section 7.2.2 of [19] for details.

3. General formula

In this section we give a formula for the standard imset on the basis of any *chain graph H* over N which is *equivalent to an acyclic directed graph*. Note that the concept of the standard imset for such a chain graph has reasonable sense owing to Lemma 2.4: it is the (shared) standard imset u_G for any acyclic directed graph G equivalent to H.

It is a well-known fact that all closure graphs $\overline{H}(C) \equiv (H_{C \cup pa_H(C)})^{mor}$ for components $C \in \mathscr{C}(H)$ in such a graph are decomposable – see Proposition 4.2 in [2]. Let $\overline{\mathscr{K}}(C)$ denote the collection of cliques of $\overline{H}(C)$ and $\overline{\mathscr{S}}(C)$ the collection of separators of $\overline{H}(C)$. Further, let $\bar{v}_C(\overline{S})$ denote the multiplicity of a separator $\overline{S}$ in $\overline{H}(C)$. The *standard imset* for H is given by the following formula:

$$u_H = \delta_N - \delta_\emptyset + \sum_{C \in \mathscr{C}(H)} \left\{ \delta_{pa_H(C)} - \sum_{\overline{K} \in \overline{\mathscr{K}}(C)} \delta_{\overline{K}} + \sum_{\overline{S} \in \overline{\mathscr{S}}(C)} \bar{v}_C(\overline{S}) \cdot \delta_{\overline{S}} \right\}. \tag{3}$$

If $H = G$ is an acyclic directed graph, then (3) obviously gives the same result as (2). The point is that the formula (3) gives the same result for equivalent chain graphs. The following result is proven as Proposition 20 in [22].

Lemma 3.1. *Let G and H be equivalent chain graphs such that there exists an acyclic directed graph equivalent to them. Then* $u_G = u_H$.

Note that the proof is based on the idea of a certain "feasible" merging operation for components in a chain graph.[5] Thus, Lemma 3.1 has the following consequence.

Corollary 3.1. *Let G be an acyclic directed graph and H the essential graph of the respective equivalence class. Then the formula (3) gives the standard imset for G.*

[4] The formula has $2n + 2$ terms. Moreover, at least one $a^* \in N$ exists with $pa_G(a^*) = \emptyset$. Thus, the terms $-\delta_\emptyset$ and $+\delta_{pa_G(a^*)}$ cancel each other.

[5] This merging operation is different from the legal merging mentioned in Section 2.2 above Lemma 2.2.

Table 1
The standard imset for the essential graph from Fig. 2

A subset of N	The respective value
$\{a,b,c,d,e,f,g\}$	$+1$
$\emptyset$	$+1$
$\{a,b,c\}$	-1
$\{d,f,g\}$	-1
$\{a,b,d\}$	-1
$\{a,e\}$	-1
$\{b\}$	-1
$\{e,f\}$	-1
$\{a,b\}$	$+2$
$\{d,f\}$	$+1$
$\{e\}$	$+1$
Other subsets of $\{a,b,c,d,e,f,g,h\}$	0

Example 2. In Table 1 we give the standard imset for the essential graph from Fig. 2.

4. Graphical characteristics

The formula (3) can be simplified for chain graphs without flags. In this section, we introduce some characteristics of these graphs that will be used in the simplified formula given in Section 5.1. Some of these characteristics appear to be invariants of independence equivalence classes of chain graphs, that is, they are shared by all graphs in such an equivalence class.

4.1. Initial components: $i(H)$

A component $C \in \mathscr{C}(H)$ in a chain graph H such that $pa_H(C) = \emptyset$ will be called an *initial component* in H. Actually, it is a counterpart of the concept of a terminal component. Let us denote by $i(H)$ the number of initial components in H. Note that this number is the same for equivalent chain graphs – this is proven as Proposition 17 in [22]. Obviously, $i(H) \geqslant 1$ for any chain graph H.

Example 3. The chain graph H in Fig. 2 has two initial components; namely $\{b\}$ and $\{a,e,f\}$. In particular, $i(H) = 2$.

4.2. Core and core-components: $core(H)$, $\mathscr{C}_{core}(H)$

We will say that a set B of nodes in a chain graph H over N is *idle* if the following two conditions hold:[6]

(i1) $\forall b_1, b_2 \in B$, $b_1 \neq b_2$, $\quad [b_1, b_2]$ is an edge in H,
(i2) $\forall a \in N \setminus B$, $\forall b \in B$, $\quad a \to b$ in H.

The condition (i2) implies that every component in a chain graph H intersecting an idle set B is contained in B; that is, every idle set is a union of some components.

One can easily show that every chain graph H over N has a unique maximal idle set of nodes (which may be empty) – see Lemma 18 in [22]. Moreover, Proposition 19 in [22] says that the largest idle set is the same for mutually equivalent chain graphs.

Let us call the complement $N \setminus B$ of the largest idle set B in N the *core* of H and denote it by $core(H)$. It follows from the definition that it is an ancestral set. The class of *core-components*, that is, the components in H contained in $core(H)$, will be denoted by $\mathscr{C}_{core}(H)$. Observe that if $core(H) \neq \emptyset$ then every initial component of H is a core-component.

4.3. Cliques and separators (for core-components): $\mathscr{K}(C)$, $\mathscr{S}(C)$

According to Lemma 2.3, if H is a chain graph without flags equivalent to an acyclic directed graph then each of its core-components $C \in \mathscr{C}_{core}(H)$ induces a decomposable graph H_C. Let us denote by $\mathscr{K}(C)$ the class of cliques of H_C, by $\mathscr{S}(C)$ the collection of its separators, and by $\nu_C(S)$ the multiplicity of $S \in \mathscr{S}(C)$ in H_C.

Example 4. The chain graph H from Fig. 2 has two idle sets, namely $\emptyset$ and $\{h\}$. The maximal idle set is $\{h\}$. In particular, the core is $core(H) = \{a,b,c,d,e,f,g\}$. The core-components are $C_1 = \{a,e,f\}$, $C_2 = \{b\}$, $C_3 = \{c\}$, $C_4 = \{d\}$ and $C_5 = \{g\}$. All these

[6] Note an explanation for this terminology: the meaning of these two conditions is that no variable in B is involved in a non-trivial conditional independence restriction given by H. That is why B is called idle.

components except for C_1 have only one clique and no separator in the respective induced subgraph. The set of cliques of H_{C_1} is $\mathscr{K}(C_1) = \{\{a,e\},\{e,f\}\}$ and the set of its separators is $\mathscr{S}(C_1) = \{\{e\}\}$; the multiplicity of the separator $\{e\}$ is 1.

The following basic observation will be utilized later.

Lemma 4.1. *Let H be a chain graph over N without flags which is equivalent to an acyclic directed graph. If $Y \in \mathscr{K}(C) \cup \mathscr{S}(C)$ for some $C \in \mathscr{C}_{core}(H)$, then $\emptyset \neq Y \subseteq Y \cup pa_H(C) \subset core(H)$.*

Proof 2. If $C \in \mathscr{C}_{core}(H)$ and $K \in \mathscr{K}(C)$ is a clique of H_C, then $K \neq \emptyset$ by definition. Moreover, the fact that $core(H)$ is an ancestral set implies $K \cup pa_H(C) \subseteq C \cup pa_H(C) \subseteq core(H)$. Assume for a contradiction that $K \cup pa_H(C) = core(H)$. This implies $K = C$ because $C \setminus K$ is a subset of $core(H)$ disjoint from $K \cup pa_H(C)$. The assumption that H has no flags implies that $a \to b$ for every $b \in K$ and $a \in pa_H(C)$. This fact allows one to show that $K \cup (N \setminus core(H))$ is an idle set, which contradicts the fact that $N \setminus core(H)$ is the largest idle set in H.

If $C \in \mathscr{C}_{core}(H)$ and $S \in \mathscr{S}(C)$ then there exists $K \in \mathscr{K}(C)$ with $S \subseteq K$. Hence, $S \cup pa_H(C) \subseteq K \cup pa_H(C)$, which has been shown to be a proper subset of $core(H)$. Observe that the fact that C is connected implies $S \neq \emptyset$ for any $S \in \mathscr{S}(C)$. Indeed, if $K_1, \ldots, K_m$, $m \geqslant 2$ is an ordering of elements of $\mathscr{K}(C)$ satisfying (1), then the hypothesis $S_m = \emptyset$ implies that there is no edge in H_C between $\bigcup_{j<m} K_j$ and K_m. In particular, there is no undirected path between nodes of these two sets, which contradicts the fact that C is connected. Another observation is that the set $C' \equiv \bigcup_{i=1}^{m-1} K_i$ is also connected because every section of a path in H_C between nodes of C' with internal nodes in $K_m \setminus S_m$ can be shortened by an edge in S_m. This allows us to use an inductive argument to show $S_i \neq \emptyset$ for $i = m, \ldots, 2$. □

Note that Lemma 4.1 need not hold for graphs with flags. This is illustrated by the following example.

Example 5. Let us consider the graph H over $\{a,b,c,d\}$ on the left-hand side of Fig. 5, which has a flag $a \to c - d$. It is equivalent to the acyclic directed graph in the middle of Fig. 5. Consider the component $C = \{c,d\}$ and observe that H_C has just one clique $K = C$. Thus, $K \cup pa_H(C) = \{a,b,c,d\} = core(H)$.

The closure graph $\overline{H}(C)$, which is shown on the right-hand side of Fig. 5 has two cliques, namely $\{a,b,c\}$ and $\{b,c,d\}$. Thus, the example also shows that the cliques of H_C need not correspond to cliques of the closure graph $\overline{H}(C)$. On the other hand, in the case of a chain graph without flags these classes of cliques are in a one-to-one correspondence – cf. the formula (3) with its later simplification (4).

4.4. Parent sets (for core-components): $\mathscr{P}_{core}(H)$

A set $P \subseteq N$ will be called a *parent set* in a chain graph H if it is **non-empty** and there exists a **core-component** $C \in \mathscr{C}_{core}(H)$ with $P = pa_H(C)$. The *multiplicity* $\tau(P)$ of a parent set P is the number of $C \in \mathscr{C}_{core}(H)$ with $P = pa_H(C)$. Let us denote the collection of parent sets in H by $\mathscr{P}_{core}(H)$.

Example 6. The parent sets for the chain graph H from Fig. 2 are $\mathscr{P}_{core}(H) = \{\{a,b\},\{d,f\}\}$. The multiplicities are $\tau(\{a,b\}) = 2$ and $\tau(\{d,f\}) = 1$.

Lemma 4.2. *Let H be a chain graph over N without flags which is equivalent to an acyclic directed graph. Then every $P \in \mathscr{P}_{core}(H)$ is a non-empty proper subset of $core(H)$.*

Proof 3. The requirement $P \neq \emptyset$ is a part of the definition of a parent set. Assuming $P = pa_H(C)$ for $C \in \mathscr{C}_{core}(H)$, the fact that $core(H)$ is an ancestral set implies $P \subset C \cup P \subseteq core(H)$. The first inclusion is strict because C is non-empty and disjoint from P. □

5. Algebraic characteristics

In this section we introduce some algebraic characteristics, more specifically, characteristics that can be read from the standard imset. Then we relate them to graphical characteristics from Section 4.

5.1. Simplified formula

For this purpose we first need to derive a special formula for the standard imset in the case of a graph without flags.

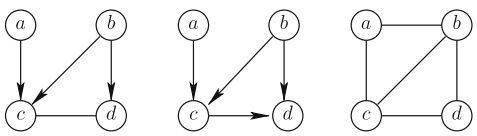

Fig. 5. Graphs from Example 5.

Lemma 5.1. *Let H be a chain graph without flags which is equivalent to an acyclic directed graph. If $core(H) = \emptyset$ then $u_H = 0$. If $core(H) \neq \emptyset$ then the standard imset for H has the form*

$$u_H = \delta_{core(H)} - \sum_{C \in \mathscr{C}_{core}(H)} \sum_{K \in \mathscr{K}(C)} \delta_{K \cup pa_H(C)} + \sum_{C \in \mathscr{C}_{core}(H)} \sum_{S \in \mathscr{S}(C)} v_C(S) \cdot \delta_{S \cup pa_H(C)} + \sum_{P \in \mathscr{P}_{core}(H)} \tau(P) \cdot \delta_P + \{i(H) - 1\} \cdot \delta_\emptyset. \quad (4)$$

In particular, $u_H \neq 0$ iff $core(H) \neq \emptyset$.

The proof is moved to the Appendix – see Section A.1. Lemma 5.1 has the following notable consequence.

Corollary 5.1. *Let u be a non-zero standard imset over N. Then the union M of all sets $L \subseteq N$ with a non-zero imset value $u(L) \neq 0$ also has a non-zero imset value.*[7] *The set M is non-empty and $u(M) = +1$. In fact, $M = core(G^*)$ where G^* is the respective essential graph.*

Proof 4. Assume $u = u_G$ for an acyclic directed graph G over N and put $M \equiv core(G)$. By Lemma 5.1, M is non-empty and the formula (4) holds for $H = G$. It follows from Lemmas 4.1 and 4.2 that any other set $L \subseteq N$ with $u(L) \neq 0$ is a proper subset of M. Finally, recall that the core is the same for equivalent chain graphs: $core(G) = core(G^*)$. □

5.2. Classes of positive and negative sets (for a standard imset)

If u is a non-zero standard imset over N, then by the *core* of u will be meant the largest set $M \subseteq N$ with $u(M) \neq 0$, whose existence was proven in Corollary 5.1. It will be denoted by $core(u)$. Recall that Corollary 5.1 says $core(u) = core(H)$ for the respective essential graph H. If $core(u) = N$ then u will be called *adapted* (to N).

The *class of negative sets* in u is the following class of sets:

$$\mathscr{T}_u = \{L \subseteq N; u(L) < 0\},$$

while by the *class of positive sets* in u will be meant the class

$$\mathscr{L}_u = \{L \subseteq N;\ u(L) > 0, \emptyset \neq L \subset core(u)\}.$$[8]

If $u = 0$ then we accept the convention that $core(u) = \emptyset$ and $\mathscr{T}_u = \mathscr{L}_u = \emptyset$.

The point is that, in the case of a non-trivial essential graph H, the terms in the formula (4) from Lemma 5.1 do not cancel each other.

Lemma 5.2. *Let H be an essential graph over N such that the corresponding standard imset is non-zero: $u_H \neq 0$.*[9] *Then, for every $L \subseteq N$,* **exclusively** *one of these six options applies:*

(a) $L = core(H)$ *and* $u_H(L) = +1$,
(b) $L = K \cup pa_H(C)$ *for* $K \in \mathscr{K}(C)$, $C \in \mathscr{C}_{core}(H)$ *and* $u_H(L) = -1$,
(c) $L = S \cup pa_H(C)$ *for* $S \in \mathscr{S}(C)$, $C \in \mathscr{C}_{core}(H)$ *and* $u_H(L) = v_C(S) > 0$,
(d) $L = P$ *for* $P \in \mathscr{P}_{core}(H)$ *and* $u_H(L) = \tau(P) > 0$,
(e) $L = \emptyset$ *and* $u_H(L) = i(H) - 1 \geqslant 0$,
(f) none of above cases occurs and $u_H(L) = 0$.

Moreover, for every set L of the type (b) only one pair $C \in \mathscr{C}_{core}(H)$, $K \in \mathscr{K}(C)$ exists such that $L = K \cup pa_H(C)$, and an analogous statement holds for sets of the type (c).

The proof is shifted to the Appendix – see Section A.2. Observe that Lemma 5.2 implies that if $u \equiv u_H \neq 0$ for an essential graph H then the class of negative sets in u takes the form

$$\mathscr{T}_u = \{K \cup pa_H(C); K \in \mathscr{K}(C), C \in \mathscr{C}_{core}(H)\} \equiv \mathscr{K}_H \quad (5)$$

and the class of positive sets in u has the following form:[10]

$$\mathscr{L}_u = \mathscr{S}_H \cup \mathscr{P}_{core}(H), \quad \text{where } \mathscr{S}_H \equiv \{S \cup pa_H(C); S \in \mathscr{S}(C), C \in \mathscr{C}_{core}(H)\}. \quad (6)$$

Thus, it follows from Corollary 5.1 and Lemma 5.2 that, given a non-zero standard imset (for an acyclic directed graph), one can immediately determine the core of the respective essential graph H, the number of its initial components, the class $\mathscr{K}_H$ and the class $\mathscr{S}_H \cup \mathscr{P}_{core}(H)$.

[7] In other words, the class of sets with non-zero imset value has the largest set M.

[8] The reader may wonder why we have excluded the core and the empty set from the class $\mathscr{L}_u$. This is because these two sets play a different special role both in our classification of sets in Lemma 5.2 and in the later reconstruction algorithm in Section 7.

[9] Recall that, by Lemma 5.1, this is equivalent to the condition $core(H) \neq \emptyset$. Actually, it follows from (3) and later Lemma 6.1 that it is also equivalent to the condition that H is not a complete undirected graph.

[10] Recall that $core(u) = core(H)$ and that both $\emptyset$ and $core(u)$ are **excluded** from the class $\mathscr{L}_u$ of positive sets.

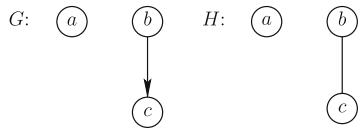

Fig. 6. Graphs from Example 7.

Note that if we analogously introduce the classes of sets $\mathscr{K}_G$ and $\mathscr{S}_G \cup \mathscr{P}_{core}(G)$ for any chain graph G equivalent to an acyclic directed graph then they need not be invariants of independence equivalence as the Example 7 below shows. Thus, the set $\mathscr{T}_u$ and $\mathscr{L}_u$ determined by the imset u are related by the relations (5) and (6) **only** to the respective essential graph; these relations need not hold for other chain graphs in the same equivalence class:

Example 7. Consider graphs G and H from Fig. 6. These are equivalent chain graphs without flags. However, $\mathscr{K}_G$ consists of three sets, namely, $\{a\}$, $\{b\}$ and $\{b,c\}$ while $\mathscr{K}_H$ has only $\{a\}$ and $\{b,c\}$. Analogously, $\mathscr{S}_G \cup \mathscr{P}_{core}(G)$ contains $\{b\}$ but $\mathscr{S}_H \cup \mathscr{P}_{core}(H)$ is empty.

6. Theoretical basis for the reconstruction

Lemma 5.2 is the basis of a two-stage reconstruction algorithm for the essential graph H based on the standard imset $u = u_H$. The main idea is to transform the problem to a simpler one by reducing the cardinality of the class of negative sets $\mathscr{T}_u$ ($\equiv \mathscr{K}_H$). Indeed, the reconstruction procedure is trivial if $|\mathscr{T}_u| = 0$ because then $u = 0$.

Lemma 6.1. *Let $u = 0$ be a zero standard imset over N. Then the respective essential graph is the complete undirected graph over N.*

Proof 5. The first step is to show that the largest idle set $B \subseteq N$ (see Section 4.2) in an essential graph H over N is a component in H, which necessarily consists of one clique. Thus, assume for contradiction that H_B has at least two components and order them into a chain $C_1, \ldots, C_k$, $k \geqslant 2$. Then put $C_u = C_{k-1}$, $C_\ell = C_k$ and observe that the conditions {**i**} and {**ii**} from Section 2.2 are fulfilled: $pa_H(C_\ell) \cap C_u = C_u = C_{k-1}$ is non-empty complete in H and $pa_H(C_\ell) \setminus C_u = core(H) \cup \bigcup_{j=1}^{k-2} C_j = pa_H(C_u)$. In particular, C_u and C_ℓ can be legally merged, which contradicts, by Lemma 2.2, the assumption that H is an essential graph.

Now, by Lemma 5.1, $u = 0$ means $core(H) = \emptyset$, that is, N is an idle set. In particular, it is a component in H. Therefore, H is an undirected graph consisting of one clique. □

Another partial step is an adaptation of a standard imset, that is, a restriction of the imset to subsets of its core.

Lemma 6.2. *Let $u = u_H$ be the standard imset corresponding to an essential graph H over N with $\emptyset \neq core(u) \equiv M \subset N$. Then the restriction w of u to subsets of M is an adapted standard imset over M. Moreover, provided G denotes the essential graph for w, the graph H can be obtained from G as follows:*

- $H_M = G$,
- $\forall x \in M,\ \forall y \in N \setminus M, \quad x \to y$ in H,
- $\forall y_1, y_2 \in N \setminus M, y_1 \neq y_2, \quad y_1 - - y_2$ in H.

Proof 6. By Corollary 5.1 we know that M is the core of H. It was shown in the proof of Lemma 6.1 that the largest idle set $N \setminus M$ consists of one component in H. Moreover, M is an ancestral set in H and this implies, by Corollary 2.1, that H_M is an essential graph over M.

Let B be the largest idle set in H_M. Observe that $B = \emptyset$ for otherwise $B \cup (N \setminus M)$ would be an idle set in H, which would contradict the assumption that $N \setminus M$ is the largest idle set in H. Recall that the fact $B = \emptyset$ means $core(H_M) = M$. Let w be the standard imset (over M) corresponding to H_M. Because $core(w) = core(H_M) = M$, w is adapted to M. The formula (4) from Lemma 5.1 applied to H and H_M implies that u and w have the same values for subsets of M, while u vanishes for sets that are not subsets of M. Since $N \setminus M$ is a complete idle set in H the graph H can be obtained from $G = H_M$ as described in Lemma 6.2. □

The main step of the reconstruction procedure is the reduction of the class of negative sets in u, which is accompanied with the respective reduction of the set of variables. Let us assume for simplicity that the imset u is adapted, that is, $core(u) = N$. The idea is to "remove" at least one set T from the class $\mathscr{T}_u$, which is known by (5) to have the form

$$\mathscr{T}_u = \{K \cup pa_H(C);\ C \in \mathscr{C}(H),\ K \in \mathscr{K}(C)\} \equiv \mathscr{K}_H.$$

Nevertheless, not every set T in this class can be removed immediately.[11] The sets in $\mathscr{T}_u$ that can simply be removed have a special form. For example, the reduction procedure is possible if T has the form $T = K \cup pa_H(C)$ where C is a terminal component in H and K is a leaf-clique of H_C. It is clear that a set of this kind always exists in $\mathscr{K}_H = \mathscr{T}_u$.

[11] Note that we wish to have a "removal" of a type such that the resulting "reduced" imset is again a standard imset over a (strict) subset of the set of variables.

Later in this paper we consider a slightly weaker condition, which can, moreover, be formulated in terms of the algebraic characteristics of u from Section 5.2. More specifically, the condition that C is a terminal component can be replaced by a weaker requirement that there is no arrow in H from the residual of the clique K. Formally, we have in mind the following condition (see Fig. 7 to illustrate):

$$\left.\begin{array}{ll} T = K \cup pa_H(C) & \text{where } C \in \mathscr{C}(H), \\ & K \in \mathscr{K}(C) \text{ is a leaf-clique of } H_C \\ & \text{and there is no arrow in } H \text{ from } R \equiv K \setminus \bigcup\limits_{K' \in \mathscr{K}(C) \setminus \{K\}} K'. \end{array}\right\} \tag{7}$$

Here is the respective result.

Lemma 6.3. *Let u be an adapted standard imset over N and H the respective essential graph. Assume that $T \in \mathscr{T}_u$ has the form (7) and $R \equiv K \setminus \bigcup_{K' \in \mathscr{K}(C) \setminus \{K\}} K'$ is the respective residual. Then there is no edge in H between R and $N \setminus T$. Let us put $M = N \setminus R$. The imset*

$$\tilde{w} = u - \delta_N - \delta_{T \setminus R} + \delta_T + \delta_M$$

vanishes for sets that are not subsets of M. The restriction w of $\tilde{w}$ to subsets of M is a standard imset over M and the respective essential graph is H_M.

The proof of Lemma 6.3 is moved to the Appendix – see Section A.3. Note that the resulting imset w need not be adapted (to M) as the following example shows.

Example 8. Let H be the graph over $N = \{a, b, c, d, e\}$ from Fig. 8. It is easy to see that H is an essential graph over N. The respective standard imset is u in Table 2. The set $T = \{a, c, e\}$ has the form (7), namely $C = K = \{e\}$ and $pa_H(C) = \{a, c\}$. Therefore, $R = \{e\}$ and $M = \{a, b, c, d\}$. The respective imset w over M is also shown in Table 2. Evidently, $core(w) = \{a, b, c\}$.

Thus, all we need for performing the reduction procedure described in Lemma 6.3 is to find $T \in \mathscr{T}_u$ satisfying (7) and determine the respective residual R. The following lemma says how to determine them on the basis of the standard imset u.

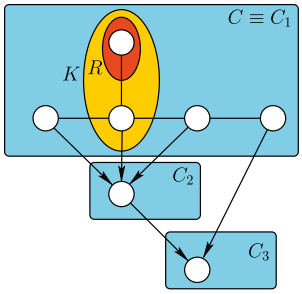

Fig. 7. A leaf-clique K of H_C with the residual R such that there is no arrow from R.

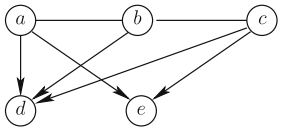

Fig. 8. The essential graph from Example 8.

Table 2
The standard imsets from Example 8

A subset of N	u	w over $\{a, b, c, d\}$
$\{a, b, c, d, e\}$	+1	0
$\{a, b\}$	−1	−1
$\{b, c\}$	−1	−1
$\{a, b, c, d\}$	−1	0
$\{a, c, e\}$	−1	0
$\{b\}$	+1	+1
$\{a, b, c\}$	+1	+1
$\{a, c\}$	+1	0
Other subsets of $\{a, b, c, d, e\}$	0	0

Lemma 6.4. *Let u be an adapted standard imset over N and H the respective essential graph. Let us denote by W_u the set of nodes which belong to at least two negative sets for u:*

$$W_u = \{x \in N;\ \exists T, T' \in \mathscr{T}_u,\ T \neq T' \text{ with } x \in T \cap T'\}.$$

Then W_u coincides with the union $\bigcup \mathscr{L}_u$ of the class of positive sets (for u). Moreover, given a negative set $T \in \mathscr{T}_u$, the condition (7) is equivalent to the condition

$$T \setminus W_u \neq \emptyset \quad \text{and} \quad \{\text{either } T \cap W_u = \emptyset \text{ or } T \cap W_u \in \mathscr{L}_u\}. \tag{8}$$

If this condition holds, then the respective residual from (7) takes the form $R = T \setminus W_u$.

The proof is shifted to the Appendix – see Section A.4. Lemma 6.4 has the following simple consequence.

Corollary 6.1. *Let u be a non-zero standard imset and $W_u = \bigcup \mathscr{L}_u$. Then there exists $T \in \mathscr{T}_u$ such that $T \cap W_u \in \mathscr{L}_u \cup \{\emptyset\}$ and $R \equiv T \setminus W_u \neq \emptyset$.*

Proof 7. Let us assume without loss of generality that u is adapted (see Lemma 6.2). As explained before Lemma 6.3, at least one $T \in \mathscr{T}_u$ exists which satisfies (7). By Lemma 6.4 it also satisfies (8). □

Lemma 6.3 has not completely answered the question of how to get the essential graph H on the basis of H_M and T. It says that the only edges to nodes in R are those within T but it does not say what are the types of these edges. The following example shows that these edges cannot be determined solely on the basis of the partition of T into $T \setminus R$ and R. It also indicates that the specification of these edges is not a local problem, in the sense that the edges cannot be determined simply on the basis of $H_{T \setminus R}$ or on the basis of the restriction of u to subsets of T.

Example 9. Let us consider the graphs G and H shown in Fig. 9. It is easy to verify using Lemma 2.1 that they are both essential graphs over $N = \{a, b, c, d, e, f, g\}$. The respective standard imsets are given in Table 3. They are quite similar. For example, the sets W_u from Lemma 6.4 are as follows: $W_{u_G} = \{b, c, e, f\}$ and $W_{u_H} = \{b, c, f\}$.

Consider the set $T = \{b, c, f, g\}$. It belongs to the class of negative sets for both standard imsets in Table 3 and satisfies the condition (8); the respective residual is $R = \{g\}$ (in both cases). Moreover, the induced subgraphs of both essential graphs for $T \setminus R = \{b, c, f\}$ coincide. On the other hand, H_T differs from G_T because one has $f - g$ in H and $f \to g$ in G.

The restrictions of considered standard imsets to subsets of T are identical – see the lower part of Table 3. This certainly does not directly indicate what type the edge should be between f and g.

Nevertheless, although the problem of reconstruction of the essential graph H is not local, H can be obtained on the basis of H_M and T. The following lemma allows us to distinguish two basic cases, namely whether $T \setminus R$ is a parent set or not.

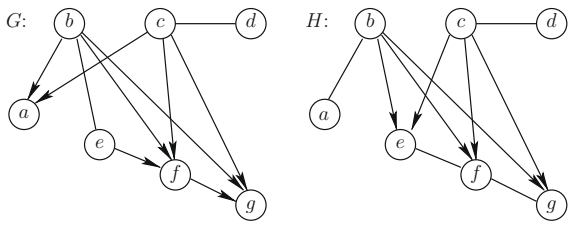

Fig. 9. The essential graphs from Example 9.

Table 3
The standard imsets for the essential graphs from Fig. 9

A subset of N	u_G	u_H
$\{a, b, c, d, e, f, g\}$	+1	+1
$\{a, b, c\}$	−1	0
$\{a, b\}$	0	−1
$\{b, e\}$	−1	0
$\{c, d\}$	−1	−1
$\{b, c, e, f\}$	−1	−1
$\{b, c, e\}$	+1	0
$\{b, c, f, g\}$	−1	−1
$\{b, c, f\}$	+1	+1
$\{b, c\}$	+1	+1
$\{b\}$	0	0
$\emptyset$	+1	+1
Other subsets of $\{a, b, c, d, e, f, g\}$	0	0

Lemma 6.5. *Let u be an adapted standard imset over N and H be the respective essential graph. Assume that $C \in \mathscr{C}(H)$ and $T = K \cup pa_H(C)$, $K \in \mathscr{K}(C)$ satisfies the condition (7). Let R denote the respective residual $K \setminus \bigcup(\mathscr{K}(C) \setminus \{K\})$; put $M = N \setminus R$. Then* **exclusively** *one of the following two cases applies:*

(p) $T \setminus R = pa_H(C)$,
(s) $T \setminus R = S \cup pa_H(C)$ *for* $S \in \mathscr{S}(C)$.

*Moreover, again under the assumption (7), the second case (***s***) is equivalent to the condition:*

$$T \setminus R \neq \emptyset \text{ and } H_{T\setminus R} \text{ has a complete component } X \text{ such that } pa_{H_M}(X) = T \setminus (R \cup X). \tag{9}$$

If (9) holds then $S = X$.

The proof is moved to Appendix – see Section A.5.

7. Reconstruction algorithm

The reconstruction algorithm for the essential graph on the basis of the standard imset is presented in this section. We first introduce basic steps of the algorithm. Then we describe it in the form of a recursive procedure. This is because it is easier to prove its correctness in this form. Later we formulate the algorithm in the form of a two-stage non-recursive procedure and illustrate it by means of an example. Finally, we give a modification of the recursive procedure that can be used to test whether a given imset is standard.

7.1. Basic subroutines

In the algorithm, we distinguish three cases: that of a zero imset, a non-zero non-adapted imset, and an adapted (non-zero) imset. Let us describe and comment on subroutines which are used if the imset is non-zero. We start with the adaptation step based on Lemma 6.2, which will be used in the case of a non-zero non-adapted imset. It is described in Table 4.

The subroutine is formulated in such a way that it works for any non-zero imset. However, Corollary 5.1 says that if u is a non-zero standard imset then the set M found in line 1 of Table 4 is uniquely determined; it is the core of u. Thus, if u is adapted then $w = u$ and if u is not adapted then w is the "restricted" standard imset mentioned in Lemma 6.2.

In the crucial case of an adapted standard imset, the imset is reduced on the basis of Lemma 6.3 (and Lemma 6.4). This is described in Table 5.

Table 4
Subroutine **Adapt**(u over $N|M, w$ over M)

Input:	u ... a standard imset over a non-empty set of variables N
Output:	M ... a subset of N
	w ... an adapted standard imset over M
1	find a maximal set $M \subseteq N$ with respect to inclusion with $u(M) \neq 0$;
2	`put` $w =$ the restriction of u to the class of subsets of M;
3	`return` M, w;

Table 5
Subroutine **Reduce**(u *over* $N|T$, M, w *over* M)

Input:	u ... an adapted standard imset over a non-empty set of variables N
Output:	T ... a proper subset of the set of variables N
	M ... a proper subset of N such that $M \cup T = N$ and $T \setminus M \neq \emptyset$
	w ... a standard imset over M
1	`put` $\mathscr{T} = \{L \subseteq N; u(L) < 0\}$;
2	$\mathscr{L} = \{L \subset N; u(L) > 0,\ L \neq \emptyset\}$;
3	$W = \bigcup \mathscr{L}$;
4	find (one arbitrary) $T \in \mathscr{T}$ such that $T \setminus W \neq \emptyset$ and $T \cap W \in \mathscr{L} \cup \{\emptyset\}$;
5	`put` $R = T \setminus W$;
6	$M = N \setminus R$;
7	$\bar{w} = u - \delta_N + \delta_M + \delta_T - \delta_{T\setminus R}$;
8	$w =$ the restriction of $\bar{w}$ to the class of subsets of M;
9	`return` T, M, w;

Table 6
Subroutine **Extend**(G over $M, T|H$ over $M \cup T$)

Input:	G ... a chain graph over a non-empty set of variables M
	T ... a set of variables with $T \setminus M \neq \emptyset$
Output:	H ... a chain graph over $M \cup T$
1	put $L = T \cap M$;
2	$R = T \setminus M$;
3	$Z = \emptyset$;
4	if $L \neq \emptyset$ then choose a terminal component X in G_L, and
5	if X is a complete component in G_L and $pa_G(X) = L \setminus X$ then put $Z = X$;
6	determine the edges in H as follows:
7	$H_M = G$,
8	$\forall y \in L \setminus Z$, $\forall z \in R$ include $y \to z$ in H,
9	$\forall x \in R \cup Z$, $z \in R$, $x \neq z$ include x—z in H;
10	return H;

Table 7
Recursive algorithm **Reconstruct**(u over $N|H$ over N)

Input:	u ... a standard imset over a non-empty set of variables N
Output:	H ... a chain graph over N
1	if $u = 0$ then put H = a complete undirected graph over N;
2	return H;
3	if $u \neq 0$ and $u(N) = 0$ then **Adapt**(u over $N|M, w$ over M);
4	**Reconstruct**(w over $M|G$ over M);
5	**Extend**(G over $M, N|H$ over N);
6	return H;
7	if $u(N) \neq 0$ then **Reduce**(u over $N|T, M, w$ over M);
8	**Reconstruct**(w over $M|G$ over M);
9	**Extend**(G over $M, T|H$ over N);
10	return H;

Note that Corollary 6.1 implies that if u is an adapted non-zero standard imset then there exists a set T satisfying the condition given in line 4 of Table 5.[12] It is easy to see that this condition is simply the condition (8) from Lemma 6.4, which is known to be equivalent to the condition (7) from Lemma 6.3. Let us emphasize that the set T chosen in line 4 is also an output of the subroutine; it will play an important role in the reconstruction of the essential graph. Note that every $T \in \mathscr{T}$ is a proper subset of N due to Corollary 5.1 and the resulting imset w is a standard imset according to Lemma 6.3.

A procedure dual to both of the above-specified ones is the extension step based on Lemma 6.5. It is used to reconstruct an essential graph over N on the basis of its induced subgraph for a subset $M \subset N$, and a special set $T \subseteq N$ with $M \cup T = N$. Depending on the situation, the procedure either extends an existing component by adding a new clique to it or creates a new component consisting of one clique. It is formally described in Table 6.

The procedure **Extend** is formally introduced and works for any chain graph on the input, but in the context of the routines which call it in this paper, it is always applied to an essential graph G. In this case, the set X satisfying the condition in line 5 of Table 6 has to be the super-terminal component in G_L because G has no flags. Note that the condition in line 5 is equivalent to the condition (9) from Lemma 6.5. The condition implies that X has to be the unique terminal component in G_L; therefore, it does not matter which terminal component X has been chosen in line 4 of Table 6.

Lemma 7.1. *Let G be a chain graph over $M \subset N$ and $T \subseteq N$ such that $T \setminus M \neq \emptyset$ and $M \cup T = N$. Then the graph H over N resulting from the procedure* **Extend**$(G, T|H)$ *is a chain graph as well. Moreover, if G is an essential graph then H is an essential graph, too.*

The proof is shifted to Appendix – see Section A.6.

7.2. Recursive formulation

In this subsection we formally introduce the entire reconstruction algorithm in a recursive form and formulate a result on its correctness. The procedure is described in Table 7.

Clearly, at least one of the cases from lines 1, 3 or 7 of Table 7 has to occur for a given standard imset u. Another observation is that the sets T and M resulting from the subroutine **Reduce** applied in line 7 satisfy $M \cup T = N$ and $T \setminus M \neq \emptyset$. The latter fact implies that the subroutine **Extend** in line 9 results in a graph over N. The main result is then formulated as follows.

[12] The subroutine will not work for input imsets that are not standard imsets unless we introduce an additional stopping rule – for details see Section 7.4.

Table 8
The first stage **Decompose** (u over $N|\tau$ over N)

Input:	u ... a standard imset over a non-empty set of variables N
Output:	τ... an ordered sequence of subsets of N
1	put $Y = N$;
2	τ = empty list;
3	if $u = 0$ then append Y as the last item in τ;
4	return τ;
5	if $u \neq 0$ and $u(Y) = 0$ then **Adapt** (u over $Y\|M, w$ over M);
6	append Y as the last item in τ;
7	go to 11;
8	if $u(Y) \neq 0$ then **Reduce** (u over $Y\|T, M, w$ over M);
9	append T as the last item in τ;
10	go to 11;
11	put $Y = M$;
12	$u = w$;
13	go to 3;

Table 9
The second stage **Compose** (τ over $N|H$ over N)

Input:	τ ... an ordered sequence $T_1, \ldots, T_n$, $n \geqslant 1$ of subsets of N satisfying (10)
Output:	H ... a chain graph over M
1	put $M = T_n$;
2	H = a complete undirected graph over M;
3	$G = H$;
4	for $j = n-1, \ldots, 1$ do **Extend** (G over $M, T_j\|H$ over $M \cup T_j$);
5	$M = M \cup T_j$;
6	$G = H$;
7	return H;

Theorem 1. *Let u be a standard imset over N. Then the result of the recursive algorithm from Table 7 is the respective essential graph H over N.*

The proof is moved to Appendix – see Section A.7.

7.3. Non-recursive formulation

In this subsection we formulate the reconstruction algorithm in a non-recursive way. There are two reasons for this approach. First, we believe that the non-recursive formulation is more transparent from the algorithmic point of view because it describes the actual order in which the subroutines are applied. Therefore, it can be easily illustrated by an example. Second, the non-recursive formulation reveals possible relationships of standard imsets and essential graphs to other potential ways of representing Bayesian network structures (see the Conclusions and [23] for more discussion).

To establish a non-recursive version of the algorithm we first need to recall the recursive formulation. It can be deduced from its description in Table 7 that in order to be able to reconstruct the desired essential graph we must remember the order in which subroutines **Extend** were applied and what their input sets were.[13] This is the main idea behind the non-recursive formulation of the reconstruction algorithm. We simply let the reconstruction algorithm run without applying the subroutines **Extend** and keep the record of their input sets instead. This is a decomposition phase. Then, in the second phase, we repeatedly apply the subroutine **Extend** to compose the essential graph. The decomposition phase of the algorithm is described in Table 8.

The result of the first phase is a sequence of input sets for **Extend** subroutines in the desired order. Observe that the resulting sequence $\tau : T_1, \ldots, T_n$, $n \geqslant 1$ is such that[14]

$$R_i \equiv T_i \Big\backslash \bigcup_{j>i} T_j \neq \emptyset \quad \text{for } i = 1, \ldots, n \quad \text{and} \quad \bigcup_{i=1}^{n} T_i = N. \tag{10}$$

The second stage of the reconstruction algorithm is presented in Table 9.

[13] Recall that the input graph for the first application of **Extend** has to be a complete undirected graph.

[14] The reason for $T_i \setminus \bigcup_{j>i} T_j \neq \emptyset$ is this: if we append $T = T_i$ as the provisional last item in τ (either in line 6 or in line 9 of Table 8 then the sets appended later to τ are contained in the respective set M obtained by calling the preceding subroutine (either in line 5 or in line 8). However, if **Adapt** subroutine is called then M is a proper subset of $T \equiv Y$ while if **Reduce** subroutine is called then one has $T \setminus M \neq \emptyset$ – cf. Table 5.

The intention here is that the input sequence of sets τ is the output of the first stage. Elements in the sequence τ are processed in reverse order. As explained above, it follows from comparison with the recursive procedure **Reconstruct** from Table 7 that the two-stage procedure gives the same result. In particular, Theorem 1 gives the following consequence.

Corollary 7.1. *Let u be a standard imset over N. If the procedure* **Decompose** *from Table 8 is applied to u and the procedure* **Compose** *from Table 9 is applied to the resulting ordered sequence τ, then the overall result is the respective essential graph H over N.*

An additional observation is that the result of the composition phase is always an essential graph.

Corollary 7.2. *Let τ be an ordered sequence of subsets of a non-empty set of variables N such that condition (10) is satisfied. Then the result of the procedure* **Compose** *from Table 9 is an essential graph over N.*

Proof 8. We can prove this result by induction on the length n of the sequence τ. Since the first "iteration" is the complete undirected graph over T_n, which is an essential graph according to Lemma 2.1, the induction hypothesis is valid. The induction step follows from Lemma 7.1. □

Remark 2. By Corollary 7.2, the procedure **Compose** results in an essential graph; Corollary 7.1 implies that every essential graph can be obtained in this way. In particular, a necessary and sufficient condition for a graph to be an essential graph over N is that it can be obtained by the procedure **Compose** applied to a sequence of sets satisfying (10). Nevertheless, this observation cannot be directly used to compute the exact number of essential graphs over N because there is no one-to-one correspondence between sequences of sets satisfying (10) and essential graphs over N. Indeed, two different sequences may result in the same essential graph as this example shows: put $N = \{a, b\}$, $\tau_1 : T_1 = N$ and $\tau_2 : T_1 = N, T_2 = \{a\}$. The resulting graph is a complete undirected graph over N in both cases.

Let us illustrate the overall procedure by means of examples.

Example 10. Let us consider the standard imset from Table 1 and apply the procedure **Decompose** from Table 8. As u is non-empty and non-adapted, in the first round of the procedure the subroutine **Adapt** is applied. Thus, $N = \{a, b, c, d, e, f, g, h\}$ becomes the first item T_1 in τ.

In the second round, the input imset w_1 is already adapted to $Y = \{a, b, c, d, e, f, g\}$ – see the respective column in Table 10. Now, the subroutine **Reduce** is applied. Note that two sets from the class $\mathscr{T}$ satisfy the condition in line 4 of Table 5, namely $\{a, b, c\}$ and $\{d, f, g\}$. Let us choose the set $\{a, b, c\}$ as the second item T_2 in τ. The resulting imset w_2 is again adapted to $Y = M = \{a, b, d, e, f, g\}$ – see Table 10.

In the third round, the reduction procedure is repeated and the only candidate for the set T is $\{d, f, g\}$. Thus, it becomes T_3 in τ and the resulting imset is w_3 from Table 10, which is again adapted.

Thus, in the fourth round, subroutine **Reduce** has to be applied. We have two candidates for the last item in τ, namely $\{a, b, d\}$ and $\{e, f\}$. Let us choose $\{a, b, d\}$ as the set T_4. The resulting imset is w_4 in the next column of Table 10.

As w_4 is adapted to $Y = \{a, b, e, f\}$, again **Reduce** is called in the fifth round and there are even three options for the set T: $\{a, e\}$, $\{b\}$ and $\{e, f\}$. Let us choose $\{a, e\}$ as T_5 in τ.

In the sixth round, when w_5 is processed, there are two options for the set T in the reduction procedure, namely $\{b\}$ and $\{e, f\}$. Note that in both cases $T \cap W$ is empty unlike in the preceding rounds. If we choose $\{b\}$ in the role of T_6 then the resulting imset w_6 is the zero imset over $\{e, f\}$ – see the last column in Table 10.

Table 10
Standard imsets from Example 10

Imset	w_1	w_2	w_3	w_4	w_5	w_6 over $\{e,f\}$
$\{a, b, c, d, e, f, g\}$	+1	0	0	0	0	0
$\{a, b, d, e, f, g\}$	0	+1	0	0	0	0
$\{a, b, d, e, f\}$	0	0	+1	0	0	0
$\{a, b, e, f\}$	0	0	0	+1	0	0
$\{b, e, f\}$	0	0	0	0	+1	0
$\emptyset$	+1	+1	+1	+1	+1	0
$\{a, b, c\}$	−1	0	0	0	0	0
$\{d, f, g\}$	−1	−1	0	0	0	0
$\{a, b, d\}$	−1	−1	−1	0	0	0
$\{a, e\}$	−1	−1	−1	−1	0	0
$\{b\}$	−1	−1	−1	−1	−1	0
$\{e, f\}$	−1	−1	−1	−1	−1	0
$\{a, b\}$	+2	+1	+1	0	0	0
$\{d, f\}$	+1	+1	0	0	0	0
$\{e\}$	+1	+1	+1	+1	0	0
Other subsets of Y	0	0	0	0	0	0

Finally, in the seventh round we just append $Y = \{e,f\}$ as the last item T_7 in τ. The resulting sequence is as follows.

$$\left.\begin{array}{l} T_1 = \{a,b,c,d,e,f,g,h\} \\ T_2 = \{a,b,c\} \\ T_3 = \{d,f,g\} \\ T_4 = \{a,b,d\} \\ T_5 = \{a,e\} \\ T_6 = \{b\} \\ T_7 = \{e,f\} \end{array}\right\} \tag{11}$$

The composition phase of the reconstruction algorithm is illustrated by means of a separate example.

Example 11. Let us consider the sequence of sets of variables from (11). Observe that it satisfies the condition (10) with $N = \{a,b,c,d,e,f,g,h\}$. The sequence is processed by the procedure from Table 9 in the reverse order.

Thus, we begin with $T_7 = \{e,f\}$ and the starting "iteration" will be the complete undirected graph over $\{e,f\}$ – see the graph G^6 in Fig. 10. Then, the subroutine **Extend** is applied to this graph and the set $T_6 = \{b\}$. This time the set L in line 1 of Table 6 is empty and no edge is included in steps 8 and 9 of the subroutine **Extend**. The resulting graph is G^5 from Fig. 10.

Now, the extension procedure is applied to that graph and $T_5 = \{a,e\}$. This time $L = \{e\}$ is non-empty and the condition in line 5 of Table 6 is fulfilled with $X = \{e\}$. For this reason we include a line between a and e in the resulting graph G^4 over $\{a,b,e,f\}$.

The next set to be processed is $T_4 = \{a,b,d\}$. In subroutine **Extend** one has $L = \{a,b\}$ and the induced subgraph for L has two terminal components. In particular, the condition in line 5 of Table 6 is not satisfied and all included edges are arrows directed towards d. A similar situation occurs when processing T_3, T_2 and T_1. The overall resulting graph G^0 is the graph from Fig. 2, which was expected.

7.4. Testing whether an imset is standard

The reconstruction algorithm can be modified to get a procedure for testing whether a given imset is a standard one. The modification is described in this subsection. We base it on the recursive version of the algorithm from Section 7.2. The basic idea is that the subroutines **Adapt** and **Reduce** are extended by suitable stopping rules. A further change is that the subroutine **Extend** is omitted. The output of the whole testing procedure is a logical variable.

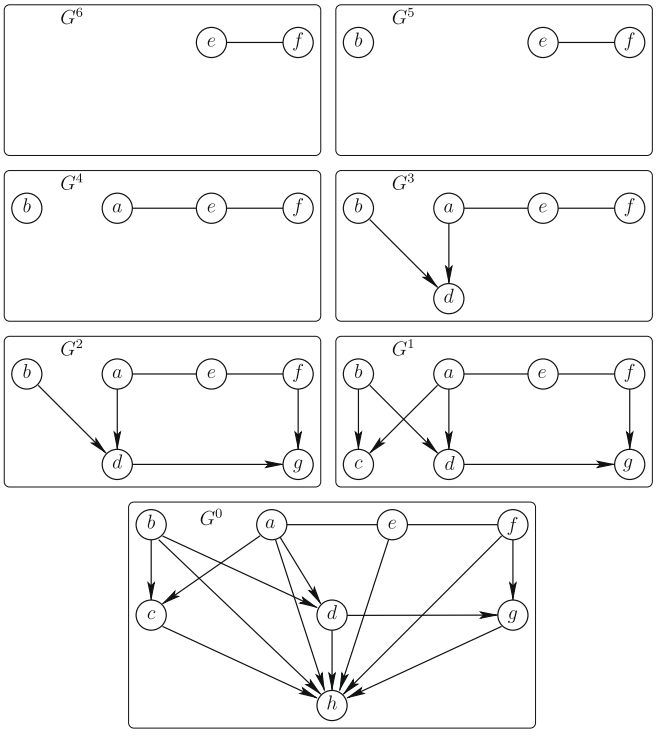

Fig. 10. Graphs from Example 11.

Table 11
Subroutine **Adapt*** (u over $N|\gamma, w$ over M)

Input:	u ... an imset over a non-empty set of variables N
Output:	γ ... a logical variable
	w ... an imset over a set M ($\emptyset \neq M \subset N$)
1	find a maximal set $M \subseteq N$ with respect to inclusion with $u(M) \neq 0$;
2	`if` $\exists Q \subseteq N\, Q \setminus M \neq \emptyset, u(Q) \neq 0$ `then` $\gamma =$ FALSE and `return` γ;
3	`put` $\gamma =$ TRUE;
4	$w =$ the restriction of u to the class of subsets of M;
5	`return` γ, w;

Table 12
Subroutine **Reduce***(u over $N|\gamma$, w over M)

Input:	u ... an imset over a non-empty set of variables N
Output:	γ ... a logical variable
	w ... an imset over a set M ($\emptyset \neq M \subset N$)
1	`put` $\mathscr{T} = \{L \subseteq N; u(L) < 0\}$;
2	$\mathscr{L} = \{L \subset N; u(L) > 0,\ L \neq \emptyset\}$;
3	$W = \bigcup \mathscr{L}$;
4	`if` $u(N) \neq 1$ or $[\forall T \in \mathscr{T}\quad T \setminus W = \emptyset\ \vee\ T \cap W \notin \mathscr{L} \cup \{\emptyset\}]$ `then` $\gamma =$ FALSE and `return` γ;
5	choose $T \in \mathscr{T}$ such that $T \setminus W \neq \emptyset$ and $T \cap W \in \mathscr{L} \cup \{\emptyset\}$;
6	`put` $R = T \setminus W$;
7	$M = N \setminus R$;
8	$\tilde{w} = u - \delta_N + \delta_M + \delta_T - \delta_{T \setminus R}$;
9	`if` $\exists Q \subseteq N\ \ Q \setminus M \neq \emptyset, \tilde{w}(Q) \neq 0$ `then` $\gamma =$ FALSE and `return` γ;
10	`put` $\gamma =$ TRUE;
11	$w =$ the restriction of $\tilde{w}$ to the class of subsets of M;
12	`return` γ, w;

Table 13
Recursive algorithm **Testing**(u over $N|\gamma$)

Input:	u ... an imset over a non-empty set of variables N
Output:	γ ... a logical variable
1	`put` $\gamma =$ TRUE;
2	`if` $\sum_{S \subseteq N} \lvert u(S) \rvert > 2\lvert N \rvert$ `then` $\gamma =$ FALSE and `return` γ;
3	`if` $\sum_{S \subseteq N} u(S) \neq 0$ or $[\exists i \in N \sum_{S,\, i \in S \subseteq N} u(S) \neq 0]$ `then` $\gamma =$ FALSE and `return` γ;
4	`if` $u = 0$ `then return` γ;
5	`if` $u \neq 0$ and $u(N) = 0$ `then` **Adapt*** (u over $N|\gamma, w$ over M);
6	`if` γ `then` **Testing** (w over $M|\gamma$);
7	`if` $u(N) \neq 0$ `then` **Reduce*** (u over $N|\gamma, w$ over M);
8	`if` γ `then` **Testing** (w over $M|\gamma$);
9	`return` γ;

The modification of the adaptation subroutine is described in Table 11. Note that the input imset for **Adapt*** subroutine will always be a non-zero imset u over N such that $u(N) = 0$ and $\sum_{S \subseteq N} u(S) = 0$. In particular, any maximal set $M \subseteq N$ with $u(M) \neq 0$ chosen in line 1 of Table 11 satisfies $\emptyset \neq M \subset N$.

The respective modification of the reduction subroutine is described in Table 12. Note that **Reduce*** will be applied to an imset u over N with $u(N) \neq 0$. Since we exclude the case $u(N) \neq 1$ in line 4 of Table 12, one has $T \subset N$ for any set $T \in \mathscr{T}$ chosen possibly in line 5 of Table 12. Such a set T exists because the other test was passed in line 4 of the procedure. Owing to the choice in line 5 we have $R \neq \emptyset$ in line 6. In particular, the set M defined in line 7 is a non-empty proper subset of N (realize that $\emptyset \neq N \setminus T \subseteq M$).

The entire testing procedure is described in Table 13.

Theorem 2. *Let u be an imset over (a non-empty set) N. Then the value of the logical variable γ obtained as the result of the procedure from Table 13 is* TRUE *iff u is a standard imset.*

The proof of this result is analogous to the proof of Theorem 1; it is moved to the Appendix – see Section A.8.

8. Remarks on complexity

The aim of this section is to explain how the procedures presented in this paper can be implemented with polynomial complexity in the number of variables. We only give plain arguments based on our (experience with the) implementation

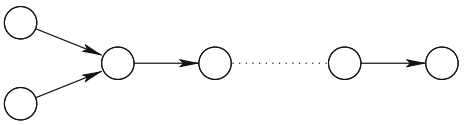

Fig. 11. An essential graph, whose standard imset has the maximal number of non-zero values.

in R. It is possible that one could derive better upper bounds on the complexity of the reconstruction algorithm than we give here.

First, let us discuss memory demands for internal computer representation of a standard imset u over N with $|N| = n \geqslant 2$. It has already been mentioned in Section 2.3 that u has at most $2n$ non-zero values. The idea is that u can be represented by a list of pairs $[L, u(L)]$, where $L \subseteq N$ is a set of variables and $0 \neq u(L) \in \mathbb{Z}$ is the corresponding non-zero value. This list has at most $2n$ items. Clearly, a subset L of N can be encoded with n bits. Moreover, it follows from Lemma 5.2 that one has $u(L) \in \{-1, 0, 1, \ldots, n-1\}$ for every $L \subseteq N$.[15]

Thus, if $u(L) \neq 0$ then it can only take one of n distinct values, which implies that n bits is enough to encode $u(L)$.[16] Consequently, every item $[L, u(L)]$ in the list can be encoded with $2n$ bits and the whole imset u with $4n^2$ bits.

An essential graph over N can be represented by means of a zero-one $N \times N$-matrix, c.f. Section 2.1.1 in [11]. It means that it can be encoded with n^2 bits. Thus, the memory demands for computer representation of a standard imset and an essential graph are alike.[17]

Note that one can possibly adjust the internal computer representation of (standard) imsets so that (most of) the steps in our subroutines from Section 7 can be done with minimal computational cost. For example, one can have u represented in the form of two separate lists of items $[L, u(L)]$: one with $u(L) > 0$ and the other with $u(L) < 0$. These lists, moreover, can be ordered so that the size of L is reflected: we put $[L_1, u(L_1)]$ before $[L_2, u(L_2)]$ whenever $|L_1| > |L_2|$. This way of (memory) organization makes the step 1 in subroutine **Adapt** and steps 1 and 2 in subroutine **Reduce** practically immediate.

The reader may wonder whether the number of edges in the essential graph, viewed as a "complexity" characteristic of the respective Bayesian network structure, is somehow reflected in the number of non-zero values of the respective standard imset, which can perhaps be interpreted as its "complexity" characteristic. The biggest Bayesian network statistical model is described by the complete undirected graph over N and this essential graph corresponds to the "simplest" standard imset, namely the zero imset over N. On the other hand, the "simplest" essential graph, namely the empty graph over N for $n = |N| \geqslant 2$, corresponds to the imset

$$u = \delta_N + (n-1) \cdot \delta_\emptyset - \sum_{a \in N} \delta_{\{a\}},$$

which has $n + 2$ non-zero values. However, this is not the standard imset with maximal possible number of non-zero values for $n \geqslant 3$. The simplest example of an essential graph whose respective standard imset achieves the maximum $2n$ of non-zero values is given in Fig. 11. Thus, there is no monotonic relation between the "complexity" of the essential graph and the "complexity" of the respective standard imset. Note there is a direct formula for the number e of edges (in the essential graph) in terms of the standard imset u:

$$\mathrm{e} = \frac{n \cdot (n-1)}{2} - \sum_{S \subseteq N} \frac{|S| \cdot (|S|-1)}{2} \cdot u(S).$$

The formula follows from Lemma 7.1 and Proposition 4.3 in [19].

We will now show why the transition from the essential graph H to the standard imset u_H is of polynomial complexity in the number of variables $n = |N|$. We base our consideration on formula (4) from Lemma 5.1. It follows from Lemma 5.2 that no terms in (4) cancel each other. In particular, (4) has at most $2n$ terms and it suffices to show that each of them is identifiable with polynomial complexity. The basic step is to identify components in H. This is a step of polynomial complexity because it can be transformed to the task of finding components in an undirected graph.[18] To determine $core(H)$ it is enough to find a terminal component in H and check whether it is an idle set.[19] Both these procedures are clearly polynominal in n. Finally, for every core-component $C \in \mathscr{C}_{core}(H)$ one has to determine $pa_H(C)$ and cliques and (multiplicities) of separators of the induced subgraph H_C. However, H_C is decomposable by Lemma 2.1 and one can apply well-known polynomial graphical procedures for determining an ordering of its cliques satisfying the running intersection property (1). For example, one can use the *maximum cardinality search* algorithm [8] for this purpose.

[15] If $u \neq 0$ then the minimal value for $u(L)$ is -1 and the maximal value is achieved for L satisfying one of the conditions (a), (c), (d) and (e). In the case (c), given a component C, the number of cliques of (decomposable graph) H_C is at most $|C| \leqslant n$, which implies the sum of multiplicities of separators of H_C is at most $n-1$. In cases (d) and (e) realize that the number of non-initial components is at most $n-1$ and the number of initial ones at most n.

[16] Actually, $u(L)$ can be encoded with $\lceil \ln_2 n \rceil$ bits.

[17] Our basic memory demands comparison leaves aside the question of addressing variables, which depends on a particular implementation.

[18] Indeed, consider provisional removal of arrows in H.

[19] It is shown in the proof of Lemma 6.1 that if H is an essential graph and $core(H) \subset N$ then the largest idle set $N \setminus core(H)$ is the super-terminal component in H, in particular, the unique terminal component in H.

Finally, we give an upper bound on the complexity of the reconstruction algorithm.

Theorem 3. *Both the recursive algorithm* **Reconstruct** *from* Table 7 *and its non-recursive version consisting of procedures* **Decompose** *and* **Compose** *from Tables* 8 *and* 9 *can be implemented with complexity (at most)* $\mathcal{O}(n^4)$.

The proof, which comes from our R implementation, can be found in the Appendix – see Section A.9, where we also give arguments why the testing algorithm **Testing** from Table 13 can be implemented with $\mathcal{O}(n^4)$, too. Let us conclude this section with a remark on a potential alternative algorithm.

Remark 3. A careful reader familiar with the literature in the field may wonder why we have not tried (instead of proposing a new algorithm) to utilize existing algorithms for our goal. Indeed, in [12] a reconstruction algorithm for the essential graph was proposed, which can use as its input the (complete) list of represented conditional independence (CI) statements.

Moreover, there are ways to check whether a given CI statement is represented in a given *structural imset*[20]: the theoretical basis for this algebraic method is described in Section 6.3 of [19]. The trouble is that practical computer testing of this independence implication is still an open research topic. Although some promising and effective algorithms have been proposed recently [5], it is still unclear how they would work with a high number of variables. Our hypothesis is that testing whether a CI statement is represented in a general *structural imset* has (at least) an exponential complexity in the number of variables.[21] On the other hand, although we conjecture that testing whether a CI statement is represented in a *standard imset* could be verified (algebraically) with polynomial complexity, we have not proven it. Therefore, we prefer to have a direct polynomial algorithm (for reconstructing the essential graph on the basis of the standard imset) which does not need to recover the respective list of CI statements. Actually, we believe that it will work better than a potential polynomial algorithm based on the reconstruction of the respective CI structure.

9. Conclusions

In the paper, we have established a connection between essential graphs and standard imsets, which are both unique representatives of Bayesian network statistical models. Note that the algorithm presented here was already mentioned (without the proof of its correctness) in [23]. In that paper we gave further motivational details and described (in Section 8 of [23]) the relationship of these two representatives to the third possible way of representing Bayesian network statistical models, namely to *hierarchical junction trees* [14].

To outline potential future application of the algorithm presented in this paper let us discuss our motives in more detail. It has already been mentioned in the Introduction that we hope the algebraic view can simplify some existing methods for learning Bayesian network structures. We have in mind procedures based on the maximization of a quality criterion $\mathcal{Q}$. A popular method for tackling this task is the method of *local search*. Let us describe it informally (see also Section 8.2 of [19]). The idea is that one introduces a specific search space, which consists of the collection of *states* and the collection of *moves* between states. The states are representatives of (Bayesian network) statistical models, typically graphs, and the moves are pre-defined "local" shifts towards "neighboring" states, typically introduced as graphical transformations. That means every particular version of the local search method has a pre-defined *neighborhood structure* in the class of representatives (cf. Section 4.3 in [4]). The goal of the method is then to find a local maximum of a quality criterion $\mathcal{Q}$ with respect to the corresponding neighborhood structure.

There are some theoretical reasons why researchers in this area consider the *inclusion neighborhood* (see [13,6] for those arguments). This concept is motivated by the conditional independence interpretation of Bayesian network statistical structures [10,3]. A standard example of the local search method is the *greedy equivalence search* (GES) algorithm proposed in [13]. This algorithm was modified in [7], where the states are represented by essential graphs and the moves are graphical transformations that together exhaust the inclusion neighborhood.

The GES algorithm, as described in [7] has one (minor) weak point: the description of the moves in terms of graphical transformations is complicated. Every move is realized as applying a special complex graphical operator to the essential graph and then running two graphical transformational algorithms: one of them transforms the resulting hybrid graph to an acyclic directed graph and the other the directed graph back into a chain graph.

The algebraic view can simplify the method. If a Bayesian network structure is represented by the respective standard imset then the move towards a neighboring state in the sense of the inclusion neighborhood corresponds to adding, or subtracting, a certain elementary imset, called a *differential imset* in Section 8.4.2 in [19]. This imset is very simple, it only has four non-zero values. Moreover, it has a conditional independence interpretation: it corresponds to a statement $a \perp\!\!\!\perp b|C$, where $a, b \in N$ are distinct and $C \subseteq N \setminus \{a, b\}$.[22] In the paper [20] the whole inclusion neighborhood of a given Bayesian network structure is characterized. More specifically, given the respective essential graph, all the moves towards its inclusion neighbors are described there. They are described in the form of triplets $a \perp\!\!\!\perp b|C$ defining differential imsets.

[20] This is a wider class of imsets – see Remark 1 concluding Section 2.3.

[21] Note that structural imsets are able to describe all probabilistic CI structures and the number of these CI structures grows super-exponentially with the number of variables – see Corollary 2.7 in [19].

[22] Note that $A \perp\!\!\!\perp B|C$ means A is conditionally independent of B given C.

Thus, one can come with the following hybrid method, which can be interpreted as a modification (simplification) of the GES algorithm. Every Bayesian network structure can be represented by a pair of representatives: by the essential graph and the standard imset. As explained above, the essential graph allows one to find all possible moves (in the sense of the inclusion neighborhood) represented uniquely in the form of differential imsets. The maximization of the increase in the criterion $\mathcal{Q}$ gives the "optimal" move. Then the standard imset representing the neighboring structure will be obtained by adding or subtracting the respective differential imset. Finally, the algorithm described in this paper provides the respective essential graph and the "greedy" search can continue. Actually, this modification of GES has already been described in [24], where some preliminary experiments were also reported.

A topic of future research can be the development of a fully algebraic method for learning Bayesian network structures. By this we mean a modification of the procedure described above in which every Bayesian network structure will be represented only by the respective standard imset. The advantage of that potential future method would be its methodological clarity. However, what is needed for this purpose is to characterize the inclusion neighborhood in terms of the standard imset. This is one of our open problems to be solved. Perhaps the algorithm presented in this paper can give a clue for dealing with that problem.

On the other hand, the algebraic point of view offers other options for the neighborhood structure, different from the inclusion neighborhood. In [21], we have introduced the concept of the *geometric neighborhood* that allows one to find the global maximum (of a score equivalent decomposable criterion) using the greedy search approach. Note that another method for finding the global optimum, motivated by dynamic programming, was presented in [16].

Acknowledgements

We are grateful to the reviewers of the paper for their comments, which helped us improve the readability of the paper. It is based on their merits that we described our motivation in more detail and incorporated the remarks on complexity.

Appendix A

A.1. The proof of Lemma 5.1

The first step is to show that, for a graph of the considered type, there is a correspondence between cliques of the closure graph for a component and cliques of the respective induced subgraph. Note that this need not be true if the graph has flags – see Example 5.

Lemma 9.1. *Let H be a chain graph without flags which is equivalent to an acyclic directed graph. Then, for every $C \in \mathscr{C}(H)$, the collection of cliques $\overline{\mathscr{K}}(C)$ of the closure graph has the form $\{K \cup pa_H(C);\ K \in \mathscr{K}(C)\}$. Moreover, $\overline{\mathscr{S}}(C) = \{S \cup pa_H(C);\ S \in \mathscr{S}(C)\}$ and $\bar{v}_C(S \cup pa_H(C)) = v_C(S)$ for every $S \in \mathscr{S}(C)$. In particular, one has*

$$u_H = \delta_N - \delta_\emptyset + \sum_{C \in \mathscr{C}(H)} \left\{ \delta_{pa_H(C)} - \sum_{K \in \mathscr{K}(C)} \delta_{K \cup pa_H(C)} + \sum_{S \in \mathscr{S}(C)} v_C(S) \cdot \delta_{S \cup pa_H(C)} \right\}. \tag{12}$$

Proof 9. The facts that $pa_H(C)$ is complete in $\overline{H}(C)$ and H has no flags imply that whenever L is complete in $\overline{H}(C)$ then $L \cup pa_H(C)$ is complete in $\overline{H}(C)$. In particular, every clique of $\overline{H}(C)$ contains $pa_H(C)$. The same two arguments imply that $K \subseteq C$ is complete in H_C iff $K \cup pa_H(C)$ is complete in $\overline{H}(C)$. This allows us to observe that the collection of cliques of $\overline{H}(C)$ is the class $\{K \cup pa_H(C);\ K \in \mathscr{K}(C)\}$. By Lemma 2.3, the graph H_C is decomposable and the class of its cliques can be ordered into a sequence $K_1, \ldots, K_m$, $m \geqslant 1$ satisfying (1). Put $\overline{K}_i \equiv K_i \cup pa_H(C)$ for $i = 1, \ldots m$ and observe that $\overline{K}_1, \ldots, \overline{K}_m$, $m \geqslant 1$ satisfies the running intersection property as well. Since the collection of separators and their multiplicities do not depend on the choice of an ordering of this type, this implies that $\overline{\mathscr{S}}(C) = \{S_i \cup pa_H(C);\ 2 \leqslant i \leqslant m\}$ with $S_i = K_i \cap (\bigcup_{j<i} K_j)$. Therefore, $\bar{v}_C(S \cup pa_H(C)) = v_C(S)$ for every $S \in \mathscr{S}(C)$. The formula (12) now easily follows from the formula (3). □

Now, the proof of Lemma 5.1 follows.

Proof 10. We start with the formula (3) and rewrite it into three special sums as follows:

$$u_H = \delta_N + \sum_{C \in \mathscr{C}(H) \setminus \mathscr{C}_{core}(H)} \{\delta_{pa_H(C)} - \sum_{\overline{K} \in \overline{\mathscr{K}}(C)} \delta_{\overline{K}} + \sum_{\overline{S} \in \overline{\mathscr{S}}(C)} \bar{v}_C(\overline{S}) \cdot \delta_{\overline{S}}\} \tag{13}$$

$$+ \sum_{C \in \mathscr{C}_{core}(H)} \{- \sum_{\overline{K} \in \overline{\mathscr{K}}(C)} \delta_{\overline{K}} + \sum_{\overline{S} \in \overline{\mathscr{S}}(C)} \bar{v}_C(\overline{S}) \cdot \delta_{\overline{S}}\} \tag{14}$$

$$- \delta_\emptyset + \sum_{C \in \mathscr{C}_{core}(H)} \delta_{pa_H(C)}. \tag{15}$$

The first step is to show that the expression on the right-hand side of (13) equals to $\delta_{core(H)}$. Let B be the largest idle set in H. Then $\mathscr{C}(H) \setminus \mathscr{C}_{core}(H)$ consists of components in H contained in B. If $B = \emptyset$ then the statement is evident. If $B \neq \emptyset$ then the fact that H_B is a chain graph implies that $\mathscr{C}(H) \setminus \mathscr{C}_{core}(H)$ can be ordered into a chain $C_1, \ldots, C_k$, $k \geqslant 1$. The fact that H is a chain

graph and the conditions (i1) and (i2) from Section 4.2 imply that $pa_H(C_1) = core(H)$, $pa_H(C_{i+1}) = C_i \cup pa_H(C_i)$ for $i = 1, \ldots, k-1$ and $C_k \cup pa_H(C_k) = N$. Moreover, for every $i = 1, \ldots, k$, the closure graph $\overline{H}(C_i)$ consists of one clique $C_i \cup pa_H(C_i)$ which implies $\sum_{K \in \mathcal{K}(C_i)} \delta_K = \delta_{C_i \cup pa_H(C_i)}$ and $\overline{\mathcal{S}}(C_i) = \emptyset$. In particular, the expression in (13) has the form

$$\delta_N + \sum_{i=1}^{k} \delta_{pa_H(C_i)} - \delta_{C_i \cup pa_H(C_i)}.$$

In this expression, all terms except for $\delta_{pa_H(C_1)} = \delta_{core(H)}$ cancel each other.

To show the first claim of Lemma 5.1 assume $core(H) = \emptyset$. The above observation then implies that the expression in (13) is $+\delta_\emptyset$. Because $\mathscr{C}_{core}(H) = \emptyset$, the expression in (14) is 0 and the expression in (15) is $-\delta_\emptyset$. In particular, $core(H) = \emptyset$ implies $u_H = 0$.

Now, assume $core(H) \neq \emptyset$. It follows from Lemma 9.1 that, for every $C \in \mathscr{C}_{core}(H)$, the expression in braces in (14) has the form

$$-\sum_{K \in \mathcal{K}(C)} \delta_{K \cup pa_H(C)} + \sum_{S \in \mathcal{S}(C)} v_C(S) \cdot \delta_{S \cup pa_H(C)}.$$

Thus, to verify the formula (4) for u_H in the case $core(H) \neq \emptyset$ it remains to show that the expression in (15) has the form

$$\sum_{P \in \mathscr{P}_{core}(H)} \tau(P) \cdot \delta_P + \{i(H) - 1\} \cdot \delta_\emptyset.$$

This follows easily from the fact that, in this case, every initial component is in $\mathscr{C}_{core}(H)$ and the definition of parent sets, including their multiplicities.

Finally, if $core(H) \neq \emptyset$ then, by Lemmas 4.1 and 4.2, the term $\delta_{core(N)}$ cannot be cancelled by any other term in the formula (4). In particular, $u_H \neq 0$ then. □

A.2. The proof of Lemma 5.2

We first prove an auxiliary observation, which will also be utilized later.

Lemma 9.2. *Let H be an essential graph over N and $u = u_H$ the respective standard imset. Assume that $C \in \mathscr{C}_{core}(H)$ and $L = X \cup pa_H(C)$ where $X \subseteq C$ is non-empty and complete in H. Then $L \notin \mathscr{P}_{core}(H)$; if $L \in \mathscr{L}_u$ then $X \in \mathscr{S}(C)$.*

Proof 11. Assume for a contradiction that $L \in \mathscr{P}_{core}(H)$, that is, $L = pa_H(C') \neq \emptyset$ for $C' \in \mathscr{C}_{core}(H)$. Let us put $C_\ell = C'$, $C_u = C$ and observe that the conditions **{i}** and **{ii}** from Section 2.2 are fulfilled: $pa_H(C_\ell) \cap C_u = pa_H(C') \cap C = L \cap C = X$ is a non-empty complete set in H and $pa_H(C_\ell) \setminus C_u = L \setminus C = pa_H(C) = pa_H(C_u)$. Thus, the components C_u and C_ℓ can be legally merged which, by Lemma 2.2, contradicts the assumption that H is an essential graph.

Therefore, if $L \in \mathscr{L}_u$, that is, $u(L) > 0$, $\emptyset \neq L \subset core(u)$, then the formula (4) in Lemma 5.1 together with facts $core(u) = core(H)$ (see Section 5.2) and $L \notin \mathscr{P}_{core}(H)$ implies $L = S \cup pa_H(C')$ for $C' \in \mathscr{C}_{core}(H)$ and $S \in \mathscr{S}(C')$. Note that $S \neq \emptyset$ by Lemma 4.1. As H_L is a chain graph without flags the facts $L = S \cup pa_H(C')$, $S \subseteq C'$ imply that S is a super-terminal component in H_L, while the facts $L = X \cup pa_H(C)$, $X \subseteq C$ imply the same conclusion for X. The uniqueness of a super-terminal component in H_L implies $S = X$. The observation $S = X$ gives $C = C'$. Thus, $X = S \in \mathscr{S}(C)$. □

Now, the proof of Lemma 5.2 follows. It is based on the formula (4) from Lemma 5.1.

Proof 12. The cases (a)–(e) from Lemma 5.2 correspond to terms in the formula (4). Therefore, it is enough to show that the cases (a)–(e) exclude each other – the case (f) is the complementary case then. What are the values of $u_H(L)$ in respective cases then follows from the formula (4).

By Lemma 5.1, $u_H \neq 0$ implies $core(H) \neq \emptyset$ and, by Lemmas 4.1 and 4.2, all sets of the form (b)–(d) are proper subsets of $core(H)$. Thus, the case (a) for a set $L \subseteq N$ excludes the cases (b)–(e) for L. Lemmas 4.1 and 4.2 also imply that the case (e), namely $L = \emptyset$, is incompatible with the cases (b)–(d). The next step is to show that the case (d) and the cases (b) and (c) cannot occur simultaneously for a set $L \subseteq N$. However, this follows from Lemma 9.2 using Lemma 4.1.

The last step is to show that if L is of type (b)–(c) then there exist unique $C \in \mathscr{C}_{core}(H)$ and $Y \in \mathscr{K}(C) \cup \mathscr{S}(C)$ with $L = Y \cup pa_H(C)$. This, together with the fact $\mathscr{K}(C) \cap \mathscr{S}(C) = \emptyset$ (see Section 2.1.1), means that the cases (b) and (c) exclude each other. It also implies the last statements in Lemma 5.2.

Thus, assume for a contradiction that one has two distinct "decompositions" of L, namely $Y \cup pa_H(C) = L = Y' \cup pa_H(C')$ for $C, C' \in \mathscr{C}_{core}(H)$, $Y \in \mathscr{K}(C) \cup \mathscr{S}(C)$ and $Y' \in \mathscr{K}(C') \cup \mathscr{S}(C')$. Observe that C and C' are distinct for otherwise we immediately get $Y = Y'$. Let us choose $x \in Y \subseteq C$. As $C \cap C' = \emptyset$ observe that $x \in L \setminus C' = pa_H(C')$ and find $y \in C'$ with $x \to y$ in H. Analogously, choose $x' \in Y' \subseteq C'$, observe $x' \in pa_H(C)$ and find $y' \in C$ with $x' \to y'$ in H. As both C and C' are connected there exists a semi-directed cycle $x \to y - \cdots - x' \to y' - \cdots - x$ in H, which contradicts the assumption that H is a chain graph. □

A.3. The proof of Lemma 6.3

We first need an auxiliary result on leaf-cliques (introduced in Section 2.1.1).

Lemma 9.3. *Let G be a decomposable graph over N and $\mathcal{K}$ the system of its cliques. Then the following two conditions are equivalent for $K \in \mathcal{K}$:*

- *K is a leaf-clique of G,*
- *there exists an ordering $K_1, \ldots, K_m$, $m \geqslant 1$ of elements of $\mathcal{K}$ satisfying the running intersection property (1) such that $K = K_m$.*

Let $\mathcal{S}$ denote the systems of separators of G and $v(S')$ the multiplicity of $S' \in \mathcal{S}$ in G. Let $K \in \mathcal{K}$ be a leaf-clique of G and $R \equiv K \setminus \bigcup(\mathcal{K} \setminus \{K\})$ the residual of K. If $|\mathcal{K}| \geqslant 2$ then the graph $G_{N\setminus R}$ has $\mathcal{K} \setminus \{K\}$ as the system of cliques and $S \equiv K \setminus R$ belongs to the system $\mathcal{S}$ of separators of G. If $v(S) = 1$ then $G_{N\setminus R}$ has $\mathcal{S} \setminus \{S\}$ as the system of separators; if $v(S) \geqslant 2$ then $\mathcal{S}$ is the system of separators of $G_{N\setminus R}$. Moreover, the multiplicity of $S' \in \mathcal{S} \setminus \{S\}$ in $G_{N\setminus R}$ is $v(S')$ and if $v(S) \geqslant 2$, then the multiplicity of S in $G_{N\setminus R}$ is $v(S) - 1$.

Proof 13. The requirement $K = K_m$ for an ordering $K_1, \ldots, K_m$, $m \geqslant 2$ satisfying (1) is clearly sufficient for the existence of $K' \in \mathcal{K} \setminus \{K\}$ with $K \cap \bigcup(\mathcal{K} \setminus \{K\}) \subseteq K'$. If $m = 1$ then $\mathcal{K} = \{K\}$ and K is also a leaf-clique.

To verify the necessity of the condition we first show that if $K \in \mathcal{K}$ is a leaf and $|\mathcal{K}| \geqslant 2$ then $\mathcal{K}' \equiv \mathcal{K} \setminus \{K\}$ is the collection of cliques of $G_{N\setminus R}$, where $R = K \setminus \bigcup \mathcal{K}'$ is the residual. Of course, every $K' \in \mathcal{K}'$ is disjoint from R. As K' is complete in $G_{N\setminus R}$ there exists a clique L of $G_{N\setminus R}$ with $K' \subseteq L$. However, L is complete in G, which implies $K' = L$ and, therefore, K' is a clique of $G_{N\setminus R}$. Conversely, assume that $\widetilde{K}$ is a clique of $G_{N\setminus R}$. As it is complete in G, there exists $K' \in \mathcal{K}$ with $\widetilde{K} \subseteq K'$. We can assume without loss of generality that $K' \neq K$, for otherwise the assumption that K is a leaf implies the existence of $K'' \in \mathcal{K}'$ such that $\widetilde{K} \subseteq K \setminus R = K \cap \bigcup \mathcal{K}' \subseteq K''$, and $K = K'$ can be replaced by K''. However, we already know that every $K' \in \mathcal{K}'$ is a complete subset in $G_{N\setminus R}$. As $\widetilde{K}$ is a clique of $G_{N\setminus R}$ it implies $\widetilde{K} = K'$ and $\widetilde{K} \in \mathcal{K} \setminus \{K\}$.

Now, since $G_{N\setminus R}$ is a decomposable graph, the system of its cliques $\mathcal{K}'$ can be ordered into a sequence $K_1, \ldots, K_{m-1}$, $m \geqslant 2$ satisfying the running intersection property. We put $K_m = K$ and obtain an ordering of $\mathcal{K}$ satisfying this property.

The statement concerning the separators is now easy to see. Let us consider an ordering $K_1, \ldots, K_m$, $m \geqslant 2$ of elements of $\mathcal{K}$ satisfying (1) and $K_m = K$. Because $K_1, \ldots, K_{m-1}$ is an ordering of cliques of $G_{N\setminus R}$ satisfying the running intersection property, the sets $S_i = K_i \cap \bigcup_{j<i} K_j$, $i < m$ are both separators of $G_{N\setminus R}$ and G. Moreover, $S_m = K_m \cap \bigcup_{j<m} K_j$ is a separator of G; and obviously $S_m = K \cap \bigcup \mathcal{K}' = K \setminus R$. □

Now, the proof of Lemma 6.3 follows.

Proof 14. To verify the first claim in Lemma 6.3 consider $a \in R$ and $b \in N$ such that $[a, b]$ is an edge in H and show $b \in T$. The assumption (7) implies that no arrow in H is directed out of R and therefore, $\neg(a \to b \text{ in } H)$. The next step is to show

$$a \in R,\ b \in N,\ a - b \text{ in } H \ \Rightarrow\ b \in K \subseteq T. \tag{16}$$

Indeed, $a \in R \subseteq K \subseteq C$ implies that $\{a, b\}$ is a complete set in H_C. Thus, there exists $\widetilde{K} \in \mathcal{K}(C)$ with $\{a, b\} \subseteq \widetilde{K}$. However, by the definition of the residual $R = K \setminus \bigcup_{K' \in \mathcal{K}(C)\setminus\{K\}} K'$ we can derive $\widetilde{K} = K$ which gives $b \in K$ and (16) is verified. Finally, if $a \leftarrow b$ in H then $a \in R \subseteq K \subseteq C$ implies $b \in pa_H(C) \subseteq T$. Thus, the first claim in Lemma 6.3 has been verified.

The fact that there is no arrow in H from R also implies that $M = N \setminus R$ is closed under parents. In particular, by Corollary 2.1, H_M is an essential graph. Let w be the standard imset corresponding to H_M and $\check{w}$ its zero extension to subsets of N.[23] Our aim is to show

$$u - \check{w} = \delta_N - \delta_M - \delta_T + \delta_{T\setminus R}. \tag{17}$$

This fact already implies the remaining statements of Lemma 6.3.[24] The first step to show (17) is to characterize the class of components in H_M. We show that

- $\mathscr{C}(H_M) = \mathscr{C}(H) \setminus \{C\}$ if $C \cap M = \emptyset$, and
- $\mathscr{C}(H_M) = \{C \cap M\} \cup \mathscr{C}(H) \setminus \{C\}$ otherwise.

Indeed, the fact (16) implies that $ne_H(R) \subseteq K$ is a complete set in $H_{C\cap M}$. Therefore, every undirected path in H_C between two nodes in $C \cap M$ which hits R can be shortened to an undirected path in $H_{C\cap M}$. This implies that the set $C \cap M$ is connected in H_M. Since every $C' \in \mathscr{C}(H) \setminus \{C\}$ is a connected set in H_M and $\{C' \cap M;\ C' \in \mathscr{C}(H)\}$ is a partition of M, it is the class of components in H_M except for the case $C \cap M = \emptyset$, in which case $C \cap M$ has to be removed.

[23] That means we put $\check{w}(Q) = w(Q)$ if $Q \subseteq M$ and $\check{w}(Q) = 0$ if $Q \subseteq N$, $\emptyset \neq Q \setminus M$.

[24] Indeed, (17) means that the imset $\check{w}$ coincides with the imset $\bar{w}$ introduced in the formulation of Lemma 6.3.

The second step to show (17) is to utilize the formula (12) from Lemma 9.1. It says that

$$u = u_H = \delta_N - \delta_\emptyset + \sum_{C' \in \mathscr{C}(H)} \left\{ \delta_{pa_H(C')} - \sum_{K' \in \mathscr{K}(C')} \delta_{K' \cup pa_H(C')} + \sum_{S \in \mathscr{S}(C')} v_{C'}(S) \cdot \delta_{S \cup pa_H(C')} \right\}, \tag{18}$$

where $v_{C'}(S)$ is the multiplicity of $S \in \mathscr{S}(C')$. If (12) is applied to $\widehat{H} \equiv H_M$[25] then it gives

$$\breve{w} = \delta_M - \delta_\emptyset + \sum_{C' \in \mathscr{C}(\widehat{H})} \left\{ \delta_{pa_{\widehat{H}}(C')} - \sum_{K' \in \widehat{\mathscr{K}}(C')} \delta_{K' \cup pa_{\widehat{H}}(C')} + \sum_{S \in \widehat{\mathscr{S}}(C')} \hat{v}_{C'}(S) \cdot \delta_{S \cup pa_{\widehat{H}}(C')} \right\}, \tag{19}$$

where the hat in $\widehat{\mathscr{K}}(C')$, $\widehat{\mathscr{S}}(C')$ and $\hat{v}_{C'}$ indicates that they correspond to $\widehat{H} = H_M$. Obviously, for every $C' \in \mathscr{C}(H) \setminus \{C\}$ one has $pa_H(C') = pa_{\widehat{H}}(C')$ and $(\widehat{H})_{C'} = H_{C'}$. Therefore, the respective terms in (18) and (19) coincide and cancel each other in $u - \breve{w}$. Hence, only terms δ_N, δ_M and the terms corresponding to $C \in \mathscr{C}(H)$ and possibly $C \cap M \in \mathscr{C}(\widehat{H})$ (if $C \cap M \neq \emptyset$) remain in the formula for $u - \breve{w}$.

Thus, if $C \cap M = \emptyset$, then $R = K = C$, which gives $\mathscr{K}(C) = \{K\}$, $\mathscr{S}(C) = \emptyset$ and no term corresponding to $C \cap M \in \mathscr{C}(\widehat{H})$ is present in (19). In particular, one has

$$u - \breve{w} = \delta_N + \{\delta_{pa_H(C)} - \delta_{K \cup pa_H(C)}\} - \delta_M.$$

As $K \cup pa_H(C) = T$ and $pa_H(C) = T \setminus K = T \setminus R$ it gives (17).

In the case $C \cap M \neq \emptyset$ we know that $pa_H(C) = pa_{\widehat{H}}(C \cap M)$[26] and, therefore,

$$u - \breve{w} = \delta_N + \left\{ - \sum_{K' \in \mathscr{K}(C)} \delta_{K' \cup pa_H(C)} + \sum_{S \in \mathscr{S}(C)} v_C(S) \cdot \delta_{S \cup pa_H(C)} \right\}$$
$$- \delta_M - \left\{ - \sum_{K' \in \widehat{\mathscr{K}}(C \cap M)} \delta_{K' \cup pa_{\widehat{H}}(C \cap M)} + \sum_{S \in \widehat{\mathscr{S}}(C \cap M)} \hat{v}_{C \cap M}(S) \cdot \delta_{S \cup pa_{\widehat{H}}(C \cap M)} \right\}.$$

We thus need to explicate the terms corresponding to $C \cap M = C \setminus R$. Lemma 9.3 applied to $G = H_C$ implies that $\widehat{\mathscr{K}}(C \cap M) = \mathscr{K}(C) \setminus \{K\}$ and that the separators of $\widehat{H}_{C \cap M} = H_{C \cap M}$ are obtained from the separators for H_C by "reducing" the multiplicity of $K \setminus R$ by 1. Thus, nearly all terms in the above formula cancel each other and we have

$$u - \breve{w} = \delta_N + \{-\delta_{K \cup pa_H(C)} + \delta_{(K \setminus R) \cup pa_H(C)}\} - \delta_M = \delta_N - \delta_M - \delta_T + \delta_{T \setminus R},$$

because $(K \setminus R) \cup pa_H(C) = (K \cup pa_H(C)) \setminus R = T \setminus R$. This completes the proof of (17). □

A.4. Proof of Lemma 6.4

Throughout the proof we utilize the observations from Section 5.2, namely (5) and (6). Recall that we also assume in Lemma 6.4 that u is adapted for which reason one has $\mathscr{C}_{core}(H) = \mathscr{C}(H)$ for the respective essential graph H.

Proof 15. I. The first claim is that $\bigcup \mathscr{L}_u = W_u$.

If $a \in \bigcup \mathscr{L}_u$ then there exists $L \in \mathscr{L}_u$ with $a \in L$. The condition (6) implies there exists $C \in \mathscr{C}(H)$ such that either $a \in pa_H(C)$ or $a \in S$ for some $S \in \mathscr{S}(C)$. If $a \in pa_H(C)$ then consider $C' \in \mathscr{C}(H)$ with $a \in C'$, find $K' \in \mathscr{K}(C')$ with $a \in K'$, choose $K \in \mathscr{K}(C)$ and observe that $a \in T \cap T'$ where $T = K \cup pa_H(C)$ and $T' = K' \cup pa_H(C')$. By (5) $T, T' \in \mathscr{T}_u$, $T \neq T'$ and, thus, $a \in W_u$. If $a \in S$ for $S \in \mathscr{S}(C)$ then there exist two distinct cliques $K, K' \in \mathscr{K}(C)$ such that $S = K \cap K'$. In particular, $a \in T \cap T'$ where $T = K \cup pa_H(C)$ and $T' = K' \cup pa_H(C)$, and (5) gives $a \in W_u$.

Conversely, if $a \in W_u$ then, by (5), there are $C, C' \in \mathscr{C}(H)$, $K \in \mathscr{K}(C)$ and $K' \in \mathscr{K}(C')$, $K \neq K'$ such that $a \in (K \cup pa_H(C)) \cap (K' \cup pa_H(C'))$. If $a \in pa_H(C) \cup pa_H(C')$ then $a \in L$ for some $L \in \mathscr{P}_{core}(H)$ and $L \in \mathscr{L}_u$ by (6). If $a \in K \cap K'$ then necessarily $C = C'$. Now, the intersection of two different cliques of a decomposable graph has to be a subset of a separator of the graph. Therefore, there exists $S \in \mathscr{S}(C)$ with $K \cap K' \subseteq S$ and, by (6), $a \in S \subseteq S \cup pa_H(C) \in \mathscr{L}_u$. Thus, in both cases $a \in \bigcup \mathscr{L}_u$.

II. A further useful observation is that $pa_H(C) \subseteq W_u$ for every $C \in \mathscr{C}(H)$.

The statement is evident if $pa_H(C) = \emptyset$. If $pa_H(C) \neq \emptyset$ then $pa_H(C) \in \mathscr{P}_{core}(H)$ because $core(H) = N$. Then, by (6), $pa_H(C) \in \mathscr{L}_u$, that is, $pa_H(C) \subseteq \bigcup \mathscr{L}_u = W_u$ by Step I.

III. Assume $T = K \cup pa_H(C)$, $C \in \mathscr{C}(H)$, $K \in \mathscr{K}(C)$ has the form (7) and show $R = T \setminus W_u$.

[25] Recall that $\breve{w}$ is the zero extension of w which means that formulas for $\breve{w}$ and w formally coincide; what is different is that in the case of $\breve{w}$ a wider domain is considered, namely the power set of N.

[26] Recall that H has no flags.

If $a \in R \equiv K \setminus \bigcup(\mathscr{K}(C) \setminus \{K\})$ then $a \in T$ and it suffices to show that $a \notin W_u$. Assume for contradiction $a \in W_u$ and consider $T' \in \mathscr{T}_u$, $T' \neq T$ with $a \in T'$, where $T' = K' \cup pa_H(C')$ for $C' \in \mathscr{C}(H)$, $K' \in \mathscr{K}(C')$. If $a \in K'$ then necessarily $C = C'$ and $K \neq K'$. The facts $a \in R$ and $a \in K'$ then contradict each other. If $a \in pa_H(C')$ then there is an arrow in H from $a \in R$ which contradicts (7). We have thus shown $R \subseteq T \setminus W_u$.

If $a \in T \setminus W_u$ then assume for contradiction $a \notin R$. As $T = K \cup pa_H(C)$ and $pa_H(C) \subseteq W_u$, by Step II, necessarily $a \in K$. Then $a \in K \setminus R = K \cap \bigcup(\mathscr{K}(C) \setminus \{K\})$ implies the existence of $K' \in \mathscr{K}(C)$, $K' \neq K$ with $a \in K'$. Hence, $a \in T' \equiv K' \cup pa_H(C)$, $T' \neq T$ and $a \in W_u$, which contradicts the assumption. Thus, we have shown $T \setminus W_u \subseteq R$.

IV. Now, we are ready to verify (7) $\Rightarrow$ (8) for $T \in \mathscr{T}_u$.

The condition (7) involves the requirement that K is a leaf of H_C and has non-empty residual. Thus, Step III gives $T \setminus W_u = R \neq \emptyset$ and $T \cap W_u = T \setminus R = (K \cup pa_H(C)) \setminus R = (K \setminus R) \cup pa_H(C)$. To verify (8) it remains to be shown that if $T \cap W_u \neq \emptyset$ then $T \cap W_u \in \mathscr{L}_u$. If $K \setminus R = \emptyset$ then $\emptyset \neq T \cap W_u = pa_H(C) \in \mathscr{P}_{core}(H) \subseteq \mathscr{L}_u$ by (6). If $K \setminus R \neq \emptyset$ then $|\mathscr{K}(C)| \geqslant 2$ and Lemma 9.3 applied to $G = H_C$ implies $S \equiv K \setminus R \in \mathscr{S}(C)$. Hence, $T \cap W_u = S \cup pa_H(C) \in \mathscr{S}_H \subseteq \mathscr{L}_u$ by (6). In both cases, we have verified (8).

V. The last step is to show that (8) $\Rightarrow$ (7) for $T \in \mathscr{T}_u$.

The condition (5) says $T = K \cup pa_H(C)$, where $C \in \mathscr{C}(H)$ and $K \in \mathscr{K}(C)$. Step II implies that there is no arrow in H from nodes in $K \setminus W_u$ and that the assumption $T \setminus W_u \neq \emptyset$ from (8) means $K \setminus W_u \neq \emptyset$. Let us distinguish two cases, namely $K \cap W_u = \emptyset$ and $K \cap W_u \neq \emptyset$.

If $K \cap W_u = \emptyset$ then observe $\mathscr{K}(C) = \{K\}$. Indeed, if we assume for contradiction $|\mathscr{K}(C)| \geqslant 2$ then there exists a separator $S \in \mathscr{S}(C)$ such that $S \subseteq K$. By (6) and Step I, $S \subseteq \bigcup \mathscr{L}_u = W_u$ for any $S \in \mathscr{S}(C)$. However, according to Lemma 4.1, $\emptyset \neq S \subseteq K \cap W_u$, which contradicts the assumption. Thus, $\mathscr{K}(C) = \{K\}$ implies that K is a leaf of H_C and we already know that there is no arrow in H from $R = K = K \setminus W_u$. In particular, (7) holds.

If $K \cap W_u \neq \emptyset$ then $L \equiv T \cap W_u \in \mathscr{L}_u$ by (8). Step II gives $pa_H(C) \subseteq W_u$, which implies $L = (K \cap W_u) \cup pa_H(C)$. Thus, the assumptions of Lemma 9.2 with $X = K \cap W_u$ are fulfilled. Hence, $X \in \mathscr{S}(C)$ is a separator of H_C and there exist at least two cliques of H_C containing X. In particular, there exists $K' \in \mathscr{K}(C) \setminus \{K\}$ with $X \subseteq K'$. The definition of W_u gives $K \cap K' \subseteq K \cap \bigcup(\mathscr{K}(C) \setminus \{K\}) \subseteq K \cap W_u = X \subseteq K \cap K'$, which implies that K is a leaf of H_C and $K \cap \bigcup(\mathscr{K}(C) \setminus \{K\}) = K \cap W_u$. In particular, the residual R of K has the form $R = K \setminus (K \cap W_u) = K \setminus W_u$. We already know, by Step II, that there is no arrow in H from $K \setminus W_u = R$. Thus, the condition (7) was verified in this case as well. □

A.5. Proof of Lemma 6.5

Proof 16. Recall that the assumption is that $T = K \cup pa_H(C)$ where $C \in \mathscr{C}(H)$, $K \in \mathscr{K}(C)$ satisfies (7). The respective residual $R = K \setminus \bigcup(\mathscr{K}(C) \setminus \{K\})$ is a subset of C, which implies

$$L \equiv T \setminus R = (K \setminus R) \cup pa_H(C).$$

Now, if $|\mathscr{K}(C)| = 1$ then $K \setminus R = \emptyset$ gives $L = pa_H(C)$; that is, the condition (**p**). On the other hand, if $|\mathscr{K}(C)| \geqslant 2$ then Lemma 9.3 applied to $G = H_C$ gives $K \setminus R \in \mathscr{S}(C)$, that is, the condition (**s**). The conditions (**s**) and (**p**) exclude each other by Lemma 5.2: the cases (c) and (d)–(e) are incompatible.

The next step is to verify (9) $\Rightarrow$ (**s**). It suffices to show that the condition (**p**) is not valid. The assumption (9) gives directly $T \setminus R \neq \emptyset$ and it remains to verify $L \notin \mathscr{P}_{core}(H)$. The condition (9) also says that H_L has a complete component X such that $pa_{H_M}(X) = L \setminus X$. By (7) there is no arrow in H from $R = N \setminus M$ and this gives $pa_{H_M}(X) = pa_H(X)$. In particular, $L = X \cup (L \setminus X) = X \cup pa_H(X)$, that is, $L = X \cup pa_H(C')$ where $C' \in \mathscr{C}(H)$ is the component in H containing X. Thus, Lemma 9.2 says $L \notin \mathscr{P}_{core}(H)$ and (**s**) holds. We have $S \cup pa_H(C) = L = X \cup pa_H(C')$. This, together with the fact that H has no flags, allows us to show that both S and X are super-terminal components in H_L and, therefore, $C = C'$ and $S = X$.

The last step is to show (**s**) $\Rightarrow$ (9). Let us assume $T \setminus R = S \cup pa_H(C)$ with $S \in \mathscr{S}(C)$. By Lemma 4.1, S is a non-empty complete subset of C. Thus, $T \setminus R \neq \emptyset$ and (**s**) implies that S is a complete component in $H_{T \setminus R}$. Moreover, as H is a chain graph without flags the fact that $pa_H(C) \setminus M = pa_H(C) \cap R = \emptyset$ implies $pa_{H_M}(S) = pa_H(C) = (S \cup pa_H(C)) \setminus S = (T \setminus R) \setminus S = T \setminus (R \cup S)$. Thus, the condition (9) holds with $X = S$. □

A.6. Proof of Lemma 7.1

Proof 17. To show that H is a chain graph, assume for a contradiction that a semi-directed cycle in H exists. As G is a chain graph, the cycle has to contain a node in the set of "new" nodes $R = T \setminus M$. The only edges between nodes in M and R are arrows from $L \setminus Z$ directed to R and possibly lines between Z and R. Every section of the cycle with internal nodes in R between nodes in $Z = X$ can be shortened by a line in X; every section in R from $y \in L \setminus Z$ to $z \in Z = X$ can be replaced by a path of the form $y \to x - z$ in G_L or by an arrow $y \to z$; this follows from the condition in line 5 of Table 6. Thus, a semi-directed cycle in G is obtained, which contradicts the assumption that G is a chain graph.

To show the second statement of the lemma we use the characterization of essential graphs from Lemma 2.2. We already know that H has to be a chain graph and we are going to verify that H satisfies the other three conditions in Lemma 2.2. The assumption that G is an essential graph implies that it has no flags. Therefore, $y \to z$ in G for every $y \in L \setminus Z$ and $z \in Z$ owing to

the condition in line 5 of Table 6 (where $Z = X$). In particular, since the procedure **Extend** just adds lines between Z and R and within R and arrows from $L \setminus Z$ to R it does not create a new flag in the resulting graph H. Thus, H has no flags.

The next goal is to show that induced subgraphs for components in H are decomposable. The difference between the components in G and H is that either R becomes a new complete component in H or the component C in G containing $Z = X$ is enlarged by R (this happens if the condition in line 5 of Table 6 is valid). In the latter case, $C \cup R$ becomes a component in H, $K \equiv Z \cup R$ becomes a clique of the induced subgraph $H_{C \cup R}$ and the intersection of K with the other cliques of $H_{C \cup R}$ is just Z. Thus, $R \cup Z$ can be placed as the last clique in an ordering of those cliques satisfying the running intersection property. This argument allows us to show what is desired (see Section 2.1.1).

To show that no pair of components in H can be legally merged (see Section 2.2) we distinguish the same two cases. If the condition in line 5 of Table 6 is fulfilled then R just enlarges an "old" component in G. Because there is no arrow in H directed out of nodes in R, one has $pa_H(C) = pa_G(C \cap M)$ for every $C \in \mathscr{C}(H)$. Hence, two components in H can be legally merged only if the corresponding components in G can be legally merged. Thus, by the assumption about G, they cannot be legally merged. If the condition in line 5 of Table 6 is not valid then $Z = \emptyset$ and R becomes a "new" complete component in H and there are no arrows in H directed out of R. We can analogously observe that no pair of "old" components can be legally merged. It remains to be verified that R cannot be legally merged with another component in H. Assume the opposite for a contradiction. Thus, there must be a component C in G such that if we put $C_u = C$ and $C_\ell = R$ then the conditions {**i**}-{**ii**} from Section 2.2 hold. Observe that then the condition in line 5 of Table 6 is valid with $X = pa_H(R) \cap C$ and $L = pa_H(R)$, which contradicts the assumption. □

A.7. Proof of Theorem 1

Proof 18. This can be proven by induction on N. If $|N| = 1$ then H is the only possible graph over N. It can be viewed as a complete undirected graph over N. By formula (3), $u = 0$ and the subroutine **Reconstruct** has to return H in line 2 of Table 7.

If $|N| \geqslant 2$ then we already use the induction hypothesis: if the procedure **Reconstruct** is applied to a standard imset w over $\emptyset \neq M \subset N$ then it returns the respective essential graph. Now, we distinguish three cases.

If $u = 0$ then Lemma 6.1 says that the respective essential graph is a complete undirected graph over N, which is just the graph returned in line 2 of Table 7.

If $u \neq 0$ is not adapted to N then $\emptyset \neq core(u) \equiv M$ by Corollary 5.1; Lemma 6.2 says that the restriction w of u to the power set of $M \subset N$ is the standard imset for the induced subgraph H_M. It was already explained in the text above Table 4 that the result of the subroutine **Adapt** is just w. Therefore, according to the induction hypothesis, the graph G obtained in line 4 of Table 7 is simply H_M. The fact that w is adapted gives $core(G) = core(w) = M$, which means G has no super-terminal component. Therefore, if the subroutine **Extend** is applied to G and $T = N$ then the condition in line 5 of Table 6 is not fulfilled.[27] In particular, the graph H returned by the subroutine is obtained from G by "adding" a super-terminal component $N \setminus M$ and Lemma 6.2 implies it coincides with the respective essential graph over N.

If u is an adapted standard imset, then Lemma 6.3 describes how to obtain the standard imset w over a (special) proper subset $M \subset N$, which corresponds to the induced subgraph H_M of the respective essential graph H. This is exactly what is described in subroutine **Reduce** from Table 5. Recall that the condition (7) from Lemma 6.3 is replaced in line 4 of Table 5 with an equivalent condition (8) from Lemma 6.4, which also says $R = T \setminus W$. Note that $M \neq \emptyset$ because it contains $N \setminus T$ and $T \subset N$ for every $T \in \mathscr{T}_u$, by Corollary 5.1. By the induction hypothesis, the induced subgraph $G \equiv H_M$ is obtained in line 8 of Table 7. Now, realize that the output sets M and T from **Reduce** are input sets for **Extend**. In particular, the set R has the same meaning in both subroutines and $L = T \setminus R$. Moreover, by Lemma 6.4, R is the residual of K from the condition (7) and has the same meaning in Lemma 6.5. The fact that the subroutine **Extend** in line 9 of Table 7 results in the desired essential graph H follows from Lemma 6.5: indeed, if the condition in line 5 of Table 6 holds then (9) is fulfilled, which is equivalent to the condition (**s**) from Lemma 6.5. Because H is a chain graph without flags, the condition (**s**) corresponds to the case that $K \equiv S \cup R = X \cup R$ becomes a super-terminal component in H_T with $T \setminus (R \cup X)$ as the set of parents in H. This is what is described in lines 7–9 of Table 6 with $L = T \setminus R$ and $Z = X$. On the other hand, if the condition in line 5 of Table 6 does not hold then (9) is not valid, which is equivalent to the condition (**p**) from Lemma 6.5. This corresponds to the case that R becomes a super-terminal component in H_T with $L = T \setminus R$ as the set of parents in H. The respective reconstruction step is also described in lines 7–9 of Table 6, this time with $Z = \emptyset$. In particular, due to the fact there is no edge in H between R and $N \setminus T$ (see Lemma 6.3), in both cases the essential graph H is returned in line 10 of Table 7. □

A.8. Proof of Theorem 2

Proof 19. First, note that a basic necessary condition for a standard imset u over N is that the sum of absolute values of its values $\sum_{S \subseteq N} |u(S)|$ is at most $2n$. This follows directly from the formula (2) in Section 2.3 and the observation that the term

[27] This follows from the fact that $G = G_L$ has no super-terminal component. Indeed, as explained in the text above Lemma 7.1, if $G = H_M$ is an essential graph then the set X in line 5 of Table 6 has to be the super-terminal component in $G_L = G$.

$-\delta_\emptyset$ in (2) cancels with $+\delta_{pa_G(a^*)}$ for some $a^* \in N$. Thus, if the condition in line 2 of Table 13 is valid then u is not standard. Further necessary conditions on a standard imset u over N are $\sum_{S\subseteq N} u(S) = 0$ and $\forall i \in N \ \sum_{S, i\in S\subseteq N} u(S) = 0$: realize that if one has $u = u_G$ where u_G is given by the formula (2) in Section 2.3 then u satisfies these two conditions. Hence, if one of the conditions in line 3 of Table 13 is valid then u is again not standard.

We prove the desired statement by induction on N. If $|N| = 1$ then the only imset over N which is not excluded in line 3 is the zero imset. However, this imset is known to be a standard imset (see Lemma 6.1); therefore, the procedure gives the right answer in this case.

If $|N| \geqslant 2$ then we can use the induction hypothesis. It says that if the procedure **Testing** is applied to an imset w over $\emptyset \neq M \subset N$ then it gives the right answer. We distinguish three cases. If $u = 0$ then u is a standard imset and the same answer is given in line 4 of Table 13.

If $u \neq 0$ but $u(N) = 0$ then the adaptation subroutine from Table 11 is applied. If u is a standard imset then Corollary 5.1 implies that the largest set $M \subset N$ with $u(M) \neq 0$ exists. Thus, the set chosen in line 1 of Table 11 coincides with this largest set M then; moreover, $u(Q) = 0$ whenever $Q \subseteq N$, $Q \setminus M \neq \emptyset$. In particular, if the condition in line 2 of Table 11 is valid then u is not a standard imset. On the other hand, if the condition in line 2 is not satisfied then the difference between u and the resulting imset w over M is just in their domains: u is the "zero" extension of w to subsets of N. By the induction hypothesis, the result of recursively calling of **Testing** in line 6 of Table 13 is correct; that is, $\gamma = \mathsf{TRUE}$ iff w is a standard imset over M. We need to show that u is a standard imset iff w is a standard imset. One implication is given by Lemma 6.2; the converse implication can be verified as follows. If w is a standard imset over M and G the respective essential graph (over M), then by the construction described in Lemma 6.2 a graph H over N is obtained. Clearly, by Lemmas 2.1 and 2.3, it is a chain graph without flags equivalent to an acyclic directed graph. Since w is adapted by construction, $M = core(w) = core(G)$. The construction of H is such that $N \setminus M$ is a super-terminal components in H. Therefore, $core(H) = M$ and the formula (4) in Lemma 5.1 for u_H gives the same values as for w with the proviso that u has zero values for sets that are not subsets of M. Hence, $u = u_H$ is a standard imset over N.

If $u(N) \neq 0$ then the subroutine **Reduce*** is applied in line 7 of Table 13. If u is a standard imset then $u(N) = 1$ by Corollary 5.1 and, by Corollary 6.1, there exists $T \in \mathscr{T} \equiv \mathscr{T}_u$ satisfying the condition in line 5 of Table 12. In particular, if one of the conditions in line 4 of Table 12 holds then u is not a standard imset. However, if none of those conditions holds then one can find the respective set T in line 5 of Table 12. If u is standard, then, by Lemma 6.4, the set T chosen in line 5 has the form (7). Therefore, the assumptions of Lemma 6.3 are fulfilled, which implies that if we perform the steps in lines 6–8 of Table 12 then we get an imset $\tilde{w}$ that vanishes for sets that are not subsets of M. In particular, if the condition in line 9 is fulfilled then u is not a standard imset.

On the other hand, if conditions in lines 4 and 9 of Table 12 are not valid then the relation between u and the resulting imset w over M from line 11 of Table 12 is that the imset $\tilde{w}$ given in line 8 of Table 12 is the "zero" extension of w to all subsets of N. By the induction hypothesis, the result of recursively calling of **Testing** in line 8 of Table 13 is correct. Thus, $\gamma = \mathsf{TRUE}$ iff w is a standard imset over M. It remains to show that u is a standard imset iff w is a standard imset. One implication is given by Lemma 6.3; the converse can be shown directly. If w is a standard imset over M then consider the respective essential graph G over M and construct a graph H over N as follows: $H_M = G$, $\forall y \in T \setminus R$ and $z \in R$ include $y \to z$ in H and $\forall z_1, z_2 \in R$ $z_1 \neq z_2 \in R$ include $z_1 - z_2$ in H. Then, by Lemmas 2.1 and 2.3, H is a chain graph without flags equivalent to an acyclic directed graph: the only difference between G and H is that R is an additional terminal component with a single clique. In particular, formula (12) in Lemma 9.1 applied to both G and H implies that the relation between u_H and the "zero" extension $\tilde{w}$ of $w = u_G$ is this: $u_H - \tilde{w} = \delta_N - \delta_M - \delta_T + \delta_{T\setminus R}$. In particular, $u = u_H$ and u is a standard imset, which was the desired conclusion. □

A.9. Proof of Theorem 3

In our R implementation, we have not tried to achieve the best possible theoretical computational efficiency. Instead, we tried to utilize the routines that are already available in R. In the implementation, we represent every imset by an environment, which consists of a collection of variables. Each subset L of N is ascribed a unique natural number code defining the name of the corresponding (integer-valued) variable. The imset value $u(L)$ is kept as the value of that variable corresponding to L. Only subsets with non-zero values are represented in the environment (of variables). We have used the hash function available in R for efficient addressing of variables in the environment.

In the proof, an additive operation with two integers (summation, subtraction, comparison) is regarded as elementary, that is, of constant complexity $\mathscr{O}(n^0)$. The test whether two nodes $a, b \in N$ coincide will be considered to be of constant complexity as well. Thus, testing whether $a \in L$ for $a \in N$, $L \subseteq N$ with $|N| = n$, is of linear complexity $\mathscr{O}(n^1)$ and, given two subsets A, B of N, the usual set operations (creating $A \cap B$, $A \setminus B$, $A \cup B$ and testing whether $A = B$ or $A \subseteq B$) are considered to be operations of quadratic complexity $\mathscr{O}(n^2)$. Transition from a subset $L \subseteq N$ to its natural number code (and setting up the corresponding integer-valued variable) will be considered of quadratic complexity $\mathscr{O}(n^2)$. Since the comparison of two integers is $\mathscr{O}(n^0)$, once subsets $A, B \subseteq N$ have set up their codes (and variables) testing whether $A = B$, $u(A) > 0$ or $u(B) < 0$, respectively, can already be done with constant complexity $\mathscr{O}(n^0)$. A hybrid graph over N is represented by an incidence matrix in our implementation. Thus, given a pair of nodes $a, b \in N$, testing whether $a—b$, $a \to b$ or $a \leftarrow b$ in the graph is also considered as the operation of constant complexity $\mathscr{O}(n^0)$.

Proof 20. In the following considerations we implicitly use the fact that if an imset u is an input for subroutines **Adapt** or **Reduce** (Tables 4 and 5) then it satisfies the following conditions:

$$\sum_{S\subseteq N}|u(S)|\leqslant 2|N|=2n \quad \text{and} \quad \sum_{S\subseteq N}u(S)=0. \tag{20}$$

This is because the input for these subroutines is always a standard imset over (a subset of) N (c.f. Sections A.7 and A.8). The condition (20) implies that the complete list of sets L with $u(L)\neq 0$ has at most $2n$ items, which is a fact used repeatedly below. Observe that the input imset for the modified versions **Adapt*** and **Reduce*** (Tables 11 and 12), which are used in the algorithm **Testing** (Table 13), also always satisfies (20).[28]

Now, let us discuss the complexity of the subroutine **Adapt** from Table 4. To find the maximal set M in line 1 one needs to make at most $2n-1$ inclusion tests, which are of quadratic complexity. Thus, the step in line 1 has $\mathcal{O}(n^3)$ complexity. In line 2 of Table 4 one goes through the list of sets L with $u(L)\neq 0$, which has at most $2n$ items. For each set one makes the test whether $L\subseteq M$ and if this is not the case, the set is "removed" from the list. Thus, the step in line 2 has $\mathcal{O}(n^3)$ complexity as well. Therefore, the whole subroutine **Adapt** has complexity $\mathcal{O}(n^3)$ and the same arguments hold for the subroutine **Adapt*** from Table 11.

As concerns the subroutine **Reduce** from Table 5, the creation of the lists $\mathscr{T}$ and $\mathscr{L}$ in lines 1 and 2 and setting up codes for the sets in the lists is at most of complexity $\mathcal{O}(n^3)$; the resulting lists will have the length at most $2n$. The step in line 3 is of $\mathcal{O}(n^3)$ since to get union of at most $2n$ sets it is enough to do $2n-1$ pairwise unions. Given a set $T\in\mathscr{T}$ (and the set W) in line 4, testing whether $T\setminus W\neq\emptyset$ is $\mathcal{O}(n^2)$ and creating $T\cap W$ is also $\mathcal{O}(n^2)$. Now, testing whether $T\cap W\in\mathscr{L}\cup\{\emptyset\}$ is already of linear complexity.[29] Since one has at most $2n$ sets T to be tested in line 4, the overall complexity of this step is $\mathcal{O}(n^3)$. The steps in lines 5 and 6 are $\mathcal{O}(n^2)$ and the step in line 7 can even be done in $\mathcal{O}(n^1)$.[30] The step in line 8 is identical with the step in line 2 of **Adapt** and has, therefore, $\mathcal{O}(n^3)$ complexity. Altogether, the whole subroutine **Reduce** has $\mathcal{O}(n^3)$ complexity. The same arguments can be applied to the subroutine **Reduce*** from Table 12.

In the subroutine **Extend** from Table 6, steps in lines 1–3 have at most $\mathcal{O}(n^2)$ complexity. The test for $L\neq\emptyset$ in line 4 is of quadratic complexity $\mathcal{O}(n^2)$ as well. The restriction to (=the creation of) the induced subgraph G_L in line 4 can be done with $\mathcal{O}(n^3)$ complexity.[31] Finding a terminal component X in G_L is also of $\mathcal{O}(n^3)$ complexity.[32] Testing whether X is complete in G_L in line 5 of Table 6 is of $\mathcal{O}(n^2)$ complexity, the same complexity as finding of $pa_G(X)$ and the test for $pa_G(X)=L\setminus X$ as well. Thus, the step in line 5 has quadratic complexity $\mathcal{O}(n^2)$. Finally, determining the graph H in lines 7–9 is again the step having at most of $\mathcal{O}(n^3)$ complexity. Therefore, the whole **Extend** subroutine has at most $\mathcal{O}(n^3)$ complexity.

Now, as concerns the recursive algorithm **Reconstruct** from Table 7, the step in line 1 has at most quadratic complexity and calling **Adapt**, **Reduce** or **Extend** is at most of $\mathcal{O}(n^3)$ complexity. The recursion depth is at most n: this is because in each round at least one variable is "removed" from the considered set of variables. Thus, complexity of the algorithm is at most $\mathcal{O}(n^4)$. The same argument holds for the non-recursive version of the algorithm, which only differs in the order of calling subroutines.

As concerns the recursive algorithm **Testing** in Table 13, the arguments are analogous. The complexity of testing the condition in line 2 of Table 13 depends on the internal computer representation of the imset u. If only non-zero values are kept, then despite the fact the memory demands for representing a general imset u over N could be exponential, testing whether the condition in line 2 is valid requires summing of at most 2n integers, which is of linear complexity.[33] Then testing in line 3 of Table 13 is of $\mathcal{O}(n^3)$ complexity. The subroutines **Adapt*** or **Reduce*** are also called at most n times, which implies that **Testing** is of (at most) $\mathcal{O}(n^4)$ complexity. □

References

[1] S.A. Andersson, D. Madigan, M.D. Perlman, A characterization of Markov equivalence classes for acyclic digraphs, Annals of Statistics 25 (1997) 505–541.
[2] S.A. Andersson, D. Madigan, M.D. Perlman, On the Markov equivalence of chain graphs, undirected graphs and acyclic digraphs, Scandinavian Journal of Statistics 24 (1997) 81–102.
[3] V. Auvrey, L. Wehenkel, On the construction of the inclusion boundary neighbourhood for Markov equivalence classes of Bayesian network structures, in: A. Darwiche, N. Friedman (Eds.), Uncertainty in Artificial Intelligence, vol. 18, Morgan Kaufmann, 2002, pp. 26–35.
[4] R.R. Bouckaert, Bayesian belief networks: from construction to inference, PhD thesis, University of Utrecht, 1995.

[28] This is because the validity of (20) is tested in lines 2 and 3 of **Testing** before those subroutines are applied.

[29] Realize it is enough to compare the natural number code for $T\cap W$ with at most $2n$ natural numbers, the codes corresponding to sets in $\mathscr{L}\cup\{\emptyset\}$.

[30] One goes through the list of sets L with $u(L)\neq 0$ and makes at most 4 modifications of constant complexity. Recall that testing whether two natural number codes coincide is of constant complexity.

[31] One goes through the entries of the incidence matrix indexed by pairs $[a,b]\in N\times N$, makes linear tests whether $a\in L$ and $b\in L$ and, if not, modifies the respective entry.

[32] The iterations of the respective procedure are connected sets in the graph. The starting iteration is a single node a. Given a connected set C, we test whether $ch(C)\equiv\{b\in N: c\to b \text{ for } c\in C\}\neq\emptyset$, which is of $\mathcal{O}(n^2)$. If yes, the next iteration will be $\{b\}$, where b is an arbitrary node from $ch(C)$. If not, the next iteration will be $C\cup ne(C)$, whose construction is also $\mathcal{O}(n^2)$. The procedure finishes if $C=C\cup ne(C)$, with a terminal component $X=C$. The procedure clearly has at most n iterations.

[33] If there are further non-zero values in the list, then $\gamma=$ FALSE, and if there is no further non-zero value then one has the whole sum and knows whether it exceeds $2n$ or not.

[5] R.R. Bouckaert, M. Studený, Racing algorithms for conditional independence inference, International Journal of Approximate Reasoning 45 (2007) 386–401.
[6] R. Castelo, The discrete acyclic digraph Markov model in data mining, PhD thesis, University of Utrecht, 2002.
[7] D.M. Chickering, Optimal structure identification with greedy search, Journal of Machine Learning Research 3 (2002) 507–554.
[8] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer-Verlag, 1999.
[9] M. Frydenberg, The chain graph Markov property, Scandinavian Journal of Statistics 17 (1990) 333–353.
[10] T. Kočka, R. Castelo, Improved learning of Bayesian networks, in: J. Breese, D. Koller (Eds.), Uncertainty in Artificial Intelligence, vol. 17, Morgan Kaufmann, 2001, pp. 269–276.
[11] S.L. Lauritzen, Graphical Models, Clarendon Press, 1996.
[12] C. Meek, Causal inference and causal explanation with background knowledge, in: P. Besnard, S. Hanks (Eds.), Uncertainty in Artificial Intelligence, vol. 11, Morgan Kaufmann, 1995, pp. 403–410.
[13] C. Meek, Graphical models, selecting causal and statistical models, PhD thesis, Carnegie Melon University, 1977.
[14] R.O. Puch, J.Q. Smith, C. Bielza, Hierarchical junction trees, conditional independence preservation and forecasting in dynamic Bayesian networks with heterogeneous evolution, in: J.A. Gámez, S. Moral, A. Salmerón (Eds.), Advances in Bayesian Networks, Springer-Verlag, 2004, pp. 57–75.
[15] A. Roverato, A unified approach to the characterization of equivalence classes for DAGs, chain graphs with no flags and chain graphs, Scandinavian Journal of Statistics 32 (2005) 295–312.
[16] T. Silander, P. Myllymäki, A simple approach for finding the globally optimal Bayesian network structure, in: R. Dechter, T. Richardson (Eds.), Uncertainty in Artificial Intelligence, vol. 22, AUAI Press, 2006, pp. 445–452.
[17] M. Studený, A recovery algorithm for chain graphs, International Journal of Approximate Reasoning 17 (1997) 265–293.
[18] M. Studený, Characterization of essential graphs by means of the operation of legal merging of components, International Journal of Uncertainty, Fuzziness and Knowledge-based Systems 12 (2004) 43–62.
[19] M. Studený, Probabilistic Conditional Independence Structures, Springer-Verlag, 2005.
[20] M. Studený, Characterization of inclusion neighbourhood in terms of the essential graph, International Journal of Approximate Reasoning 38 (2005) 283–309.
[21] M. Studený, J. Vomlel, A geometric approach to learning BN structures, in: M. Jaeger, T.D. Nielsen (Eds.), Proceedings of the Fourth European Workshop on Probabilistic Graphical Models, Hirtshals, Denmark, September 2008, pp. 281–288.
[22] M. Studený, A. Roverato, Š. Štěpánová, Two operations of merging and splitting components in a chain graph, Kybernetika 45 (2009), in press.
[23] J. Vomlel, M. Studený, Graphical and algebraic representatives of conditional independence models, in: P. Lucas, J.A. Gámez, A. Salmerón (Eds.), Advances in Bayesian Networks, Springer, Verlag, 2007, pp. 55–80.
[24] J. Vomlel, M. Studený, Using imsets for learning Bayesian networks, in: Proceedings of the 10th Czech-Japan Seminar on Data Analysis and Decision Making under Uncertainty, Liblice, Czech Republic, September 2007, pp. 178–189.
[25] T. Verma, J. Pearl, Equivalence and synthesis of causal models, in: P.P. Bonissone, M. Henrion, L.N. Kanal, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, vol. 6, Elsevier, 1991, pp. 220–227.