Journal of Machine Learning Research 11 (2010) 3453-3479 Submitted 3/10; Revised 9/10; Published 11/10

Efficient Algorithms for Conditional Independence Inference

Remco Bouckaert[*] REMCO@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikato
Hamilton 3240, New Zealand

Raymond Hemmecke HEMMECKE@MA.TUM.DE
Silvia Lindner SLINDNER@MA.TUM.DE
Zentrum Mathematik
Technische Universtität Munich
Boltzmannstrasse 3
85747 Garching, Germany

Milan Studený STUDENY@UTIA.CAS.CZ
Institute of Information Theory and Automation of the ASCR
Pod Vodárenskou věží 4
18208 Prague, Czech Republic

Editor: Rina Dechter

Abstract

The topic of the paper is computer testing of (probabilistic) *conditional independence* (CI) implications by an algebraic method of structural imsets. The basic idea is to transform (sets of) CI statements into certain integral vectors and to verify by a computer the corresponding algebraic relation between the vectors, called the *independence implication*. We interpret the previous methods for computer testing of this implication from the point of view of polyhedral geometry. However, the main contribution of the paper is a new method, based on *linear programming* (LP). The new method overcomes the limitation of former methods to the number of involved variables. We recall/describe the theoretical basis for all four methods involved in our computational experiments, whose aim was to compare the efficiency of the algorithms. The experiments show that the LP method is clearly the fastest one. As an example of possible application of such algorithms we show that testing inclusion of Bayesian network structures or whether a CI statement is encoded in an acyclic directed graph can be done by the algebraic method.

Keywords: conditional independence inference, linear programming approach

1. Introduction

First, we explain the motivation and mention some previous work. Then we describe the aim and structure of the paper.

1.1 Motivation

Conditional independence (CI) is a highly important concept in statistics and artificial intelligence. Properties of probabilistic CI provide theoretical justification for the method of local computation

∗. Also at the University of Auckland, New Zealand.

(Cowel et al., 1999) which is at the core of probabilistic expert systems (Jensen, 2001), successfully applied in numerous areas. The importance of CI is given by its interpretation in terms of *relevance among symptoms* or variables in consideration (Pearl, 1988); that's why it is crucial in probabilistic reasoning. Traditional methods for describing (statistical models of) CI structures use graphs whose nodes correspond to variables in consideration; it leads to popular *graphical models* (Lauritzen, 1996).

Formal properties of probabilistic CI and the attempts to describe probabilistic *CI inference* in terms of mathematical logic have been traditional research topics since the late 1970's. Basic properties of CI, now known as the *semi-graphoid properties*, were accentuated in statistics by Dawid (1979). Pearl (1988) interpreted those properties, formulated in the form of simple implications between CI statements, as axioms for the relevance relation. Moreover, Pearl proposed to view graphs as "inference engines" devised for efficient representing and manipulating relevance relationships. His idea (see Pearl, 1988, page 14) was to use an *input list of CI statements* to construct a graph and then, by using a special graphical separation criterion, to read from the graph additional CI statements, implied by the input list through the axioms. The goal of such inference procedures is to enable one to determine, at any state of knowledge, what information is relevant to the task at hand and what can be ignored.

Pearl's intention led him to a conjecture that CI inference for discrete probabilistic distributions can be characterized by the semi-graphoid properties. The conjecture was refuted in Studený (1989); actually, it has been shown later that there is no finite system of properties of semi-graphoid type characterizing (discrete probabilistic) CI inference (Studený, 1992). Thus, the question of computer testing of CI inference became a topic of research interest.

In Studený (2005), the method of *structural imsets* has been proposed as a non-graphical algebraic device for describing probabilistic CI structures; its advantage over graphical approaches is that it allows one to describe any discrete probabilistic CI structure. The idea is to use, instead of graphs, certain special integral vectors (of high dimension), called *imsets*, as tools for describing CI structures and implementing the "inference engine" for CI implication. This is because the corresponding algebraic relation between structural imsets, called *independence implication*, gives a sufficient condition for (probabilistic) CI inference. The topic of this paper is computer testing of this implication. The intended use is

- computer testing of implications between CI statements, and
- checking equivalence of structural imsets by an algebraic method.

Another (indirect) source of motivation for this paper is learning *Bayesian network* (BN) structures by score and search methods (Neapolitan, 2004). The point is that every BN structure can be described uniquely by a simple algebraic representative, called the *standard imset*. It was shown in Studený (2005, Chapter 8) that every reasonable scoring function (for learning BN structures) can be viewed as an affine function of the standard imset. This observation opens the way to the application of efficient *linear programming* (LP) methods in this area (Studený, Vomlel, and Hemmecke, 2010). Nevertheless, if a graphical model of CI structure is described by an imset, a natural question arises: is there any criterion, a counterpart of the graphical separation criterion, which allows one to read from this algebraic representative all CI statements encoded in it? This question again leads to the task of (computer) testing independence implication.

1.2 Some Former Algorithms

Given structural imsets u and v (over a set of variables N), the intuitive meaning of the implication $u \rightharpoonup v$ ($\equiv$ u independence implies v) is that the CI structure induced by u contains the CI structure induced by v. In Studený (2005, § 6.2) two algebraic characterizations of the relation $u \rightharpoonup v$ were given. They established the theoretical basis for the first-generation algorithms for computer testing of $u \rightharpoonup v$. The limitation of these algorithms is that to implement them fully one needs some additional information obtainable as the result of computations, which were performed only for $|N| \leq 5$ (Studený, Bouckaert, and Kočka, 2000).

The theoretical aspects of their implementation were analyzed in Studený (2004), while their practical implementations were described in detail in Bouckaert and Studený (2007). Basically, there are two algorithms, which can be interpreted as mutually complementary procedures. One of them, based on so-called "direct" characterization of $u \rightharpoonup v$ (see Studený, 2005, § 6.2.1), is suitable to confirm the implication; that's why it was called the *verification algorithm* in Bouckaert and Studený (2007). The other algorithm, based on so-called "skeletal" characterization of $u \rightharpoonup v$ (see Studený, 2005, § 6.2.2), fits to disproving the implication; that's why it was named the *falsification algorithm*.

The main idea of Bouckaert and Studený (2007) was to combine both algorithms to get a more effective tool. Owing to the computations from Studený et al. (2000), the implemented versions of both these algorithms are guaranteed to give a decisive answer to any independence implication problem for $|N| \leq 5$. Nevertheless, the combined version has also been implemented for $|N| = 6$, although without a guaranteed response to each implication problem. Recently, the first-generation algorithms from Bouckaert and Studený (2007) have been applied (in a modified form) in connection with the lattice-theoretic approach to CI inference (Niepert, van Gucht, and Gyssens, 2008).

1.3 Aim of the Paper

In this paper, we bring a new view on the problem of testing CI inference. First, we interpret the previous methods from the point of view of polyhedral geometry. Second, this geometric interpretation and the *linear programming approach* lead to new methods and, consequently, to the second-generation algorithms for testing CI inference, which appear to be more efficient than the first-generation algorithms.

More specifically, the geometric view has recently helped to solve an open problem that was very closely related to the topic of CI inference (see Studený, 2005, Question 7). It was the question whether every structural imset is a *combinatorial imset*, that is, whether it can be written as the sum of elementary imsets (= imsets corresponding to elementary CI statements). The question has been answered negatively in Hemmecke et al. (2008), where an example of a structural imset over 5 variables was found that is not combinatorial.

This fact naturally leads to a more advanced question motivated by the topic of CI inference: what is the so-called *Hilbert basis* of the cone generated by standard imsets (cf., Studený, 2005, Theme 10). A recent achievement is that the Hilbert basis for $|N| = 5$ has been found as a result of computations by Bruns et al. (2010). This makes it possible to design two modifications of the verification algorithm for 5 variables. In fact, the CI inference problem is transformed to testing whether a given imset is combinatorial, respectively structural.

Another important idea brought by this paper is that every testing $u \rightharpoonup v$ task can be re-formulated as a special LP problem. Note that the LP approach was mentioned as a potential method for testing

this implication already in Studený (2004, § 5). However, in the present paper, we re-formulate this implication problem in a different way. The LP approach has the following advantages:

- it allows one to go far beyond the limit of 5 variables,
- the second-generation algorithms are much faster than the first-generation ones,
- the re-formulation makes it possible to apply highly effective software packages developed to solve LP problems.

The goal of this paper is to describe the idea of the new approach, the corresponding algorithms and the experiments, whose aim was to compare the new algorithms (with the old ones).

Moreover, to give an example of the possible use of the algorithms we prove a result related to learning Bayesian networks. We characterize the inclusion of BN structures in terms of their algebraic representatives, standard imsets. Given acyclic directed graphs G and H over N, we show that the BN structure induced by H is contained in the one induced by G iff the difference $u_G - u_H$ of their standard imsets is a combinatorial imset, which happens iff it is a structural imset. Thus, testing the inclusion can be transformed to testing combinatorial imsets (or structural ones). A consequence of this observation is an elegant algebraic procedure for reading CI statements represented in a standard imset u_G, a kind of counterpart of the graphical separation criterion. Since the standard imset u_G is quite simple we have reasons to conjecture that our procedure can be implemented with polynomial complexity with respect to $|N|$.

1.4 Structure of the Paper

Section 2 is devoted to the terminology for CI inference using structural imsets, while basic concepts from polyhedral geometry are recalled in Section 3. In Section 4, the methods we are using are explained and, in Section 5, the experiments we performed are described. In Conclusions we discuss the perspectives and formulate some open problems. Appendices contain some proofs, a table of types of the Hilbert basis elements for 5 variables, an illustrative example and a comment on possible interpretation of the LP method.

2. Concepts Concerning CI Inference

The symbol N will be used to denote a non-empty finite set of *variables* in consideration. Given disjoint sets of variables $A, B \subseteq N$, the juxtaposition AB will denote their union.

2.1 Conditional Independence

Let P be a discrete probability distribution over N specified by its density $p : \mathsf{X}_N \to [0,1]$, where $\mathsf{X}_N \equiv \prod_{i \in N} \mathsf{X}_i$ is the joint sample space, a product of non-empty finite individual sample spaces. Given $A \subseteq N$ and $x \in \mathsf{X}_N$, let x_A denote the respective marginal configuration of x and p_A the marginal density of p; by convention $p_\emptyset(-) = 1$. Given pairwise disjoint $A, B, C \subseteq N$, we say that A is *conditionally independent of B given C* with respect to P if

$$\forall x \in \mathsf{X}_N \quad p_{A \cup B \cup C}(x_{A \cup B \cup C}) \cdot p_C(x_C) = p_{A \cup C}(x_{A \cup C}) \cdot p_{B \cup C}(x_{B \cup C}) \,.$$

We write $A \perp\!\!\!\perp B|C\,[P]$ then and call it a *CI statement* with respect to P; if P is not material we just use the symbol $A \perp\!\!\!\perp B|C$. The *CI structure* induced by P consists of all CI statements valid with respect to P.

Now, let L be a set of CI statements, called an *input list*, and t a CI statement outside L. We say L *probabilistically implies* t if, for every discrete probability distribution P, whenever all statements in L are valid with respect to P then t is valid with respect to P, too. Classic examples of valid probabilistic CI implications are the *semi-graphoid properties* (Pearl, 1988):

Symmetry	$A \perp\!\!\!\perp B\|C$	$\Longrightarrow$	$B \perp\!\!\!\perp A\|C$,
Decomposition	$A \perp\!\!\!\perp BD\|C$	$\Longrightarrow$	$A \perp\!\!\!\perp D\|C$,
Weak union	$A \perp\!\!\!\perp BD\|C$	$\Longrightarrow$	$A \perp\!\!\!\perp B\|DC$,
Contraction	$A \perp\!\!\!\perp B\|DC$ & $A \perp\!\!\!\perp D\|C$	$\Longrightarrow$	$A \perp\!\!\!\perp BD\|C$.

A CI statement $A \perp\!\!\!\perp B|C$ is called *elementary* if both A and B are singletons and it is called *trivial* if either $A = \emptyset$ or $B = \emptyset$. Since trivial statements are always valid, it follows from the semi-graphoid properties that, for any discrete probability distribution P, the CI structure induced by P is uniquely determined by the list of elementary CI statements valid with respect to P.

2.2 Imsets

An *imset* over N is an integer-valued function on the power set of N, denoted by $\mathcal{P}(N)$ in the rest of the paper.[1] It can be viewed as a vector whose components, indexed by subsets of N, are integers. An easy example is the *zero imset*, denoted by 0; another simple example is the identifier δ_A of a subset $A \subseteq N$:

$$\delta_A(S) = \begin{cases} 1 & \text{if } S = A, \\ 0 & \text{if } S \subseteq N,\ S \neq A. \end{cases}$$

An imset associated with a CI statement $A \perp\!\!\!\perp B|C$ is then the combination

$$u_{\langle A,B|C\rangle} \equiv \delta_{ABC} + \delta_C - \delta_{AC} - \delta_{BC}\,.$$

The class of *elementary imsets* (over N), denoted by $\mathcal{E}(N)$, consists of imsets associated with elementary CI statements (over N).

A *combinatorial imset* is an imset u that can directly be decomposed into elementary imsets, that is,

$$u = \sum_{w \in \mathcal{E}(N)} k_w \cdot w \qquad \text{for some } k_w \in \mathbb{Z}_+\,.$$

We denote the class of combinatorial imsets over N by $\mathcal{C}(N)$.

An imset u over N will be called *structural* if there exists $n \in \mathbb{N}$ such that the multiple $n \cdot u$ is a combinatorial imset, that is,

$$n \cdot u = \sum_{w \in \mathcal{E}(N)} k_w \cdot w \quad \text{for some } n \in \mathbb{N},\ k_w \in \mathbb{Z}_+\,.$$

In other words, a structural imset is an imset which is a combination of elementary ones with non-negative rational coefficients. The class of structural imsets over N will be denoted by $\mathcal{S}(N)$; this

1. The word **imset** is an abbreviation for **i**nteger-valued **m**ulti**set**.

class of imsets is intended to describe CI structures. Clearly, $\mathcal{E}(N) \subseteq \mathcal{C}(N) \subseteq \mathcal{S}(N)$. One has $\mathcal{C}(N) = \mathcal{S}(N)$ for $|N| \leq 4$ (Studený, 1991), but $\mathcal{S}(N) \neq \mathcal{C}(N)$ for $|N| = 5$ (Hemmecke et al., 2008).

However, one can show (see Studený, 2005, Lemma 6.3) that there exists a smallest $n_* \in \mathbb{N}$, depending on $|N|$, such that $u \in \mathcal{S}(N) \Leftrightarrow n_* \cdot u \in \mathcal{C}(N)$ for every imset u over N. One has $n_* = 1$ for $|N| \leq 4$ (Studený, 1991) and one of the news brought by this paper is that $n_* = 2$ for $|N| = 5$ (see Proposition 6 in Appendix B). An open question is what is the exact value of n_* for every $|N|$ (cf., Studený, 2005, Theme 11).

2.3 Independence Implication

Let u, v be structural imsets over N. We say that u *independence implies* v and write $u \rightharpoonup v$ if there exists $k \in \mathbb{N}$ such that $k \cdot u - v$ is a structural imset:

$$u \rightharpoonup v \quad \text{iff} \quad \exists k \in \mathbb{N} \quad k \cdot u - v \in \mathcal{S}(N)\,. \tag{1}$$

Now, given $u \in \mathcal{S}(N)$ the corresponding CI structure $\mathcal{M}(u)$ consists of all CI statements $A \perp\!\!\!\perp B | C$ such that $u \rightharpoonup u_{\langle A,B|C\rangle}$. The importance of structural imsets follows from the fact that, for every discrete probability distribution P over N, there exists $u \in \mathcal{S}(N)$ such that the CI structure induced by P coincides with $\mathcal{M}(u)$ (use Studený, 2005, Theorem 5.2).

An equivalent characterization of independence implication is as follows. A real set function $m : \mathcal{P}(N) \to \mathbb{R}$ is called *supermodular* iff

$$m(C \cup D) + m(C \cap D) \geq m(C) + m(D) \qquad \text{for every } C, D \subseteq N.$$

Such a function can be viewed as a real vector whose components are indexed by subsets of N. The *scalar product* of m and an imset u over N is then

$$\langle m, u \rangle \equiv \sum_{S \subseteq N} m(S) \cdot u(S)\,.$$

The dual definition of independence implication is this (see Studený, 2005, Lemma 6.2): one has $u \rightharpoonup v$ iff, for every supermodular function m over N,

$$\langle m, u \rangle = 0 \quad \Longrightarrow \quad \langle m, v \rangle = 0\,. \tag{2}$$

The independence implication can be viewed as a sufficient condition for probabilistic CI implication. More specifically, given an input list L of CI statements, let us "translate" every statement s in L into the associated imset u_s and introduce a combinatorial imset $u_L \equiv \sum_{s \in L} u_s$. Then one has:

Proposition 1 Let L be an input list of CI statements and t another CI statement over N. If $u_L \rightharpoonup u_t$ then L probabilistically implies t.

The proof, based on observations from Studený (2005), is given in Appendix A. Note that the above-mentioned condition is a sufficient condition for the probabilistic CI implication, but not a necessary one. On the other hand, the analysis in the case of four variables suggests that it is quite good approximation of probabilistic implication. For $|N| = 4$, one has 22108 (formal) CI structures induced by structural imsets, while the actual number of (discrete) probabilistic CI structures is 18478 (Šimeček, 2007), and there are only about 20 types of probabilistic CI implications that are not derivable through Proposition 1.

The reader may be interested in whether there exists a limit for the factor k in (1). If we assume that the "input" u is a combinatorial imset and the "output" v an elementary one then such a limit exists. One can show (cf., Studený, 2005, Lemma 6.4) that the least $k_* \in \mathbb{N}$ exists such that

$$u \rightharpoonup v \quad \text{iff} \quad k_* \cdot u - v \in \mathcal{S}(N)\,.$$

It depends on $|N|$: $k_* = 1$ for $|N| \leq 4$ and $k_* = 7$ for $|N| = 5$ (Studený et al., 2000). It is an open question what is the exact value of k_* for $|N| \geq 6$ (cf., Studený, 2005, Theme 12).

2.4 Bayesian Network Translation

Bayesian networks (Pearl, 1988; Neapolitan, 2004) can be interpreted as graphical models of CI structures assigned to acyclic directed graphs. More specifically, given an acyclic directed graph G having N as the set of its nodes, the corresponding graphical separation criterion (for the definition see Lauritzen, 1996, § 3.2.2) defines the collection $I(G)$ of CI restrictions given by G. The corresponding statistical model then consists of probability distributions on X_N that are *Markovian with respect to G*, that is, satisfy those CI restrictions. Given two acyclic directed graphs G and H over N, we say they are *independence equivalent* if $I(G) = I(H)$; it essentially means they define the same statistical model.

Given an acyclic directed graph G over N, the *standard imset* for G, denoted by u_G, is given by the formula

$$u_G = \delta_N - \delta_\emptyset + \sum_{i \in N} \{\, \delta_{\mathrm{pa}_G(i)} - \delta_{\{i\} \cup \mathrm{pa}_G(i)} \,\}, \tag{3}$$

where $\mathrm{pa}_G(i) \equiv \{j \in N;\, j \to i \ \text{ in } G\}$ denotes the set of *parents* of a node i in G. Lemma 7.1 in Studený (2005) says that u_G is always a combinatorial imset and $\mathcal{M}(u_G) = I(G)$. Thus, the graphical separation criterion applied to G can be replaced by an algebraic criterion applied to u_G. Moreover, Corollary 7.1 in Studený (2005) adds that $u_G = u_H$ iff G and H are independence equivalent. That means, the standard imset is a unique representative of the equivalence class of graphs.

Note that u_G need not be the only combinatorial imset defining $I(G)$; it is the simplest such imset, a kind of "standard" representative. Using standard imsets we can easily characterize the inclusion for Bayesian networks:

Proposition 2 Let G and H be acyclic directed graphs over N. Then $I(H) \subseteq I(G)$ if and only if $u_G - u_H \in \mathcal{C}(N)$, which is also equivalent to $u_G - u_H \in \mathcal{S}(N)$.

Proof The equivalence $I(H) \subseteq I(G) \Leftrightarrow u_G - u_H \in \mathcal{C}(N)$ is proved in (Studený, 2005, Lemma 8.6). Clearly, $u_G - u_H \in \mathcal{C}(N)$ implies $u_G - u_H \in \mathcal{S}(N)$. Conversely, if $u_G - u_H \in \mathcal{S}(N)$ then, by (1), $u_G \rightharpoonup u_H$ and it makes no problem to show $\mathcal{M}(u_H) \subseteq \mathcal{M}(u_G)$. This means $I(H) \subseteq I(G)$. ■

We have some reasons to conjecture that testing whether a difference of two standard imsets is a combinatorial imset can be done with polynomial complexity in $|N|$; see Conclusions for the discussion. Moreover, let us emphasize that as far as we are aware of, there is no direct graphical characterization of independence inclusion; in our view, it is quite difficult task to describe this in graphical terms. Note that indirect transformational graphical description of independence inclusion by Chickering (2002) does not provide an algorithm for testing the inclusion for a given pair of

acyclic directed graphs G and H. Of course, a different situation is with testing independence equivalence, in which case a simple graphical characterization and algorithms are available.

Proposition 2 has the following consequence, which allows one to replace the graphical separation criterion by testing whether an imset is combinatorial.

Corollary 3 Given an acyclic directed graph G over N, the class $I(G)$ coincides with the collection of CI statements $A \perp\!\!\!\perp B|C$ such that $u_G - u_{\langle A,B|C\rangle} \in \mathcal{C}(N)$, respectively $u_G - u_{\langle A,B|C\rangle} \in \mathcal{S}(N)$.

Proof It is enough to construct, for every CI statements $A \perp\!\!\!\perp B|C$, an acyclic directed graph H such that $u_{\langle A,B|C\rangle} = u_H$. To this end, consider a total order of nodes in N in which C is followed A, B and $N \setminus ABC$, direct edges of the complete undirected graph over N according to this order and remove the arrows between A and B. ■

Of course, as mentioned above, there are elegant graphical separation criteria for testing whether a CI statement is represented in an acyclic directed graph. There are no doubts that these criteria can be implemented with polynomial complexity in $|N|$ (see Geiger, Verma, and Pearl, 1989) and that they are also appropriate for human users. However, our method proposed in Corollary 3 may appear to be suitable for computer testing, in particular in the situation when the Bayesian network structure is internally represented in a computer by the standard imset, as suggested in Studený, Vomlel, and Hemmecke (2010).

3. Basic Concepts from Polyhedral Geometry

Here we recall some well-known concepts and facts from the theory of convex polyhedra. Proofs can be found in textbooks on linear programming, for example Schrijver (1986).

Vectors are regarded as column vectors. Given two vectors $\mathbf{x}, \mathbf{y}$, their scalar product will be denoted either by $\langle \mathbf{x}, \mathbf{y} \rangle$ or, using matrix multiplication notation, by $\mathbf{x}^\intercal \mathbf{y}$, where $\mathbf{x}^\intercal$ is the transpose of $\mathbf{x}$. The inequality $\mathbf{x} \leq \mathbf{y}$ is meant in all components, and $\mathsf{0}$, respectively $\mathsf{1}$, is a vector all whose components are zeros, respectively ones.

3.1 Polyhedra, Polyhedral Cones and Farkas' Lemma

Systems of linear inequalities appear in many areas of mathematics. Let $\mathbf{A}$ be a $d \times n$-matrix and $\mathbf{b}$ a d-vector. The set of all solutions in $\mathbb{R}^n$ to the linear system $\mathbf{Ax} \leq \mathbf{b}$ is called a *polyhedron*. If the defining matrix $\mathbf{A}$ and the right-hand side vector $\mathbf{b}$ are rational, the polyhedron is said to be *rational*. Polyhedra defined via homogeneous inequalities $\mathbf{Ax} \leq \mathsf{0}$ are called *polyhedral cones*.

A non-trivial fundamental result in polyhedral geometry states that every polyhedral cone C can equivalently be introduced as the *conical hull* $\mathrm{cone}(\mathsf{V})$ of finitely many points $\mathsf{V} \subseteq \mathbb{R}^n$, that is, the set of all finite conical combinations $\sum_t \lambda_t \cdot \mathbf{v}_t$ (with $\lambda_t \geq 0$ for all t) of elements $\mathbf{v}_t \in \mathsf{V}$ (see Schrijver, 1986, Corollary 7.1a). If the polyhedral cone is *rational* then it is the conical hull of finitely many *rational* points. A classic algorithmic task is to change between the *inner description* $\mathsf{C} = \mathrm{cone}(\mathsf{V})$ and the *outer description* $\mathsf{C} = \{\mathbf{x} \in \mathbb{R}^n; \mathbf{Ax} \leq \mathsf{0}\}$, and back. Standard software packages that allow one to switch between both descriptions are for example 4TI2 (4ti2 team, 2008), CDD (Fukuda, 2008), or CONVEX (Franz, 2009).

In our studies below, it will be important to decide whether a given polyhedron P is empty or not. Clearly, a certificate for $\mathsf{P} = \{\mathbf{x} \in \mathbb{R}^n; \mathbf{Ax} \leq \mathbf{b}\} \neq \emptyset$ is any $\mathbf{v} \in \mathsf{P}$. Given $\mathbf{v} \in \mathbb{R}^n$, we can easily

check whether $\mathbf{Av} \leq \mathbf{b}$ and thus, $\mathbf{v} \in \mathsf{P}$. However, is there also a simple certificate for the case that $\mathsf{P} = \emptyset$? Indeed, there is such a certificate (see Schrijver, 1986, Corollary 7.1e):

Lemma 4 (Farkas' lemma) *The system $\boldsymbol{Ax} \leq \boldsymbol{b}$ is solvable iff every $\boldsymbol{u} \in \mathbb{R}^d$ with $\boldsymbol{A}^\intercal \boldsymbol{u} = 0$, $\boldsymbol{u} \geq 0$ satisfies $\boldsymbol{u}^\intercal \boldsymbol{b} \geq 0$.*

Thus, any $\mathbf{u} \in \mathbb{R}^d$ satisfying $\mathbf{A}^\intercal \mathbf{u} = 0$, $\mathbf{u} \geq 0$ and $\mathbf{u}^\intercal \mathbf{b} < 0$ gives a certificate that P is empty. Indeed, given $\mathbf{x} \in \mathsf{P}$, write $0 = 0^\intercal \mathbf{x} = (\mathbf{A}^\intercal \mathbf{u})^\intercal \mathbf{x} = (\mathbf{u}^\intercal \mathbf{A})\mathbf{x} = \mathbf{u}^\intercal (\mathbf{Ax}) \leq \mathbf{u}^\intercal \mathbf{b} < 0$ to get a contradiction.

To decide using a computer whether $\mathsf{P} \neq \emptyset$ or not, we can make use of well-developed tools from the theory of linear programming (see below, Section 3.4).

3.2 Dimension and Faces of a Polyhedron

The *dimension* $\dim(\mathsf{P})$ of a polyhedron $\mathsf{P} \subseteq \mathbb{R}^n$ is the dimension of its *affine hull* $\mathrm{aff}(\mathsf{P})$, which is the set of all finite affine combinations $\sum_t \lambda_t \cdot \mathbf{v}_t$ (with $\lambda_t \in \mathbb{R}$ for all t and $\sum_t \lambda_t = 1$) of elements $\mathbf{v}_t \in \mathsf{P}$. By convention, the dimension of the empty set is -1. A polyhedron is *full-dimensional* if $\dim(\mathsf{P}) = n$.

Given $\mathbf{v} \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, we call the inequality $\langle \mathbf{v}, \mathbf{x} \rangle \leq \alpha$ *valid* for a polyhedron $\mathsf{P} \subseteq \mathbb{R}^n$ if it is satisfied for every $\mathbf{x} \in \mathsf{P}$. If the inequality $\langle \mathbf{v}, \mathbf{x} \rangle \leq \alpha$ is valid for the polyhedron P, we call the set $\mathsf{F} = \mathsf{P} \cap \{\mathbf{x} \in \mathbb{R}^n; \langle \mathbf{v}, \mathbf{x} \rangle = \alpha\}$ a *face* of P.

The sets $\emptyset$ and P are always faces of a polyhedron P, defined by the valid inequalities $\langle 0, \mathbf{x} \rangle \leq 1$ and $\langle 0, \mathbf{x} \rangle \leq 0$, respectively. Every polyhedron P has only finitely many faces and all of them are polyhedra, too (see Schrijver, 1986, § 8.3). They can be classified by their dimension. Faces of dimension 0 are points in $\mathbb{R}^n$, called *vertices*. Faces of dimension 1 are called *edges*: these are either line-segments, half-lines or lines. Faces of dimension $\dim(\mathsf{P}) - 1$ are called *facets*.

If a polyhedron is full-dimensional then facet-defining inequalities establish a unique (up to positive scalar factors) inclusion-minimal inequality system defining P (for details see Schrijver, 1986, § 8.4). Since every (proper) face F of P is the intersection of facets containing it, facets implicitly determine all faces.

A polyhedron $\mathsf{P} = \{\mathbf{x}; \mathbf{Ax} \leq \mathbf{b}\}$ is *pointed* if the only vector $\mathbf{y}$ with $\mathbf{Ay} = 0$ is the zero vector $\mathbf{y} = 0$ (see Schrijver, 1986, § 8.2). A non-empty pointed polyhedron has at least one vertex. A pointed polyhedral cone C has just one vertex 0, its edges are half-lines, called *extreme rays*. Non-zero representatives of the extreme rays of C then provide its inclusion-minimal inner description (Schrijver, 1986, § 8.8).

3.3 Hilbert Basis

A *lattice point* (in $\mathbb{R}^n$) is an integral vector $\mathbf{x} \in \mathbb{Z}^n$, that is, a vector whose components are integers. If C is a pointed rational polyhedral cone then each of its extreme rays contains a non-zero lattice point; this lattice point is unique if it is *normalized*, that is, if its components have no common integer divisor. Then one can show that every lattice point in C is a linear combination of these unique representatives (of extreme rays) with non-negative rational coefficients (Studený, 1993, Lemma 10). However, in general, not every lattice point in C can be obtained as such a combination of those representatives with non-negative *integral* coefficients. Therefore, the following concept is important.

By a *Hilbert basis* of a rational polyhedral cone C (see Schrijver, 1986, § 16.4) is meant a finite set $\mathsf{H} \subseteq \mathsf{C}$ of lattice points such that every lattice point in C is a non-negative integer linear combination of elements in H. A relevant result in polyhedral geometry (Schrijver, 1986, Theorem 16.4) says that every rational polyhedral cone possesses a Hilbert basis. If the cone is in addition pointed, there is a unique inclusion-minimal Hilbert basis. Clearly, any Hilbert basis of C has to include the above mentioned normalized representatives of extreme rays of C.

Again, it is a challenging algorithmic task to compute the minimal integral Hilbert basis of a given (pointed) rational polyhedral cone. However, there are a few software packages that allow to do so in some simpler cases, for example 4TI2 (4ti2 team, 2008) or NORMALIZ (Bruns, Ichim, and Söger, 2009).

3.4 LP Problems and the Simplex Method

A *linear programming* (LP) problem is the task to maximize/minimize a linear function $\mathbf{x} \mapsto \langle \mathbf{c}, \mathbf{x} \rangle$, where $\mathbf{c} \in \mathbb{R}^n$, over a polyhedron $\mathsf{P} = \{\mathbf{x} \in \mathbb{R}^n;\, \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$. A classic tool to deal with this problem is the *simplex method*. Nevertheless, it is not the aim to describe here the details of this method; they can be found in textbooks on linear programming; for example Schrijver (1986, Chapter 11).

We just recall its main features to explain the geometric interpretation. The plain version of the simplex method (see Schrijver, 1986, § 11.1) assumes one has a vertex of (a pointed polyhedron) P as a starting iteration. The basic idea is to move from vertex to vertex along the edges of P until an optimal vertex is reached or an edge is found on which the linear function is unbounded. Each particular variant of the simplex method uses a special *pivoting operation* to choose the next iteration, which should have a better value of the linear objective function than the previous one.

However, it may be the case that one is not sure whether the polyhedron P is non-empty. Then the simplex method has two phases. In Phase I, one finds at least one $\mathbf{x} \in \mathsf{P}$, called a *feasible vertex solution*, provided that it exists, or one concludes that $\mathsf{P} = \emptyset$ otherwise. If a feasible solution is available, Phase II consists of finding an optimal solution as mentioned above. A standard trick to deal with Phase I is to re-formulate it as an auxiliary LP problem, in which a feasible vertex solution is known.

For example, consider the task to optimize a linear function over a special polyhedron of the form $\mathsf{Q} = \{\mathbf{x} \in \mathbb{R}^n;\, \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$, where $\mathbf{A} \in \mathbb{R}^{d \times n}$ and $\mathbf{b} \in \mathbb{R}^d$, which task is, in fact, equivalent to the general case (see Schrijver, 1986, § 7.4). Moreover, one can assume without loss of generality $\mathbf{b} \geq 0$ here.[2] Then the auxiliary LP problem is as follows:

$$\min \{\, \langle 1, \mathbf{z} \rangle;\ (\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{n+d},\ \mathbf{A}\mathbf{x} + \mathbf{z} = \mathbf{b},\ \mathbf{x} \geq 0,\ \mathbf{z} \geq 0 \,\}. \tag{4}$$

This LP problem has the vector $(0, \mathbf{b})$ as a feasible vertex solution[3] and it has the property that its optimal value $\langle 1, \mathbf{z} \rangle$ is 0 iff $(\mathbf{x}, \mathbf{z}) = (\mathbf{x}, 0)$ and $\mathbf{x} \in \mathsf{Q}$. Thus, testing whether the polyhedron Q is non-empty can be done in this way.

4. Methods

In this section, we describe a few methods that can be used to test CI inference. The task is as follows. An input list L of *elementary* CI statements and another elementary CI statement t over

2. Indeed, if the i-th component of $\mathbf{b}$ is negative, one can replace the i-th row of $\mathbf{A}$ and the i-th component of $\mathbf{b}$ with their multiples by -1.
3. Actually, $(0, \mathbf{b})$ is a vertex of the region, defined by valid inequality $\langle 1, \mathbf{x} \rangle \geq 0$.

N are given and the goal is to decide whether $u_L \rightharpoonup u_t$ (cf., Proposition 1). We are going to give a computational comparison of these methods for $|N| = 5$ in Section 5.1 to see which of them is the fastest.

4.1 Method 1: using Skeletal Characterization

This is the first ever implemented method for testing independence implication (Studený et al., 2000, § 7.4). The motivation source for this method was the dual definition (2) of independence implication in terms of supermodular functions (see Section 2.3). The point is that, in (2), written in contrapositive form as [4]

$$\langle m, v\rangle > 0 \quad \Longrightarrow \quad \langle m, u\rangle > 0\,,$$

one can limit oneself to ℓ-standardized supermodular functions $m : \mathcal{P}(N) \to \mathbb{R}$, that is, to those that satisfy $m(S) = 0$ if $|S| \leq 1$. This is a pointed rational polyhedral cone. The class of normalized integral representatives of its extreme rays was called the *ℓ-skeleton* in Studený (2005) and denoted by $\mathcal{K}^{\diamond}_{\ell}(N)$.

The *skeletal characterization* of the implication $u \rightharpoonup v$, for $u, v \in \mathcal{S}(N)$, is as follows:

$$\forall\, m \in \mathcal{K}^{\diamond}_{\ell}(N) \qquad \langle m, v\rangle > 0 \quad \Longrightarrow \quad \langle m, u\rangle > 0 \tag{5}$$

(see Studený, 2005, Lemma 6.2). The ℓ-skeleton was computed for $|N| = 5$ (Studený et al., 2000): it consists of 117978 imsets, which break into 1319 permutational types.

Thus, the corresponding algorithm consists in checking whether each of the implications in (5) is valid or not. To disprove the implication $u \rightharpoonup v$ it is enough to find at least one $m \in \mathcal{K}^{\diamond}_{\ell}(N)$ violating (5); to confirm it one has to check that all of them are valid. By ordering the elements of the ℓ-skeleton such that those $m \in \mathcal{K}^{\diamond}_{\ell}(N)$ which are more likely to cause violation are tried earlier one can speed up the disproval (but not the confirmation). This suggested to sort elements of $\mathcal{K}^{\diamond}_{\ell}(N)$ on the basis of zeros in $\{\langle m, w\rangle\,;\; w \in \mathcal{E}(N)\}$ (for details see Bouckaert and Studený, 2007, § 3.1).

Example 1 *To illustrate the method consider a trivial example with $N = \{a, b, c\}$: take the input $L = \{\, a \perp\!\!\!\perp b \,|\, c\,,\; a \perp\!\!\!\perp c \,|\, \emptyset\,\}$ and $t : a \perp\!\!\!\perp b \,|\, \emptyset$. We already know by semi-graphoid properties that L probabilistically implies t (see Section 2.1). However, the aim of this example is to get this conclusion through Proposition 1 (see Section 2.3) using the skeletal characterization. We have*

$$u_L = u_{\langle a,b|c\rangle} + u_{\langle a,c|\emptyset\rangle} = \delta_{abc} - \delta_{bc} - \delta_a + \delta_\emptyset \quad \textit{and} \quad u_t = u_{\langle a,b|\emptyset\rangle} = \delta_{ab} - \delta_a - \delta_b + \delta_\emptyset\,.$$

In the case of 3 variables, the ℓ-skeleton has 5 elements, listed in rows of Table 1. The columns in the table correspond to structural imsets u_t and u_L, the items are corresponding scalar products. The condition (5) evidently holds for $v = u_t$ and $u = u_L$. Thus, $u_L \rightharpoonup u_t$.

The interpretation of the method from the point of view of polyhedral geometry is as follows. The independence implication can equivalently be defined in terms of (inclusion of) facets of the cone $\mathcal{R}(N) \equiv \mathrm{cone}\,(\mathcal{E}(N))$, the cone spanned by elementary imsets. Let F_u denote the face of $\mathcal{R}(N)$ generated by $u \in \mathcal{S}(N) \subseteq \mathcal{R}(N)$, that is, the least face of $\mathcal{R}(N)$ containing u ($\equiv$ the intersection of all faces of $\mathcal{R}(N)$ containing u). Then, for $u, v \in \mathcal{S}(N)$, one has $u \rightharpoonup v$ iff the face F_u contains v, which means, $F_v \subseteq F_u$ (see Studený, 2005, Remark 6.2).

4. Note that $\langle m, u\rangle \geq 0$ for any m supermodular and $u \in \mathcal{S}(N)$ (see Studený, 2005, Proposition 5.1).

	$u_t = \delta_{ab} - \delta_a - \delta_b + \delta_\emptyset$	$u_L = \delta_{abc} - \delta_{bc} - \delta_a + \delta_\emptyset$
$m_1 = \delta_{abc}$	0	1
$m_2 = \delta_{abc} + \delta_{ab}$	1	1
$m_3 = \delta_{abc} + \delta_{ac}$	0	1
$m_4 = \delta_{abc} + \delta_{bc}$	0	0
$m_5 = 2 \cdot \delta_{abc} + \delta_{ab} + \delta_{ac} + \delta_{bc}$	1	1

Table 1: Scalar products with elements of $\mathcal{K}^\diamond_\ell(N)$ for $N = \{a,b,c\}$.

The method is based on the computation of all facets of the cone $\mathcal{R}(N)$. These facets correspond to the extreme rays of the (dual) cone of ℓ-standardized supermodular functions. Thus, basically, one is checking whether every facet containing F_u also contains F_v. The problem with this approach is that it can hardly be extended beyond five variables because computing these facets seems to be computationally infeasible for $|N| \geq 6$.

4.2 Method 2: Racing Algorithms

The idea of the paper (Bouckaert and Studený, 2007) was to combine two algorithms for testing the independence implication $u \rightharpoonup v$. One of them, called the *verification algorithm*, was based on (1) and appeared to be suitable to confirm the implication provided it holds. However, it may spend a long time before it gives a response if the implication does not hold. The other algorithm, called the *falsification algorithm* and based on (2), was designed to disprove the implication if it does not hold. However, it is not able to confirm $u \rightharpoonup v$ provided it holds.

The combined procedure starts with two threads, the verification one and the falsification one. Once one thread finds its proof, it stops the other and returns its outcome. This approach makes it possible to go beyond 5 variables, but may not give a decisive response (in reasonable time) to some complex implication problems. On the other hand, empirical evidence from Bouckaert and Studený (2007) suggests that this method is, on average, faster than the method described in Section 4.1.

4.2.1 VERIFICATION: DECOMPOSING INTO ELEMENTARY IMSETS

Consider a combinatorial imset $u \in \mathcal{C}(N)$, an elementary imset $v \in \mathcal{E}(N)$ and the task to decide whether $u \rightharpoonup v$. That is, by (1), testing whether $k \cdot u - v$ is a structural imset for some $k \in \mathbb{N}$. Observe that (1) is equivalent to

$$\exists l \in \mathbb{N} \qquad l \cdot u - v \in \mathcal{C}(N)\,. \tag{6}$$

Indeed, $\mathcal{C}(N) \subseteq \mathcal{S}(N)$ gives (6)$\Rightarrow$(1). Now, (1) implies that $n \cdot (k \cdot u - v)$ is a combinatorial imset for some $k, n \in \mathbb{N}$ (see Section 2.2). As $(n-1) \cdot v \in \mathcal{C}(N)$, it gives $(n \cdot k) \cdot u - v \in \mathcal{C}(N)$. Moreover, it follows from concluding remarks in Sections 2.2 and 2.3 that $n_* \cdot k_*$ is an upper limit for l in (6). In general, we do not know what is the least such upper limit for l. Even in case $|N| = 5$, we only know it is a number between 7 (see Studený et al., 2000) and $14 = 2 \cdot 7 = n_* \cdot k_*$.

The characterization (6) allows one to transform testing independence implication to the task to decide whether a given candidate imset $y = l \cdot u - v$ is combinatorial. A combinatorial imset y may have many decompositions $y = \sum_{w \in \mathcal{E}(N)} k_w \cdot w$, $k_w \in \mathbb{Z}_+$ into elementary imsets. However, the number $\sum_{w \in \mathcal{E}(N)} k_w$ of summands, called the *degree* of y, is the same for any such decomposition (see Studený, 2005, § 4.2.2). Because there is a simple formula for the degree of the candidate imset

y, the search space, the tree of potential decompositions, is known. This space could be big, but it can be limited by introducing additional sanity checks, which allow one to cut off some blind alleys in the tree. Moreover, the search can be guided by suitable heuristics and this can speed up the resulting algorithm (for details see Bouckaert and Studený, 2007, § 3.2).

The decomposition itself is quite fast, but what can slow down the whole procedure is the factor l from (6), depending on a particular pair u, v. The point is that the degree of a candidate imset $y = l \cdot u - v$ grows (linearly) with l; consequently, the size of the corresponding tree of possible decompositions grows exponentially with l. Typically, if $u \rightharpoonup v$ holds then one often finds a decomposition of y with $l = 1$. However, for $|N| = 5$, there are a few cases when the decomposition of y exists for $1 < l \leq 7$ and not for $l - 1$ (for examples see Studený et al., 2000, § 4.3). In these cases and also when $u \rightharpoonup v$ does not hold one has to search through a huge space, which makes the method infeasible for efficient disproving implications.

Example 2 *Consider $N = \{a, b, c, d\}$, the input list*

$$L = \{\, a \perp\!\!\!\perp b \,|\, c\,,\ a \perp\!\!\!\perp c \,|\, d\,,\ a \perp\!\!\!\perp d \,|\, b\,,\ b \perp\!\!\!\perp c \,|\, ad\,\}\,, \tag{7}$$

and another CI statement $t : a \perp\!\!\!\perp c \,|\, b$. We are going to show $u_L \rightharpoonup u_t$ by the decomposition method. Actually, we show that (6) holds with $l = 1$. More specifically, we have

$$\begin{aligned} u_L &= u_{\langle a,b|c\rangle} + u_{\langle a,c|d\rangle} + u_{\langle a,d|b\rangle} + u_{\langle b,c|ad\rangle} \\ &= \delta_{abcd} + \delta_{abc} - \delta_{ab} - \delta_{ac} - \delta_{bc} - \delta_{bd} - \delta_{cd} + \delta_b + \delta_c + \delta_d\,, \end{aligned}$$

and, as $u_t = \delta_{abc} - \delta_{ab} - \delta_{bc} + \delta_b$, we know that

$$y \equiv 1 \cdot u_L - u_t = \delta_{abcd} - \delta_{ac} - \delta_{bd} - \delta_{cd} + \delta_c + \delta_d\,.$$

The task is to test whether y is a combinatorial imset. If this is the case then the degree of y must be 3, which means we search for a decomposition into elementary imsets with 3 summands. Clearly, at least one summand v has to satisfy $v(abcd) > 0$. There are 6 elementary imsets over $\{a, b, c, d\}$ with this property and two of them are excluded by sanity checks. For example, for $v' = u_{\langle c,d|ab\rangle}$ and $y' = y - v'$ one has $-1 = \sum_{\{c,d\} \subseteq T} y'(T) < 0$, which is impossible for a combinatorial imset in place of y'. However, if we subtract $\tilde{v} = u_{\langle a,b|cd\rangle}$ then

$$\tilde{y} = y - \tilde{v} = \delta_{acd} + \delta_{bcd} - \delta_{ac} - \delta_{bd} - 2 \cdot \delta_{cd} + \delta_c + \delta_d$$

is a good direction. Again, at least one of two summands v in the searched decomposition of $\tilde{y}$ must satisfy $v(acd) > 0$ and the choice $v = u_{\langle a,d|c\rangle}$ leads to the final decomposition

$$y = u_{\langle a,b|cd\rangle} + u_{\langle a,d|c\rangle} + u_{\langle b,c|d\rangle}\,.$$

Thus, the implication $u_L \rightharpoonup u_t$ has been confirmed by the decomposition method.

To explain the geometric interpretation (of the method) note that the cone $\mathcal{R}(N) = \mathrm{cone}\,(\mathcal{E}(N))$ is slightly special. The lattice points in this cone are just structural imsets.[5] Moreover, there exists

5. As mentioned in Section 3.3, lattice points in $\mathcal{R}(N)$ are just non-negative rational combinations of elementary imsets $\mathcal{E}(N)$, that is, structural imsets (see Section 2.2).

a hyperplane in $\mathbb{R}^{\mathcal{P}(N)}$ which intersects all extreme rays of $\mathcal{R}(N)$ just in its normalized integral representatives $\mathcal{E}(N)$ and these are the only lattice points in the intersection of this hyperplane with $\mathcal{R}(N)$. In particular, every structural imset belongs to one of parallel hyperplanes (to this basic one) and its "degree" says how far it is from the origin (= zero imset). Now, the condition (1) means that at least one multiple of u (by $k \in \mathbb{N}$) has the property that it remains a lattice point within the cone $\mathcal{R}(N)$ even if v is subtracted. The condition (6) is a minor modification of (1): it requires a multiple of u minus v is a sum of elementary imsets (with possible repetition). The algorithm, therefore, looks for the decomposition and the degree (of the candidate imset y) serves as the measure of its complexity.

4.2.2 FALSIFICATION: RANDOMLY GENERATING SUPERMODULAR FUNCTIONS

Falsification is based on the characterization (2). To disprove the implication $u \rightharpoonup v$ it is enough to find a supermodular function $m : \mathcal{P}(N) \to \mathbb{R}$ such that $\langle m, u \rangle = 0$ and $\langle m, v \rangle > 0$. Actually, one can limit oneself to ℓ-standardized integer-valued supermodular functions, that is, *supermodular imsets*. These imsets have a special form which allows one to generate them randomly. The idea is

- to randomly generate a collection of subsets of N,
- to assign them randomly selected positive integer values (and the zero values to remaining sets), and
- to modify the resulting function to make it a supermodular imset.

The details of this procedure can be found in Bouckaert and Studený (2007, § 3.3). The procedure allows one to disprove (= falsify) the implication $u \rightharpoonup v$ even for $|N| \geq 6$, but it is clearly not able to confirm it provided it holds. Therefore, it has to be combined with a verification procedure.

Example 3 *Consider $N = \{a, b, c, d\}$ and the same input list (7) as in Example 2, but a different CI statement $t : b \perp\!\!\!\perp c \,|\, a$ to be derived. If our random procedure generates the supermodular imset*

$$m = 2 \cdot \delta_{abcd} + \delta_{abc} + \delta_{abd} + \delta_{acd} + 2 \cdot \delta_{bcd} + \delta_{bc} + \delta_{bd} + \delta_{cd} \,,$$

then one can observe that $\langle m, u_L \rangle = 0$ while $\langle m, u_{\langle b,c|a \rangle} \rangle = 1$. In particular, by (2), the respective implication is not valid: $\neg(u_L \rightharpoonup u_{\langle b,c|a \rangle})$.

The geometric interpretation of the algorithm is similar to the interpretation of the method from Section 4.1. Supermodular functions correspond to faces of the cone $\mathcal{R}(N)$. Thus, the procedure consists in random generating faces of $\mathcal{R}(N)$ and the aim to find a face of $\mathcal{R}(N)$ which contains u but not v.

4.3 Method 3: Decomposition via Hilbert Basis

An alternative to testing whether an imset is combinatorial is testing whether it is structural. Since structural imsets coincide with the lattice points in the cone $\mathcal{R}(N) \equiv \mathrm{cone}\,(\mathcal{E}(N))$, each of them can be written as a sum (with possible repetition) of the elements of the Hilbert basis $\mathcal{H}(N)$ of the cone $\mathcal{R}(N)$ (see Section 3.3). For $|N| \leq 4$ one has $\mathcal{H}(N) = \mathcal{E}(N)$ (Studený, 1991); however, the results from Hemmecke et al. (2008) imply that the set of elementary imsets does not constitute a Hilbert basis of $\mathcal{R}(N)$ for $|N| \geq 5$.

Computation of the Hilbert basis is a very hard task. Recently, the Hilbert basis of $\mathcal{R}(N)$ for $|N|=5$ has been obtained as a result of sophisticated computations (Bruns et al., 2010). In Appendix B we provide a list of its representatives. Altogether, the obtained Hilbert basis of $\mathcal{R}(N)$ has 1300 elements, falling into 35 (permutation equivalence) types.

Thus, having the Hilbert basis of $\mathcal{R}(N)$ at hand, one can test the independence implication $u \rightharpoonup v$ for $u,v \in \mathcal{S}(N)$ through (1): the task is to find out whether there exists a decomposition of $y \equiv k \cdot u - v$ into Hilbert basis elements for some $k \in \mathbb{N}$. This is analogous to the decomposition approach from Section 4.2.1, where the set of elementary imsets $\mathcal{E}(N)$ was used instead of $\mathcal{H}(N)$.

Thus, the interpretation of this new method is the same, the difference is that we have now a wider class of leaves of the (potential) decomposition tree. On the other hand, the advantage of the Hilbert basis decomposition for $|N|=5$ should be that, in some complex cases, a much simpler decomposition of y may exist that involves also the elements of $\mathcal{H}(N) \setminus \mathcal{E}(N)$ (simpler = with a less number of summands). We are also sure that the upper limit for the constant k is only $k_* = 7$. In particular, the depth of the tree of potential decompositions should be smaller, while the tree itself is expected to be wider.

4.4 Method 4: Linear Programming

The basic idea is to re-formulate (the definition of) independence implication in terms of the (pointed rational polyhedral) cone $\mathcal{R}(N) \equiv \mathrm{cone}(\mathcal{E}(N))$ spanned by elementary imsets. More specifically, given $u,v \in \mathcal{S}(N)$, the condition (1) can be expressed in this way:

$$u \rightharpoonup v \quad \text{iff} \quad \exists k \in [0,\infty) \quad k \cdot u - v \in \mathcal{R}(N)\,. \tag{8}$$

Indeed, since $\mathcal{S}(N) \subseteq \mathcal{R}(N)$ the implication (1)⇒(8) is evident. Conversely, provided (8) holds with k, it holds with any $k' \geq k$ because $\mathcal{R}(N)$ is a cone and $(k'-k) \cdot u \in \mathcal{R}(N)$. Therefore, there exists $k' \in \mathbb{N}$ with $k' \cdot u - v \in \mathcal{R}(N)$. As $k' \cdot u - v$ is an imset, it belongs to $\mathcal{S}(N)$, and (1) holds.

The geometric interpretation of the condition (8) is clear. It means that the ray (= half-line) with the origin in $-v$ and the direction given by u intersects the cone $\mathcal{R}(N)$. The point is that testing whether this happens can be viewed as an LP problem:

Lemma 5 Given $u,v \in \mathcal{S}(N)$ one has $u \rightharpoonup v$ iff the system of equalities

$$\sum_{w \in \mathcal{E}(N)} \lambda_w \cdot w(S) - k \cdot u(S) = -v(S) \qquad \text{for any } S \subseteq N\,, \tag{9}$$

has a non-negative solution in λ_w, $w \in \mathcal{E}(N)$ and k.

Proof The cone $\mathcal{R}(N)$ consists of conic combinations of representatives of its extreme rays, that is, of elementary imsets. Thus, (8) can be re-written as the requirement for the existence of $k \geq 0$ and $\lambda_w \geq 0$ with $k \cdot u - v = \sum_{w \in \mathcal{E}(N)} \lambda_w \cdot w$. This imset equality, specified for any $S \subseteq N$, yields (9). ■

Non-negative solutions to (9) form a polyhedron of the type $\mathsf{Q} = \{\mathbf{x}; \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$, $\mathbf{A} \in \mathbb{R}^{d \times n}$, $\mathbf{b} \in \mathbb{R}^d$. Indeed, the rows of $\mathbf{A}$ correspond to subsets of N, while the columns to elementary imsets and the factor k. Thus, $d = |\mathcal{P}(N)| = 2^{|N|}$ and $n = |\mathcal{E}(N)| + 1 = \binom{|N|}{2} \cdot 2^{|N|-2} + 1$. As explained in Section 3.4, testing whether Q is non-empty is equivalent to solving the auxiliary LP problem (4) in

$(n+d)$-dimensional space. An illustrative example of the application of this LP method in the case $|N| = 3$ is given in Appendix C.

The advantage of this approach is that it is not limited to a particular number of variables as the previous ones. To get an impression of the implication problem complexity at hand, let us have a look at Table 2. Clearly, these (comparably small) linear programs should be solvable quickly in practice. We give computational evidence to this claim in Section 5.2. Another comment is that this LP method can be interpreted as a kind of decomposition procedure (analogous to the one from Section 4.2.1); see Appendix D for an explanation.

$\|N\|$	3	4	5	6	7	8	9	10
$n+d$	15	41	113	305	801	2049	5121	12545

Table 2: The dimensions of LP problems to be solved in order to decide $u \rightharpoonup v$.

Let us finally note that there are alternative ways to re-formulate testing of $u \rightharpoonup v$ as an LP problem. For example, consider the cone of ℓ-standardized supermodular functions. We know the outer description of this (pointed) cone:

$$\mathcal{K}_\ell(N) = \{m \in \mathbb{R}^{\mathcal{P}(N)};\ m(S) = 0 \text{ for } |S| \leq 1,\ \langle m, w\rangle \geq 0 \text{ for } w \in \mathcal{E}(N)\}.$$

Since, in (2), one can limit oneself to ℓ-standardized supermodular functions, $u \rightharpoonup v$ is equivalent to the requirement

$$\sup\{\langle m, v\rangle;\ m \in \mathcal{K}_\ell(N),\ \langle m, u\rangle = 0\} = 0. \tag{10}$$

This is an LP problem with a feasible region.[6] Thus, Phase II of the simplex method can be applied to solve it. Note that (10), mentioned in Studený (2004, § 5), can be viewed, after appropriate modifications, as the dual LP problem to (9) in the sense of the LP theory; however, we omit the details in this paper.

5. Experiments

In this section, we describe the results of our computational experiments. We performed two separate bunches of experiments. One of them was done in New Zealand and the aim was to compare various methods in case $|N| = 5$ (see Section 5.1).

As the result was that the LP method performs best in case $|N| = 5$, the aim of the other group of experiments (see Section 5.2), done in Germany, was to test the LP approach in case $|N| > 5$. These latter experiments were based on the commercial optimization software CPLEX (IBM Ilog team, 2009).

5.1 Comparison for Five Variables

Empirical evaluation of the methods can give insight in the practical behavior of the various methods. First, we considered the case of five variables, so that we can compare the new methods with techniques based on skeletal representations (see Section 4.1). The experimental set up from Bouckaert and Studený (2007) was used for the five-variable tests. In short, a thousand random input lists

6. Indeed, $m \equiv 0$ is a feasible solution to (10).

containing 3, up to 11 elementary CI statements were generated and, for each elementary CI statement outside the input list, it was verified whether it was implicated or not. So, the thousand 3-input cases result in verification of a thousand times the total number of elementary CI statements (80 for 5 variables) minus the 3 statements already given in the input list, that is, $1000 \times (80-3) = 77000$ inference problems. Table 3 lists the numbers of inference problems in its last column.

The algorithms considered were:

- Skeletal algorithm and sorted skeletal algorithm (see Section 4.1).
- Racing algorithms (see Section 4.2). There are situations where both the verification and the falsification algorithm perform poor, resulting in unacceptable running times. These outliers distort the typical behavior of the algorithm, so we put a cap on total calculation time and report the number of problems where a solution is not found within this deadline as the number of unresolved cases in Table 3.
- *Hilbert basis* (HB) decomposition algorithm (see Section 4.3). Initial experimentation showed that the HB approach is very good at verifying the implication. However, when a CI statement is not implied, it takes a long time to traverse the search space. So, the average time for the HB approach is expected to be very poor. Again, we capped the allowed time for the algorithm to run and when no solution was found within that time the problem was as marked unresolved.
- The observation that the HB algorithm performs well for implication but poor on falsification immediately gives rise to another algorithm where the HB decomposition algorithm and falsification algorithm are raced against each other. This algorithm is called the *Hilbert racer*. Like the racer algorithm, this algorithm was time constrained.
- Now we have two algorithms that appear to perform well for verification, it is natural to combine them with the falsifier and thus get a three horse race. This algorithm is called the *triple racer*, and it is time constrained as well.
- The last algorithm under consideration is the LP method described in Section 4.4.

Input size	Hilbert capped		Hilbert racer		racer		triple racer		Total
3	5127	(6.7%)	0	(0%)	0	(0%)	0	(0%)	77000
4	10098	(13.3%)	0	(0%)	0	(0%)	0	(0%)	76000
5	12793	(17.1%)	0	(0%)	6	(0.01%)	9	(0.01%)	75000
6	15139	(20.5%)	11	(0.01%)	28	(0.04%)	30	(0.04%)	74000
7	16475	(22.6%)	38	(0.05%)	110	(0.15%)	88	(0.12%)	73000
8	18042	(25.1%)	85	(0.12%)	238	(0.33%)	215	(0.30%)	72000
9	18308	(25.8%)	181	(0.25%)	377	(0.53%)	306	(0.43%)	71000
10	17738	(25.3%)	211	(0.30%)	489	(0.70%)	386	(0.55%)	70000
11	16798	(24.3%)	230	(0.33%)	495	(0.72%)	349	(0.51%)	69000

Table 3: Numbers of inference problems that were not resolved due to time-outs.

All algorithms were implemented in Java, run under Sun Java 1.6.0_12 in server mode on a Intel dual Core 3.00GHz CPU with 2GB memory, though memory is not an issue for any of the algorithms.

5.1.1 RESULTS AND DISCUSSION

Table 3 shows the number of unresolved cases when time was capped at 1 second per problem. Only the algorithms that did not resolve all problems are shown there. The numbers in brackets are percentages, the last column the total number of problems for a particular input size.

Given the total number of problems (around 70 thousand per input size) this restricted calculation took around 20 hours per input size. Clearly, the HB decomposition method when capped leaves a large proportion of implication problems unresolved, up to a quarter for the larger input sizes.

Combining the HB algorithm with the falsification algorithm reduces the number of unresolved problems to less than a third of a percent. Comparing the Hilbert racer with the original racer algorithm shows that more problems are resolved in the time available, so the HB decomposition algorithm appears to perform better at resolving problems. Finally, the triple racer, which combines both algorithms, performs in between the two algorithms in terms of number of unresolved problems. At first sight, one would expect the triple racer to perform at least as good as the Hilbert racer, however, since the test was run on a dual core processor instead of a machine with at least three cores, the various algorithms were too busy battling for processor access and lost time to make the deadline.

	User time						
Input size	skeletal	skeletal sorted	Hilbert	racer	Hilbert racer	triple racer	LP
3	286	351	5955	132	14	163	29
4	680	648	10320	207	23	318	31
5	1476	1122	12868	414	107	667	32
6	2748	1713	15225	707	1418	1050	32
7	5385	2652	16584	1690	5601	1915	32
8	8583	3625	18171	2884	12145	3159	32
9	12086	4771	18476	6099	24274	4602	32
10	16180	6439	18089	19238	28227	11919	31
11	19530	8225	17507	20962	32452	17145	31
	Elapsed time						
Input size	skeletal	skeletal sorted	Hilbert	racer	Hilbert racer	triple racer	LP
3	288	353	12745	90	28	152	31
4	681	650	16471	162	38	277	32
5	1477	1124	18636	313	105	562	33
6	2749	1714	20606	516	876	855	33
7	5385	2652	21665	1016	3120	1548	34
8	8582	3625	22977	1889	6864	2522	34
9	12085	4771	23156	4610	13509	3693	33
10	16177	6439	22749	17080	16350	10509	33
11	19528	8224	22118	18255	18221	15250	32

Table 4: Total computer times for the experiments in seconds.

Table 4 shows total computer time to resolve all problems for a given input size. This is a rough indication of the average computation time per implication problem, given that the number of problems per input size changes slightly (less than 10%) from the smallest 3-input problems to the largest 11-input problems. The user time is time the processor spent on the problem, while elapsed

time is actual clock time. Since the racing algorithms are multi-threaded, having multiple jobs run in parallel, elapsed time can be less than user time. For the single threaded algorithms, user time is approximately equal to elapsed time.

Comparing the unsorted and sorted skeleton approaches, it shows that ordering the skeletons can result in considerable time savings.

The HB decomposition algorithm is outperformed, clearly due to its bad performance in falsifications. There is a relatively large gap between user and elapsed time for the HB algorithm, which is due to the way the time-out is implemented. This causes the central processing unit to be idle for some of the time in some cases. Note the high correlation between user time and number of unresolved problems in Table 3. This indicates that the HB decomposition algorithm finds a resolution for a large number of problems very quickly, but performs very poor on a large number of others.

The racer algorithm performs very well on the smaller input sizes, but looses out on the larger ones. The Hilbert racer performs even better on the smaller problems, but again looses out on the larger input sizes. The triple racer is the best performer on the larger problems. However, taking in account the number of unresolved problems, the Hilbert racer is winning out, so it is to be expected that a slightly longer time-out period will bring the performance of the triple racer on the same level as the Hilbert racer.

The most remarkable result is the performance of the LP algorithm. Unlike all the other algorithms, it does not suffer from performance degradation with increasing input sizes. Furthermore, the calculation times are just a fraction of the times for the other algorithms. Considering that this is a straightforward Java implementation of the algorithm where only limited effort went in optimizing the code, its performance compared to the other algorithms is overwhelmingly better.

5.2 Experiments for Higher Number of Variables

To perform the experiments for $|N| > 5$, the only chance is to use the LP method presented in Section 4.4. The experiments were analogously defined as in Section 5.1. If L is an input list of elementary CI statements, then one single experiment comprises the testing of $|\mathcal{E}(N)|$ many independence implications $u_L \rightharpoonup u$ for all $u \in \mathcal{E}(N)$.[7] For $|N| = 4,5,6$ we have considered input lists L containing 3 up to 11 different elementary CI statements over N and performed 1000 such experiments for each combination of $|N|$ and $|L|$. Thus, we made 9000 experiments in total for $|N| = 4,5,6$. For $|N| = 7,8,9,10$ we only considered 1000 experiments with L containing 3 elementary CI statements.

The number of LP problems to be tested within a single experiment and the dimension of these problems depends on the number $|\mathcal{E}(N)| = \binom{|N|}{2} \cdot 2^{|N|-2}$ of elementary imsets and on the number $2^{|N|}$ of subsets of N (compare with Table 2 in Section 4.4). The running times are averaged over all 9000 experiments for $|N| = 4,5,6$ and over all 1000 experiments for $|N| = 7,8,9,10$, respectively. The running times include the set up of the LP method and the actual LP computation. The tests are done using the commercial optimization software CPLEX 9.100 (IBM Ilog team, 2009) on a Sun Fire V890 Ultra Sparc IV processor.

Table 5 illustrates the growth of the running times of the computations for $|N| = 4$ up to $|N| = 10$. Therein, we relate the running times with the growth of $|N|$ and the amount $|\mathcal{E}(N)|$ of LP problems to be tested for each experiment.

7. Contrary to Section 5.1, we involved "unnecessary" testing $u_L \rightharpoonup u_t$ for $t \in L$ in the experiments.

$\|N\|$	4	5	6	7	8	9	10
LPs/experiment	24	80	240	672	1792	4608	11520
time/experiment	4 ms	18 ms	115 ms	658 ms	5037 ms	150.38s	3862.10s

Table 5: The growth of computation times for $|N| = 4$ to $|N| = 10$.

5.2.1 RESULTS AND DISCUSSION

Table 5 illustrates a natural increase in the time needed for one experiment as $|N|$ increases. This is clear as both more LP problems have to be solved and they are of higher dimensions. For small values of $|N|$ the increase in time occurs mainly because of the LP set up, not because of the computation itself. For higher $|N|$ this proportion changes. But in spite of that, even for $|N| = 7$ all 1000 experiments together could be done within minutes. For $|N| = 10$ it took more than one month to perform all 1000 experiments; however, it took only about an hour for each single experiment. More specifically, on average, only about one third of a second was enough to test $u_L \rightharpoonup u_t$ for $|N| = 10$.

Just looking at the increase of the dimensions of the LP problems, it is reasonable to assume that one can test independence implication for up to $|N| = 15$ by directly plugging the corresponding LP problem into CPLEX. However, testing all $|\mathcal{E}(N)|$ inference problems for one experiment would be too expensive. Recall that one experiment already required about one hour of computation time for $|N| = 10$. To go even beyond $|N| = 15$ for testing $u_L \rightharpoonup u_t$, one can still exploit the known structure of the matrix $\mathbf{A}$ of the LP problem and employ column generation techniques in order to solve these much bigger LP problems to optimality. Of course, this approach has to be coded and tested for bigger $|N|$ to determine its efficiency.

6. Conclusions

Our computational experiments confirmed that the LP approach both overcomes the barriers of previous methods set by the number of variables in consideration and, moreover, it is also much faster. The new method also has the perspective to go even beyond the present limits, provided special LP techniques are used.

For small CI inference problems, involving a limited number of variables, one can enter data manually through web interfaces (based on LP method) available at

`http://www.cs.waikato.ac.nz/~remco/ci/.`

For bigger inference problems, in which the input is expected in the form of a file, one can use efficient commercial LP software instead.

We showed that the potential application of computer testing of CI inference is in the area of Bayesian networks (see Section 2.4). More specifically, by Proposition 2 (and Corollary 3), testing independence inclusion (and reading CI restrictions from an acyclic directed graph) can be transformed to the task whether a difference of two standard imsets is a combinatorial imset. Since, by (3), every standard imset has at most $2 \cdot |N|$ non-zero values, and, also, its degree is at most $|N| - 1$, we have good reasons to believe that the decomposition algorithm from Section 4.2.1 can be implemented with polynomial complexity in this special case. Actually, a relevant result (Studený and Vomlel, 2011, Lemma 3) implies that if the difference of two standard imsets is a combinatorial imset then it is a plain sum of elementary imsets (= a combination with coefficients $+1$). Thus, we

dare to conjecture that testing whether a difference of two standard imsets is a combinatorial imset can be done with polynomial complexity in $|N|$ using the procedure from Section 4.2.1.

After finishing this paper, we learned about a conference paper by Niepert (2009), which also comes with the idea of application of the LP approach to probabilistic CI inference. The reader may be interested in what is the relation of both approaches. The LP problem in Niepert (2009, Proposition 4.9) is, in fact, equivalent to our task (9) with fixed factor $k = 1$, if one uses a suitable one-to-one linear transformation to involved imsets u, v and w. This transformation has a nice property that the transformed LP problem $\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}$, $\mathbf{x} \geq 0$ has 0-1 matrix $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{b}} \geq 0$. However, since the transformation is linear and one-to-one, it is essentially the same LP problem. Our approach is more general because it allows one to consider the multiplicative factor $k \geq 0$ in (9), which results in a wider class of derivable CI implications (cf., Section 4.2.1). We hope that the algorithms presented in our paper will find their application, perhaps in the research on stable CI statements (Niepert, van Gucht, and Gyssens, 2010).

The results presented in the paper lead to further open theoretical problems. The most difficult one is probably to find out what is the exact value of the constant n_* from Section 2.2 for $|N| > 5$, as it seems to be related to finding the Hilbert basis $\mathcal{H}(N)$, which is a hard task. Another group of open questions concerns the verification of the implication $u \rightharpoonup v$ for $u \in \mathcal{C}(N)$ and $v \in \mathcal{E}(N)$. We would like to know the minimal bounds for the factors in (1), (6) and (8). Perhaps computing these bounds can be formulated as a mixed (integer) LP problem. Finally, there are general questions of complexity, like whether it is NP hard to decide CI implications with respect to the input size.

Acknowledgments

The research of Milan Studený has been supported by the grants GAČR n. 201/08/0539 and MŠMT n. 1M0572. We thank the reviewers for their comments, which helped to improve the readability of the paper.

Appendix A. Proof of Proposition 1

Recall that L is an input list of CI statements over N and t another CI statement over N. The basic tool for our arguments is the *multiinformation function* $m_P : \mathcal{P}(N) \to \mathbb{R}$ ascribed to any discrete probability distribution P over N (for technical details see Studený, 2005, § 2.3.4). Corollary 2.2 in Studený (2005) says that m_P is always a supermodular function and, for every CI statement $s \equiv A \perp\!\!\!\perp B\,|\,C$ over N, one has

$$\langle m_P, u_s \rangle = 0 \quad \text{iff} \quad A \perp\!\!\!\perp B\,|\,C\,[P]\,. \tag{11}$$

Now, assuming $u_L \rightharpoonup u_t$, the equivalent characterization (2) of the independence implication implies that, for every discrete probability distribution P over N,

$$\text{whenever } \langle m_P, u_L \rangle = 0 \quad \text{then} \quad \langle m_P, u_t \rangle = 0\,. \tag{12}$$

Assuming P is a distribution such that every s in L is valid with respect to P, one has $\langle m_P, u_s \rangle = 0$ by (11), and, hence, by the linearity of the scalar product, $\langle m_P, u_L \rangle = \sum_{s \in L} \langle m_P, u_s \rangle = 0$. This implies, by (12), $\langle m_P, u_t \rangle = 0$, which means, by (11), that t is a valid CI statement with respect to P.

Appendix B. Hilbert Basis for Five Variables

As said in Section 4.3, the Hilbert basis $\mathcal{H}(N)$ for $|N| = 5$ falls into 35 types. Nevertheless, there is a further symmetry within it. Consider a bijective *reflection* map $\iota : \mathcal{P}(N) \to \mathcal{P}(N)$, given by $\iota(A) = N \setminus A$ for $A \subseteq N$. Observe that elementary imsets are closed under (the composition with) reflection. Thus, the cone $\mathcal{R}(N)$ and its Hilbert basis $\mathcal{H}(N)$ are closed under the reflection, too. In particular, some of 35 permutation equivalence classes are reflections of others. If the reflection is taken into consideration, the number of resulting equivalence classes in $\mathcal{H}(N)$ is lower, namely 21. Their representatives are given in Table 6.

$\emptyset$	0	0	2	2	1	1	0	0	1	0	0	1	0	2	0	0	1	2	3	3	2
$\{a\}$	0	0	0	0	0	0	1	1	0	1	0	1	1	-1	1	1	0	0	-1	1	1
$\{b\}$	0	0	0	0	0	1	1	1	0	1	1	0	0	0	1	1	0	0	-1	-1	0
$\{c\}$	0	0	0	0	1	1	0	0	1	1	1	1	1	0	1	1	1	0	1	-1	0
$\{d\}$	0	0	0	-2	0	0	1	1	0	1	1	0	1	-1	1	1	2	0	-1	-1	0
$\{e\}$	0	0	-2	0	0	-1	1	1	0	1	1	-1	1	-1	1	1	-1	-2	-2	-2	-2
$\{a,b\}$	0	0	-1	-1	0	-1	-1	-1	-1	-2	0	-1	0	0	-2	-1	-1	-1	0	-1	-1
$\{a,c\}$	0	0	-1	-1	-1	1	0	0	-1	-1	0	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
$\{a,d\}$	0	0	1	1	-1	-1	-2	-2	0	-1	0	-1	1	1	-1	-1	-1	1	0	-1	-1
$\{a,e\}$	0	0	1	0	2	1	1	0	2	1	0	0	-2	1	0	-1	1	2	1	2	2
$\{b,c\}$	0	0	0	0	-1	-2	0	0	-1	-1	-1	-1	0	-1	-1	-1	-1	1	-1	0	-1
$\{b,d\}$	0	0	-1	1	2	0	0	0	2	1	-1	1	0	1	0	-1	-1	-1	0	0	-1
$\{b,e\}$	0	0	1	-1	-1	0	-2	-2	0	-1	-1	1	0	1	-1	-1	1	1	1	1	1
$\{c,d\}$	0	0	-1	1	-1	-1	0	0	-1	-1	-1	-1	-2	1	-1	-1	-2	-1	-1	0	-1
$\{c,e\}$	0	0	1	-1	-1	0	0	0	-1	-1	-1	0	1	1	-1	-1	0	1	1	1	1
$\{d,e\}$	0	1	1	1	0	1	-1	-1	-1	-2	-1	1	-1	0	-2	-1	0	1	1	1	1
$\{a,b,c\}$	0	0	1	1	0	0	-1	-1	1	1	-1	1	-1	1	1	0	2	0	1	1	1
$\{a,b,d\}$	0	0	0	-1	-1	1	1	1	-1	0	-1	0	-2	-2	1	0	1	0	1	1	1
$\{a,b,e\}$	0	0	0	1	-1	-1	0	1	-1	0	1	-1	1	-1	1	2	0	-1	-1	-1	-1
$\{a,c,d\}$	0	0	0	0	2	0	1	1	1	1	1	1	0	-1	1	2	1	0	1	1	1
$\{a,c,e\}$	0	0	-1	1	-1	-2	-2	-1	-1	-1	-1	1	0	-2	0	0	-1	-1	0	-1	-1
$\{a,d,e\}$	0	0	-2	-1	-1	0	0	1	-1	0	-1	-1	0	0	1	0	-1	-2	-1	-1	-1
$\{b,c,d\}$	0	0	1	-1	-1	1	-1	-1	-1	-1	1	0	1	-1	0	1	1	0	1	1	2
$\{b,c,e\}$	0	0	-1	1	2	1	1	1	1	1	1	-1	-2	-1	1	1	-1	-2	0	-1	-1
$\{b,d,e\}$	0	-1	-1	0	-1	-1	1	1	-1	0	1	-1	-1	-1	1	1	-1	-1	-1	-1	-1
$\{c,d,e\}$	1	-1	0	-1	0	-1	-1	-1	1	1	1	-1	0	-1	1	1	1	-1	0	-1	-1
$\{a,b,c,d\}$	0	0	-1	0	0	-1	0	0	0	0	0	-1	1	1	-1	-1	-2	0	-2	-2	-2
$\{a,b,c,e\}$	0	0	0	-2	0	1	1	0	0	0	0	0	1	1	-1	-1	0	1	-1	0	0
$\{a,b,d,e\}$	0	0	1	0	1	0	-1	-2	1	0	0	1	1	1	-2	-1	0	1	1	0	0
$\{a,c,d,e\}$	-1	0	1	0	0	1	1	0	0	0	0	0	0	1	-1	-1	0	1	-1	0	0
$\{b,c,d,e\}$	-1	1	0	0	0	0	0	0	0	0	-2	1	1	1	-1	-2	0	2	-1	1	1
$\{a,b,c,d,e\}$	1	0	1	2	1	1	1	2	1	1	2	1	0	0	3	3	2	0	3	2	2

Table 6: The 21 columns correspond to representatives of equivalence classes of $\mathcal{H}(N)$.

The consequence of finding $\mathcal{H}(N)$ for $|N| = 5$ in Bruns et al. (2010) is as follows:

Proposition 6 For $|N| = 5$, the least factor $n_* \in \mathbb{N}$ such that, for any imset u over N, it holds $u \in \mathcal{S}(N) \Leftrightarrow n_* \cdot u \in \mathcal{C}(N)$, is $n_* = 2$.

Proof Using a computer, we checked that every element $v \in \mathcal{H}(N)$ for $|N| = 5$ has the property that $2 \cdot v$ is a combinatorial imset. Now, given $u \in \mathcal{S}(N)$, consider its Hilbert basis decomposition $u = \sum_{v \in \mathcal{H}(N)} k_v \cdot v$, $k_v \in \mathbb{Z}_+$ and write $2 \cdot u = \sum_{v \in \mathcal{H}(N)} k_v \cdot (2 \cdot v)$, where each summand $2 \cdot v$ is known to be a combinatorial imset. In particular, $2 \cdot u \in \mathcal{C}(N)$. ■

Appendix C. Example of the Application of the LP Method

The aim of this text is to illustrate the LP method from Section 4.4 by an example. We assume that the reader is familiar with the details of the tableau form of the simplex method (see, for example, Schrijver, 1986, § 11.2). Consider the same situation as in Example 1:

$$N = \{a,b,c\}, \quad L = \{\, a \perp\!\!\!\perp b \,|\, c\,,\ a \perp\!\!\!\perp c \,|\, \emptyset\,\}, \ \text{ and } \ t : a \perp\!\!\!\perp b \,|\, \emptyset\,,$$

which leads to the task to verify/disprove $u \rightharpoonup v$ for

$$u = u_L = \delta_{abc} - \delta_{bc} - \delta_a + \delta_\emptyset \ \text{ and } \ v = u_t = \delta_{ab} - \delta_a - \delta_b + \delta_\emptyset\,.$$

By Lemma 5, this is equivalent to solving the system of equations (9). In this case, it means searching for a non-negative solution $\mathbf{x} \geq 0$ of $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is 8×7-matrix and $\mathbf{b}$ a column 8-vector specified as follows:

A	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	**b**
$\emptyset$	0	0	0	+1	+1	+1	−1	−1
a	0	0	+1	−1	−1	0	+1	+1
b	0	+1	0	−1	0	−1	0	+1
c	+1	0	0	0	−1	−1	0	0
ab	0	−1	−1	+1	0	0	0	−1
ac	−1	0	−1	0	+1	0	0	0
bc	−1	−1	0	0	0	+1	+1	0
abc	+1	+1	+1	0	0	0	−1	0

Here, the first 6 columns of $\mathbf{A}$ are elementary imsets, the 7-th column is $-u$, while $\mathbf{b}$ encodes $-v$. Moreover, the columns of $\mathbf{A}$ are named by the respective (searched) non-negative coefficients, while the rows are named by corresponding subsets of N. To get the standard case $\mathbf{b} \geq 0$ we modify it by multiplying by (-1) the rows of $\mathbf{A}$ and components of $\mathbf{b}$ corresponding $\emptyset$ and ab. The modified system is $\mathbf{A}'\mathbf{x} = \mathbf{b}'$, where one has:

$\mathbf{A}'$	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	$\mathbf{b}'$
$\emptyset$	0	0	0	−1	−1	−1	+1	+1
a	0	0	+1	−1	−1	0	+1	+1
b	0	+1	0	−1	0	−1	0	+1
c	+1	0	0	0	−1	−1	0	0
ab	0	+1	+1	−1	0	0	0	+1
ac	−1	0	−1	0	+1	0	0	0
bc	−1	−1	0	0	0	+1	+1	0
abc	+1	+1	+1	0	0	0	−1	0

We are going to solve the auxiliary LP problem (4) (see Section 3.4), to which purpose we create a tableau with additional identity submatrix and cost vector.

	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	$\emptyset$	a	b	c	ab	ac	bc	abc	
cost	0	0	0	0	0	0	0	+1	+1	+1	+1	+1	+1	+1	+1	0
$\emptyset$	0	0	0	−1	−1	−1	+1	+1	0	0	0	0	0	0	0	+1
a	0	0	+1	−1	−1	0	+1	0	+1	0	0	0	0	0	0	+1
b	0	+1	0	−1	0	−1	0	0	0	+1	0	0	0	0	0	+1
c	+1	0	0	0	−1	−1	0	0	0	0	+1	0	0	0	0	0
ab	0	+1	+1	−1	0	0	0	0	0	0	0	+1	0	0	0	+1
ac	−1	0	−1	0	+1	0	0	0	0	0	0	0	+1	0	0	0
bc	−1	−1	0	0	0	+1	+1	0	0	0	0	0	0	+1	0	0
abc	+1	+1	+1	0	0	0	−1	0	0	0	0	0	0	0	+1	0

The next step is to "recompute" the cost vector to get zeros above the identity submatrix. In our case it means subtracting the sum of 8 rows that correspond to sets from the previous version of the cost vector. In the additional row, bullets indicate the current basis and the corresponding (starting) vertex is given by the coefficients indicated.

	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	$\emptyset$	a	b	c	ab	ac	bc	abc	
cost	0	−2	−2	+4	+2	+2	−2	0	0	0	0	0	0	0	0	−4
$\emptyset$	0	0	0	−1	−1	−1	+1	+1	0	0	0	0	0	0	0	+1
a	0	0	+1	−1	−1	0	+1	0	+1	0	0	0	0	0	0	+1
b	0	+1	0	−1	0	−1	0	0	0	+1	0	0	0	0	0	+1
c	+1	0	0	0	−1	−1	0	0	0	0	+1	0	0	0	0	0
ab	0	+1	+1	−1	0	0	0	0	0	0	0	+1	0	0	0	+1
ac	−1	0	−1	0	+1	0	0	0	0	0	0	0	+1	0	0	0
bc	−1	−1	0	0	0	+1	+1	0	0	0	0	0	0	+1	0	0
abc	+1	+1	+1	0	0	0	−1	0	0	0	0	0	0	0	+1	0
								•	•	•	•	•	•	•	•	
vert								1	1	1	0	1	0	0	0	

Now, the proper algorithm can start. The strategy for the choice of the pivoting position from Schrijver (1986, § 11.2) leads to pivoting on $abc \times \lambda_{\langle a,c|b\rangle}$-cell, which results in the following tableau:

	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	$\emptyset$	a	b	c	ab	ac	bc	abc	
cost	+2	0	0	+4	+2	+2	−4	0	0	0	0	0	0	0	+2	−4
$\emptyset$	0	0	0	−1	−1	−1	+1	+1	0	0	0	0	0	0	0	+1
a	0	0	+1	−1	−1	0	+1	0	+1	0	0	0	0	0	0	+1
b	−1	0	−1	−1	0	−1	+1	0	0	+1	0	0	0	0	−1	+1
c	+1	0	0	0	−1	−1	0	0	0	0	+1	0	0	0	0	0
ab	−1	0	0	−1	0	0	+1	0	0	0	0	+1	0	0	−1	+1
ac	−1	0	−1	0	+1	0	0	0	0	0	0	0	+1	0	0	0
bc	0	0	+1	0	0	+1	0	0	0	0	0	0	0	+1	+1	0
abc	+1	+1	+1	0	0	0	−1	0	0	0	0	0	0	0	+1	0
		•						•	•	•	•	•	•	•		
vert		0						1	1	1	0	1	0	0		

The vertex and the value of the cost function in it remain unchanged, just the vertex is expresses using another basis. The same procedure now leads to the choice of another pivoting cell: the column corresponds to k and the row to $\emptyset$.

	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	$\emptyset$	a	b	c	ab	ac	bc	abc	
cost	+2	0	0	0	−2	−2	0	+4	0	0	0	0	0	0	+2	0
$\emptyset$	0	0	0	−1	−1	−1	+1	+1	0	0	0	0	0	0	0	+1
a	0	0	+1	0	0	+1	0	−1	+1	0	0	0	0	0	0	0
b	−1	0	−1	0	+1	0	0	−1	0	+1	0	0	0	0	−1	0
c	+1	0	0	0	−1	−1	0	0	0	0	+1	0	0	0	0	0
ab	−1	0	0	0	+1	+1	0	−1	0	0	0	+1	0	0	−1	0
ac	−1	0	−1	0	+1	0	0	0	0	0	0	0	+1	0	0	0
bc	0	0	+1	0	0	+1	0	0	0	0	0	0	0	+1	+1	0
abc	+1	+1	+1	−1	−1	−1	0	+1	0	0	0	0	0	0	+1	+1
		•					•		•	•	•	•	•	•		
vert		1					1		0	0	0	0	0	0		

Now, we observe the decrease in the value of the cost function: the corresponding vertex is given by $k = 1$, $\lambda_{\langle a,c|b\rangle} = 1$ and remaining coefficients vanishing. The value of the cost function is already 0, a further terminating iteration (= pivoting on $b \times \lambda_{\langle a,c|\emptyset\rangle}$-cell) does not change the vertex and, hence, the value of the cost function in it. Thus, we have successfully found a non-negative solution to (9). In particular, we confirmed $u_L \rightharpoonup u_t$ by the LP method.

Appendix D. Decomposition Interpretation of the LP Method

In spite of the different formulations, the procedure from Section 4.2.1 and the LP approach from Section 4.4 have common points. Given $u \in \mathcal{C}(N)$ and $v \in \mathcal{E}(N)$, both methods try to find a decomposition of $k \cdot u - v$ into elementary imsets. A formal difference is that k and the searched coefficients are integers in Section 4.2.1 but reals in Section 4.4. We try now to explain the analogy to the reader familiar with the details of the simplex method.

To see the analogy let us assume that k is a fixed natural number, as in Section 4.2.1. For example, let k be the upper limit for the factor l in (6). This ensures that $k \cdot u - v \in \mathcal{C}(N)$ iff $u \rightharpoonup v$. Put $b(S) = k \cdot u(S) - v(S)$ for $S \subseteq N$ and search for a non-negative solution in λ_w, $w \in \mathcal{E}(N)$ to the system

$$\sum_{w \in \mathcal{E}(N)} \lambda_w \cdot w(S) = b(S) \qquad \text{for any } S \subseteq N.$$

The set of such solutions is a polyhedron $\hat{\mathsf{Q}} = \{\mathbf{x} \in \mathbb{R}^n; \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$, with $\mathbf{A} \in \mathbb{R}^{d \times n}$ and $\mathbf{b} \in \mathbb{R}^d$. Here, the rows of $\mathbf{A}$ correspond to subsets of N ($d = |\mathcal{P}(N)|$), the columns to elementary imsets ($n = |\mathcal{E}(N)|$) and $\mathbf{b}$ has $b(S)$, $S \subseteq N$ as components. The only difference from the situation in Section 4.4 is that we have one column less.

As mentioned in Section 3.4, $\mathbf{b}$ (and $\mathbf{A}$) can be modified so that it has non-negative integers as components and entries in $\mathbf{A}$ remain in $\{-1, 0, 1\}$. The simplex method takes $(0, \mathbf{b}) \equiv (\mathbf{x}_0, \mathbf{z}_0)$ as its starting iteration and generates a sequence $(\mathbf{x}_0, \mathbf{z}_0), \ldots, (\mathbf{x}_m, \mathbf{z}_m)$, $m \geq 0$ of points in the feasible region of (4). The final iteration $(\mathbf{x}_m, \mathbf{z}_m)$ then minimizes the function $(\mathbf{x}, \mathbf{z}) \mapsto \langle 1, \mathbf{z} \rangle$ over that region. As explained in Section 3.4, $\hat{\mathsf{Q}} \neq \emptyset$ iff $\mathbf{z}_m = 0$.

Now, the move from $(0, \mathbf{b}) \equiv (\mathbf{x}_0, \mathbf{z}_0)$ to $(\mathbf{x}_1, \mathbf{z}_1)$ means the following. One finds $w \in \mathcal{E}(N)$, corresponding to a (column) component x_w, and a set $S' \subseteq N$, corresponding to a component $z_{S'}$, such that $b(S') - \lambda_w \cdot w(S') = 0$ with $\lambda_w \geq 0$. The components of $\mathbf{z}_1$ are then $b(S) - \lambda_w \cdot w(S)$ for $S \subseteq N$. Since $\mathbf{b} = \mathbf{z}_0$ has integral components, $\lambda_w \in \mathbb{Z}^+$. In other words, the first step of the simplex method, the move from $\mathbf{z}_0$ to $\mathbf{z}_1$, means subtracting a non-negative integral multiple of an elementary imset w from $\mathbf{b} = \mathbf{z}_0$ so that the result remains a non-negative vector.

Although further steps already cannot be interpreted as elementary imset subtraction, the following still holds. If $\hat{\mathsf{Q}} \neq \emptyset$, the simplex method finds iteratively the decomposition of $\mathbf{b}$ into elementary imsets with non-negative coefficients. Moreover, in every iteration, one set (= a component of $\mathbf{z}$) is "vanished". This is analogous to the procedure from Section 4.2.1, the difference is that the heuristic for finding the next set S' to be "vanished" is given by the pivoting operation.

References

Remco R. Bouckaert and Milan Studený. Racing algorithms for conditional independence inference. *International Journal of Approximate Reasoning*, 45(2):386–401, 2007.

Winfried Bruns, Bogdan Ichim, and Christof Söger. NORMALIZ, a tool for computations in affine monoids, vector configurations, lattice polytopes and rational cones. Available electronicly at `http://www.math.uos.de/normaliz/`, 2009.

Winfried Bruns, Raymond Hemmecke, Bogdan Ichim, Matthias Köppe, and Christof Söger. Challenging computations of Hilbert bases of cones associated with algebraic statistics. To appear in *Experimental Mathematics*, e-print available at `arxiv.org/abs/1001.4145v1`, 2010.

David M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(2):507–554, 2002.

Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer Verlag, 1999.

A. Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society B*, 41(1):1–31, 1979.

Matthias Franz. CONVEX, a Maple package for convex geometry. Available electronicly at `www.math.uwo.ca/~mfranz/convex/`, 2009.

Komei Fukuda. CDD and CDD+, an implementation of the double description method. Available electronicly at `www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html`, 2008.

Dan Geiger, Tom Verma, and Judea Pearl. D-separation: from theorems to algorithms. In *Proceedings of the 5th Conference on Uncertainty in Artificial Intelligence*, pages 118–125, Elsevier, 1989.

Raymond Hemmecke, Jason Morton, Anne Shiu, Bernd Sturmfels, and Oliver Wienand. Three counter-examples on semi-graphoids. *Combinatorics, Probability and Computing*, 17(2):239–257, 2008.

IBM Ilog team. CPLEX, a mathematical programming optimizer. Available electronicly at `www-01.ibm.com/software/integration/optimization/cplex/`, 2009.

Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer Verlag, 2001.

Steffen L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

Richard E. Neapolitan. *Learning Bayesian Networks*. Pearson Prentice Hall, 2004.

Mathias Niepert, Dirk van Gucht, and Marc Gyssens. On the conditional independence implication problem, a lattice theoretic approach. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 435–443, AUAI Press, 2008.

Mathias Niepert. Logical inference algorithms and matrix representations for probabilistic conditional independence. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 428–435, AUAI Press, 2009.

Mathias Niepert, Dirk van Gucht, and Marc Gyssens. Logical and algorithmic properties of stable conditional independence. *International Journal of Approximate Reasoning*, 51(5):531–543, 2010.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley, 1986.

Petr Šimeček. *Independence models* (in Czech). PhD thesis, Charles University, Prague, 2007.

Milan Studený. Multiinformation and the problem of characterization of conditional independence relations. *Problems of Control and Information Theory*, 18(1):3–16, 1989.

Milan Studený. Convex set functions I and II. Research reports n. 1733 and 1734, Institute of Information Theory and Automation, Prague, November 1991.

Milan Studený. Conditional independence relations have no finite complete characterization. In *Information Theory, Statistical Decision Functions and Random Processes, Transactions of the 11th Prague Conference, vol. B* (S. Kubík, J. Á. Víšek eds.), pages 377–396, Kluwer, 1992.

Milan Studený. Convex cones in finite-dimensional real vector spaces. *Kybernetika*, 29(2):180–200, 1993.

Milan Studený, Remco R. Bouckaert, and Tomáš Kočka. Extreme supermodular set functions over five variables. Research report n. 1977, Institute of Information Theory and Automation, Prague, January 2000.

Milan Studený. Structural imsets, an algebraic method for describing conditional independence structures. In *Proceedings of IPMU 2004* (B. Bouchon-Meunier, G. Coletti, R. R. Yager eds.), pages 1323–1330, 2004.

Milan Studený. *Probabilistic Conditional Independence Structures*. Springer Verlag, 2005.

Milan Studený, Jiří Vomlel, and Raymond Hemmecke. A geometric view on learning Bayesian network structures. *International Journal of Approximate Reasoning*, 51(5):573–586, 2010.

Milan Studený and Jiří Vomlel. On open questions in the geometric approach to structural learning Bayesian nets. Accepted in *International Journal of Approximate Reasoning*, to appear in 2011.

4ti2 team. 4TI2, a software package for algebraic, geometric and combinatorial problems on linear spaces. Available electronicly at `www.4ti2.de`, 2008.