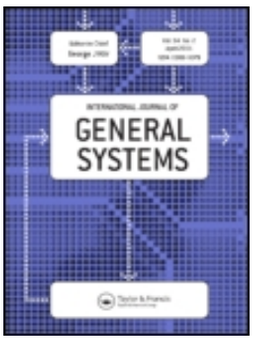

International Journal of General Systems

ISSN: 0308-1079 (Print) 1563-5104 (Online) Journal homepage: http://www.tandfonline.com/loi/ggen20

Computing minimal sets of descriptive conditions for binary data

Radim Belohlavek & Vilem Vychodil

To cite this article: Radim Belohlavek & Vilem Vychodil (2014) Computing minimal sets of descriptive conditions for binary data, International Journal of General Systems, 43:5, 521-534, DOI: 10.1080/03081079.2014.885166

To link to this article: http://dx.doi.org/10.1080/03081079.2014.885166

International Journal of General Systems, 2014
Vol. 43, No. 5, 521–534, http://dx.doi.org/10.1080/03081079.2014.885166

Taylor & Francis
Taylor & Francis Group

Computing minimal sets of descriptive conditions for binary data

Radim Belohlavek* and Vilem Vychodil

Data Analysis and Modeling Lab, Department of Computer Science, Palacky University, Olomouc, Czech Republic

(Received 31 December 2013; accepted 13 January 2014)

Suppose we are given a set of binary attributes. Each attribute is described by a finite set of objects to which the attribute applies. We consider the following problem. Is it possible to find a small set of conditions, or factors, associated to the attributes in such a way that an attribute applies to an object if and only if the object satisfies all conditions associated to the attribute? In this sense, we look for minimal sets of descriptive conditions for a set of binary attributes. We show that this problem can equivalently be expressed by so-called triangular decompositions of binary matrices. Small sets of descriptive conditions correspond to decompositions with small inner dimension. The conditions, or factors, used in these decompositions correspond to I-beam-shaped patterns in binary matrices. We utilize results on triangular decompositions and provide an efficient approximation algorithm for computing optimal decompositions. We present an example demonstrating usefulness of discovering descriptive conditions and evaluate performance of the algorithm.

Keywords: binary data; matrix decomposition; triangular product; closure operator

1. Problem description and related work

1.1. Motivation and problem description

Matrix decompositions play a fundamental role in data analysis. Namely, the decomposition methods allow us to reveal hidden factors explaining the input object-variable matrix. The factors may be regarded as new variables that are better structured and more economic in describing the given objects. The parsimonity of data description and the resulting dimensionality reduction is one benefit of revealing the factors. Importantly, as the factors may be considered as variables more fundamental than the original ones, the knowledge of the factors themselves and of the way the original variables are associated to the factors is valuable to understand the data.

In this paper, we are concerned with decompositions of binary matrices, i.e. matrices whose entries are 0 or 1. Unlike some of the recent approaches, which are mentioned below in this section and which seek to decompose a given $n \times m$ binary matrix I into the ordinary Boolean matrix product $I = A \circ B$ of binary matrices A and B, we are interested in triangular product decompositions, i.e. decompositions

$$I = A \triangleleft B,$$

where $\triangleleft$ is the operation of a so-called triangular product of binary matrices defined by

$$(A \triangleleft B)_{ij} = \min_{l=1}^{k}(A_{il} \rightarrow B_{lj}). \tag{1}$$

*Corresponding author. Email: radim.belohlavek@acm.org

Here, $\to$ denotes the truth function of logical implication (i.e. $0 \to 0 = 1, 0 \to 1 = 1, 1 \to 1 = 1$, and $1 \to 0 = 0$). Our aim is to look for decompositions with k (the number of factors) as small as possible. Triangular products were studied in a number of papers by Bandler and Kohout, see e.g. (Kohout and Bandler 1985; Kohout and Kim 2002; Lee, Kim, and Kohout 2004).

Compared to the ordinary Boolean matrix product $\circ$, the triangular product $\triangleleft$ provides us with a conceptually different way to construe a binary relation (represented by I) from two relations (represented by A and B). To illustrate the meaning of decomposition $I = A \triangleleft B$ and to compare it to the meaning of the ordinary product decomposition $I = A \circ B$, let the $n \times m$ binary matrix I describe n attributes and m objects. $I_{ij} = 1/0$ means that attribute i applies/does not apply to object j. Recall that $A \circ B$ is defined by

$$(A \circ B)_{ij} = \max_{l=1}^{k}(A_{il} \otimes B_{lj}) \tag{2}$$

where $\otimes$ denotes the truth function of of conjunction. First, $(A \circ B)_{ij} = 1$ means that there exists l such that $A_{il} = 1$ and $B_{lj} = 1$. Let us interpret $A_{il} = 1/0$ as indicating if attribute i is one of the particular manifestations of factor l and $B_{lj} = 1/0$ as indicating if factor l applies to j. Then, the meaning of the factor model $I = A \circ B$ is the following: attribute i applies to object j iff there exists factor l which applies to j and of which i is a particular manifestation. Contrary to that, $(A \triangleleft B)_{ij} = 1$ means that for each l for which $A_{il} = 1$, we have $B_{lj} = 1$. Let us interpret $B_{lj} = 1/0$ again as indicating if factor l applies to j and $A_{il} = 1/0$, differently from the case of $\circ$, as indicating that factor l is one of the necessary conditions for having attribute i. Then, the meaning of the factor model $I = A \triangleleft B$ is the following: attribute i applies to object j iff object j satisfies all the conditions l necessary for i. Therefore, decomposition $I = A \triangleleft B$ corresponds to revealing necessary conditions describing the original attributes and its meaning is diferent from that of $I = A \circ B$ which corresponds to revealing sufficient conditions.

Note also that the numbers of factors necessary for the decompositions based on $\circ$ and $\triangleleft$ may be different. As an example, for $I = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ we have

$$I = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \triangleleft \begin{pmatrix} 0 & 1 \end{pmatrix} \quad \text{and} \quad I = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

showing a $\triangleleft$-decomposition of I into a 2×1 and 1×2 matrices, i.e. using 1 factor, and a $\circ$-decomposition of I into a 2×2 and 2×2 matrices, i.e. using 2 factors. One may easily check that no $\circ$-decomposition of I using 1 factor exists.

The problem described is analogous to the problem of factor analysis (McDonald 1985). However, the model is technically different from the ordinary factor analysis models because we deal with binary matrices and with a matrix product which is different from the usual product of linear algebra used in the ordinary factor analysis. Clearly, particularly interesting are the decompositions with the number k of conditions (factors) smaller than the number n of attributes. In this case, the conditions represent new, perhaps more fundamental, attributes which provide a more concise description of the objects.

1.2. Related work

Learning various types of rules and conditional statements, which is a general theme related to our paper, has been the topic of many research activities. The approach we propose in this paper connects the problem of conditional statements to triangular decompositions of binary matrices. Binary matrices are a particular case of matrices with entries from residuated lattices, whose triangular decompositions were studied in (Belohlavek 2009). In this paper, we utilize the results from (Belohlavek 2009). Related to our paper is the recent work on decompositions of

binary matrices into binary matrices. This work includes associated to the input binary matrix. Decompositions of binary matrices into binary matrices are the subject of recent papers on data mining, see e.g. (Geerts, Goethals, and Mielikäinen 2006; Miettinen et al. 2006). Various aspects of the problem of decomposition of binary matrices are discussed in (Tatti et al. 2006; Vaidya, Atluri, and Guo 2007).

1.3. Contribution and outline of the paper

In view of the NP-hardness of computing decompositions of an input binary matrix I into $A \triangleleft B$ with the least possible inner dimension k, see Section 3, we utilize the description of optimal triangular decompositions of I provided in (Belohlavek 2009) and design an efficient greedy algorithm for computing (sub)optimal decomposition of I. This algorithm is experimentally evaluated on several well-known data-sets and is shown to deliver good decompositions. An important advantageous feature of the algorithm is the fact that it generates the factors of the decomposition from the most important to the least important. As a result, it can naturally be used to compute approximate decompositions by stopping it whenever a prescribed percentage of data have been explained by the factors computed so far. Using an example regarding mental health disorder, we demonstrate that computing the new type of decomposition helps us reveal a useful knowledge about the input data. In addition, we obtain an interesting experimental observation regarding the well-known MUSHROOM data-set containing 119 binary attributes and 8192 objects. Namely, MUSHROOM is decomposable using 85 factors only. That is, using the new product, MUSHROOM may be embedded into an 85 dimensional Boolean space without any loss.

The paper is organized as follows. Section 2 provides theoretical results regarding the optimal decompositions. Section 4 contains an example illustrating usefulness of the factors corresponding to the decompositions. In Section 3, we present a greedy algorithm for computing decompositions and its experimental evaluation. Section 5 concludes the paper and outlines future research.

2. Optimal conditions are minimal I-beams

Optimal triangular decompositions of matrices over residuated lattices are described in (Belohlavek 2009). Since binary matrices are a particular kind of such matrices, the results from (Belohlavek 2009) apply to binary matrices. As we utilize these results in the subsequent sections of our paper, we recall the relevant results from (Belohlavek 2009) in this section. For reader's convenience, we present the results in the particular setting of binary matrices, which is needed in our paper, and provide examples.

Definition 1 An $n \times m$ binary matrix J is called an I-beam if there exist binary matrices C and D of dimensions $n \times 1$ (column) and $1 \times m$ (row), such that such that $J = C \triangleleft D$.

The term comes from the shape of such matrices. For instance, for

$$C = \begin{pmatrix} 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \end{pmatrix}^{\mathrm{T}}, \quad \text{and} \quad D = \begin{pmatrix} 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \end{pmatrix},$$

the corresponding I-beam $C \triangleleft D$ is

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix}.$$

Let

$$\mathcal{F} = \{\langle C_1, D_1\rangle, \ldots, \langle C_k, D_k\rangle\}$$

be a set of $1 \times n$ and $1 \times m$ binary vectors C_l and D_l, respectively, and let us fix the indexing of this set. Define $n \times k$ and $k \times m$ matrices $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ by

$$(A_{\mathcal{F}})_{il} = (C_l)_i \quad \text{and} \quad (B_{\mathcal{F}})_{lj} = (D_l)_j. \tag{3}$$

That is, the lth column of $A_{\mathcal{F}}$ is C_l^{T} and the lth row of $B_{\mathcal{F}}$ is D_l. It is easy to see (Belohlavek 2009) that $I = A_{\mathcal{F}} \triangleleft B_{\mathcal{F}}$ iff I is the intersection of I-beams $C_l^{\mathrm{T}} \triangleleft D_l$, $l = 1, \ldots, k$. Note that an intersection $\min_k J_k$ of I-beams J_k is defined componentwise using minima, i.e. by $(\min_k J_k)_{ij} = \min_k (J_k)_{ij}$. Therefore, conditions/factors in the $\triangleleft$-decompositions of I can be identified with I-beams.

Example 1 For the following decomposition of a 4×5 matrix I:

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \triangleleft \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix},$$

the corresponding representation of I using I-beams is

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \wedge \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \wedge \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \wedge \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

For the I-beams J^1, J^2, J^3, J^4 of this representation, J^l results as a $\triangleleft$-product of the lth column of A and the lth row of B. Note that the I-beam shape of J^ls becomes apparent after permutation of rows and columns.

We see that in order to find a suitable decomposition of I into $A \triangleleft B$, we need to find suitable I-beams whose intersection yields I. Note first that the number of I-beams which cover a given $n \times m$ Boolean matrix I can be quite large. For instance, there are at least 2^n I-beams covering I since for every $1 \times n$ vector C, we can consider a corresponding $1 \times m$ vector D that has 1s at positions j such that $I_{ij} = 1$ for some i with $C_i = 1$. One can easily check that the corresponding I-beam $C^{\mathrm{T}} \triangleleft D$ covers I, i.e. $I_{ij} \leq (C^T \triangleleft D)_{ij}$. The reason for such large a number of beams is that there is a redundancy involved when we consider all I-beams. Namely, we can have two beams one inside the other. Such a redundancy may be removed if we restrict ourselves to particular beams that are fixed points of the operators we now introduce.

Let $X = \{1, 2, \ldots, n\}$ and $Y = \{1, 2, \ldots, m\}$. This notation will be used from now on. Define operators ${}^{\cap} : 2^X \to 2^Y$ and ${}^{\cup} : 2^Y \to 2^X$ by letting for $C \subseteq X$ and $D \subseteq Y$,

$$C^{\cap} = \{j \in Y \mid \text{for some } i \in C : I_{ij} = 1\}, \tag{4}$$

$$D^{\cup} = \{i \in X \mid \text{for each } j \in Y : \text{if } I_{ij} = 1 \text{ then } j \in D\}. \tag{5}$$

Denote by $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$ the sets of fixpoints of $\langle {}^{\cap}, {}^{\cup}\rangle$. That is,

$$\mathcal{B}(X^{\cap}, Y^{\cup}, I) = \{\langle C, D\rangle \mid C^{\cap} = D,\ D^{\cup} = C\}.$$

Let the characteristic vectors of $C \subseteq X$ and $D \subseteq Y$ be denoted by $c(C)$ and $c(D)$. That is, $c(C)_i = 1$ if $i \in C$ and $c(C)_i = 0$ if $i \notin C$; the same for D. For $\mathcal{F} \subseteq \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ we put

$$c(\mathcal{F}) = \{\langle c(C), c(D)\rangle \mid \langle C, D\rangle \in \mathcal{F}\}. \tag{6}$$

We now introduce a particular partial order which results in a suitable containment of I-beams mentioned above. For $C_1, C_2 \subseteq X$ and $D_1, D_2 \subseteq Y$, put

$$\langle C_1, D_1\rangle \leq_{\mathrm{I}} \langle C_2, D_2\rangle \quad \text{iff } C_1 \supseteq C_2 \,\&\, D_1 \subseteq D_2.$$

In terms of I-beams, $\langle C_1, D_1\rangle \leq_{\mathrm{I}} \langle C_2, D_2\rangle$ means that the I-beam $c(C_1)^{\mathrm{T}} \triangleleft c(D_1)$ corresponding to $\langle C_1, D_1\rangle$ is contained in the I-beam $c(C_2)^{\mathrm{T}} \triangleleft c(D_2)$ corresponding to $\langle C_2, D_2\rangle$, i.e. that $(c(C_1) \triangleleft c(D_1))_{ij} \leq (c(C_2) \triangleleft c(D_2))_{ij}$ for every i and j. It can be shown (Belohlavek 2009) that $\langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ are just the pairs $\langle C, D\rangle$ that are minimal with respect to $\leq_{\mathrm{I}}$ and for which $I_{ij} \leq (c(C)^T \triangleleft c(D))_{ij}$ for all i and j. That is, $\langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ are just minimal I-beams covering I.

Denote for an $n \times m$ binary matrix I by $\rho_{\triangleleft}(I)$ the least k such that there exist an $n \times k$ and a $k \times m$ binary matrices A and B for which $I = A \triangleleft B$. The following theorem describes the role of $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$ for triangular decompositions.

THEOREM 1 [optimality, (Belohlavek 2009)] *Let I be an $n \times m$ binary matrix. Then $\rho_{\triangleleft}(I) \leq \min(n, m)$. Moreover, there exists $\mathcal{F} \subseteq \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ with $|\mathcal{F}| = \rho_{\triangleleft}(I)$ such that for $c(\mathcal{F})$ defined by (6) and the $n \times |\mathcal{F}|$ and $|\mathcal{F}| \times m$ binary matrices $A_{c(\mathcal{F})}$ and $B_{c(\mathcal{F})}$ defined by (3) we have*

$$I = A_{c(\mathcal{F})} \triangleleft B_{c(\mathcal{F})}.$$

Next, we show that certain minimal I-beams, which we call mandatory, need to be included in every decomposition of I. Note that a corresponding theorem does not hold in the general setting of matrices over residuated lattices. They correspond to the subset $\mathcal{M}(X, Y, I)$ of fixpoints defined by

$$\mathcal{M}(X, Y, I) = \{\langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I) \mid \langle C, D\rangle = \langle (Y - \{y\})^{\cup}, \{x\}^{\cap}\rangle \text{ for some } x \in X, y \in Y\}.$$

THEOREM 2 *If $I = A_{c(\mathcal{F})} \triangleleft B_{c(\mathcal{F})}$ for $\mathcal{F} \subseteq \mathcal{B}(X^{\cap}, Y^{\cup}, I)$, then $\mathcal{M}(X, Y, I) \subseteq \mathcal{F}$.*

Proof We know from the above considerations that $I = \min_{\langle C,D\rangle \in \mathcal{F}} c(C)^{\mathrm{T}} \triangleleft c(D)$. We prove that $\mathcal{M}(X, Y, I) \subseteq \mathcal{F}$ by showing that if $\langle C, D\rangle \in \mathcal{M}(X, Y, I)$, i.e. $\langle C, D\rangle = \langle (Y - \{j\})^{\cup}, \{i\}^{\cap}\rangle$ for some $i \in X$, $j \in Y$, then $\langle C, D\rangle$ is the only fixpoint from $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$ with $(c(C)^{\mathrm{T}} \triangleleft c(D))_{ij} = 0$. In such a case, $\langle C, D\rangle$ needs indeed to be contained in $\mathcal{F}$ because otherwise $I_{ij} = 0$ but $(A_{c(\mathcal{F})} \triangleleft B_{c(\mathcal{F})})_{ij} = 1$. Therefore, let $\langle C_1, D_1\rangle$ be another fixpoint with $(c(C_1)^{\mathrm{T}} \triangleleft c(D_1))_{ij} = 0$. Then $i \in C_1$ and $j \notin D_1$. Furthermore, since $\{i\} \subseteq C_1$, we have $C = \{i\}^{\cap\cup} \subseteq C_1^{\cap\cup} = C_1$, and thus $D = C^{\cap} \subseteq C_1^{\cap} = D_1$. Since $j \notin D_1$, $D_1 \subseteq Y - \{j\}$. Therefore, $D_1 = D_1^{\cup\cap} = (Y - \{j\})^{\cup\cap} = D$. To sum up, $D = D_1$, whence $\langle C, D\rangle = \langle C_1, D_1\rangle$, which is a contradiction to the assumption that $\langle C, D\rangle \neq \langle C_1, D_1\rangle$. □

3. Algorithms

3.1. Greedy algorithm

In this section, we present an approximation algorithm for finding an optimal decomposition $I = A \triangleleft B$ which uses minimal I-beams covering I as conditions/factors.

3.1.1. Connection to set cover

Observe first a connection between the problem of decomposition to a dual version of the well-known set covering optimization problem for which we refer to Cormen et al. (2001). The dual problem can be formulated as follows. We are given a set U and a collection $\mathcal{S}$ of supersets of U whose intersection is U, i.e. $\bigcap \mathcal{S} = U$. The task is to find the smallest subcollection $\mathcal{C}$ of $\mathcal{S}$ with $\bigcap \mathcal{C} = U$. It is easily seen that the dual problem is reducible to the set covering optimization problem and *vice versa*. The set covering problem (hence, the dual problem as well) is NP-hard but there exists an efficient greedy approximation algorithm which achieves an approximation ratio $\leq \ln(|U|) + 1$, see (Cormen et al. 2001). A straightforward modification of this algorithm gives us an approximation algorithm for the dual problem with the same approximation ratio. Such an algorithm, called "naive", is the starting point of our greedy algorithm, denoted Algorithm 1.

The problem of finding an optimal decomposition $I = A \triangleleft B$ is reducible to the dual version of the set covering optimization problem by putting $U = \{\langle i, j\rangle \mid I_{ij} = 1\}$ and

$$\mathcal{S} = \{S_{\langle C,D\rangle} \mid \langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I)\}$$

where $S_{\langle C,D\rangle} = \{\langle i, j\rangle \mid (c(C) \triangleleft c(D))_{ij} = 1\}$. That is, U is the set of all pairs for which the corresponding entry I_{ij} is 1, and $\mathcal{S}$ consists of sets corresponding to fixpoints from $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$. Namely, each $S_{\langle C,D\rangle}$ in $\mathcal{S}$ corresponding to $\langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ contains just pairs $\langle i, j\rangle$ for which $(c(C) \triangleleft c(D))_{ij} = 1$. Then, according to the observations from Section 2, $\mathcal{C}$ is a solution to the dual problem if and only if

$$\mathcal{F} = \{\langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I) \mid S_{\langle C,D\rangle} \in \mathcal{C}\}$$

is a smallest set for which $I = A_{c(\mathcal{F})} \triangleleft B_{c(\mathcal{F})}$.

3.1.2. "Naive" algorithm with known approximation ratio

The above-mentioned greedy approximation algorithm for the set covering problem (and for its dual version which we need) can be adopted for computing optimal decompositions $I = A \triangleleft B$. Namely, the operator ${}^{\cap\cup} : 2^X \to 2^X$ sending $C \subseteq X$ to $(C^{\cap})^{\cup}$ is a closure operator, i.e. the following conditions are satisfied: $C \subseteq C^{\cap\cup}$; $C_1 \subseteq C_2$ implies $C_1^{\cap\cup} \subseteq C_2^{\cap\cup}$; and $C^{\cap\cup} = (C^{\cap\cup})^{\cap\cup}$. Note that while ${}^{\cap\cup} : 2^X \to 2^X$ is a closure operator, the compound operator ${}^{\cup\cap} : 2^Y \to 2^Y$ is an interior operator, i.e. it satisfies $D \supseteq D^{\cup\cap}$; $D_1 \subseteq D_2$ implies $D_1^{\cup\cap} \subseteq D_2^{\cup\cap}$; and $D^{\cup\cap} = (D^{\cup\cap})^{\cup\cap}$. Furthermore, we easily see that $\mathcal{B}(X^{\cap}, Y^{\cup}, I) = \{\langle C, C^{\cap}\rangle | C = C^{\cap\cup}\}$. Therefore, $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$ can be recovered from the fixpoints of ${}^{\cap\cup}$. Hence, $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$, which can be identified with $\mathcal{S}$ in the above description, can be computed using algorithms for computing sets of fixpoints of closure operators. Theorem 2 allows us to speed up the algorithm for finding an optimal set $\mathcal{F}$ by inserting the mandatory factors from $\mathcal{M}(X, Y, I)$ to $\mathcal{F}$ in the beginning. Putting together the above observations yield an approximation algorithm for computing optimal decompositions, i.e. for computing minimal sets of conditions describing the original attributes.

An apparent drawback of this algorithm, called "naive" algorithm in what follows, is the fact that we first need to compute the set $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$ of all fixpoints of $\langle {}^{\cap}, {}^{\cup}\rangle$. This set may be large.

Algorithm 1 Optimal decomposition

INPUT: I (binary matrix)
OUTPUT: $\mathcal{F}$ (set of conditions/factors for I) for which $I = A_{c(\mathcal{F})} \triangleleft B_{c(\mathcal{F})}$
set U ***to*** $\{\langle i, j\rangle \mid I_{ij} = 0\}$
set $\mathcal{F}$ ***to*** $\emptyset$
while $(U \neq \emptyset)$:
  set C ***to*** $\emptyset$
  set V ***to*** 0
  while there is $i \notin C$ ***such that*** $|C \oplus i| > V$:
    do select $i \notin C$ ***that maximizes*** $|C \oplus i|$:
      set D ***to*** $(C \cup \{i\})^{\cap}$
      set C ***to*** $D^{\cup}$
      set V ***to*** $|(C \times \overline{D}) \cap U|$
  add $\langle C, D\rangle$ ***to*** $\mathcal{F}$
  for each $\langle i, j\rangle \in C \times \overline{D}$:
    remove $\langle i, j\rangle$ ***from*** U
return $\mathcal{F}$

On the other hand, an advantage of the "naive" algorithm is that it has a known approximation ratio, that is, an estimate of how well the delivered number of factors approximates the optimal number of factors. For this reason, we keep the "naive" algorithm for reference in experiments.

3.1.3. Greedy algorithm

In the following, we propose another greedy algorithm for computing optimal decompositions $I = A \triangleleft B$. The algorithm is derived from Algorithm 2 described in (Belohlavek and Vychodil 2010) – an algorithm for computing (sub)optimal decomposition of a binary matrix into a Boolean product (i.e. max-min product) of two binary matrices.

The algorithm takes I as its input and delivers a set $\mathcal{F} \subseteq \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ of fixpoints (minimal I-beams) for which $I = A_{c(\mathcal{F})} \triangleleft B_{c(\mathcal{F})}$. The algorithm starts with the input matrix I, an empty set $\mathcal{F}$ of minimal I-beams which cover I, and a set U representing entries of I which contain 0. At every point, the algorithm uses a heuristic for selecting a minimal I-beam covering I which is added to $\mathcal{F}$. The heuristic consists in maximizing the area of intersection of two areas. The first one is the area of I which contains 0s outside the I-beam. The second one is the area of 0s from I which were not "ruled out" by previously included I-beams. The latter area is represented by U, which is being updated after selecting every particular I-beam.

Technically, the algorithm uses the fact that for every fixpoint $\langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ we have $C = \bigcup_{x \in C} \{x\}^{\cap\cup}$ and that if $x \notin C$ then $\langle (C \cup \{x\})^{\cap\cup}, (C \cup \{x\})^{\cap}\rangle$ is a fixpoint from $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$ for which $C \subset (C \cup \{x\})^{\cap\cup}$. As a consequence, any fixpoint from $\mathcal{B}(X^{\cap}, Y^{\cup}, I)$ given by $C \subseteq X$ can be obtained by incremental addition of elements from X. Namely, for any $\langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I)$, there is a finite sequence $\langle C_0, D_0\rangle, \ldots, \langle C_k, D_k\rangle$ of fixpoints and elements $x_1, \ldots, x_k$ from X such that $C_0 = \emptyset^{\cap\cup}$, $C_k = C$, $C_i = (C_{i-1} \cup \{x_i\})^{\cap\cup}$ and $C_{i-1} \subset C_i$ for all $i = 1, \ldots, k$. This incremental construction of fixpoints which begins with $\langle \emptyset^{\cap\cup}, \emptyset^{\cap}\rangle$ and ends up with $\langle C, D\rangle$ is exploited by the greedy algorithm. Furthermore, in order to express the intersection of areas mentioned above, we introduce the following notation. For $C \subseteq X = \{1, \ldots, n\}$ and $x \in X$ ($\overline{P}$ denotes the complement of set P), we put

$$C \oplus x = \left[(C \cup \{x\})^{\cap\cup} \times \overline{(C \cup \{x\})^{\cap}}\right] \cap U.$$

Algorithm 2 Computing I-beams

INPUT: $\langle C, D\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ and $i \notin C$
OUTPUT: $\langle C', D'\rangle \in \mathcal{B}(X^{\cap}, Y^{\cup}, I)$ with $C' = (C \cup \{i\})^{\cap\cup}$
set D' ***to*** $D \cup \{i\}^{\cap}$
set C' ***to*** C
for each $x \in X$ ***such that*** $x \notin C$:
  if $\{x\}^{\cap} \subseteq D'$ ***then***:
    set C' ***to*** $C' \cup \{x\}$
return $\langle C', D'\rangle$

We can easily see that $(C \cup \{x\})^{\cap\cup} \times \overline{(C \cup \{x\})^{\cap}}$ is the area consisting of 0s which are outside the area of the I-beam corresponding to $\langle (C \cup \{x\})^{\cap\cup}, (C \cup \{x\})^{\cap}\rangle$. Therefore, $|C \oplus x|$ is the number of 0s in I which are outside of the I-beam and are in U. Algorithm 1 contains the pseudocode of our algorithm. As in the case of the "naive" algorithm, one may speed up Algorithm 1 by inserting the mandatory factors from $\mathcal{M}(X, Y, I)$ to $\mathcal{F}$ in the beginning. Due to Theorem 2, this guarantees a speedup if an exact decomposition of I is looked for (cf. Approximate Decompositions below).

3.1.4. Greedy algorithm vs. "naive" algorithm

Compared to the "naive" algorithm based on the almost direct adoption of the greedy approximation algorithm for the set covering problem which we described earlier in this section, Algorithm 1 is much faster (see Section 4.2 for details). For example, for matrices of dimension 8000×100, Algorithm 1 finishes in the order of tens of seconds while the "naive" algorithm needs tens of minutes. Nevertheless, the quality of decomposition which Algorithm 1 delivers is comparable to those delivered by the "naive" algorithm both in terms of the number k of conditions/factors which the algorithms finds and in terms of the sizes of the I-beams which the algorithms compute. Namely, both algorithms tend to produce I-beams from the smallest ones to the largest ones and the sizes of the I-beams produced by the algorithms tend to be perfectly correlated. This is an interesting observation because, as mentioned above, the "naive" algorithm has a guaranteed approximation ratio.

3.1.5. Approximate decompositions

In addition to the exact decompositions of I into $A \triangleleft B$, one is also interested in approximate decomposition $A \triangleleft B$ of I, by which we mean looking for conditions/factors such that the corresponding $A \triangleleft B$ coincides with at least a given portion of the data matrix I. The approximate decompositions become interesting in particular for large data-sets. An important property of Algorithm 1 is that approximate decompositions can naturally be obtained by a simple modification. Namely, we stop updating $\mathcal{F}$ in Algorithm 1 once I and $A_{c(\mathcal{F})} \triangleleft B_{c(\mathcal{F})}$ coincide on a prescribed portion of entries. That is, condition ***while*** $(U \neq \emptyset)$ in the pseudocode is replaced by ***while*** $(|U| < \varepsilon)$ for some ε. Experiments demonstrating approximate decompositions are presented in Section 4.2.

3.1.6. Computing I-beams

Another useful property of Algorithm 1 is that it can further be optimized. One source of optimization is an efficient computation of I-beams which are used as candidates for factors. Namely, Algorithm 1 repeatedly computes $C \oplus i$ which is apparently the most time-consuming operation in

the whole "while-loop" in Algorithm 1. According to its definition, $C \oplus i$ depends on determining the I-beam

$$\langle C', D' \rangle = \langle (C \cup \{i\})^{\cap\cup}, (C \cup \{i\})^{\cap} \rangle,$$

where $i \notin C$. An efficient algorithm for computing $\langle C', D' \rangle$ can use the fact that the previously computed $\langle C, C^{\cap} \rangle = \langle C, D \rangle$ is an I-beam as well. Indeed, during each step of the while-loop in Algorithm 1, $\langle C, D \rangle$ is an I-beam. Therefore,

$$D' = (C \cup \{i\})^{\cap} = C^{\cap} \cup \{i\}^{\cap} = D \cup \{i\}^{\cap},$$

i.e. D' can be obtained as a union of D and a subset of Y given by the ith row of I. If D and I are represented by bit arrays (e.g. arrays of 32 or 64 bit integers encoding blocks of 32 or 64 Boolean values), this operation can be done by performing logical OR on the bit arrays (i.e. bitwise OR) which is very efficient.

Moreover, C' can be computed by adding new values $x \in X$ to C. Clearly, $C \subseteq C'$. Therefore, one can initially set C' to C and iterate over all $x \in X$ which are not in C and check whether x should be added to C'. Since $C' = (D \cup \{i\}^{\cap})^{\cup} = (D')^{\cup}$, $x \in C'$ iff for each $y \in Y$, $I_{xy} = 1$ implies $y \in D'$. The latter is true iff $\{x\}^{\cap} \subseteq D'$, i.e. iff all nonzero entries in the row $I_{x\_}$ corresponding to x are included among the nonzero entries of the characteristic vector of $c(D')$. Hence, it suffices to check a simple inclusion-based condition to determine whether $x \in C'$. Such a test can be performed using bitwise NOT and OR. Altogether, if I, C, and D are represented by bit arrays, the I-beam $\langle C', D' \rangle$ can be computed using fast low-level Boolean operations which are implemented in arithmetic logic units (ALUs) of contemporary computers. Using such low-level operations significantly improves the performance of factorization. The procedure for computing I-beams we have just described is formalized by Algorithm 2.

4. Examples and experiments

In this section, we present an example which illustrates the new type of decomposition. Furthermore, we compare the new type of decomposition with the standard Boolean decomposition (2) on selected standard benchmark data-sets.

4.1. Illustrative example

To ilustrate the meaning of the new type of decomposition, we consider data describing a set of students and their abilities. In particular, consider matrix I with 8 rows representing selected keywords that appear as notions in subjects related to computer science: 1: algorithm, 2: closure, 3: completeness, 4: implication, 5: index, 6: matrix, 7: polynomial, and 8: recursion. The matrix consists of five columns corresponding to students (denoted $1, \ldots, 5$). Each matrix entry indicates whether a student has a difficulty with understanding/recognizing the particular keyword/notion. Following this interpretation, we can say that $I_{ij} = 1$ iff the student j is unfamiliar with keyword/notion i. Such data can be gathered, e.g. from questionnaires filled by freshmen students. Let us assume that I has the following form:

$$I = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

The matrices A and B in a decomposition $I = A \triangleleft B$ can be interpreted as follows: Matrix A describes relationship between keywords and *topics* (e.g. subjects) in which the keywords/notions appear. Matrix B indicates which students may have difficulties with the topics. Indeed, it is natural to consider that student j is unfamiliar with keyword/notion i whenever he is weak in every topic which exploits/uses the keyword/notion i. Following this interpretation, in order to discover the critical topics of interest, one needs to find a decomposition $I = A \triangleleft B$.

Using Algorithm 1 presented in Section 3, we obtain from I a set $\mathcal{F} = \{F_1, F_2, F_3\}$ of the following fixpoints (I-beams):

$$\begin{aligned} F_1 &= \langle \{2, 5, 6, 7\}, \{2\} \rangle, \\ F_2 &= \langle \{1, 4, 5, 7, 8\}, \{1, 3, 5\} \rangle, \\ F_3 &= \langle \{2, \ldots, 8\}, \{2, \ldots, 5\} \rangle. \end{aligned}$$

The attribute-condition (i.e. keyword-topic) matrix $A_{c(\mathcal{F})}$ and the condition-object (i.e. topic-student) matrix $B_{c(\mathcal{F})}$ induced by $\mathcal{F}$, which are defined by (3), are:

$$A_{c(\mathcal{F})} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad B_{c(\mathcal{F})} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The I-beams corresponding to F_1, F_2, and F_3, are the following:

$$F_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \; F_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \; F_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The original matrix I equals $A_{c(\mathcal{F})} \triangleleft B_{c(\mathcal{F})} = F_1 \wedge F_2 \wedge F_3$. The factors discovered by Algorithm 1 can be naturally interpreted as subjects related to computer science. Namely, factor F_1, which is manifested by keywords/notions closure, index, matrix, and polynomial can be interpreted as a subject "(linear) algebra". Analogously, F_2 manifested by keywords algorithm, implication, index, polynomial, and recursion can be interpreted as a subject "algorithms and programming".

Finally, F_3 can be seen as a subject "mathematics" since it is manifested by all keywords except for algorithms. A factorization of this type can help increase efficiency in education by assigning students to class sections according the abilities and weaknesses.

4.2. *Factorization of selected standard data-sets*

In order to compare the efficiency of the new type of decomposition, we compared Algorithm 1 with an algorithm computing the standard Boolean decomposition (2). We used standard data-sets CONNECT, CHESS, MUSHROOM, and TIC-TAC-TOE from the UCI Machine Learning Repository. For these data-sets, we compared Algorithm 1 with the greedy approximation algorithm for the standard Boolean decomposition described in (Belohlavek and Vychodil 2010). These data-sets often appear in papers dealing with dependency discovery (e.g. association rules mining). The choice of the algorithm from (Belohlavek and Vychodil 2010) is appropriate because it has a guaranteed approximation ratio and unlike other algorithms for computing standard Boolean decompositions, it can always achieve an exact factorization. Currently, the algorithm from (Belohlavek and Vychodil 2010) seem to be the best algorithm for computing max-min decompositions in terms of quality of decompositions.

Before we discuss details of results for the particular data-sets, let us note that a general observation on using the usual and the new type of decompositions for various real-world data-sets we have made is that *neither of the two types of decompositions can be claimed as superior to the other.* There are real-world data-sets where the usual max-min decomposition yields better factorizations (with smaller number of factors) and *vice versa*. Also, there are data-sets where the two factorizations produce almost the same numbers of factors and in addition the factors explain almost the same amounts of data. Such observations, which may be intuitively expected, are important as they support the need to investigate further types of decompositions and show that the usual max-min decomposition is not the "universal one", i.e. the one which always yields the smallest number of factors.

The difference between the new type of decomposition and the standard Boolean decomposition is considerable in case of the MUSHROOM data-set which describes 8192 selected mushrooms and 119 of their characteristic attributes. The density of MUSHROOM, i.e. the number of 1s in the data-set divided by the total number of entries, is 0.19. Algorithm 1 produced an (exact) $\triangleleft$-decomposition of I into a 119×85 binary matrix A and a 85×8192 matrix B. This decomposition reveals an interesting, previously unknown fact: MUSHROOM contains 85 (hidden) conditions/factors which completely explain the 119 (explicitly present) attributes. That is, the data can be represented in a 85-dimensional Boolean space of factors instead of the 119-dimensional Boolean space of original attributes. Importantly, such decomposition is a lossless one, i.e. I is exactly equal to $A \triangleleft B$. In case of the standard Boolean decomposition $\circ$, the algorithm from (Belohlavek and Vychodil 2010) does not produce an exact decomposition with the number of factors strictly smaller than the number of the original attributes. Thus, MUSHROOM is an example of a data-set for which $\triangleleft$-decompositions yield considerably smaller numbers of factors than $\circ$-decompositions.

Interestingly, we may obtain a very good approximation of MUSHROOM using a small number of factors. Figure 1 shows the approximation by the first 50 factors. The gray dashed line in the figure denotes the approximation by factors of the $\circ$-decomposition, whereas the solid black line denotes the approximation by factors of the $\triangleleft$-decomposition. One can observe from Figure 1 that the percentage of the data in MUSHROOM that are explained by the first n factors is growing rapidly for the first 10 factors: more than 50% of the data are explained by three factors, more than 70% of the data are explained by six factors, and more than 80% of the data are explained

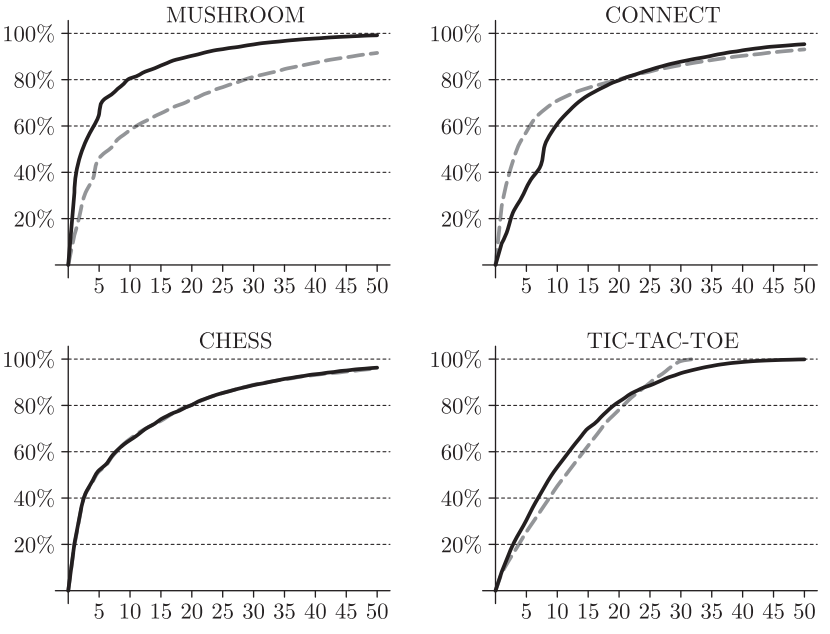

Figure 1. Approximation by first 50 factors in various data-sets.

by 10 factors. Then, the growth of the approximation is slower but still, a large portion of the data can be explained by a number of factors much smaller than 85. For instance, 95% of the data is explained by 30 factors. Table 1 contains the numbers of factors that are needed to achieve a prescribed approximation of the data. For each data-set, we include two lines, the upper and lower lines contain numbers of factors of the $\circ$-decomposition and $\triangleleft$-decompositions, respectively. In case of MUSHROOM, one can see that not only the numbers of factors using $\triangleleft$-decompositions are smaller but in addition they explain considerably greater parts of the data.

Similar experiments have been made with CONNECT (67, 557 objects and 129 attributes, density 0.33), CHESS (3196 objects and 75 attributes, density 0.49), and TIC-TAC-TOE (958 and 29 attributes, density 0.34). From Figure 1 and Table 1, one can see that in case of CHESS, both types of factorization have roughly the same performance. It is also apparent that CHESS does not seem to be (exactly) factorizable by either of the methods although the $\triangleleft$-decomposition performs slightly better. On the other hand, TIC-TAC-TOE is an example of a data-set where the $\circ$-decomposition yields better results. Notice that in this case, the first 24 factors obtained by the $\triangleleft$-decomposition are better (in terms of the coverage) than those obtained by the $\circ$-decomposition, but then a relatively small part of the data (about 10%) are covered by a considerably larger number of factors than in case of the $\circ$-decomposition. The CONNECT data-set shows an opposite behavior to that of TIC-TAC-TOE. The first 21 factors of the $\circ$-decomposition are better than the first 21 factors of the $\triangleleft$-decomposition but then the situation changes.

Let us add that both the types of decompositions can be useful not only in cases where the exact factorization reveals a number of factors smaller than the original attributes (as in case of MUSHROOM and the $\triangleleft$-decomposition) but also in cases where one wants to simplify the data-set by considering an approximate decomposition. For instance, from Table 1 it is apparent that, e.g. 95% of data are explained by considerably smaller number of factors than the original attributes (in three of the four cases, $\triangleleft$-decompositions deliver better results).

Table 1. Numbers of factors that approximate $p\%$ of data-sets.

		10%	20%	30%	40%	50%	60%	70%	80%	90%	92%	95%	98%	99%	100%
MUSHROOM	○	1	2	3	5	7	12	19	29	46	52	62	76	85	120
	◁	1	1	1	2	3	5	6	10	20	23	30	42	48	85
CONNECT	○	1	1	2	3	4	6	10	20	39	46	59	83	99	214
	◁	2	3	5	7	8	10	14	21	35	39	49	68	84	209
CHESS	○	1	2	2	3	5	8	13	20	33	37	47	62	72	124
	◁	1	1	2	3	5	8	13	20	33	37	45	61	71	121
TIC-TAC-TOE	○	2	4	7	9	12	15	18	21	26	27	28	30	30	32
	◁	2	3	5	7	10	12	15	20	27	28	32	38	42	57

5. Conclusions and future research

We presented a novel approach to decomposition of binary data. The decomposition can be interpreted as providing new attributes (conditions/factors) which describe the original ones. Utilizing recent theorems that describe optimal decompositions, we proposed a greedy approximation algorithm for computing a minimal set of conditions/factors. We provided an illustrative example and performed experiments evaluating the quality of the decompositions delivered compared to the standard Boolean decompositions.

Future research will focus on two directions. One is a further development of the decomposition algorithms, utilizing possibly further heuristics and new theoretical insight into the decompositions. This includes a study of various interestingness criteria for factors, including psychologically motivated criteria. The other direction is an exploitation of the new type of decomposition as a method of dimensionality reduction. For example, a recent observation (Outrata 2010) that the standard Boolean decomposition may help improve classification of binary data offers a natural question of the role of triangular decompositions. Since, as shown by the experiments, the triangular decomposition may achieve better dimensionality reduction compared to the standard Boolean decomposition, the triangular decomposition has a potential to be successfully employed for classification in a similar scenario as the one in (Outrata 2010). The last direction for future research consists in extending the methods to triangular decompositions from binary to ordinal data (see e.g. (Belohlavek 2009; Konecny 2011)) and as well as to more general frameworks involving binaru and ordinal data, see e.g. (Bartl and Klir 2014; Belohlavek and Osicka 2012; Belohlavek and Vychodil 2012).

Funding

Supported by [grant number P202/10/0262] of the Czech Science Foundation.

Notes on contributors

Radim Belohlavek received his PhD degree in Computer Science from the Technical University of Ostrava (Czech Republic) in 1998, PhD degree in Mathematics from Palacky University in 2001, and DSc degree in Informatics and Cybernetics from the Academy of Sciences of the Czech Republic in 2008. He is professor of Computer Science at Palacky University, Olomouc. His academic interests are in the areas of uncertainty and information, fuzzy logic and fuzzy sets, applied algebra and logic, and data analysis and formal concept analysis. He published two books, *Fuzzy Relational Systems: Foundations and Principles* (Kluwer, 2002) and *Fuzzy Equational Logic* (Springer, 2005, with Vilem Vychodil), and over 150 papers in conference proceedings and journals. He is a senior member of IEEE (Institute of Electrical and Electronics Engineers) and a member of ACM (Association

for Computing Machinery), AMS (American Mathematical Society), and is a member of editorial boards of several international journals.

Vilem Vychodil received his MSc degree in computer science (2002) and PhD (2004) degree in algebra and geometry (specialization: fuzzy logic) from Palacky University (Olomouc, Czech Republic). During 2007–2009, he was with the Dept. of Systems Science and Industrial Engineering (SUNY Binghamton). He is currently working as an associate professor in Dept. of Computer Science and as a researcher in Data Analysis and Modeling Laboratory (DAMOL), Palacky University. His research is focused on relational data analysis and logic in computer science.

References

Bandler, W., and L. J. Kohout. 1980a. "Semantics of Implication Operators and Fuzzy Relational Products." *International Journal of Man-Machine Studies* 12: 89–116.

Bandler, W., and L. J. Kohout. 1980b. "Fuzzy Relational Products as a Tool for Analysis and Synthesis of the Behaviour of Complex Natural and Artificial Systems." In *Fuzzy Sets: Theory and Applications to Policy Analysis and Information Systems*, edited by P. P. Wang and S. K. Chang, 341–367. New York: Plenum Press.

Bartl, E., and G. J. Klir. 2014. "Fuzzy Relational Equations in General Framework." *International Journal General Systems* 43 (1): 1–18.

Belohlavek, R. 2009. "Optimal Triangular Decompositions of Matrices with Entries from Residuated Lattices." *International Journal of Approximate Reasoning* 50 (8): 1250–1258.

Belohlavek, R., and P. Osicka. 2012. "Triadic Concept Lattices of Data with Graded Attributes." *International Journal of General Systems* 41 (2): 93–108.

Belohlavek, R., and M. Trnecka. 2013. *From-below Approximations in Boolean Matrix Factorization: Geometry and New Algorithm*. In revision, arXiv:1306.4905 [cs.NA].

Belohlavek, R., and V. Vychodil. 2010. "Discovery of Optimal Factors in Boolean Factor Analysis Via Novel Method of Binary Matrix Decomposition." *Journal of Computer and System Sciences* 76 (1): 3–20.

Belohlavek, R., and V. Vychodil. 2012. "Formal Concept Analysis and Linguistic Hedges." *International Journal of General Systems* 41 (5): 503–532.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms*. 2nd ed. Cambridge, MA: MIT Press.

Ganter, B., and R. Wille. 1999. *Formal Concept Analysis. Mathematical Foundations*. Berlin: Springer.

Geerts, F., B. Goethals, and T. Mielikäinen. 2006. "Tiling Databases." *Proceedings of Discovery Science* 2004: 278–289.

Kohout, L. J., and W. Bandler. 1985. "Relational-product Architectures for Information Processing." *Information Sciences* 37 (1–3): 25–37.

Kohout, L. J., and E. Kim. 2002. "The Role of BK-products of Relations in Soft Computing." *Soft Computing* 6 (2): 92–115.

Konecny, J. 2011. "Isotone Fuzzy Galois Connections with Hedges." *Information Sciences* 181 (10): 1804–1817.

Lee, Y., Y.-G. Kim, and L. J. Kohout. 2004. "An Intelligent Collision Avoidance System for AUVs Using Fuzzy Relational Products." *Information Sciences* 158: 209–232.

McDonald, R. P. 1985. *Factor Analysis and Related Methods*. London: Lawrence Erlbaum Associates.

Miettinen, P., T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. 2006. "The Discrete Basis Problem." *Proceedings of PKDD* 2006: 335–346.

Outrata, J. 2010. Boolean Factor Analysis for Data Preprocessing in Machine Learning. In *Proceedings of ICMLA*, edited by S. Draghici, T. M. Khoshgoftaar, V. Palade, V. Pedrycz, M. A. Wani, X. Zhu, 899–902. Washington, DC: IEEE Computer Society.

Tatti, N., T. Mielikäinen, A. Gionis, and H. Mannila. 2006. "What is the Dimension of Your Binary Data?" *Proceedings of ICDM* 2006: 603–612.

Vaidya, J., V. Atluri, and Q. Guo. 2007. "The Role Mining Problem: Finding a Minimal Descriptive Set of Roles." *Proceedings of ACM Symposium on Access Control Models and Technologies* 2007: 175–184.