



Factorization of Matrices With Grades With Overcovering

Radim Belohlavek , Senior Member, IEEE, and Marketa Trneckova 

Abstract—We present a novel algorithm for factorization of matrices with grades or, equivalently, for decomposition of fuzzy relations. The algorithm is inspired by a recent factorization method for Boolean matrices and develops two ideas in the setting of ordinal attributes. First, it uses formal concepts associated with the factorized matrix, or fuzzy relation, as essential components around which factors are built. Second, it steps back when computing new factors to check whether some computed factors may be eliminated or improved given the subsequently generated factors. The new algorithm thus uses convenient properties of formal concepts utilized by previous factorization algorithms. Still, unlike the previous algorithms, our algorithm allows for more general and therefore more precise factorizations due to a possible overcovering of the input data, which our new algorithm admits. We provide an experimental evaluation of the new algorithm and compare it to some existing algorithms for factorization of data with grades. The evaluation reveals that our new algorithm outperforms the current algorithms in terms of quality of factorization. We also present observations and improvements for factorization of Boolean matrices. We conclude with a discussion regarding open research topics.

Index Terms—Decomposition of fuzzy relations, factorization, fuzzy concept lattice, fuzzy logic, fuzzy relation.

I. PROBLEM DESCRIPTION

A. Matrices With Grades

WE DEAL with $n \times m$ matrices I whose entries I_{ij} are elements of an ordered scale L ; the set of all such matrices shall be denoted $L^{n \times m}$. We suppose that the elements a in L represent truth degrees and hence the matrix I represents a fuzzy relation (graded relation) between n objects (rows) and m attributes (columns). The entry $I_{ij} \in L$ shall be interpreted as the degree to which the object i has the attribute j . We assume that in general, L is equipped with a partial order \leq with respect to which it forms a complete lattice bounded by 0 and 1, and that a binary operation \otimes is defined on L , which is commutative (i.e., $a \otimes b = b \otimes a$), associative (i.e., $a \otimes (b \otimes c) = (a \otimes b) \otimes c$), has 1 as its neutral element (i.e., $a \otimes 1 = a = 1 \otimes a$), and is distributive over arbitrary suprema (i.e., $a \otimes (\bigvee_{j \in J} b_j) = \bigvee_{j \in J} (a \otimes b_j)$). This allows one to regard \otimes as a reasonable many-valued conjunction [10], [11]. Note

that the assumed structure on L may be described in terms of other well-known structures; e.g., L equipped with \vee and \otimes forms an integral commutative quantale.

The operation \otimes and the supremum in L (maximum if L is a chain) allow one to define the well-known sup- \otimes -product $A \circ B$ of matrices $A \in L^{n \times k}$ and $B \in L^{k \times m}$ by

$$(A \circ B)_{ij} = \bigvee_{l=1}^k (A_{il} \otimes B_{lj}). \quad (1)$$

The grades in scales L are conveniently represented by numbers, such as the Likert scale $\{1, \dots, 5\}$. We assume that these numbers are normalized and hence are in the unit interval $[0, 1]$. Hence, the Likert scale is represented by $L = \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$. Due to the well-known Miller's 7 ± 2 phenomenon [13], small scales with up to 7 ± 2 degrees are preferable to use because humans can understand and use such scales easily.

Example 1: Examples of scales L equipped with operations \otimes are known in fuzzy logic; among them L being the real unit interval $[0, 1]$ or its finite equidistant subinterval, i.e., $L = \{0, \frac{1}{n}, \dots, \frac{n-1}{n}, 1\}$, which we use in our examples. In particular, the Łukasiewicz operation \otimes is given by

$$a \otimes b = \max(0, a + b - 1).$$

With the Łukasiewicz \otimes one has, for instance

$$\begin{pmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \\ 0.5 & 0.5 \end{pmatrix} \circ \begin{pmatrix} 0.5 & 1.0 & 0.0 \\ 0.0 & 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.5 & 1.0 & 0.0 \\ 0.0 & 0.5 & 0.5 \\ 0.0 & 0.5 & 0.0 \end{pmatrix}.$$

■

B. Factorization Problem

We are interested in finding a good factorization of a given object-attribute matrix $I \in L^{n \times m}$ into an object-factor matrix A and a factor-attribute matrix B . Basically, this means to find $A \in L^{n \times k}$ and $B \in L^{k \times m}$ such that I approximately equals $A \circ B$, i.e.,

$$I \approx A \circ B \quad (2)$$

and the number k of factors is small.

To assess approximate equality $I \approx J$ of matrices $I, J \in L^{n \times m}$, it is common to use

$$I \approx J = \frac{\sum_{i,j=1}^{n,m} I_{ij} \leftrightarrow J_{ij}}{n \cdot m} \quad (3)$$

Manuscript received 19 June 2023; revised 16 October 2023; accepted 2 November 2023. Date of publication 7 November 2023; date of current version 2 April 2024. Recommended by Associate Editor C. Cornelis. (Corresponding author: Marketa Trneckova.)

The authors are with the Department Computer Science, Palacky University, 779 00 Olomouc, Czech Republic (e-mail: radim.belohlavek@acm.org; marketa.trneckova@gmail.com).

Digital Object Identifier 10.1109/TFUZZ.2023.3330760

where \leftrightarrow is the so-called biresiduum, which plays the role of many-valued logical equivalence.¹ The biresiduum measures closeness of truth degrees and is defined by

$$I_{ij} \leftrightarrow J_{ij} = \min(I_{ij} \rightarrow J_{ij}, J_{ij} \rightarrow I_{ij})$$

with \rightarrow being the residuum in L , i.e., the many-valued implication associated with \otimes by $a \rightarrow b = \bigvee \{c \in L \mid a \otimes c \leq b\}$; see [10], [11].

Example 2: For the Łukasiewicz operation \otimes (see above), we have $a \rightarrow b = \min(1, 1 - a + b)$ for any $a, b \in [0, 1]$. One then obtains $a \leftrightarrow b = 1 - |a - b|$, hence $I \approx J$ may be interpreted as the sum of approximate equalities of the corresponding entries in I and J divided by the number of all entries. ■

Remark 1: For $L = \{0, 1\}$, i.e., 0 and 1 are the only truth degrees, our setting becomes the setting of factorization of Boolean matrices. In particular, \otimes and \leftrightarrow become classical conjunction and classical equivalence, respectively, and $A \circ B$ becomes the Boolean matrix product. One easily observes that $I \approx J = 1 - \frac{E(I, J)}{n \cdot m}$, where $E(I, J) = \sum_{i,j=1}^{n,m} |I_{ij} - J_{ij}|$ is the Hamming distance of Boolean matrices I and J . ■

Two particular problems—reflecting two basic views of the factorization problem—are recognized in the literature: the discrete basis problem (DBP_L) and the approximate factorization problem (AFP_L) are as follows.

- DBP_L: for a given matrix $I \in L^{n \times m}$ and a positive integer k , compute matrices $A \in L^{n \times k}$ and $B \in L^{k \times m}$ that maximize $I \approx A \circ B$.
- AFP_L: for a given matrix $I \in L^{n \times m}$ and $\varepsilon \in L$, compute matrices $A \in L^{n \times k}$ and $B \in L^{k \times m}$ with $I \approx A \circ B \geq \varepsilon$ such that k is as small as possible.

In DBP_L, one prescribes the number k of factors and seeks the best possible factorization involving k factors (emphasis is on a good factorization by a limited number of factors); in AFP_L, one prescribes a required precision ε and seeks a factorization with the least number of factors that satisfies the prescribed precision (emphasis is on a very precise factorization).

C. Our Contributions

We provide a significant improvement of the idea of using formal concepts of the input matrix I as factors for factorization of I . The idea of using formal concepts appeared in previous contributions to factorization of matrices with grades and leads to well-behaving factorization algorithms. We argue, however, that such an approach results in an unnecessarily restricted class of factorizations. We analyze the error committed by general factorizations and propose in Section II to use formal concepts not as factors but rather as starting blocks from which factors are computed by an iterative process. The new approach leads to factorizations not restricted in the manner described previously. In Section III we demonstrate that the new approach indeed outperforms the previous approaches. In Section IV we discuss future problems revealed by the present perspective.

¹Alternatively, one may normalize using an appropriately defined size $|I|$ of I , i.e., use $\frac{\sum_{i,j=1}^{n,m} I_{ij} \leftrightarrow J_{ij}}{|I|}$ instead of (3). We use this alternative in our experiments and return to this question later.

II. GRECOND_L+: THE NEW ALGORITHM

A. Preparatory Considerations

1) *Formal Concepts of I as Possible Factors:* Recall first the rationale for using formal concepts of I as factors for decomposition of I . Formal concepts in $I \in L^{n \times m}$ are defined as fixpoints of certain operators associated with the object-attribute matrix I [1]: Let

$$X = \{1, \dots, n\} \text{ and } Y = \{1, \dots, m\}$$

represent objects (rows) and attributes (columns), respectively. A formal concept of I is any pair $\langle C, D \rangle$ of fuzzy sets $C \in L^X$ of objects and $D \in L^Y$ of attributes satisfying $C^\uparrow = D$ and $D^\downarrow = C$, where the operators $\uparrow: L^X \rightarrow L^Y$ and $\downarrow: L^Y \rightarrow L^X$ (called concept-forming operators) are defined as follows:

$$C^\uparrow(j) = \bigwedge_{i \in X} (C(i) \rightarrow I_{ij}) \text{ and } D^\downarrow(i) = \bigwedge_{j \in Y} (D(j) \rightarrow I_{ij})$$

for each $i \in X$ and $j \in Y$. The set of all formal concepts of I is denoted by $\mathcal{B}(X, Y, I)$. The fuzzy sets C and D are interpreted as the extent and the intent of the formal concept D , respectively. Accordingly, $C(i) \in L$ and $D(j) \in L$ are interpreted as the degree to which the concept applies to the object i and the degree to which the attribute j is characteristic (is a manifestation) of the concept.

One easily observes that due to (1), $I \approx A \circ B$ may be rewritten as

$$I \approx A_{\cdot 1} \circ B_{1\cdot} \vee \dots \vee A_{\cdot k} \circ B_{k\cdot}$$

where for each $l = 1, \dots, k$, $A_{\cdot l}$ and $B_{l\cdot}$ denote the l th column of A and the l th row of B , respectively. Consequently, factor l in a decomposition $I \approx A \circ B$ is naturally represented as the pair $\langle A_{\cdot l}, B_{l\cdot} \rangle$.

The rationale of using formal concepts of I as factors consists in the following property proved in [2].

Lemma 1: Let $I \in L^{n \times m}$ and let $I = A \circ B$ be a decomposition involving k factors. Then, there exists a set $\mathcal{F} = \{\langle C_1, D_1 \rangle, \dots, \langle C_K, D_K \rangle\}$ consisting of $K \leq k$ formal concepts of I such that for the matrices $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ defined by $(A_{\mathcal{F}})_{il} = C_l(i)$ and $(B_{\mathcal{F}})_{lj} = D_l(j)$ one has

$$I = A_{\mathcal{F}} \circ B_{\mathcal{F}}.$$

In this sense, formal concepts of I are optimal factors for exact factorizations of I .

2) *Analyzing $I \approx A \circ B$: Undercovering and Overcovering:* When it comes to approximate rather than exact factorizations, formal concepts still provide reasonable results; see e.g., [3], [5], [6]. However, they no longer provide optimal factorizations, since using formal concepts as factors leads to a limited class of factorizations. The reason becomes apparent when analyzing the approximate equality $I \approx A \circ B$ and, in particular, when analyzing the committed error associated with an approximate factorization.

When we desire exact factorization, using formal concepts as factors proved to be beneficial. However, when an approximate factorization is needed, a fundamental limitation of using formal

concepts appears, namely that factorizations using formal concepts never commit overcovering—one of two kinds of error: For factor model (2) and matrix entry $\langle i, j \rangle$, we speak of

uncovering when $I_{ij} > (A \circ B)_{ij}$

and

overcovering when $I_{ij} < (A \circ B)_{ij}$.

For our present purpose, it is more natural to reason in terms of error of decomposition rather than approximate equality (3); see Remark 1. Defining the error $E(I, A \circ B)$ naturally by

$$E(I, A \circ B) = \sum_{i,j=1}^{n,m} (1 - (I_{ij} \leftrightarrow (A \circ B)_{ij})) \quad (4)$$

one obtains

$$I \approx A \circ B = 1 - \frac{E(I, A \circ B)}{nm}.$$

Now, since with our assumptions we have

$$a \leftrightarrow b = \begin{cases} a \rightarrow b & \text{for } a > b, \text{ and} \\ b \rightarrow a & \text{for } a < b, \text{ and} \\ 1 & \text{for } a = b, \end{cases}$$

we obtain

$$E(I, A \circ B) = E_u(I, A \circ B) + E_o(I, A \circ B)$$

where

$$E_u(I, A \circ B) = \sum_{i,j=1, I_{ij} > (A \circ B)_{ij}}^{n,m} (1 - (I_{ij} \rightarrow (A \circ B)_{ij})) \quad (5)$$

and

$$E_o(I, A \circ B) = \sum_{i,j=1, I_{ij} < (A \circ B)_{ij}}^{n,m} (1 - ((A \circ B)_{ij} \rightarrow I_{ij})). \quad (6)$$

Clearly, E_u and E_o are parts of E due to uncovering and overcovering, respectively. Note also that in (5) and (6), the conditions $I_{ij} > (A \circ B)_{ij}$ and $I_{ij} < (A \circ B)_{ij}$ may be omitted, because $1 - (I_{ij} \rightarrow (A \circ B)_{ij}) = 0$ for $I_{ij} \leq (A \circ B)_{ij}$ and $1 - ((A \circ B)_{ij} \rightarrow I_{ij}) = 0$ for $I_{ij} \geq (A \circ B)_{ij}$.

Remark 2: In the Boolean case, i.e., $L = \{0, 1\}$, since $I_{ij} \rightarrow (A \circ B)_{ij}$ and $(A \circ B)_{ij} \rightarrow I_{ij}$ can only be equal to 0 and 1, one observes that E_u is the number of entries $\langle i, j \rangle$ for which $I_{ij} > (A \circ B)_{ij}$ and E_o is the number of entries for which $I_{ij} < (A \circ B)_{ij}$. In the general case involving intermediate grades, the situation is more involved. Namely, $1 - (I_{ij} \rightarrow (A \circ B)_{ij})$ and $1 - ((A \circ B)_{ij} \rightarrow I_{ij})$ represent degrees of uncovering and overcovering, respectively.

The following lemma provides an important observation used in the design of our new algorithm. Namely, as new factors are added, E_u is decreasing while E_o may only increase.

Lemma 2: Let $I \in L^{n \times m}$ be a given object-attribute matrix, $A \in L^{n \times k}$ and $B \in L^{k \times m}$ be an object-factor and factor-attribute matrices, respectively. Let $A' \in L^{n \times (k+1)}$ and $B' \in L^{(k+1) \times m}$ result by adding a column to A and a row to B (i.e.,

adding a $(k+1)$ th factor to the existing k factors). Then

$$E_u(I, A \circ B) \geq E_u(I, A' \circ B')$$

$$E_o(I, A \circ B) \leq E_o(I, A' \circ B').$$

Proof 1: The proof follows by a straightforward verification using the facts that for each $\langle i, j \rangle$, we have $(A \circ B)_{ij} \leq (A' \circ B')_{ij}$ and that for each $a, a', b \in L$ with $a \leq a'$ we have $a \rightarrow b \geq a' \rightarrow b$. ■

B. GRECOND_L + Algorithm

1) *Basic Idea:* Our new algorithm, GRECOND_L+, generates factors one-by-one in a greedy manner that is based on a geometric insight into the factorization problem. This is due to the NP-hardness of the factorization problem. Each factor is generated by a process that has three steps. In the first step, a formal concept $\langle C, D \rangle$ of the input matrix I is obtained by a particular procedure as a basis of the constructed factor; this basis is called the *nucleus* of the constructed factor in what follows. Since, as argued previously, such a formal concept does not commit any overcovering error, which may be severely limiting when approximate factorizations are sought, the second step consists in extending the nucleus to a candidate new factor $\langle C \cup E, D \cup F \rangle$ by adding—to certain degrees—both objects and attributes to C and D . In general, the extended candidate factor $\langle C \cup E, D \cup F \rangle$ commits both uncovering and overcovering error, but commits a smaller overall error compared to its nucleus $\langle C, D \rangle$. The third step is directly motivated by the fact that the overcovering error, E_o , may never decrease when computing factors (see Lemma 2). To alleviate this property, and thus make possible a drop in the overcovering error, the algorithm revisits and modifies the previously generated factors in the third step. The latter idea was first used in an old, generally unknown factorization algorithm 8M [8, pp. 933–945] and was recently utilized in [4].

The above idea is expected to result in an algorithm able to produce factorizations with small uncovering error E_u , because formal concepts, which commit no E_o , are used as substantial parts of the constructed factors, and a relatively small overcovering error E_o , because overcover is allowed only as a part of fine-tuning of the constructed factors.

2) *Detailed Description:* A pseudocode of our algorithm is represented by Algorithms 1 and 2. In the following, we provide a detailed description of the pseudocode. In the algorithm, \mathcal{F} represents the set of constructed factors (i.e., pairs $\langle C, D \rangle \in L^X \times L^Y$) and \mathcal{U} stores the set of all entries $\langle i, j \rangle$ not covered by the factors in that $I_{ij} > (A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij}$.

The factors are consecutively generated within the main loop in Algorithm 1 (l. 2–27). The loop ends when the undercover error, E_u , reaches 0, i.e., when $I_{ij} \leq (A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij}$ for each entry $\langle i, j \rangle$, which corresponds to $\mathcal{U} = \emptyset$. Clearly, the stopping condition may be changed as desired, for example to ensure that $I \approx A_{\mathcal{F}} \circ B_{\mathcal{F}}$ exceeds a prescribed precision threshold ε , corresponding to the AFP_L, or to end the loop after k prescribed factors are generated, corresponding to the DBP_L, cf. the end of Section I-B.

The first step, i.e., computing a formal concept $\langle C, D \rangle$ of I , which shall play the role of a nucleus of the constructed factor (see Section II-B1) proceeds in l. 4–9. The initially empty fuzzy set D is being extended to $D \cup \{^a/j\}$ until the corresponding formal concept $\langle D \cup \{^a/j\}^\downarrow, (D \cup \{^a/j\})^\uparrow \rangle$ stops having a better coverage. This approach is borrowed from the GRECOND_L algorithm [7]. Here

$$D \oplus_a j = \{ \langle k, l \rangle \in \mathcal{U} \mid I_{kl} \leq (D \cup \{^a/j\})^\downarrow(k) \\ \otimes (D \cup \{^a/j\})^\uparrow(l) \}$$

is the number of previously uncovered entries that are covered by the formal concept corresponding to the examined extension $D \cup \{^a/j\}$, and $|D \oplus_a j|$ denoted the number of these entries.

Note at this point that instead of maximizing the number of covered entries in the above-mentioned construction of the nucleus $\langle C, D \rangle$, i.e., instead of minimizing E_u , one could minimize $E_u + E_o$ and thus take both the uncovering and overcovering into account. However, since $\langle C, D \rangle$ is a formal concept, it commits no overcovering, hence minimizing E_u is equivalent to minimizing $E_u + E_o$.

In the second step, the resulting formal concept $\langle C, D \rangle$ is considered as a nucleus, for which its expansion is computed in l. 10 by EXPANSION (Algorithm 2) as explained in the following. The expansion may commit overcovering, which is penalized using a parameter $w > 0$ —a weight of the overcover error. The expanded pair $\langle C \cup E, D \cup F \rangle$ is then stored in \mathcal{F} as a new candidate factor in l. 11.

The extension $\langle E, F \rangle$ of the nucleus $\langle C, D \rangle$ proceeds as follows. One computes the extension F of attributes by starting with empty F (l. 1 in Algorithm 2) and subsequently adding attributes j to degrees a that are higher than the present degrees $(D \cup F)(j)$, until such addition is possible (l. 2–7). When doing so, attribute j and degree a are selected with maximal $gain(\{^a/j\})$. The value $gain(\{^a/j\})$ reflects a possible improvement of the approximate equality $I \approx A \circ B$ in that it rewards positively a decrease of the undercover error E_u and penalizes with weight $w > 0$ the increase in the overcover error E_o .

The situation is conceptually more involved compared to the extension proposed in [4] for the binary case. Namely, one needs to distinguish three cases. For our purpose, let

$$cur_{ij} = (A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij} \vee C_i \otimes (D \cup F)_j$$

i.e., cur_{ij} is the value of entry $\langle i, j \rangle$ of the matrix reconstructed from the factors in \mathcal{F} computed so far and the currently considered factor $\langle C, D \cup F \rangle$. Moreover, let

$$new_{ij} = (A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij} \vee C_i \otimes (D \cup F \cup \{^a/j\})_j$$

i.e., new_{ij} is the analogous value with $\langle C, D \cup F \rangle$ extended to $\langle C, D \cup F \cup \{^a/j\} \rangle$. The three cases to distinguish are as follows.

- 1) $new_{ij} \leq I_{ij}$: In this case, since $cur_{ij} \leq new_{ij}$, the change from cur_{ij} to new_{ij} represents a decrease in E_u . Hence, the entry $\langle i, j \rangle$ contributes to $gain(\{^a/j\})$ by $new_{ij} - cur_{ij}$.

- 2) $cur_{ij} < I_{ij} < new_{ij}$: In this case, the change involves a decrease $I_{ij} - cur_{ij}$ in E_u but also increase $new_{ij} - I_{ij}$ in E_o . The entry $\langle i, j \rangle$ thus contributes to $gain(\{^a/j\})$ by $I_{ij} - cur_{ij} - w(new_{ij} - I_{ij})$.
- 3) $I_{ij} \leq cur_{ij}$: In this case, E_o is increasing by $new_{ij} - cur_{ij}$. The considered entry hence contributes to $gain(\{^a/j\})$ by $-w(new_{ij} - cur_{ij})$.

The abovementioned considerations lead to the following definition of $gain(\{^a/j\})$:

$$gain(\{^a/j\}) = \sum_{i=1, new_{ij} \leq I_{ij}}^n new_{ij} - cur_{ij} \\ + \sum_{i=1, cur_{ij} < I_{ij} < new_{ij}}^n (I_{ij} - cur_{ij}) \\ - w \cdot (new_{ij} - I_{ij}) - w \cdot \sum_{i=1, I_{ij} \leq cur_{ij}}^n new_{ij} - cur_{ij}.$$

The loop in l. 12–15 ensures that all matrix entries covered by this factor are removed from \mathcal{U} .

The third step (l. 16–26) represents the revisiting and modification of the computed candidate factors stored in \mathcal{F} . First, in l. 16–18, if removing $\langle C, D \rangle$ from \mathcal{F} does not make worse the overall error, then $\langle C, D \rangle$ is removed from \mathcal{F} , because it is redundant.

If $\langle C, D \rangle$ may not be removed in l. 16–18, one tries in l. 20–24 to reduce the candidate factor $\langle C, D \rangle$ in that degrees $D(j)$ are reduced to nucleus(D)(j), if possible, i.e., if the coverage due to $D(j)$ is achieved by the factors other than $\langle C, D \rangle$. Reducing $\langle C, D \rangle$ to its nucleus, i.e., to a formal concept of I , is motivated by the fact that a formal concept does not commit any overcovering error E_o , which is a consequence of the following observation.

Lemma 3: For any formal concept $\langle C, D \rangle \in \mathcal{B}(X, Y, I)$ we have $C(i) \otimes D(j) \leq I_{ij}$.

In particular, $D(j)$ gets reduced to nucleus(D)(j) if this reduction does not make worse the overall error.

III. EXPERIMENTAL EVALUATION

We now present an experimental evaluation of the new GRECOND_L+ algorithm, for which purpose we use both synthetic and real data. Using synthetic data makes it possible to test the compared algorithms on data with known characteristics. Real data allow us to assess the compared algorithms in terms of the interpretability of the computed factors. In addition to GRECOND_L+, we present for the purpose of comparison two other factorization algorithms, namely GRECOND_L [7] and ASSO_L [6], [12], which are representative for solving the AFP_L and DBP_L problems, respectively. While the principle of GRECOND_L has been mentioned in Section II-B, ASSO_L proceeds by computing from an input matrix $I \in L^{n \times m}$ a so-called association matrix \mathcal{A} with m columns. To compute a decomposition $I \approx A \circ B$, ASSO_L uses the rows of \mathcal{A} as candidate rows of B . For each such candidate, the best corresponding candidate column of A is computed; the best thus obtained pair

Algorithm 1: GRECOND+.

Input: $n \times m$ matrix I , number w
Output: set \mathcal{F} of factors

```

1  $\mathcal{U} \leftarrow \{\langle i, j \rangle \mid I_{ij} \neq 0\}; \mathcal{F} \leftarrow \emptyset$ 
2 while  $\mathcal{U} \neq \emptyset$  do
3    $D \leftarrow \emptyset; V \leftarrow 0$ 
4   while exists  $\{a/j\} \notin D$  such that  $|D \oplus_a j| > V$  do
5     select  $\{a/j\} \notin D$  that maximizes  $|D \oplus_a j|$ 
6      $D \leftarrow (D \cup \{a/j\})^{\downarrow \uparrow}$ 
7      $V \leftarrow |D \oplus_a j|$ 
8   end
9    $C \leftarrow D^{\downarrow}$ 
10   $\langle E, F \rangle \leftarrow \text{EXPANSION}(\langle C, D \rangle, w)$ 
11  add  $\langle C \cup E, D \cup F \rangle$  to  $\mathcal{F}$ 
12  for  $\langle i, j \rangle \in \mathcal{U}$  do
13    if  $I_{ij} \leq (C \cup E)_i \otimes (D \cup F)_j$  then
14       $\mathcal{U} \leftarrow \mathcal{U} - \langle i, j \rangle$ 
15    end
16  foreach factor  $\langle C, D \rangle \in \mathcal{F}$  do
17    if  $\sum_i \sum_j (I_{ij} \leftrightarrow (A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij}) \leq$   

        $\sum_i \sum_j (I_{ij} \leftrightarrow (A_{\mathcal{F}'} \circ B_{\mathcal{F}'})_{ij})$  where  

        $\mathcal{F}' = \mathcal{F} - \{\langle C, D \rangle\}$  then
18      remove  $\langle C, D \rangle$  from  $\mathcal{F}$ 
19    else
20      foreach  $j$  such that  $D_j > \text{nucleus}(D)_j$  do
21        if  $\sum_i \sum_j (I_{ij} \leftrightarrow (A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij}) \leq$   

          $\sum_i \sum_j (I_{ij} \leftrightarrow (A_{\mathcal{F}'} \circ B_{\mathcal{F}'})_{ij})$  where  

          $\mathcal{F}' = \mathcal{F} - \{\langle C, D \rangle\} \cup \{\langle C, D' \rangle\}$   

         where  $D'_j = \text{nucleus}(D)_j$  and  

          $D'_k = D_k$  for all  $k \neq j$  then
22           $D_j \leftarrow \text{nucleus}(D)_j$ 
23        end
24      end
25    end
26  end
27 end
28 return  $\mathcal{F}$ 

```

Algorithm 2: EXPANSION.

Input: pair $\langle C, D \rangle$, number w
Output: expansion $\langle E, F \rangle$ of $\langle C, D \rangle$

```

1  $E \leftarrow \emptyset; F \leftarrow \emptyset$ 
2 repeat
3   select column  $j$  and  $a \in L$  s. t.  $(D \cup F)_j < a$   

    maximizing  $\text{gain}(\{a/j\})$ 
4   if  $\text{gain}(\{a/j\}) > 0$  then
5     add  $\{a/j\}$  to  $F$ 
6   end
7 until  $F$  did not change
8 return  $\langle E, F \rangle$ 

```

of candidates is then added as a new column of A and a new row of B .

In order to describe performance of the observed algorithms, we utilize the error $E(I, A_{\mathcal{F}} \circ B_{\mathcal{F}})$, see (4), which describes closeness of the input $n \times m$ matrix I and the matrix $A_{\mathcal{F}} \circ B_{\mathcal{F}}$ reconstructed from the set \mathcal{F} of computed factors with $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ defined as in Lemma 1. For the reasons mentioned in Section II-A2, we also use the uncover error E_u and the overcover error E_o , see (5) and (6), respectively.

In particular, we use the normalized versions of the errors

$$e(I, A_{\mathcal{F}} \circ B_{\mathcal{F}}) = \frac{E(I, A_{\mathcal{F}} \circ B_{\mathcal{F}})}{|I|}$$

$$e_u(I, A_{\mathcal{F}} \circ B_{\mathcal{F}}) = \frac{E_u(I, A_{\mathcal{F}} \circ B_{\mathcal{F}})}{|I|}$$

$$e_o(I, A_{\mathcal{F}} \circ B_{\mathcal{F}}) = \frac{E_o(I, A_{\mathcal{F}} \circ B_{\mathcal{F}})}{|I|}$$

where $|I|$ is defined by

$$|I| = \sum_{i,j=1}^{n,m} I_{ij}. \quad (7)$$

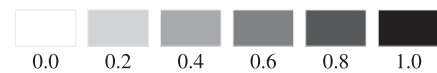
If I is considered as a fuzzy relation, $|I|$ is the so-called scalar cardinality of this fuzzy relation. In the Boolean case, i.e., if $L = \{0, 1\}$, $|I|$ is the number of 1s in I and in the general case $|I|$ measures the size of the incidence relation represented by I . Clearly, we have $e = e_u + e_o$. The reason for normalizing by $|I|$ rather than by $n \cdot m$ is that for sparse matrices, normalization by $n \cdot m$ results in a very small values of relative error and tends to be less informative.

We typically observe the previous kinds of error for sets \mathcal{F} consisting of the first computed factor, the first two computed factors, etc., up to a set consisting of k computed factors, where k is chosen, as described in the text.

A. Illustrative Example

We start with an illustrative example with data originally presented in [6]. The purpose is to illustrate the typical features of the factorization algorithms that we compare in the next sections, as well as to demonstrate a general usefulness of factorizations.

The input data, presented in Table I, describe five most popular dog breeds and their 11 attributes scaled to the matrix with the degrees taken from $L = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$.² Note that the full set of 151 breeds is explored later. We consider the Łukasiewicz operation \otimes on L (see Example 1) in our experiments due to their known convenient properties. For the purpose of visualization, we represent the grades in L by shades of gray as follows.



The degrees may naturally be assigned linguistic labels: “not at all” to 0, “somewhat” to 0.2, “rather not” to 0.4, “rather yes” to 0.6, “almost fully” to 0.8, and “fully” to 1.

²[Online]. Available: <http://www.petfinder.com/>

TABLE I
FIVE MOST POPULAR DOG BREEDS

	Energy	Playfulness	Friendliness toward dogs	Friendliness toward strangers	Friendliness toward other pets	Protection ability	Exercise	Affection	Ease of training	Watchdog ability	Grooming
Labrador Retriever	0.8	1.0	0.8	1.0	1.0	0.4	0.6	1.0	1.0	0.8	0.4
Golden Retriever	0.6	1.0	1.0	1.0	1.0	0.4	0.6	1.0	1.0	0.6	0.6
Yorkshire Terrier	0.8	0.8	0.4	0.6	0.4	0.2	0.2	0.6	0.4	1.0	0.8
German Shepherd	0.6	0.4	0.2	0.4	0.6	1.0	0.8	0.6	1.0	1.0	0.4
Beagle	0.6	0.6	1.0	1.0	1.0	0.2	0.6	1.0	0.2	0.8	0.2

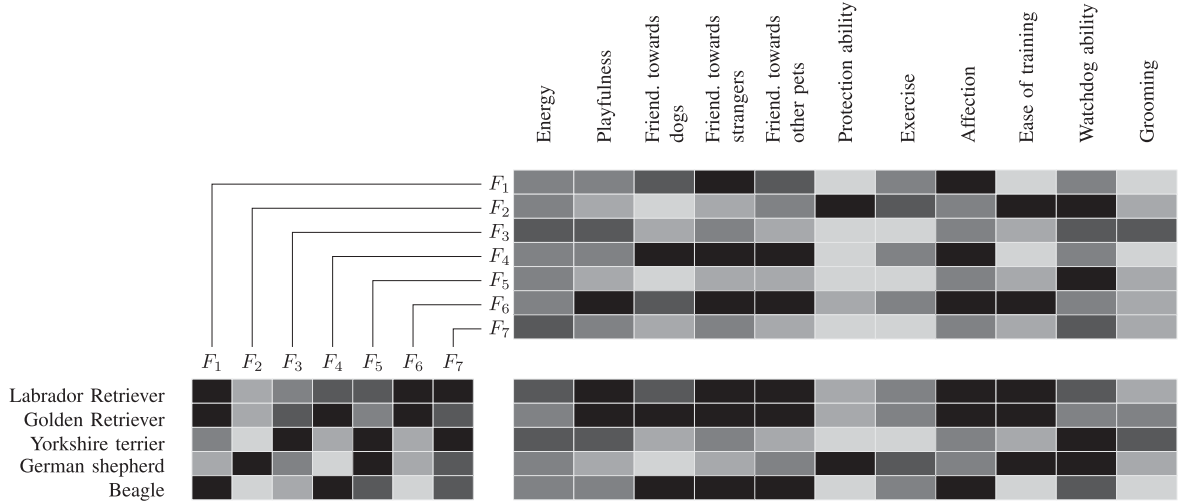


Fig. 1. GRECOND_L: Decomposition $I = A_F \circ B_F$. I , A_F , and B_F are the bottom-right, bottom-left, and top matrix, respectively.

The data in Table I may thus be represented by a 5×11 object-attribute matrix I . This matrix, along with its decompositions into A_F and B_F into the object-factor and factor-attribute matrices A_F and B_F , which are computed by the observed algorithms, are shown below in this section.

In the following visualization, every factor F_l is represented by the l th column in A_F and the l th row in B_F , as usual in the literature. The entry $(A_F)_{il}$ indicates the degree to which factor l applies to breed i , while $(B_F)_{lj}$ represents the degree to which attribute j is a particular manifestation of factor l ; cf. Lemma 1.

1) *Results for GRECOND_L*: The results of factorizing I by the GRECOND_L algorithm are displayed in Fig. 1.

The first computed factor, F_1 , is manifested significantly by the three kinds of “Friendliness” and “Affection” (attributes with high degrees in the first row of B_F) and applies in particular to Labrador, Golden Retriever, and Beagle (breeds with high degrees in the first column of A_F), and to some extent also to Yorkshire. The factor may hence be termed *friendliness*. On the other hand, the three attributes with the highest degree in the row of F_2 and a high degree of “Exercise” tell us that this factor is naturally interpreted as *guardian dog*. The corresponding column shows that F_2 applies to German Shepherd and separates this breed clearly from the other breeds. Factor F_3 may be interpreted as *dogs suitable for kids*, because it is manifested to a

great degree by “Friendliness,” “Playfulness,” “Affection,” and “Ease of training,” and applies to Golden Retriever (to degree 1) and Labrador Retriever (to degree 0.8).

The relative errors e committed by the first factor, the first two factors, and the first three factors are 0.26, 0.14, and 0.07, respectively. Since GRECOND_L does not commit overcovering, e_o remains 0. If we let the algorithm compute further factors, we obtain further error values 0.05, 0.04, 0.01, and 0, i.e., an exact factorization is obtained in the end.

2) *Results for ASSO_L*: In Fig. 2, we present the first three factors obtained by ASSO_L when the parameters of this algorithm are set in such a way that the uncover and overcover arrow obtain the same significance (e.g., by $w_0 = 1$ and $w_1 = 1$), and the parameter controlling the matrix of associations is set to $\tau = 0.9$.

The first factor, the first two, and the first three factors have the relative error e equal to 0.28, 0.16, and 0.156, respectively. The relative overcovering error e_o obtains 0.0073, 0.0327, and 0.04, respectively. While the error is reasonably small, a problem, which often appears with ASSO_L, is that the obtained factors are difficult to interpret. For example, the first factor (i.e., the most important from the point of view of the method) does not carry any important information. Such factors tend to appear in ASSO_L because when $|L| > 2$ (non-Boolean case), the factors correspond to rectangles with values “around the middle” in L , such as 0.4 and 0.6. Such factors have a small error even though

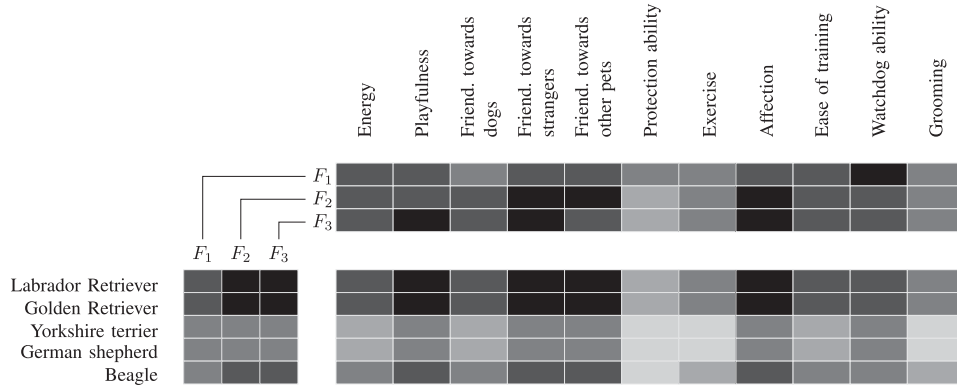


Fig. 2. ASSO_L: Decomposition $I \approx A_F \circ B_F$. $A_F \circ B_F$, A_F , and B_F are the bottom-right, bottom-left, and top matrix, respectively.

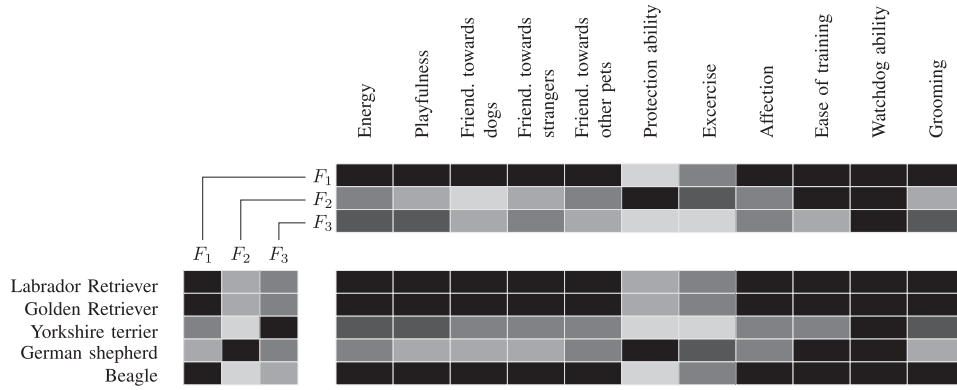


Fig. 3. GRECOND_{L+}, $w = 0.5$: Decomposition $I \approx A_F \circ B_F$. $A_F \circ B_F$, A_F , and B_F are the bottom-right, bottom-left, and top matrix, respectively.

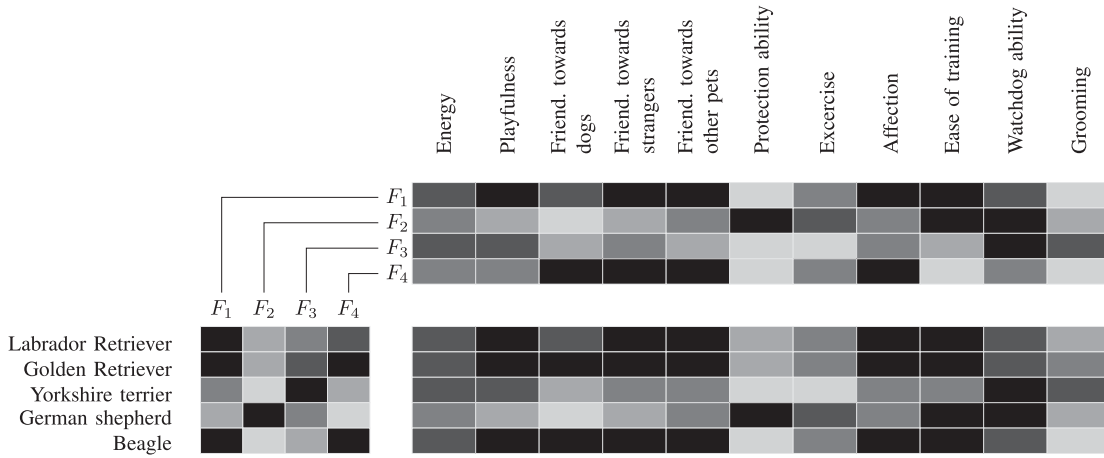


Fig. 4. GRECOND_{L+}, $w = 1$: Decomposition $I \approx A_F \circ B_F$. $A_F \circ B_F$, A_F , and B_F are the bottom-right, bottom-left, and top matrix, respectively.

they are difficult to interpret; for more details of this property see [6].

3) *Results for GRECOND_{L+}*: We computed factorizations for various values of the parameter w . For instance, with $w = 0.5$ we obtained three factors with the error e of the first factor, the first two factors, and the first three factors being 0.2, 0.12, and 0.1, respectively, and the corresponding e_o being (when rounded off) 0.1, 0.1, and 0.1, respectively. With these three factors, the uncover error e_u drops to 0. With $w = 1$, we obtain four factors with which we eventually obtain $e_u = 0$.

The error e of the first factor, the first two factors, the first three factors, and the first four factors equals 0.19, 0.09, 0.05, and 0.04, and the corresponding e_o being (when rounded off) equal to 0.04, 0.04, 0.04, and 0.04. We thus observe that for both values of w , the values of e are reasonably small. Nevertheless, the higher value of w makes the overcover e_o smaller, which is in accordance with the intended role of w .

The computed factorizations for $w = 0.5$ and $w = 1$ are shown in Figs. 3 and 4, respectively. As we can see in these figures, the computed factors not only reasonably explain

the input data in terms of having a high coverage but also are easily interpretable and provide natural groupings of the dog breeds and their attributes. For instance, the first two factors produced both for $w = 0.5$ and $w = 1$ may be termed *friendliness* and *guardian dog* and are similar to the corresponding factors obtained by GRECOND_L.

This introductory example indicates that while both ASSO_L and GRECOND_L produce general decompositions, i.e., may commit both uncovering and overcovering, and produce reasonably precise decompositions, GRECOND_L does not suffer the difficulty when interpreting factors, often encountered with ASSO_L. Moreover, as we shall see, unlike ASSO_L, GRECOND_L is able to compute very precise decompositions. In addition, the example indicates that GRECOND_L provides very precise factorizations with smaller error when compared to GRECOND_L. These properties shall be examined in the rest of this section.

B. Synthetic Data

In our experiments, we prescribe a value ε of error and observe the minimum number of factors produced by a particular examined algorithm, which are necessary to obtain a factorization whose error does not exceed ε . The prescribed values of error and the obtained numbers of factors are provided in the tables as follows.

In particular, we prescribe the part e_u of the overall relative error e , and observe the minimal number of factors needed to obtain a factorization whose e_u does not exceed the prescribed value ε , for which factorization we display the actual errors e_u , e_o , and e .

Our reason to prescribe a desired value of e_u , rather than e , consists in the fact that e_u represents the extent of data yet unexplained by the considered factorization. It also corresponds to the purpose of our new algorithm, i.e., to explain all of the input data or almost all of the input data (i.e., achieve small e_u) with a reasonably small number of factors, while possibly committing a small e_o .

Remark 3: Prescribing e_u corresponds to a realistic scenario, which we now describe in the Boolean setting (the kind of application we use naturally extends to a setting with fuzzy attributes). Suppose that $I_{ij} = 1/0$ denotes that person i has/has not passed test j . The tests may involve various parts regarding mathematical reasoning, logical reasoning, language proficiency, factual knowledge, etc. The discovered factors are expected to correspond to general skills, such as logical skills, verbal intelligence, etc. A natural goal is to obtain a factorization with $e_u = 0$ because nonzero e_u indicates that certain successful results in the tests are left unexplained. Committing e_o does not matter that much because it might have been the case that $I_{ij} = 0$ because person i made a silly mistake but actually has the skills needed to solve test j . While formally symmetric, e_u is considerably more important than e_o in this kind of application. This is why it is compelling to prescribe a desired value of e_u rather than e or e_o .

We used synthetic data with varying size and other parameters. The data we chose to display in our experiments were organized

TABLE II
SYNTHETIC DATA

dataset	size	$ L $	k	avg $ I > 0 $	avg $ I $
Set 1	50×50	5	10	2493	2085
Set 2	100×100	5	20	10000	9275
Set 3	200×100	11	25	20000	16413
Set 4	200×100	11	30	19998	14677

in collections denoted Sets 1–4, each consisting of 500 $n \times m$ matrices. The characteristics of these datasets are described in Table II. Every matrix is obtained as a product of $n \times k$ and $k \times m$ randomly generated matrices A and B in which entries from scale L are selected according to a prescribed probability distribution. For instance, in Set 1 we used a five-element scale $L = \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$ with the probabilities $p(a)$ of the degrees $a \in L$ in A and B being $p(0) = p(\frac{1}{4}) = \frac{1}{8}$ and $p(\frac{1}{2}) = p(\frac{3}{4}) = p(1) = \frac{1}{4}$. The same distribution is used for Set 2, while for Set 3 and Set 4 we used distributions represented by vectors

$$\left[\frac{1}{11} \frac{1}{11} \frac{1}{11} \frac{1}{11} \frac{1}{11} \frac{1}{11} \frac{1}{11} \frac{1}{11} \frac{1}{11} \frac{1}{11} \frac{1}{11} \right]$$

and

$$\left[\frac{2}{17} \frac{2}{17} \frac{2}{17} \frac{2}{17} \frac{2}{17} \frac{2}{17} \frac{2}{17} \frac{1}{17} \frac{1}{17} \frac{1}{17} \frac{1}{17} \right]$$

respectively. The probability distributions generalize the commonly considered densities of Boolean matrices, e.g., for $L = \{0, 1\}$ the distribution $[\frac{1}{4} \frac{3}{4}]$ corresponds to density 0.75. The table also contains the average number of nonzero entries (avg $|I > 0|$) and average scalar cardinality (avg $|I|$) over all matrices in the sets.

The results of our experiments with datasets Sets 1–4 are displayed in Table III. As indicated previously, we present for each Set i and for the prescribed values $\varepsilon = 0.5, \dots, 0.00$ of e_u the observed quantities for each algorithm we examined, namely the smallest number of factors needed to obtain a factorization whose e_u does not exceed ε under which we display the actual values of e_u , e_o , and e for this factorization. For each Set i , all the observed numbers are actually averages over the 500 matrices in the text; the number of factors is moreover rounded to integers. Thus for instance, for Set 2, and $\varepsilon = 0.1$, GRECOND_L with $w = 0.5$ needs on average three factors, for which the factorization obtains $e_u = 0.073$, $e_o = 0.034$, and hence $e = e_u + e_o \approx 0.108$.

One can see from the results in Table III that while ASSO_L, which we ran with its parameters $w_0 = w_1 = 1$ and $\tau = 0.8$, achieves the prescribed coverage with just one factor, it commits a considerably large overcover error compared to the other algorithms. Moreover, as in the case of the first factor described in Section III-A2, a problem with this one factor is interpretability due to a typically “flat shape” of the factor, as described in that section. This confirms previous studies [6] observing that ASSO_L provides good dimensionality reduction at the expense of mediocre interpretability.

In comparison, GRECOND_L yields factorizations, which are of expected quality and size given the previous studies. As apparent from the table, the new algorithm GRECOND_L+, which may be

TABLE III
NUMBERS OF FACTORS NEEDED FOR PRESCRIBED VALUES OF e_u (SYNTHETIC DATA)

dataset	e_u	ASSO _L	GRECOND _L	GRECOND _L + ($w = 0.5$)	GRECOND _L + ($w = 1$)	GRECOND _L + ($w = 4$)
Set 1	0.50	1 (0.000, 0.193, 0.193)	1 (0.445, 0.000, 0.445)	1 (0.339, 0.020, 0.359)	1 (0.357, 0.015, 0.373)	1 (0.435, 0.002, 0.437)
	0.20	1 (0.000, 0.193, 0.193)	3 (0.184, 0.000, 0.184)	2 (0.187, 0.030, 0.217)	3 (0.142, 0.021, 0.162)	3 (0.180, 0.002, 0.182)
	0.15	1 (0.000, 0.193, 0.193)	4 (0.128, 0.000, 0.128)	3 (0.120, 0.035, 0.155)	3 (0.142, 0.021, 0.162)	4 (0.127, 0.002, 0.129)
	0.10	1 (0.000, 0.193, 0.193)	5 (0.091, 0.000, 0.091)	4 (0.082, 0.038, 0.120)	4 (0.099, 0.021, 0.121)	5 (0.089, 0.002, 0.091)
	0.05	1 (0.000, 0.193, 0.193)	7 (0.043, 0.000, 0.043)	6 (0.040, 0.040, 0.080)	6 (0.050, 0.022, 0.071)	7 (0.042, 0.002, 0.044)
	0.00	1 (0.000, 0.193, 0.193)	12 (0.000, 0.000, 0.000)	12 (0.000, 0.040, 0.041)	12 (0.000, 0.022, 0.022)	12 (0.000, 0.002, 0.002)
Set 2	0.50	1 (0.000, 0.078, 0.078)	1 (0.367, 0.000, 0.367)	1 (0.178, 0.021, 0.198)	1 (0.193, 0.019, 0.213)	1 (0.269, 0.008, 0.277)
	0.20	1 (0.000, 0.078, 0.078)	3 (0.199, 0.000, 0.199)	1 (0.178, 0.021, 0.198)	1 (0.193, 0.019, 0.213)	2 (0.192, 0.009, 0.201)
	0.15	1 (0.000, 0.078, 0.078)	5 (0.136, 0.000, 0.136)	2 (0.109, 0.028, 0.137)	2 (0.122, 0.025, 0.146)	4 (0.126, 0.009, 0.135)
	0.10	1 (0.000, 0.078, 0.078)	7 (0.100, 0.000, 0.100)	3 (0.073, 0.034, 0.108)	3 (0.086, 0.028, 0.115)	6 (0.092, 0.010, 0.101)
	0.05	1 (0.000, 0.078, 0.078)	12 (0.049, 0.000, 0.049)	5 (0.039, 0.042, 0.080)	6 (0.044, 0.033, 0.077)	11 (0.045, 0.010, 0.055)
	0.00	1 (0.000, 0.078, 0.078)	29 (0.000, 0.000, 0.000)	22 (0.000, 0.047, 0.048)	29 (0.000, 0.035, 0.035)	29 (0.000, 0.010, 0.010)
Set 3	0.50	1 (0.000, 0.210, 0.210)	2 (0.398, 0.000, 0.398)	1 (0.418, 0.012, 0.430)	1 (0.417, 0.013, 0.430)	1 (0.463, 0.006, 0.468)
	0.20	1 (0.000, 0.210, 0.210)	6 (0.199, 0.000, 0.199)	3 (0.188, 0.032, 0.220)	4 (0.166, 0.029, 0.195)	5 (0.190, 0.008, 0.198)
	0.15	1 (0.000, 0.210, 0.210)	9 (0.136, 0.000, 0.136)	4 (0.146, 0.039, 0.185)	5 (0.140, 0.030, 0.170)	7 (0.145, 0.008, 0.153)
	0.10	1 (0.000, 0.210, 0.210)	12 (0.097, 0.000, 0.097)	6 (0.094, 0.051, 0.145)	8 (0.094, 0.033, 0.127)	10 (0.100, 0.008, 0.108)
	0.05	1 (0.000, 0.210, 0.210)	18 (0.048, 0.000, 0.048)	11 (0.046, 0.059, 0.105)	14 (0.044, 0.036, 0.080)	16 (0.049, 0.008, 0.057)
	0.00	1 (0.000, 0.210, 0.210)	37 (0.000, 0.000, 0.000)	24 (0.000, 0.064, 0.064)	26 (0.000, 0.038, 0.042)	31 (0.000, 0.009, 0.009)
Set 4	0.50	1 (0.001, 0.329, 0.330)	3 (0.461, 0.000, 0.461)	1 (0.324, 0.081, 0.405)	1 (0.479, 0.026, 0.505)	3 (0.450, 0.003, 0.453)
	0.20	1 (0.001, 0.329, 0.330)	11 (0.194, 0.000, 0.194)	4 (0.172, 0.096, 0.268)	8 (0.183, 0.036, 0.219)	10 (0.196, 0.003, 0.200)
	0.15	1 (0.001, 0.329, 0.330)	14 (0.144, 0.000, 0.144)	5 (0.141, 0.101, 0.242)	10 (0.147, 0.038, 0.185)	13 (0.143, 0.003, 0.146)
	0.10	1 (0.001, 0.329, 0.330)	18 (0.099, 0.000, 0.099)	8 (0.090, 0.108, 0.198)	14 (0.098, 0.040, 0.138)	17 (0.097, 0.003, 0.101)
	0.05	1 (0.001, 0.329, 0.330)	24 (0.049, 0.000, 0.049)	13 (0.048, 0.113, 0.161)	21 (0.046, 0.043, 0.089)	23 (0.048, 0.004, 0.052)
	0.00	1 (0.001, 0.329, 0.330)	37 (0.000, 0.000, 0.000)	28 (0.000, 0.117, 0.117)	33 (0.000, 0.044, 0.044)	34 (0.000, 0.004, 0.004)

TABLE IV
REAL DATA

dataset	size	$ L $	$ I > 0 $	$ I $
Breeds	151×11	6	1963	1177
Decathlon	28×10	5	266	147
IPAQ	4510×16	3	41624	36296
Music	900×26	7	20377	4579
Rio	87×31	4	402	288

regarded as an extension of GRECOND_L, actually outperforms GRECOND_L. First, it needs less factors in most cases to achieve the prescribed e_u , i.e., the prescribed extent to which it explains the data. Second, while GRECOND_L commits no overcover error e_o due to its sole usage of formal concepts as factors, the usage of extensions of formal concepts as factors by GRECOND_L+ leads to overcover error e_o , but this error is reasonably small. Observe also that larger w typically leads to smaller e_o , which is the intended role of w . The ability of GRECOND_L+ to provide factorizations with a small number of factors compared to GRECOND_L may even be observed for very small e_u , i.e., for factorizations that are required to explain the input data to a great extent. This is an interesting property given that GRECOND_L is particularly designed for exact and almost exact factorizations.

C. Real Data

In this section we present factorization results for selected real data, which are described in Table IV. In this table, $|L|$ denotes the number of truth degrees in the scale L for the particular data, $|I > 0|$ denotes the number of nonzero entries in the data matrix I , and $|I|$ denotes the scalar cardinality (7).

Dog breeds is a data,³ which describes 151 popular breeds using 11 attributes regarding the breed characteristics, such as

“Playfulness,” “Friendliness toward dogs,” “Friendliness toward strangers,” “Protection ability,” “Watchdog ability,” “Ease of training,” and the like. *Decathlon* is a data⁴ describing the performance of 28 athletes in the ten disciplines of decathlon in the 2004 Olympics. The *IPAQ* data⁵ describes results of an international questionnaire examining physical activity of population. It involves 4510 respondents and 16 questions, which include questions regarding respondents’ age, sex, sports activity, walking activity, health, body mass index, and the like. The *Music* data comes from an inquiry of how people perceive the speed of song [9]. In particular, the aim was to examine how the perception of speed of a given song depends on various features of the song. The *Rio* data⁶ represents countries and their success in terms of obtained medals at the 2016 Olympics in Rio de Janeiro.

As regards the meaning of computed factors, the results confirm that the factors obtained by ASSO_L are generally not easy to interpret compared to those obtained by GRECOND_L and GRECOND_L+. This again results from a good interpretability of formal concepts, which are utilized by GRECOND_L and GRECOND_L+. We also observed that the factors obtained by GRECOND_L and GRECOND_L+ are often similar.

The results of our experiments are displayed in Table V, which is organized like the above table for synthetic data. That is, the tables display for the various prescribed values of e_u the number of factors computed by the examined algorithms that are needed for the actual e_u not to exceed the prescribed value, as well as the actual values e_u , e_o , and e of relative error. As with the synthetic data, the first factor produced by ASSO_L tends to have

⁴[Online]. Available: <http://www.sports-reference.com/>

⁵[Online]. Available: <http://www.ipaq.ki.se/>

⁶[Online]. Available: <https://www.rio2016.com/en/medal-count>

³[Online]. Available: <http://www.petfinder.com/>

TABLE V
NUMBERS OF FACTORS NEEDED FOR PRESCRIBED VALUES OF e_u (REAL DATA)

dataset	e_u	ASSO _L	GRECOND _L	GRECOND _L + ($w = 0.5$)	GRECOND _L + ($w = 1$)	GRECOND _L + ($w = 4$)
Breeds	0.50	1 (0.055, 0.291, 0.345)	2 (0.429, 0.000, 0.429)	4 (0.496, 0.000, 0.496)	5 (0.412, 0.000, 0.412)	4 (0.464, 0.000, 0.464)
	0.20	1 (0.055, 0.291, 0.345)	6 (0.172, 0.000, 0.172)	9 (0.171, 0.000, 0.171)	9 (0.161, 0.000, 0.161)	9 (0.172, 0.000, 0.172)
	0.15	1 (0.055, 0.291, 0.345)	7 (0.145, 0.000, 0.145)	10 (0.135, 0.000, 0.135)	10 (0.111, 0.000, 0.111)	10 (0.123, 0.000, 0.123)
	0.10	1 (0.055, 0.291, 0.345)	9 (0.096, 0.000, 0.096)	11 (0.093, 0.000, 0.093)	11 (0.082, 0.000, 0.082)	11 (0.082, 0.000, 0.082)
	0.05	2 (0.041, 0.293, 0.334)	13 (0.044, 0.000, 0.044)	12 (0.044, 0.000, 0.044)	12 (0.042, 0.000, 0.042)	12 (0.043, 0.000, 0.043)
	0.00	—	25 (0.000, 0.000, 0.000)	13 (0.000, 0.000, 0.000)	15 (0.000, 0.000, 0.000)	15 (0.000, 0.000, 0.000)
Decathlon	0.50	1 (0.042, 0.513, 0.555)	2 (0.375, 0.000, 0.375)	1 (0.463, 0.070, 0.533)	3 (0.440, 0.000, 0.440)	3 (0.363, 0.000, 0.363)
	0.20	1 (0.042, 0.513, 0.555)	5 (0.183, 0.000, 0.183)	4 (0.185, 0.114, 0.299)	6 (0.195, 0.000, 0.195)	6 (0.194, 0.000, 0.194)
	0.15	1 (0.042, 0.513, 0.555)	7 (0.126, 0.000, 0.126)	5 (0.124, 0.115, 0.239)	7 (0.146, 0.000, 0.146)	8 (0.136, 0.000, 0.136)
	0.10	1 (0.042, 0.513, 0.555)	8 (0.088, 0.000, 0.088)	6 (0.061, 0.117, 0.178)	8 (0.095, 0.000, 0.095)	9 (0.087, 0.000, 0.087)
	0.05	1 (0.042, 0.513, 0.555)	10 (0.046, 0.000, 0.046)	7 (0.049, 0.117, 0.166)	10 (0.000, 0.000, 0.000)	13 (0.000, 0.000, 0.000)
	0.00	—	15 (0.000, 0.000, 0.000)	11 (0.000, 0.117, 0.117)	10 (0.000, 0.000, 0.000)	13 (0.000, 0.000, 0.000)
IPAQ	0.50	1 (0.418, 0.228, 0.646)	5 (0.460, 0.000, 0.460)	8 (0.387, 0.073, 0.459)	9 (0.453, 0.000, 0.453)	10 (0.402, 0.000, 0.402)
	0.20	8 (0.189, 0.239, 0.428)	11 (0.195, 0.000, 0.195)	11 (0.168, 0.135, 0.302)	15 (0.168, 0.000, 0.168)	14 (0.120, 0.000, 0.120)
	0.15	—	13 (0.131, 0.000, 0.131)	12 (0.145, 0.135, 0.280)	16 (0.084, 0.000, 0.084)	14 (0.120, 0.000, 0.120)
	0.10	—	15 (0.090, 0.000, 0.090)	13 (0.061, 0.135, 0.196)	16 (0.084, 0.000, 0.084)	15 (0.058, 0.000, 0.058)
	0.05	—	18 (0.047, 0.000, 0.047)	14 (0.000, 0.135, 0.135)	17 (0.000, 0.000, 0.000)	16 (0.000, 0.000, 0.000)
	0.00	—	31 (0.000, 0.000, 0.000)	14 (0.000, 0.135, 0.135)	17 (0.000, 0.000, 0.000)	16 (0.000, 0.000, 0.000)
Music	0.50	10 (0.475, 0.231, 0.706)	10 (0.500, 0.000, 0.500)	11 (0.466, 0.000, 0.466)	10 (0.499, 0.000, 0.499)	10 (0.500, 0.000, 0.500)
	0.20	—	17 (0.189, 0.000, 0.189)	18 (0.180, 0.000, 0.180)	17 (0.189, 0.000, 0.189)	17 (0.189, 0.000, 0.189)
	0.15	—	18 (0.149, 0.000, 0.149)	20 (0.144, 0.000, 0.144)	18 (0.149, 0.000, 0.149)	18 (0.149, 0.000, 0.149)
	0.10	—	20 (0.091, 0.000, 0.091)	21 (0.096, 0.000, 0.096)	20 (0.091, 0.000, 0.091)	20 (0.091, 0.000, 0.091)
	0.05	—	23 (0.037, 0.000, 0.037)	23 (0.037, 0.000, 0.037)	23 (0.037, 0.000, 0.037)	23 (0.037, 0.000, 0.037)
	0.00	—	26 (0.000, 0.000, 0.000)	26 (0.000, 0.000, 0.000)	26 (0.000, 0.000, 0.000)	26 (0.000, 0.000, 0.000)
Rio	0.50	8 (0.468, 0.141, 0.609)	8 (0.499, 0.000, 0.499)	9 (0.462, 0.000, 0.462)	9 (0.476, 0.000, 0.476)	8 (0.499, 0.000, 0.499)
	0.20	—	16 (0.188, 0.000, 0.188)	19 (0.193, 0.005, 0.198)	16 (0.193, 0.000, 0.193)	18 (0.191, 0.000, 0.191)
	0.15	—	19 (0.135, 0.000, 0.135)	20 (0.144, 0.005, 0.148)	19 (0.139, 0.000, 0.139)	21 (0.135, 0.000, 0.135)
	0.10	—	22 (0.094, 0.000, 0.094)	24 (0.088, 0.005, 0.093)	23 (0.086, 0.000, 0.086)	25 (0.089, 0.000, 0.089)
	0.05	—	26 (0.050, 0.000, 0.050)	26 (0.050, 0.005, 0.054)	26 (0.050, 0.000, 0.050)	28 (0.031, 0.000, 0.031)
	0.00	—	35 (0.000, 0.000, 0.000)	31 (0.000, 0.005, 0.005)	32 (0.000, 0.000, 0.000)	32 (0.000, 0.000, 0.000)

a relatively small error. In addition to mediocre interpretability of the first factor, the real data—which has naturally a rather different character compared to our synthetic data—reveals another shortcoming of ASSO_L, namely, the inability to produce factorizations with low prescribed error e_o . This is because with ASSO_L, the overcover error e_o may only increase as new factors are computed, which does not allow the algorithm to add further factors, which would possibly decrease the uncover error e_u : Decreasing e_u would take place on the expense of increasing e_o , which move would result in the increase of the total error e , and is hence not permitted by the algorithm. This shows that ASSO_L not only suffers from mediocre interpretability of its first factors but also from the inability to produce precise factorizations.⁷ GRECOND_L+ displays a similar behavior if related to GRECOND_L, even though for real data, the numbers of needed factors produced by GRECOND_L are smaller compared to GRECOND_L+ more often than what we observed for the synthetic data. Nevertheless, we observe a nice behavior of our new algorithm regarding how the error evolves: Namely, as new factors are computed, both parts of error e_u and e_o tend to decrease, in most cases to the point when both e_u and e_o , and hence the overall e , drop to zero. We consider this a rather interesting property, not envisioned when we have been designing the algorithm. The resulting factorization by GRECOND_L+ is hence often an exact factorization which, interestingly, involves a smaller (sometimes significantly smaller) number of factors compared to the exact factorization obtained by GRECOND_L. Note at this point that, as is well known, GRECOND_L is capable to compute an exact factorization for each input matrix I . Note also that the sometimes significantly smaller number of

factors obtained by GRECOND_L+ is partly due to the fact that GRECOND_L does not perform any test of possible reducibility of the computed factors. That is to say, GRECOND_L itself can partly be improved by revisiting all the computed factors once exact factorization is computed and removing the factors that are not needed in face of all the factors generated (remove a factor when the remaining factors commit the same error as the original set of factors).

D. Improvement to the Binary Case

While studying the algorithm GRECOND+ from [4], which is designed for binary data, we made several observations some of which are reported in this section. For one, we suggest an improvement to the original algorithm if small overcover error is desired. Second, we present a comparison of GRECOND+ with the basic GRECOND algorithm, which is of interest because GRECOND+ is actually an extension of GRECOND.

Note first that since our new algorithm works for matrices with entries from a general scale L , setting $L = \{0, 1\}$, we obtain an algorithm for binary (Boolean) data, which may be compared to the original GRECOND+ designed in [4]. In this regard, our new algorithm—while inspired by the one described in [4]—actually uses a different way to remove an attribute j from D . The reason we use this different way is that it leads to factorizations with a smaller overcover error, which we find important. In particular, 1. 20–24 of GRECOND_L+, when performed in the binary case, is different from the corresponding procedure described in [4]. The procedure described in [4] removes j from D if each 1 in the column j that appears in some of the rows corresponding to C is covered by some $\langle G, H \rangle \in \mathcal{F} - \{\langle C, D \rangle\}$, i.e., if for each $\langle i, j \rangle \in C \times D$ with $I_{ij} = 1$ there exists $\langle G, H \rangle \in \mathcal{F} - \{\langle C, D \rangle\}$ with $\langle i, j \rangle \in G \times H$. Our approach, when translated to the binary

⁷The strength of ASSO_L consists in providing a very good reduction of dimensionality by its first few factors, which is a significant property when interpretability of factors is not of primary concern.

TABLE VI
NUMBERS OF FACTOR NEEDED FOR PRESCRIBED e_u (BINARY DATA)

dataset	e_u	GRECOND+ ($w = 0.5$)	GRECOND _L + ($w = 0.5$)	GRECOND+ ($w = 4$)	GRECOND _L + ($w = 4$)	GRECOND
Firewall	0.75	1 (0.279, 0.022, 0.301)	1 (0.310, 0.016, 0.326)	1 (0.311, 0.004, 0.315)	1 (0.339, 0.000, 0.340)	1 (0.349, 0.000, 0.349)
	0.50	1 (0.279, 0.022, 0.301)	1 (0.310, 0.016, 0.326)	1 (0.311, 0.004, 0.315)	1 (0.339, 0.000, 0.340)	1 (0.349, 0.000, 0.349)
	0.25	2 (0.128, 0.022, 0.150)	2 (0.159, 0.016, 0.175)	2 (0.158, 0.004, 0.162)	2 (0.186, 0.000, 0.187)	2 (0.196, 0.000, 0.196)
	0.1	3 (0.046, 0.029, 0.075)	3 (0.087, 0.018, 0.105)	3 (0.088, 0.004, 0.092)	4 (0.086, 0.000, 0.086)	4 (0.090, 0.000, 0.090)
	0.05	3 (0.046, 0.029, 0.075)	5 (0.044, 0.018, 0.062)	5 (0.042, 0.007, 0.049)	6 (0.043, 0.000, 0.044)	6 (0.047, 0.000, 0.047)
	0.00	53 (0.000, 0.056, 0.056)	58 (0.000, 0.018, 0.018)	62 (0.000, 0.029, 0.029)	64 (0.000, 0.000, 0.000)	66 (0.000, 0.000, 0.000)
DBLP	0.75	1 (0.707, 0.149, 0.855)	4 (0.740, 0.000, 0.740)	4 (0.719, 0.000, 0.719)	4 (0.719, 0.000, 0.719)	3 (0.656, 0.000, 0.656)
	0.50	3 (0.448, 0.251, 0.700)	9 (0.449, 0.000, 0.449)	8 (0.479, 0.000, 0.479)	8 (0.479, 0.000, 0.479)	6 (0.482, 0.000, 0.482)
	0.25	7 (0.242, 0.295, 0.537)	15 (0.175, 0.000, 0.175)	13 (0.243, 0.000, 0.243)	13 (0.243, 0.000, 0.243)	12 (0.232, 0.000, 0.232)
	0.1	12 (0.082, 0.295, 0.377)	17 (0.044, 0.000, 0.044)	17 (0.038, 0.000, 0.038)	17 (0.038, 0.000, 0.038)	17 (0.083, 0.000, 0.083)
	0.05	14 (0.038, 0.295, 0.332)	17 (0.044, 0.000, 0.044)	17 (0.038, 0.000, 0.038)	17 (0.038, 0.000, 0.038)	19 (0.038, 0.000, 0.038)
	0.00	16 (0.000, 0.295, 0.295)	19 (0.000, 0.000, 0.000)	19 (0.000, 0.000, 0.000)	19 (0.000, 0.000, 0.000)	21 (0.000, 0.000, 0.000)
Chess	0.75	1 (0.098, 0.341, 0.440)	1 (0.695, 0.066, 0.761)	1 (0.619, 0.022, 0.641)	2 (0.614, 0.006, 0.620)	2 (0.660, 0.000, 0.660)
	0.50	1 (0.098, 0.341, 0.440)	2 (0.442, 0.133, 0.575)	3 (0.466, 0.023, 0.489)	4 (0.489, 0.007, 0.497)	5 (0.485, 0.000, 0.485)
	0.25	1 (0.098, 0.341, 0.440)	5 (0.217, 0.206, 0.423)	14 (0.238, 0.023, 0.261)	14 (0.248, 0.011, 0.258)	16 (0.250, 0.000, 0.250)
	0.1	1 (0.098, 0.341, 0.440)	16 (0.098, 0.226, 0.324)	29 (0.099, 0.023, 0.122)	32 (0.097, 0.011, 0.108)	33 (0.096, 0.000, 0.096)
	0.05	5 (0.038, 0.380, 0.418)	31 (0.050, 0.236, 0.286)	42 (0.049, 0.023, 0.072)	46 (0.049, 0.011, 0.061)	47 (0.048, 0.000, 0.048)
	0.00	26 (0.000, 0.384, 0.384)	58 (0.000, 0.237, 0.237)	97 (0.000, 0.023, 0.023)	103 (0.000, 0.011, 0.011)	124 (0.000, 0.000, 0.000)
Mushroom	0.75	1 (0.329, 0.565, 0.894)	2 (0.718, 0.049, 0.766)	3 (0.666, 0.002, 0.668)	3 (0.717, 0.000, 0.717)	3 (0.677, 0.000, 0.677)
	0.50	1 (0.329, 0.565, 0.894)	4 (0.401, 0.142, 0.543)	6 (0.491, 0.008, 0.500)	8 (0.491, 0.000, 0.491)	7 (0.493, 0.000, 0.493)
	0.25	3 (0.250, 0.606, 0.855)	17 (0.243, 0.171, 0.414)	25 (0.244, 0.008, 0.253)	26 (0.246, 0.000, 0.246)	24 (0.241, 0.000, 0.241)
	0.1	19 (0.093, 0.657, 0.750)	44 (0.100, 0.184, 0.284)	48 (0.090, 0.008, 0.098)	58 (0.097, 0.000, 0.097)	46 (0.100, 0.000, 0.100)
	0.05	28 (0.048, 0.675, 0.724)	66 (0.049, 0.186, 0.235)	63 (0.049, 0.008, 0.058)	73 (0.049, 0.000, 0.049)	62 (0.048, 0.000, 0.048)
	0.00	70 (0.000, 0.677, 0.677)	90 (0.000, 0.186, 0.186)	—	113 (0.000, 0.000, 0.000)	120 (0.000, 0.000, 0.000)

case, uses a more relaxed condition, namely it removes j whenever the removal of j improves the overall error. This relaxed condition actually leads to a smaller overall error of the produced factorizations.

Table VI presents selected results of our experiments for binary data; the layout of the table is the same as in the above-mentioned tables for synthetic and real data with fuzzy attributes. In particular, we selected some of the datasets used in [4], which describe various domains. The *Firewall* data describe 365 firewall records (objects) by 709 system attributes; the *DBLP* data are the 19×6980 data describing a selection of 19 renowned computer science conferences and 6980 authors who published in these conferences according to the DBLP database; the *Chess* data are the 3196×76 data involving chess end-game samples including information whether white can win or not; the *Mushroom* data describe 8124 mushrooms using 119 physical characteristics. For these data, the new algorithms, GRECOND+ and our new GRECOND_L+, when compared to GRECOND, tend to produce smaller numbers of factors needed to achieve a prescribed level of uncover error e_u , i.e., needed to explain a prescribed portion of the input data. This is particularly apparent with GRECOND+ and the weight w , which represents penalty for overcovering, set to a small value. Observe also that the new algorithms tend to produce exact factorizations or almost exact factorizations with smaller numbers of factors (sometimes significantly smaller; again as in the case with fuzzy attributes, this is partly because the original GRECOND does not perform any reconsideration of the computed factors). Comparing GRECOND+ with our new GRECOND_L+, we see that while GRECOND+ usually achieves the prescribed e_u with a smaller number of factors, it does so at the expense of producing a considerably larger overcovering, and hence a larger total error e .

IV. CONCLUSION

In this article, we proposed a new algorithm GRECOND_L+ for the factorization of matrices over ordinal scales. This algorithm, unlike other algorithms based on using formal concepts as factors, commits the so-called overcover error—one of two types of error observed when factorizing matrices over ordinal scales. The algorithm is based on utilizing formal concepts as basic building blocks from which the actual factors are constructed by reasonable overcover-committing extension. Our experimental evaluation reveals that due to the usage of formal concepts, the algorithm produces easily interpretable factors. Moreover, the algorithm tends to produce smaller factorizations compared to the other existing algorithms. Interestingly, the management of the overcover error results in a gradual drop of this error as new factors are computed, leading eventually to highly precise factorizations. We also provide new observations on factorization of binary data and propose an improvement to reduce the overcover error of the produced factorizations.

We believe that our results reveal the promise of the main novelty that our new algorithm puts forward, namely constructing factors as extensions of easily interpretable basic blocks, and revisiting and modifying previously generated factors in order to make smaller the otherwise growing overcover error. This approach has a general appeal and may be applied to existing algorithms as well as to new factorization strategies to be designed in the future.

REFERENCES

- [1] R. Belohlavek, "Concept lattices and order in fuzzy logic," *Ann. Pure Appl. Log.*, vol. 128, no. 1–3, pp. 277–298, 2004.
- [2] R. Belohlavek, "Optimal decompositions of matrices with entries from residuated lattices," *J. Log. Comput.*, vol. 22, pp. 1405–1425, 2012.

- [3] R. Belohlavek and M. Krmelova, "Factor analysis of ordinal data via decomposition of matrices with grades," *Ann. Math. Artif. Intell.*, vol. 72, pp. 23–44, 2014.
- [4] R. Belohlavek and M. Trnecka, "A new algorithm for Boolean matrix factorization which admits overcovering," *Discrete Appl. Math.*, vol. 249, pp. 36–52, 2018.
- [5] R. Belohlavek and M. Trneckova, "Factorization of matrices with grades via essential entries," *Fuzzy Sets Syst.*, vol. 360, pp. 97–116, 2019.
- [6] R. Belohlavek and M. Trneckova, "The discrete basis problem and Asso algorithm for fuzzy attributes," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 7, pp. 1417–1427, Jul. 2019.
- [7] R. Belohlavek and V. Vychodil, "Factor analysis of incidence data via novel decomposition of matrices," in *Proc. Int. Conf. Formal Concept Anal.*, 2009, pp. 83–97.
- [8] W. J. Dixon, ed., *BMDP Statistical Software Manual*. Berkeley, CA, USA: Univ. California Press, 1992.
- [9] K. Flaska and P. Cakirpaloglu, "Identification of the multidimensional model of subjective time experience" *Int. Studies in Time Perspective*, Imprensa da Universidade de Coimbra, Coimbra, Portugal, pp. 259–273, 2013.
- [10] S. Gottwald, *A Treatise on Many-Valued Logics*. Baldock, U.K.: Research Studies Press, 2001.
- [11] P. Hájek, *Metamathematics of Fuzzy Logic*. Norwell, MA, USA: Kluwer, 1998.
- [12] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila, "The discrete basis problem," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 10, pp. 1348–1362, Oct. 2008.
- [13] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychol. Rev.*, vol. 63, pp. 81–97, 1956.



Radim Belohlavek (Senior Member, IEEE) is a professor of computer science with Palacky University Olomouc, Olomouc, Czech Republic. He was a tenured professor with the State University of New York, Albany, NY, USA. He has authored or coauthored four books (Kluwer, Springer, MIT Press, Oxford University Press) and more than 250 journal and conference papers in these areas. His research interests include uncertainty and information, data analysis, fuzzy logic and fuzzy sets, applied algebra, and logic.

Dr. Belohlavek is a Member of editorial boards of several journals including *J. Computer and System Sciences*, *Fuzzy Sets and Systems*, *Int. J. General Systems*, and *Fundamenta Informaticae*.



Marketa (Krmelova) Trneckova received the Ph.D. degree from the Palacky University Olomouc, Olomouc, Czech Republic.

Her research interests include data analysis, fuzzy logic, and digital image processing. She has authored or coauthored several papers in journals and conferences, including the *Annals of Mathematics and Artificial Intelligence*, *Knowledge-Based Systems*, *IEEE TRANSACTIONS ON FUZZY SYSTEMS*, and *IEEE ICDM*.

Dr. Trneckova is a Member of ACM.