Int J Data Sci Anal (2016) 1:199–214
DOI 10.1007/s41060-016-0012-3

REGULAR PAPER

Exact and approximate Boolean matrix decomposition with column-use condition

Yuan Sun[1] · Shiwei Ye[2] · Yi Sun[2] · Tsunehiko Kameda[3]

Received: 28 December 2015 / Accepted: 5 June 2016 / Published online: 17 June 2016
© Springer International Publishing Switzerland 2016

Abstract An arbitrary $m \times n$ Boolean matrix M can be decomposed exactly as $M = U \circ V$, where U (resp. V) is an $m \times k$ (resp. $k \times n$) Boolean matrix and $\circ$ denotes the Boolean matrix multiplication operator. The minimum k is called the Boolean rank of M, and it is known to be NP-hard to find it. With the interpretability issue in data mining applications in mind, we impose the *column-use condition* that the columns of U form a subset of the columns of the given M, and employ commonly used heuristics to find as small a k as possible.To this end, we first derive an exact closed-form formula, $J = \overline{\overline{M}^{\mathrm{T}} \circ M}$, such that $M = M \circ J^{\mathrm{T}}$ holds, where J is maximal in the sense that if any 0 element in J is changed to a 1; then, this equality no longer holds. We measure the performance (in minimizing k) of our algorithms on several real benchmark datasets. The results demonstrate that one of our proposed algorithms performs as well or better on all but one of them than other representative heuristic algorithms, which do not impose the column-use condition and thus theoretically should find a smaller k.Boolean matrix decomposition with the column-use condition has wide applications. In educational databases, for example, the "ideal item response matrix" R, the "knowledge state matrix" A, and the "Q-matrix" Q play important roles. As they are related exactly by $\overline{R} = \overline{A} \circ Q^{\mathrm{T}}$, given R, we can find A and Q with a small number (k) of interpretable "knowledge states," using our heuristics.

Keywords Boolean matrix decomposition · Boolean rank · Efficient algorithm · Educational database

This work was supported by the Ministry of Education of Japan under Scientific Research Grants 25280121 and 26560134 awarded to Yuan Sun.

✉ Tsunehiko Kameda
tiko@sfu.ca

Yuan Sun
yuan@nii.ac.jp

Shiwei Ye
shiwye@ucas.ac.cn

Yi Sun
sunyi@ucas.ac.cn

1 Information and Society Research Division, National Institute of Informatics, Tokyo, Japan

2 School of Electrical and Communication Engineering, University of China Academy of Science, Beijing, China

3 School of Computing Science, Simon Fraser University, Vancouver, Canada

1 Introduction

Matrix decomposition, a.k.a. *matrix factorization*, has a long history and is an indispensable tool in matrix algebra [14]. Many applications of matrix decomposition to data mining are described in a recent book on massive data mining by Rajaraman et al. [39]. The well-known *singular value decomposition* (SVD), for example, is now a well-established technique and has been applied in diverse areas, ranging from statistics, image processing, and signal processing to data analytics, to name a few. Although SVD provides a powerful tool in many applications, it suffers from the lack of *interpretability* in some applications [31]. To address the interpretability issue, researchers investigated *nonnegative factorization* (NMF) [3,25,26,47]. In applications such as digital image analysis, DNA analysis, and chemical spectral analysis, for example, it is required that the factor matrices have only nonnegative elements.

To deal with categorical data in data mining, there have recently been intensive research activities in *Boolean matrix* *decomposition* (BMD). This problem has appeared and been investigated in many different guises. A good overview can be found in the Ph.D. thesis of Miettinen [32], and Vaidya [45] surveys many equivalent problems to BMD. The essence of these problems can be abstracted as *formal concept analysis* (FCA) [12]. The *bi-clique cover* problem [1,6,9,17,29] is a particularly nice equivalent formulation of BMD. Unfortunately, the bi-clique covering of a bipartite graph, hence BMD, is an NP-hard problem [38] even for the chordal bipartite graphs [35]. However, it can be solved in polynomial time for some other subclasses of bipartite graphs [11,30,35].

In connection with data mining, BMD has attracted a great deal of research interest in recent years, as evidenced by a large number of recent publications. The seminal work by Miettinen et al. [32,34] was a catalyst to ignite a wave of interest in BMD and its applications to data mining, for example, [2,4,5,21,33,34,40,43,46,49].

By $M \in \{0, 1\}^{m\times n}$, we mean that M is an $m \times n$ Boolean matrix. BMD aims to find two matrices $U \in \{0, 1\}^{m\times k}$ and $V \in \{0, 1\}^{k\times n}$ such that the difference $\|M - U \circ V\|_L$ under some norm L is minimized with a given k or as small a k as possible. The minimum possible k is called the *Boolean rank* of M. It is known that the Boolean rank of a binary matrix may be larger or smaller than its real rank [15]. Moreover, the rank of any real matrix can be computed efficiently by Gaussian elimination, while finding the Boolean rank of a binary matrix is NP-hard [37]. The minimization of $\|M - U \circ V\|_L$ under the Hamming norm L for a given k is called the *discrete basis problem* (DBP) [34].

We can divide $\|M - U \circ V\|_L$ into two components [4], E_u,[1] which is the number of 1's in M that are 0's in $U \circ V$, and E_o,[2] which is the number of 0's in M that are 1's in $U \circ V$. In this paper, we require that $E_o = 0$ or $U \circ V \leq M$, in other words, if an element of M is a 0, then the corresponding element of $U \circ V$ must also be a 0. This condition is called *from-below* approximation in [4,5]. We initially require that $\|M - U \circ V\|_L = 0$ under any norm L, namely we are interested in *exact* BMD. Later in this paper we relax the requirement of exact decomposition, and also discuss from-below approximation to BMD. Unless otherwise specified, $\|M\|$ (resp. $\|\boldsymbol{v}\|$) shall denote the number of non-0 elements in matrix M (resp. vector $\boldsymbol{v}$), i.e., we adopt the l_0 norm. Since BMD is an NP-hard problem, it is impractical to insist that we discover U and V with the minimum k, especially when the size of M is very large.

Geerts et al. [13] formulate the problem as follows. A *tile* consists of a set of 1's in a Boolean matrix that appear at every intersection of a set of rows and a set of columns, and the number of those 1's is called the *area* of the tile. A tile is also called a *combinatorial rectangle* in a communications context [22], and a maximal tile corresponds to a *formal concept* in FCA [12]. A set of tiles is called a *tiling*. Geerts et al. [13] investigate several tiling problems cast in the context of databases. We paraphrase some of them as problems of *covering* 1's in a given matrix M.

– *Minimum tiling* Find a tiling containing the smallest number of tiles that together cover all the 1's in M.
– *Maximum k-tiling* Find a tiling consisting of at most k tiles covering the largest number of 1's in M.
– *Large tile mining* (LTM) Given a minimum threshold σ, find all tiles whose area is at least σ.

Thus, the difference between maximum k-tiling and the discrete basis problem is that the former imposes the from-below approximation condition, but the latter does not. Geerts et al.'s main interest is in designing an algorithm for maximum k-tiling, which can be used to solve minimum tiling problem.

We mentioned nonnegative factorization (NMF) earlier in connection with the interpretability issue. To address this issue from a different angle, Drineas et al. [7,8] introduced CX- and CUR-decompositions. In the CX-decomposition, a given matrix M is decomposed into two matrices C and X such that the "difference" $\|M - C \circ X\|_L$ is minimized, with the column-use condition that the columns of C must be a subset of the columns of M. Note that in CX-decomposition, a parameter k is given and it is required that C have no more than k columns.

In the CUR-decomposition, on the other hand, a given matrix M is decomposed into three matrices C, U, and R, with the condition that the columns of C (resp. rows of R) must be a subset of the columns (resp. rows) of M. Miettinen [31] applies CX- (resp. CUR)-decomposition to BMD, where all the factor matrices are Boolean, and proposes heuristic algorithms. They calls it BCX- (resp. BCUR)-decomposition, and imposes the *column-use condition* that the columns of C form a subset of the columns of M.

We also adopt the column-use condition that the set of columns of the factor matrix U form a subset of the columns of M in decomposing M into $U \circ V$. Arguments in support of imposing this condition in some data mining applications can be found in [18,31]. *Role mining* problem [46], which is also equivalent to BMD, is particularly useful to explain/justify the column-use condition. The human resources department of a company may assign certain permissions to its employees. These permissions can be represented by a Boolean matrix M, where the rows (resp. columns) represent the employees (resp. permissions). Since it is constructed by the management, each permission has a well-defined specific purpose, namely it is clearly interpretable. We now quote a paragraph from Ene et al. [9], which gives some support to the column-use condition. A "role" corresponds to a tile.

[1] The subscript "u" stands for "uncovered" or positive error.

[2] The subscript "o" stands for "overcovered" or negative error.

"We have ignored the qualitative but important question of whether or not these roles are meaningful. Indeed, this is the biggest barrier we have encountered to getting the results of role mining to be used in practice; customers are unwilling to deploy roles that they can't understand. In practice, role mining alone is not sufficient."

Our main goal is to solve the minimum tiling problem defined above (or from-below BMD) with the column-use condition. Imposing the column-use condition has a beneficial effect of greatly reducing the search space for candidate tiles. As commented in [4], the number of maximal tiles may be exponential in $n+m$. The major effort in [13] is on pruning tiles that are not good candidates. Thanks to the column-use condition that we impose, we are spared of that task and our search space consists only of $O(n)$ tiles. Selecting the best k from among a set of candidates is common to both their algorithms and ours and they both use essentially the same set-covering heuristic.

It is clear that exact BMD is easily reducible to the *set cover* problem (SC). Feige [10] shows that SC can be solved approximately with the guaranteed approximation ratio of $O(\log n)$ in the worst case. Umetani et al. [44] give a survey on SC algorithms, but new heuristics are still being proposed, e.g., [23]. Bělohlávek et al. [5] comment that using a SC heuristic (without any modification) to solve BMD is not very effective. In another context, Miettinen [31] also states that in practice algorithms without provable approximation factors performed better.

1.1 Main contributions of this paper

We present algorithms for minimum tiling (or exact BMD) in a limited search space, although our algorithms can be modified only slightly to solve the maximum k-tiling problem (or from-below BMD) as well.

Using elementary matrix calculus, we first derive a simple closed-form formula for matrix J satisfying $M = M \circ J^{\mathrm{T}}$, where J is maximal in the sense that if any 0 in J is changed to a 1, then this equality is violated. We then propose two heuristic algorithms for decomposing $M \in \{0,1\}^{m\times n}$ into $U \circ V$, where $U \in \{0,1\}^{m\times k}$ and $V \in \{0,1\}^{k\times n}$, such that U satisfies the column-use condition and k is minimized. Matrix J greatly facilitates finding the set of all candidate tiles.

Two important performance criteria are (i) how close the common dimension k of the generated U and V is to the (Schein) rank of M, which is the minimum possible, and (ii) how fast U and V can be computed. We demonstrate that our algorithms do rather well in these aspects in comparison with other known algorithms without the column-use condition [4, 5, 13, 49]. Obviously, without the column-use condition, one should be able to achieve a smaller (not larger to be exact) k. When the objective is *exact* BMD, in spite of this restriction, our algorithms do as well as or better than the others (without the column-use condition) on four out of the five popular datasets we have tested,[3] which we find rather surprising.

We apply one of our algorithms to educational data mining. The "ideal item response matrix" R, the "knowledge state matrix" A, and the "Q-matrix" Q play important roles. As they are related exactly by $\overline{R} = \overline{A} \circ Q^{\mathrm{T}}$, given R, we can find A and Q with a small number (k) of interpretable "knowledge states," using our heuristics.

Our algorithms can be slightly modified to find from-below approximation with competitive *coverage* (i.e., the fraction of the 1's covered by the selected tiles). Since matrix operations are available in popular mathematical software packages such as `MATLAB`, `Maple`, and the `R`-language, we made special efforts to state our algorithms in matrix operations. We believe that it has helped to enhance readability.

1.2 Related work

Geerts et al. [13] concentrate on 'maximum k-tiling' and 'large tile mining' mentioned before. Their algorithm, which we call `Tiling`, uses the well-known greedy SC heuristic to iteratively find tiles that cover the most uncovered 1's in the given matrix M. What is new is the way they choose the candidate tiles. Miettinen et al. [34] designed an algorithm, named `Asso`, to solve the DBP. As such, it is not designed to produce exact BMD with the minimum dimension.

Work by Bělohlávek et al. [4,5] addresses exact, as well as approximate, BMD. They make use of ideas from FCA [12], and propose two heuristic algorithms, named `GreConD` and `GreEss`, which find good from-below approximation as well as exact BMD. They do not impose the column-use condition. In [4], they compare the performance of their algorithms with other known algorithms. Keprt and Snášel [19] also discuss BMD, from the viewpoint of *concept lattice* [12].

Another group of researchers, Xiang et al., worked on the "summarization" of a database [49]. Essentially, they also try to find a tiling that covers all 1's in a given transactional database, which can be represented by a Boolean matrix. However, the objective function that they want to minimize is not the number of tiles in the tiling, but the total size of the "description length," based on the *Minimum Description Length* (MDL) *Principle.* (See Grünwald's book [16].) They equate the "description length" of a tile with the sum of the number of 1's in a row of the tile and the number of 1's in a column of the tile. They propose a heuristic algorithm, named `Hyper`, to minimize this objective function, and claim that it also tends to minimize the number of tiles, which is the dimension k in our model.

[3] See Table 2 in Sect. 5. The rows labeled 100 % show the data for exact decomposition.

Ene et al. [9] have proposed a very effective heuristic algorithm for the bi-clique cover problem. Their main contribution is to find a small set of candidate tiles in polynomial time, and they use a simple heuristic used in [13,46] to select the smallest subset from among them. Therefore, it is not relevant to our work reported here, because, thanks to the column-use condition that we adopt, we already have a small number (i.e., $O(n)$) of candidates.

1.3 Paper organization

The rest of the paper is organized as follows. Section 2 gives some basic definitions which will be used throughout the paper, and reviews a minimal set of Boolean algebra facts needed to understand this paper. Section 3 derives a formula, which forms the theoretical basis for our algorithms. In Sect. 4, we describe two algorithms for decomposing a given M into the unknown U and V, and illustrate them with simple examples. Section 5 presents some experimental results, which are very encouraging. In Sect. 6, as an example of possible practical applications, we show how to apply our algorithms to educational data mining. Section 7 concludes the paper with some discussions.

2 Preliminaries

In this section, the basic notations and definitions used throughout this paper are given. We also cite some basic formulae of Boolean matrix theory. Some standard terms in matrix theory are used without definition since they are readily available, for example, in books by Golub and Van Loan [14] and Kim [20].

2.1 Notations and definitions

Let $M = [\mu_{ij}] \in \{0, 1\}^{m\times n}$. Although there is no intrinsic size or magnitude attribute in the value 0 (`False`) and 1 (`True`), we assume that the "larger than" ($>$) relation $1 > 0$ holds and $1 - 0 = 1$, $1 - 1 = 0 - 0 = 0$. In an expanded form, it is represented as

$$M = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_m \end{pmatrix} = \begin{pmatrix} \mu_{11} & \mu_{12} & \cdots & \mu_{1n} \\ \mu_{21} & \mu_{22} & \cdots & \mu_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{m1} & \mu_{m2} & \cdots & \mu_{mn} \end{pmatrix} \tag{1}$$

where $\boldsymbol{\mu}_i = [\mu_{i1}, \mu_{i2}, \ldots, \mu_{in}]$ is called the ith *row vector*, and $[\mu_{1j}, \mu_{2j}, \ldots, \mu_{mj}]^{\mathrm{T}}$ is called the jth *column vector* of M. We also often use $M[i, :]$ (resp. $M[:, j]$) to denote the i-th row(resp. j-th column) vector of M. The matrix whose (i, j) elements is $\overline{\mu}_{ij}$, where $\overline{0} = 1$ and $\overline{1} = 0$, is called the *complement* of M and is denoted by $\overline{M}$. The matrix whose (i, j) elements is μ_{ji} is called the *transpose* of M, and is denoted by M^{T}. The $n \times n$ identity matrix is denoted by $I_{n\times n}$, and $[0]_{m\times n}$ shall denote an $m \times n$ matrix whose elements are all 0's. Let $\mathbb{R}$ (resp. $\mathbb{N}$) denote the set of all real numbers (resp. natural numbers, including 0).

Definition 1 Let $\boldsymbol{p} = [p_1, p_2, \ldots, p_n] \in \{0, 1\}^{1\times n}$ and $\boldsymbol{q} = [q_1, q_2, \ldots, q_n] \in \{0, 1\}^{1\times n}$. We say that $\boldsymbol{p}$ *dominates* $\boldsymbol{q}$ if $p_i \geq q_i$ for all $i = 1, \ldots, n$, and write $\boldsymbol{p} \geq \boldsymbol{q}$. We write $\boldsymbol{p} > \boldsymbol{q}$ if $\boldsymbol{p} \geq \boldsymbol{q}$ and $p_i > q_i$ for some i $(1 \leq i \leq n)$, and say that $\boldsymbol{p}$ *strictly dominates* $\boldsymbol{q}$. Dominance relation is similarly defined for a pair of column vectors.

Definition 2 We define a partial order "$\leq$" on a pair of binary matrices $P = [p_{ij}] \in \{0, 1\}^{m\times n}$ and $Q = [q_{ij}] \in \{0, 1\}^{m\times n}$. We write $P \leq Q$, if $p_{ij} \leq q_{ij}$, for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

Definition 3 Let $P = [p_{ij}] \in \{0, 1\}^{m\times n}$ and $Q = [q_{ij}] \in \{0, 1\}^{m\times n}$ such that $P \leq Q$. We say that P *covers* the set of 1 entries in Q at $\{(i, j) \mid p_{ij} = 1\}$.

Definition 4 If $U = [u_{ij}] \in \{0, 1\}^{m\times n}$ and $V = [v_{ij}] \in \{0, 1\}^{m\times n}$, the *element-wise Boolean sum* of U and V is defined by

$$U \vee V = [u_{ij} \vee v_{ij}] \in \{0, 1\}^{m\times n},$$

and *element-wise Boolean product* of U and V is defined by

$$U \wedge V = [u_{ij} \wedge v_{ij}] \in \{0, 1\}^{m\times n},$$

where $0 \vee 0 = 0$, $1 \vee 0 = 0 \vee 1 = 1 \vee 1 = 1$, $0 \wedge 0 = 1 \wedge 0 = 0 \wedge 1 = 0$, and $1 \wedge 1 = 1$.

For $U = [u_{ij}] \in \{0, 1\}^{m\times k}$ and $V = [v_{ij}] \in \{0, 1\}^{k\times n}$, their ordinary *arithmetic product* is defined by

$$P = UV = [p_{ij}] \in \mathbb{R}^{m\times n}, \quad p_{ij} = \sum_{t=1}^{k} u_{it} v_{tj}. \tag{2}$$

Their *Boolean product* is defined by

$$B = U \circ V = [b_{ij}] \in \{0, 1\}^{m\times n}, \quad b_{ij} = \vee_{t=1}^{k} (u_{it} \wedge v_{tj}). \tag{3}$$

In a Boolean product, 1's and 0's are considered as Boolean values, while in an arithmetic product, they are treated as integers. Let M be given by (1) and c be a constant. The matrix whose (i, j) element is $c\mu_{ij}$ is called a *scalar multiple* of M and is denoted by $c \cdot M$.

2.2 Brief review of matrix algebra relevant to this paper

The materials in this subsection, except Lemma 1, can be found in [14,20].

Proposition 1 *Associativity.*

(a) $(UV)W = U(VW)$
(b) $(U \circ V) \circ W = U \circ (V \circ W)$.

We can thus write UVW (resp. $U \circ V \circ W$) for (a) (resp. (b)) without ambiguity.

Proposition 2 *Transpose of product.*

(a) *For* $U \in \{0,1\}^{m \times k}$ *and* $V \in \{0,1\}^{k \times n}$, $(U \circ V)^{\mathrm{T}} = V^{\mathrm{T}} \circ U^{\mathrm{T}}$ *holds.*
(b) *For* $U \in \mathbb{R}^{m \times k}$ *and* $V \in \mathbb{R}^{k \times n}$, $(UV)^{\mathrm{T}} = V^{\mathrm{T}} U^{\mathrm{T}}$ *holds.*

Proposition 3 *Product expansion.*

$$\begin{aligned} M &= U \circ V = U[:,1] \circ V[1,:] \vee U[:,2] \circ V[2,:] \vee \cdots \\ &\quad \vee U[:,k] \circ V[k,:] \\ &= \vee_{t=1}^{k} \{U[:,t] \circ V[t,:]\} \end{aligned} \tag{4}$$

The following proposition follows directly from (3).

Proposition 4 *Let* $\boldsymbol{p} = [p_1\ p_2 \ldots p_m]$ *and* $\boldsymbol{q} = [q_1\ q_2 \ldots q_n]$ *be two Boolean row vectors. We have*

$$\boldsymbol{p}^{\mathrm{T}} \circ \boldsymbol{q} = \left[q_1 \cdot \boldsymbol{p}^{\mathrm{T}}\ \ q_2 \cdot \boldsymbol{p}^{\mathrm{T}} \ldots q_n \cdot \boldsymbol{p}^{\mathrm{T}} \right] \tag{5}$$

$$= \begin{pmatrix} p_1 \cdot \boldsymbol{q} \\ p_2 \cdot \boldsymbol{q} \\ \vdots \\ p_m \cdot \boldsymbol{q} \end{pmatrix} \in \{0,1\}^{m \times n}. \tag{6}$$

For example, if $\boldsymbol{p} = [0\ 1\ 0\ 1\ 0\ 1]$ and $\boldsymbol{q} = [0\ 1\ 0\ 1\ 1]$, then

$$\boldsymbol{p}^{\mathrm{T}} \circ \boldsymbol{q} = \begin{pmatrix} . & . & . & . & . \\ . & 1 & . & 1 & 1 \\ . & . & . & . & . \\ . & 1 & . & 1 & 1 \\ . & . & . & . & . \\ . & 1 & . & 1 & 1 \end{pmatrix} \tag{7}$$

Thus, $\boldsymbol{p}^{\mathrm{T}} \circ \boldsymbol{q}$ represents a tile. We identify $\boldsymbol{p}^{\mathrm{T}} \circ \boldsymbol{q}$ with the tile it represents, and sometimes call this expression itself a tile. The formula in the following lemma will be used to simplify our algorithms later.

Lemma 1 *Let* $\boldsymbol{p} = [p_1\ p_2 \ldots p_m]$ *and* $\boldsymbol{q} = [q_1\ q_2 \ldots q_n]$ *be two Boolean row vectors, and let* $C \in \{0,1\}^{n \times m}$. *Then, the following equality holds.*

$$\|C \wedge (\boldsymbol{p}^{\mathrm{T}} \circ \boldsymbol{q})\| = \boldsymbol{q} C \boldsymbol{p}^{\mathrm{T}}. \tag{8}$$

Proof The quantity $\|C \wedge (\boldsymbol{p}^{\mathrm{T}} \circ \boldsymbol{q})\|$ is clearly the number of 1 elements of C such that the corresponding element of $\boldsymbol{p}^{\mathrm{T}} \circ \boldsymbol{q}$ is also a 1. By Proposition 4, the (i, j) element of $\boldsymbol{p}^{\mathrm{T}} \circ \boldsymbol{q}$ is a 1 if $p_i = q_j = 1$, and a 0 otherwise. Note that $C\boldsymbol{p}^{\mathrm{T}} \in \mathbb{N}^{n \times 1}$ on the right hand side of (8) is a column vector such that its ith element is the number of 1's in the ith row of C, which are counted if it is in column j satisfying $C[i, j] = p_j = 1$. Now, $\boldsymbol{q}(C\boldsymbol{p}^{\mathrm{T}})$ adds the ith element of $C\boldsymbol{p}^{\mathrm{T}}$ provided $q_i = 1$ and computes their total. □

3 BMD theorems

In the rest of this paper, we refer to matrix $U \in \{0,1\}^{m \times k}$ defined by

$$U = \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_m \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mk} \end{pmatrix} \tag{9}$$

and matrix $V \in \{0,1\}^{k \times n}$ defined by

$$V = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_k \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kn} \end{pmatrix} \tag{10}$$

The following lemma follows easily from the fact that $1 \vee 1 = 1$.

Lemma 2 *Define matrices* $G = [g_{ij}] = UV \in \mathbb{N}^{m \times n}$ *and* $H = [h_{ij}] = U \circ V \in \{0,1\}^{m \times n}$. *Then, for* $i = 1, 2, \ldots, m$ *and* $j = 1, 2, \ldots, n$ *we have*

$$\begin{aligned} g_{ij} = 0 &\Leftrightarrow h_{ij} = 0 \\ g_{ij} \geq 1 &\Leftrightarrow h_{ij} = 1. \end{aligned} \tag{11}$$

The following proposition follows easily from definition.

Proposition 5 *Let* $\boldsymbol{p}, \boldsymbol{q} \in \{0,1\}^{1 \times a}$ *be two Boolean row vectors. Then "*$\boldsymbol{p}$ *dominates* $\boldsymbol{q}$*" can be expressed as*

$$\boldsymbol{p} \geq \boldsymbol{q} \Leftrightarrow \overline{\boldsymbol{p}} \circ \boldsymbol{q}^{\mathrm{T}} = \boldsymbol{q} \circ \overline{\boldsymbol{p}}^{\mathrm{T}} = 0 \Leftrightarrow \overline{\overline{\boldsymbol{p}} \circ \boldsymbol{q}^{\mathrm{T}}} = \overline{\boldsymbol{q} \circ \overline{\boldsymbol{p}}^{\mathrm{T}}} = 1. \tag{12}$$

Lemma 4 below plays an important role in what follows. In order to prove it, we first need to show a technical lemma.

Lemma 3 *Let $P \in \{0,1\}^{a\times p}$ be an arbitrary Boolean matrix.*

(a) *For any two row vectors $\boldsymbol{u}, \boldsymbol{v} \in \{0,1\}^{1\times a}$ we have*

$$[\overline{\boldsymbol{v}} = \overline{(\boldsymbol{u}\circ P)} \circ P^{\mathrm{T}}] \Rightarrow \boldsymbol{v} \geq \boldsymbol{u} \tag{13}$$

(b) *For any two matrices $U, V \in \{0,1\}^{b\times a}$ we have*

$$[\overline{V} = \overline{U\circ P} \circ P^{\mathrm{T}}] \Rightarrow V \geq U. \tag{14}$$

Proof (a) Suppose $\overline{\boldsymbol{v}} = \overline{(\boldsymbol{u}\circ P)} \circ P^{\mathrm{T}}$ holds. Then $\overline{v}_j = 0$ (i.e., $v_j = 1$) if and only if

$$\overline{(\boldsymbol{u}\circ P)} \circ P[j,:]^{\mathrm{T}} = 0.$$

By Proposition 5, this implies that $\boldsymbol{u}\circ P$ dominates the jth column of P^{T}, i.e., the jth row of P. Since this clearly happens if $u_j = 1$, we have $u_j = 1 \Rightarrow v_j = 1$. It follows that $\boldsymbol{v} \geq \boldsymbol{u}$.

(b) Let $\boldsymbol{u}_i$ (resp. $\boldsymbol{v}_i$) be the ith row vector of matrix U (resp. V), as in (9) (res. (10)). Then (13) holds for each i $(1 \leq i \leq b)$, namely,

$$[\overline{\boldsymbol{v}}_i = \overline{(\boldsymbol{u}_i\circ P)} \circ P^{\mathrm{T}}] \Rightarrow \boldsymbol{v}_i \geq \boldsymbol{u}_i,$$

and (14) follows. □

Without loss of generality, we assume from now on that the given matrix M has no all-0 row or all-0 column. We now prove the following theorem, which provides a basis for the algorithms given in the next section.

Lemma 4 *Let $M \in \{0,1\}^{m\times n}$, $U \in \{0,1\}^{m\times k}$, and $V \in \{0,1\}^{k\times n}$ satisfy $M = U\circ V$, and define*

$$K \equiv \overline{\overline{M}^{\mathrm{T}} \circ U} \tag{15}$$

Then we have

(a) $V \leq K^{\mathrm{T}}$, *and*
(b) $M = U\circ K^{\mathrm{T}}$

Proof (a) From (15), we get

$$\overline{K} = \overline{M}^{\mathrm{T}} \circ U \tag{16}$$

Plugging $M = U\circ V$ into (16) and using Proposition 2(a) and the fact that the order of complementation and transpose is reversible, we obtain

$$\overline{K} = \overline{U\circ V}^{\mathrm{T}} \circ U = \overline{V^{\mathrm{T}}\circ U^{\mathrm{T}}} \circ U. \tag{17}$$

If we set $U = P^{\mathrm{T}}$ in (17), we get

$$\overline{K} = \overline{V^{\mathrm{T}}\circ P} \circ P^{\mathrm{T}}.$$

Thus we get $K \geq V^{\mathrm{T}}$ using (14).

(b) Define $N = U\circ K^{\mathrm{T}}$. We want to show that $N = M$. From (15), we get

$$N^{\mathrm{T}} = K\circ U^{\mathrm{T}} = \overline{\overline{M}^{\mathrm{T}} \circ U} \circ U^{\mathrm{T}}, \tag{18}$$

which yields $\overline{N}^{\mathrm{T}} \geq \overline{M}^{\mathrm{T}}$ or $N \leq M$ by (14). On the other hand, from $V \leq K^{\mathrm{T}}$ (see part (a)), we get $M = U\circ V \leq U\circ K^{\mathrm{T}} = N$. It follows that $M = N$. □

Example 1 Let

$$U = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}; \quad V = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Then we have

$$M = U\circ V = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix};$$

$$K = \overline{\overline{M}^{\mathrm{T}} \circ U} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Clearly, Lemma 4(a) holds, and we can verify Lemma 4(b) as well.

From now on, we consider the special case in Lemma 4, where $U = M$, hence

$$J = \overline{\overline{M}^{\mathrm{T}} \circ M} \in \{0,1\}^{n\times n}. \tag{19}$$

Lemma 4 has the following important implication.

Corollary 1 *Given an arbitrary matrix $M \in \{0,1\}^{m\times n}$, let J be defined by (19). Then, $V \leq J^{\mathrm{T}}$ holds for any matrix $V \in \{0,1\}^{n\times n}$ satisfying $M = M\circ V$.* □

Matrix J has a number of other important properties.

Theorem 1 *For any $M \in \{0,1\}^{m \times n}$, matrix J defined by (19) has the following properties.*

(a) $J[i,j] = 1 \Leftrightarrow M[:,i] \ge M[:,j]$, i.e., column i dominates column j of M.
(b) $J[i,j] = J[j,i] = 1 \Leftrightarrow M[:,i] = M[:,j] \Leftrightarrow J[:,i] = J[:,j]$ and $J[i,:] = J[j,:]$.
(c) $J[i,j] = 1 > J[j,i] = 0 \Leftrightarrow M[:,i] > M[:,j] \Rightarrow J[:,j] > J[:,i]$ and $J[j,:] < J[i,:]$.

Proof (a) If we let $\boldsymbol{p} = M^{\mathrm{T}}[:,i]$ and $\boldsymbol{q} = M^{\mathrm{T}}[:,j]$ in (12), then we get $M^{\mathrm{T}}[i,:] \ge M^{\mathrm{T}}[j,:]$ if and only if

$$\overline{\overline{M}^{\mathrm{T}}[i,:] \circ M[:,j]} = 1,$$

which holds if and only if $J[i,j] = 1$ by (19). Note that $M^{\mathrm{T}}[i,:] \ge M^{\mathrm{T}}[j,:]$ is equivalent to $M[:,i] \ge M[:,j]$.

(b) By interchanging i and j in part (a), we get $J[j,i] = 1 \Leftrightarrow M[:,i] \le M[:,j]$. It follows that $J[i,j] = J[j,i] = 1 \Leftrightarrow M[:,i] = M[:,j]$. Thus for any column $M[:,k]$ we have $M[:,k] \ge M[:,j] \Leftrightarrow M[:,k] \ge M[:,i]$, i.e., any column that dominates $M[:,j]$ also dominates $M[:,i]$, and vice versa. This implies $J[:,i] = J[:,j]$. Similarly, for any column $M[:,k]$ we have $M[:,k] \le M[:,j] \Leftrightarrow M[:,k] \le M[:,i]$, i.e., any column that is dominated by $M[:,j]$ is also dominated by $M[:,i]$, and vice versa, which implies $J[i,:] = J[j,:]$.

(c) $J[i,j] = 1 > J[j,i] = 0$ implies that $M[:,i]$ strictly dominates $M[:,j]$, i.e., $M[:,i] > M[:,j]$. Therefore, for any column $M[:,k]$ we have $M[:,k] \ge M[:,i] \Rightarrow M[:,k] > M[:,j]$, which implies $J[:,j] > J[:,i]$. Similarly, for any column $M[:,k]$ we have $M[:,k] \le M[:,j] \Rightarrow M[:,k] < M[:,i]$, which implies $J[j,:] < J[i,:]$. □

Example 2 The properties proved in Theorem 1 can be verified for the matrix M in Example 1, for which matrix J defined by (19) is

$$J = \overline{\overline{M}^{\mathrm{T}} \circ M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

We now prove another useful property of matrix J defined by (19).

Lemma 5 *Given an arbitrary matrix $M \in \{0,1\}^{m \times n}$, let J be defined by (19). If any 0-element in J is changed to a 1, then $M = M \circ J^{\mathrm{T}}$ no longer holds.*

Proof Assume that J does not have the maximum number of 1's and assume that $J[i,j] = 0$, $1 \le i,j \le n$, can be changed from 0 to 1 without violating Lemma 4(b) with $U = M$, i.e., $M = M \circ J^{\mathrm{T}}$. Let $\boldsymbol{w}_j = J[j,\cdot]$, so that $(\boldsymbol{w}_j)^{\mathrm{T}}$ is the jth column of J^{T}. If the ith element of $\boldsymbol{w}_j$ is 0, i.e., $J[j,i] = 0$, then $M[\cdot,j] \not\ge M[\cdot,i]$ by Theorem 1(a). Let $\boldsymbol{w'}_j$ be obtained from $\boldsymbol{w}_j$ by changing its ith element from 0 to 1. Since $M \circ (\boldsymbol{w}'_j)^{\mathrm{T}} \ge M[\cdot,i]$, we have $M[\cdot,j] \not\ge M \circ (\boldsymbol{w}'_j)^{\mathrm{T}}$, a contradiction. We conclude that if any element in J is changed from a 0 to a 1, then $M = M \circ J^{\mathrm{T}}$ is violated. □

If we change $J[3,1] = 0 \to 1$ in Example 2, for example, then the [3, 3] element of $M \circ J^{\mathrm{T}}$ become a 1, while $M[3,3] = 0$, and $M = M \circ J^{\mathrm{T}}$ no longer holds.

Theorem 2 *Let $M = U \circ V$ be an optimal decomposition of M, satisfying the column-use condition,[4] where $U \in \{0,1\}^{m \times k}$, $V \in \{0,1\}^{k \times n}$ and k is the minimum possible. Then for each $i = 1, 2, \ldots, k$, we have $U[:,i] \circ V[i,:] \in \{M[:,t] \circ J[t,:] \mid t = 1, \ldots, n\}$.*

Proof Let

$$U \circ V = \vee_{i=1}^{k} \{U[:,i] \circ V[i,:]\},$$

and consider a particular term $U[:,i] \circ V[i,:]$ in it. Since U consists of columns of M, there is an h such that $U[:,i] = M[:,h]$. By Corollary 1, $J[h,:]$ is the maximal row vector such that $U[:,i] \circ J[h,:] \le M$, hence $V[i,:] \le J[h,:]$. We thus have $U[:,i] \circ V[i,:] \le U[:,i] \circ J[h,:] = M[:,h] \circ J[h,:]$. □

Intuitively, Theorem 2 implies that the search space for an optimal decomposition of M under the column-use condition can be limited to $\{M[:,t] \circ J[t,:] \mid t = 1, \ldots, n\}$. When the column-use condition is not imposed, the counterpart to Theorem 2 is proved in [4], using FCA.

In the next section, we design heuristic algorithms for exact BMD, based on Theorem 2.

4 Heuristic BMD algorithms

4.1 Algorithm description

In this section, we propose new algorithms for finding factor matrices $U \in \{0,1\}^{m \times k}$ and $V \in \{0,1\}^{k \times n}$ from matrix $M \in \{0,1\}^{m \times n}$. By Theorem 2, we want to find a subset of $\{M[:,t] \circ J[t,:] \mid t = 1, \ldots, n\}$ that provides the optimal tiling. Since an exhaustive search is obviously impractical, we want to devise a heuristic algorithm that yields a good suboptimal tiling.

[4] By definition, this means that the columns of U are some of the columns of M.

Suppose there exists an l satisfying

$$U \circ V = \vee_{i=1, j \neq l}^{k} \{U[:, i] \circ V[i, :]\},$$

in other words,

$$\vee_{i=1, j \neq l}^{k} \{U[:, i] \circ V[i, :]\} \geq U[:, l] \circ V[l, :]. \tag{20}$$

Then, we can safely eliminate the lth column $U[:, l]$ and the lth row $V[l, :]$ from U and V, respectively, which helps reduce the dimension k. The condition (20) is equivalent to $\|\mathcal{T}\| = \|\mathcal{T} - T_l\|$, where $\mathcal{T} = UJ^{\mathrm{T}}$ (arithmetic matrix product defined by (2)) with J given by (19), and $T_l = U[:, l] \circ V[:, l]$. There may be several indices l that satisfy (20). Therefore, we need to decide in which order the eliminations should be carried out. We thus define the *selection index* σ_i as follows:

$$\sigma_i = \|U[:, i]\| \times \|V[:, i]\|,$$

where, as the reader recalls, $\|V\|$ represents the number of 1's (l_0 norm) in vector V. Clearly, σ_i is the number of 1 entries in M that are covered by G_i. Given the initial matrices U and V, satisfying $M = U \circ V$, we generate the new matrix J by (19). There are at least two straightforward strategies that appear reasonable, regarding which attribute we should process first.

(a) `Remove-Smallest`: Remove attribute j such that σ_j is the smallest, provided the removal does not affect M.
(b) `Pick-Largest`: Retain attribute j such that σ_j is the largest.

Strategy (b) has been used before by other researchers, including Geerts et al. [13] and Vaiya et al. [46]. Our first heuristic algorithm adopts strategy (a). After deleting one attribute, we update U and V, and repeat the elimination process until there is no more attribute that can be deleted.

Algorithm 1 `Remove-Smallest` Input: *Response matrix* $M \in \{0, 1\}^{m \times n}$.

```
1. Initialize $U = M$ and $k = n$.
2. Compute
       $$V^{\mathrm{T}} = J = \overline{\overline{M}^{\mathrm{T}} \circ M}. \tag{21}$$
3. Compute [5]
       $\mathcal{T} = UV.$
4. For $i = 1, 2, \ldots, k$ compute the size of the maximal tile for $i$th attribute ($\alpha_i$) by
       $\sigma_i = \|U[:, i]\| \times \|V[:, i]\|,$
   and rename the attributes so that $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_k$ holds.
5. For $j = 1, 2, \ldots, k$, do
   (a) Compute
           $T_j = U[:, j] \circ V[j, :];$
   (b) If $\|\mathcal{T}\| = \|\mathcal{T} - T_j\|$ then (i) remove column $U[:, j]$ from $U$ and row $V[j, :]$ from $V$; and (ii) set $\mathcal{T} = \mathcal{T} - T_j$; $k = k - 1$.
6. Output $U$ and $V$.
```

[5] Intuitively, $\mathcal{T}[i, j]$ is the number of tiles in $U \circ V$ that cover $M[i, j]$.

Our second algorithm adopts strategy (b).

Algorithm 2 `Pick-Largest` Input: *Response matrix* $M \in \{0, 1\}^{m \times n}$.

```
1. Initialize $U = M$ and $k = n$.
2. Compute
       $$V^{\mathrm{T}} = J = \overline{\overline{M}^{\mathrm{T}} \circ M}. \tag{22}$$
3. Initialize[6] $C = [0]_{m \times n} \in \{0, 1\}^{m \times n}$.
4. For $i = 1, 2, \ldots, k$ compute the size of the maximal tile for the $i$th attribute ($\alpha_i$) by
       $\sigma_i = \|U[:, i]\| \times \|V[:, i]\|.$
5. For each $i$ such that $\alpha_i$ has not been picked or discarded, compute (see (8))
       $\delta_i = \sigma_i - U[:, i]^{\mathrm{T}} C V[:, i].$
   If $\delta_i = 0$ then remove $\alpha_i$ by deleting $U[:, i]$ (resp. $V[i, :]$) from $U$ (resp. $V$).
6. Let $\delta_j = \max_i\{\delta_i\}$ and compute
       $T_j = U[:, j] \circ V[j, :].$
   Replace matrix $C$ by $C \vee T_j$, and delete $U[:, j]$ (resp. $V[j, :]$) from $U$ (resp. $V$). If there are still attributes remaining, then go to Step 5.
7. Output $U$ and $V$.
```

[6] Matrix C keeps track of the 1 elements in M covered by the products that have been picked so far. We have $C = M$ when this algorithm completes.

Note that we compute σ_i just once in Step 4, but it is effectively updated in Step 5. The correctness of the above algorithms is implied by Theorems 1 and 2.

4.2 Simple example

Example 3 Let us consider the following matrix M, and carry out Steps 2) and 4), which are common to both algorithms.

$$M = \begin{pmatrix} 0&0&0&0&0&0&0\\ 1&0&1&1&0&1&1\\ 0&1&1&0&1&0&1\\ 0&0&0&1&0&0&0\\ 0&1&1&0&1&0&1\\ 0&1&1&0&1&0&1\\ 1&0&1&1&0&1&1\\ 1&1&1&1&1&1&1 \end{pmatrix}$$

$$V^{\mathrm{T}} = \overline{\overline{M}^{\mathrm{T}} \circ M} = \begin{pmatrix} 1&0&0&0&0&1&0\\ 0&1&0&0&1&0&0\\ 1&1&1&0&1&1&1\\ 1&0&0&1&0&1&0\\ 0&1&0&0&1&0&0\\ 1&0&0&0&0&1&0\\ 1&1&1&0&1&1&1 \end{pmatrix}$$

Step 3 of `Remove-Smallest` computes

$$\mathcal{T} = UV = \begin{pmatrix} 0&0&0&0&0&0&0\\ 2&0&4&3&0&2&4\\ 0&2&4&0&2&0&4\\ 0&0&0&1&0&0&0\\ 0&2&4&0&2&0&4\\ 0&2&4&0&2&0&4\\ 2&0&4&3&0&2&4\\ 2&2&6&3&2&2&4 \end{pmatrix} \tag{23}$$

If we order the columns of U from the smallest to the largest according to the value of σ_i, we get 4,3,7,1,6,2,5. Thus, `Remove-Smallest` processes the columns of U in this order.
Step 5(a): Compute T_4.

$$T_4 = U[:,4] \circ V[4,:] = \begin{pmatrix} 0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0 \end{pmatrix}$$

Step 5(b): $\|\mathcal{T}\| > \|\mathcal{T} - T_4\| \Rightarrow$ Cannot remove attribute 4.

Step 5(a): Now try the next smallest attribute 3, and compute T_3.

$$T_3 = U[:,3] \circ V[3,:] = \begin{pmatrix} 0&0&0&0&0&0&0\\ 0&0&1&0&0&0&1\\ 0&0&1&0&0&0&1\\ 0&0&0&0&0&0&0\\ 0&0&1&0&0&0&1\\ 0&0&1&0&0&0&1\\ 0&0&1&0&0&0&1\\ 0&0&1&0&0&0&1 \end{pmatrix}$$

Step 5(b): $\|\mathcal{T}\| = \|\mathcal{T} - T_3\| \Rightarrow$ Remove attribute 3, and update $\mathcal{T}$.

$$\mathcal{T} = \mathcal{T} - T_3 = \begin{pmatrix} 0&0&0&0&0&0&0\\ 2&0&3&3&0&2&3\\ 0&2&3&0&2&0&3\\ 0&0&0&1&0&0&0\\ 0&2&3&0&2&0&3\\ 0&2&3&0&2&0&3\\ 2&0&3&3&0&2&3\\ 2&2&5&3&2&2&3 \end{pmatrix}$$

Similarly, attributes (columns of M) 7, 1, and 5 are removed.
Step 6: generates

$$U = \begin{pmatrix} 0&0&0\\ 1&1&0\\ 0&0&1\\ 1&0&0\\ 0&0&1\\ 0&0&1\\ 1&1&0\\ 1&1&1 \end{pmatrix}; \quad V = \begin{pmatrix} 0&0&0&1&0&0&0\\ 1&0&1&1&0&1&1\\ 0&1&1&0&1&0&1 \end{pmatrix} \tag{24}$$

The columns of U are columns 4, 6, and 2 of M, and $M = U \circ V$. Let us now apply Algorithm `Pick-Largest` to matrix M. We already illustrated the first four steps above. From Table 1, we see that $\delta_5 = \sigma_5 = 16$ is the largest (tied with $\delta_2 = \sigma_2 = 16$). Since $\delta_i = 0$ holds for no i, we proceed to Step 6.

Table 1 Computing σ_i

i	1	2	3	4	5	6	7
$\|\|U[:,i]\|\|$	3	4	6	4	4	3	6
$\|\|V[i,:]\|\|$	5	4	2	1	4	5	2
σ_i	15	16	12	4	16	15	12

$$T_5 = U[:,5] \circ V[5,:] = \begin{pmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&1&1&0&1&0&1\\ 0&0&0&1&0&0&0\\ 0&1&1&0&1&0&1\\ 0&1&1&0&1&0&1\\ 0&0&0&0&0&0&0\\ 0&1&1&0&1&0&1 \end{pmatrix}$$

We set $C = C \vee T_5$ to remember the 1's that are now covered by the picked product term. Although this algorithm does not use $\mathcal{T}$ in (23), it is instructive to interpret Steps 5 and 6 of `Pick-Largest` in terms of $\mathcal{T}$. We have

$$\mathcal{T} = \mathcal{T} - T_5 = \begin{pmatrix} 0&0&0&0&0&0&0\\ 2&0&4&3&0&2&4\\ 0&1&3&0&1&0&3\\ 0&0&0&0&0&0&0\\ 0&1&3&0&1&0&3\\ 0&1&3&0&1&0&3\\ 2&0&4&3&0&2&4\\ 2&1&5&3&1&2&3 \end{pmatrix}$$

In Step 5, we update $\{\delta_i\}$. For example, let us compute $CV[i,:]^{\mathrm{T}}$ for $i = 2$. We get

$$CV[2,:]^{\mathrm{T}} = [0\ 0\ 4\ 0\ 4\ 4\ 0\ 4] \quad \text{and} \quad U[:,2]^{\mathrm{T}} CV[:,2] = 16.$$

Therefore, $\delta_2 = \sigma_2 - 16 = 0$. This implies that $T_2 \le C$. We can simply remove attribute 2 (i.e, $U[:,2]$ and $V[2,:]$). Updating C by $C = C \vee T_2$ does not change C.

$$\mathcal{T} = \mathcal{T} - T_2 = \begin{pmatrix} 0&0&0&0&0&0&0\\ 2&0&4&3&0&2&4\\ 0&0&2&0&0&0&2\\ 0&0&0&1&0&0&0\\ 0&0&2&0&0&0&2\\ 0&0&2&0&0&0&2\\ 2&0&4&3&0&2&4\\ 2&0&4&3&0&2&2 \end{pmatrix}$$

This computation can be done by matrix operation, although it is not the most efficient, since it computes elements that are of no use to us. Construct a column vector Us whose ith element is $\|U[:,i]\|$, and a row vector Vs whose ith element is $\|V[:,i]\|$. Compute matrix $P = Us \circ Vs$.

$$P = \begin{pmatrix} 15&12&6&3&12&15&6\\ 20&16&8&4&16&20&8\\ 30&24&12&6&24&30&12\\ 20&16&8&4&16&20&8\\ 20&16&8&4&16&20&8\\ 15&12&6&3&12&15&6\\ 30&24&12&6&24&30&12 \end{pmatrix}$$

Thus, the diagonal elements of P are $\|U[:,i]\| \times \|V[:,i]\|$, which are listed in Table 1. Note that the ith diagonal element of $U^{\mathrm{T}} \circ C \circ V^{\mathrm{T}}$ is the number of 1's in $U[:,i] \circ V[:,i]$ that are already covered by C.

$$U^{\mathrm{T}} \circ C \circ V = \begin{pmatrix} 2&4&2&0&4&2&2\\ 8&16&8&0&16&8&8\\ 8&16&8&0&16&8&8\\ 2&4&2&0&4&2&2\\ 8&16&8&0&16&8&8\\ 2&4&2&0&4&2&2\\ 8&16&8&0&16&8&8 \end{pmatrix}$$

Thus, the amounts $\{\delta_i\}$ can be found on the diagonal of $P - U^{\mathrm{T}} \circ C \circ V$, and they are 13, 0, 4, 4, 0, 13, 4. So $\delta_2 = 13$ and $\delta_6 = 13$ are the largest. Let us pick attribute 6,[7] update $C = C \vee T_6$, and recompute $P - U^{\mathrm{T}} \circ C \circ V$. Since updated $\delta_1 = 0$, we discard attribute 1. (Step 5.) We then get $\delta_7 = 4$, so pick attribute 7. Since $\delta_3 = 0$, we discard attribute 3. Finally, we need to pick attribute 4. For this particular example, `Pick-Largest` generates the same decomposition as `Remove-Smallest` given in (24).

Comment 3 *Although computing $U^{\mathrm{T}} \circ C \circ V$ is a conceptually neat way of finding $\{\delta_i\}$, the time to compute the off-main diagonal elements is wasted. Thus, we do not use it in* `Pick-Largest`.

5 Performance

5.1 Complexity analysis

The time complexity of both algorithms is dominated by the time to compute matrix V of (21) and (22), respectively, in their Step 2. By Proposition 3, it can be expanded into n (column vector, row vector) pairs of sizes m and n, respectively. Then, evaluating $\overline{M}^{\mathrm{T}} \circ M$ takes time proportional to

$$\sum_{i=1}^{m} \|\overline{M}^{\mathrm{T}}[:,i]\| \times \|M[i,:]\| \le n \sum_{i=1}^{m} \|M[i,:]\| = n\|M\|.$$

This implies that (21) and (22) can be evaluated in $O(n\|M\|)$ time. Note that in terms of $\mathcal{T}$ defined in Step 3 of Algorithm `Remove-Smallest`, we have

$$\|\mathcal{T}\|_{l_1} = \sum_{i=1}^{n} \|U[:,i]\| \times \|V[:,i]\| \le m \sum_{i=1}^{n} \|U[:,i]\| = m\|M\|,$$

[7] When there is a tie in the sizes δ_i, as in this example, there are choices as to which one we remove or pick first. A particular choice may affect the coverage performance. We randomly select one.

where $\|\mathcal{T}\|_{l_1}$ (l_1 norm) represents the sum of the elements of $\mathcal{T}$.

Theorem 4 *Both Algorithms* `Remove-Smallest` *and* `Pick-Largest` *run in* $O(m\|M\|)$ *time.*

Proof We can consider that every operation in Algorithm `Remove-Smallest` essentially accesses/modifies some element of $\mathcal{T}$ and the (i, j) element is accessed $\mathcal{T}[i, j]$ times. Therefore, the total time is given by $O(\|\mathcal{T}\|_{l_1}) = O(m\|M\|)$.

As for Algorithm `Pick-Largest`, although $\mathcal{T}$ is not used in it, imagine that it was defined. We use $U[:, i]^{\mathrm{T}} C V[:, i]$ to describe Step 5, but it is used for only for the purpose of a concise description, and this step can be implemented more efficiently without matrix multiplication. All we need is a way to keep track of which 1 elements of M has already been covered. Therefore, the total time is still given by $O(\|\mathcal{T}\|_{l_1}) = O(m\|M\|)$, as reasoned above. □

The above theorems imply that our algorithms run faster if the given matrix M is sparse. If we use a sophisticated algorithm, matrix multiplication can be done in $O(m^{2.373})$ time, assuming $m \geq n$ [24,48]. We should mention that another important performance measure for heuristic algorithms of the *approximation ratio* relative to the optimum. We have not looked into this performance measure yet.

5.2 Experiments on real datasets

To evaluate the performance of our heuristic algorithms, `Pick-Largest` and `Remove-Smallest`, we have tested them on several real datasets, which have been used by other authors as benchmarks. They are `Mushroom` [27], `DBLP`,[8] `DNA` [36], `Chess` [27], and `Paleo`.[9] Table 2 in the next page lists the results of our experiments and compares them with `Tiling` [13], `Asso` [34], `Hyper` [49], and `GreConD` [5], and `GreEss` [5]. All but the last two columns of Table 2 are from [4]. The common dimension k of the factor matrices, generated by the exact BMD heuristics mentioned above are listed. The numbers in bold face indicate the best value in each row. The rows labeled 100 % shows the data for exact decomposition. `Asso` is not meant for exact BMD, as commented earlier.

Among the datasets we used, `Mushroom` consists of 8,124 objects and 23 “nominal” attributes. If a “nominal” attribute y takes $h > 2$ values, $\{v_1, \ldots, v_h\}$, we expanded y, replacing it by h Boolean attributes $\{y_{v_1}, \ldots, y_{v_h}\}$. In row i, the value of the column corresponding to y_{v_j} is has a 1 if the value of the attribute y in row i in the original dataset is equal to v_j.

[8] http://www.informatik.uni-trier.de/~ley/db/.

[9] http://www.helsinki.fi/science/now/.

Note that only our algorithms impose the column-use condition. In spite of this restriction, `Pick-Largest` achieves the smallest tiling size (or dimension k) for exact (i.e., 100 %) coverage for four out of the five datasets in Table 2, which was somewhat unexpected. Incidentally, we have found a decomposition without the column-use condition with $k = 101$ by some other means, so none of the algorithms in Table 2 can find the optimal decomposition for the `Mushroom` dataset. As can be seen from Table 2, `Pick-Largest` and `Remove-Smallest` performed equally well in finding the exact decomposition. Another observation on Table 2 is that for 100 % coverage some results are peculiar in that $k > n$, i.e., the common dimension of the factor matrices is larger than n. Our algorithms are the only ones that never produce such results.

Although our original intention was to design algorithms for exact BMD, our algorithms can also be used for “from-below” approximation [5]. In the from-below approximation, an important performance criterion is the *coverage*, defined as the number of 1’s covered by the product $U \circ V$ over the total number of 1’s in the given matrix M [13]. The coverage is given in the second column of Table 2. Each entry in the table is the number of tiles used, which is the same as the common dimension k of U and V. Figure 1 plots the coverage of Algorithm `Pick-Largest` as a function of the number of attributes contained in U and V. The attributes are arranged in the order they were picked.

In most applications, high coverage, say, more than 90 %, would be of interest, and we have collected coverage data in Table 3 in this range for `Pick-Largest` and `Remove-Smallest`, but unfortunately not for the others, since we haven’t had the time to program the other algorithms. For `Mushroom`, however, it is stated in [13] that `Tiling` needs 45 tiles to attain 90 % coverage vs. 47 for `Pick-Largest`. Table 3 shows that to attain 100 % coverage, `Tiling` needs 119 tiles vs. 109 for `Pick-Largest`. We have some evidence to suggest that our algorithms perform better than others especially at higher coverages.

Another important aspect of performance is the efficiency of the algorithm in terms of speed and memory use. Table 4 shows the time it took for them to decompose M (of `Mushroom`) into U and V and the amount of memory used.

Belohlavek et al. [5] carried out extensive tests of their algorithms `GreConD` and `GreEss`, which can be used for exact BMD, on `Mushroom`, and measured the time and memory requirement. Their data for exact BMD are given in Table 4. We should mention that the platforms we used to produce our results are different from theirs, as shown in Table 5. Probably, it is safe to say that there is not a huge difference between the two. Unfortunately, we do not have similar data for other algorithms, since they are not published.

Table 2 Coverage comparison of BMD algorithms for five datasets

	Coverage (%)	Tiling	Asso	Hyper	GreConD	GreEss	Remove-Smallest	Pick-Largest
Mushroom [27]	50	7	**6**	19	7	8	37	10
(8124 × 119)	75	**24**	36	37	**24**	26	59	27
	100	119	N/A	122	120	**105**	109	109
DBLP	50	**5**	**5**	**5**	**5**	**5**	6	6
(6980 × 19)	75	**10**	**10**	**10**	11	**10**	11	11
	100	21	19	**19**	20	**19**	**19**	**19**
DNA [36]	50	32	**27**	67	33	41	67	58
(4590 × 392)	75	94	**80**	155	96	105	155	123
	100	489	N/A	392	496	408	**368**	**368**
Chess [27]	50	5	**2**	26	4	6	26	12
(3196 × 75)	75	16	**15**	39	15	17	44	26
	100	124	N/A	90	124	113	**72**	**72**
Paleo	50	39	40	**38**	39	**38**	39	39
(501 × 139)	75	75	76	**73**	76	**73**	75	74
	100	151	N/A	**139**	152	145	**139**	**139**

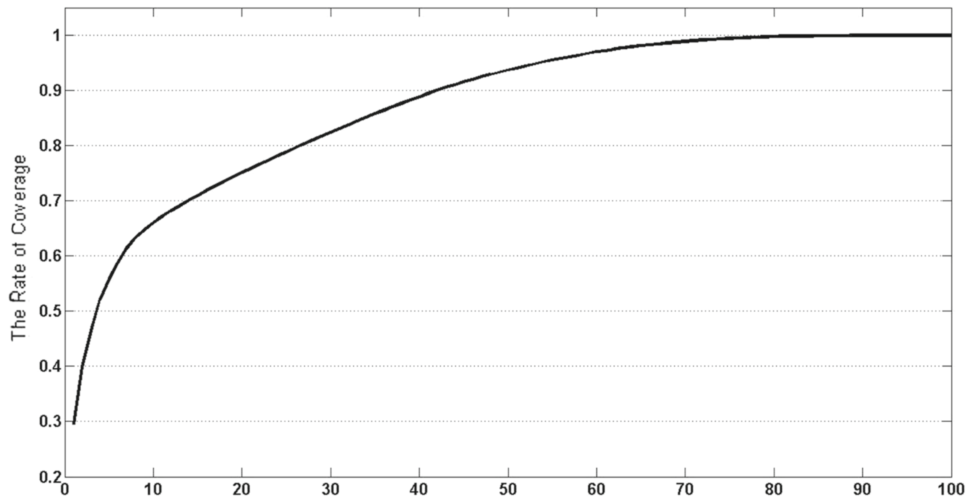

Fig. 1 Coverage of Algorithm Pick-Largest for Mushroom

Table 3 Comparison of Remove-Smallest and Pick-Largest at high coverage ratios

	Coverage (%)	Mushroom	DBLP	DNA	Paleo
Rem.- Smallest	90	76	15	243	107
	95	85	17	292	112
	98	100	19	332	132
Pick-Largest	90	47	15	197	107
	95	62	17	242	112
	98	81	19	285	132

Table 4 Performance comparison

	GreConD [4]	GreEss [4]	Remove-S.	Pick-L.
Time	18 min 5.7 s	12.47 s	7.39 s	10.71 s
Memory	97 MB	2 MB	2.25 MB	1.72 MB

Table 5 Running platforms

	Ours	Belohlavek et al.'s [5]
CPU	AMD Athlon X2 350 Dual Core Processor (3.5 GHz)	INTEL Xcon 4 (3.2 GHz)
Memory	4 GB (1.6 GHz)	1 GB
OS	Windows 7 Professional	Not mentioned in [5]
Program	MATLAB Version R2012b	MATLAB (partially hand-coded in C)

6 Application to educational data mining

Educational data mining has been attracting increasing interest in recent years. It aims to discover students' mastery of knowledge, or skills which are itemized as *attributes*. In the widely studied *Rule Space Model* [43] in cognitive assessment in education, a Boolean matrix, named the *Q-matrix*, is used to represent hypothetical sets of attributes which would be needed to answer the test items correctly. To explain the relevance of exact BMD to the educational *Q-matrix theory* developed by Tatsuoka [42], let us introduce new symbols for matrices.

Attribute or skill set: We assume that the students' *knowledge* can be represented by the *knowledge state matrix* $A = [a_{ij}] \in \{0,1\}^{m\times k}$, where $a_{ij} = 1$ (resp. $a_{ij} = 0$) indicates that the ith student possesses (resp. does not possess) knowledge represented by the jth attribute. For $i = 1, 2, \ldots, m$, the *knowledge state* of student i is represented by a row vector

$$\boldsymbol{a}_i = [a_{i1}, a_{i2}, \ldots, a_{ik}].$$

Q-matrix: It is denoted by $Q = [q_{ij}] \in \{0,1\}^{n\times k}$, where $q_{ij} = 1$ (resp. $q_{ij} = 0$) indicates that answering test item i correctly requires (resp. does not require) knowing or understanding attribute (=*concept*) j. Define a row vector by

$$\boldsymbol{q}_i = [q_{i1}, q_{i2}, \ldots, q_{ik}].$$

Response matrix: Given m students and n test items, the test results can be represented by a matrix $R \in \{0,1\}^{m\times n}$, where $R[i,j] = 1$ (resp. $R[i,j] = 0$) indicates that the ith student solved the jth test item correctly (resp. incorrectly). Theoretically, student i should be able to answer question j if $\boldsymbol{a}_i \geq \boldsymbol{q}_j$ or $\overline{\boldsymbol{a}_i} \circ \boldsymbol{q}_j = 0$. We thus define the *ideal item response* $R[i,j]$ by

$$R[i,j] = \begin{cases} 1 & \boldsymbol{a}_i \geq \boldsymbol{q}_j \\ 0 & otherwise \end{cases} \tag{25}$$

If both Q and A were known, then the students' test performance, called the *ideal item response pattern* [43], could be theoretically predicted. The following result was announced in [41] without proof. Here we provide a simple but formal proof.

Theorem 5 *The ideal item response matrix R, the knowledge state matrix A and the Q-matrix Q are related as follows:*

$$R = \overline{\overline{A} \circ Q^{\mathrm{T}}}. \tag{26}$$

Proof The fact that student i has enough knowledge to answer question j can be represented by $\boldsymbol{a}_i \geq \boldsymbol{q}_j$, which is equivalent to $\overline{\boldsymbol{a}}_i \circ \boldsymbol{q}_j^{\mathrm{T}} = 0$ hence $\overline{\overline{\boldsymbol{a}_i} \circ \boldsymbol{q}_j^{\mathrm{T}}} = 1$ by Proposition 5. If he/she does not, i.e., $\boldsymbol{a}_i \not\geq \boldsymbol{q}_j$, on the other hand, then $\overline{\boldsymbol{a}}_i \circ \boldsymbol{q}_j^{\mathrm{T}} = 1$, and $\overline{\overline{\boldsymbol{a}_i} \circ \boldsymbol{q}_j^{\mathrm{T}}} = 0$. □

If R is given but the underlying matrices Q and A are unknown, we want to mine Q and A out of R. Thanks to Theorem 5, by finding decomposition $\overline{R} = \overline{A} \circ Q^{\mathrm{T}}$, we can learn students' knowledge state matrix A and the Q-matrix Q from the test responses in R. We simply set $M = \overline{R}$, $U = \overline{A}$, and $V = Q^{\mathrm{T}}$, and decompose M. Thus, the Q-matrix learning problem can be transformed into *exact* (i.e., not approximate) Boolean matrix decomposition problem. Here we assume that R has no "noise," namely it correctly represents the students' knowledge, and mine Q and A from it. Clearly, the set of collected test responses, $\mathcal{R}$, is likely to be "noisy," because students may be able to guess correct answers by luck, or may make silly mistakes (called "slips" [43]). Therefore, the discovered factors A' and Q' of $\mathcal{R}$ are just approximations to the true A and Q. This problem is a main issue in *Rule space* model [28,41–43,50], but is beyond the scope of this paper.

Example 4 Here we use the dataset of Example 3.9 in [43]. Table 6 shows the ideal item response pattern matrix R for $m = 12$ students and $n = 11$ test items,

The knowledge state matrix $A^{12\times4}$ and Q-matrix $Q^{11\times4}$ (each with $k = 4$ attributes) are given by

Table 6 The ideal item response matrix $R^{12\times 11}$ [43]

	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	0	1	0	1	0	0	0	0
3	1	1	0	1	0	1	0	1	0	0	0
4	1	1	0	0	0	0	0	0	0	0	0
5	1	0	1	1	1	1	0	0	1	1	0
6	1	0	1	0	1	0	0	0	0	0	0
7	1	0	0	1	0	1	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0
9	0	0	1	1	0	0	0	0	1	0	0
10	0	0	1	0	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0

$$A=\begin{pmatrix}1&1&1&1\\1&1&1&0\\1&1&0&1\\1&1&0&0\\1&0&1&1\\1&0&1&0\\1&0&0&1\\1&0&0&0\\0&0^*&1&1\\0&0^*&1&0\\0&0^*&1&0\\0&0^*&0&0\end{pmatrix};\quad Q=\begin{pmatrix}1&0&0&0\\1&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&1&0\\1&0&0&1\\1&1&1&0\\1&1&0&1\\0&0&1&1\\1&0&1&1\\1&1&1&1\end{pmatrix}$$

In [43], they constructed R from the given A and Q. Here, taking R as the input, Algorithms `Remove-Smallest` and `Pick-Largest` were able to recover A and Q.

Comment 6 *In the above example, note that $Q[:,1]$ dominates column $Q[:,2]$. This means that any test item that tests concept 2 automatically tests concept 1, in other words, attribute 1 is a prerequisite for concept 2* [43]. *Students 9 to 12 have not mastered concept 1, which are tested in test items 1,2, 5∼8, and 10∼11. Thus $R[s,1]=0$ (they cannot answer test items testing concept 1) for $s=9\sim12$. As for any test items that has a 0 in both columns 1 and 2 of Q, i.e., $Q[3,:]$, $Q[4,:]$, and $Q[9,:]$, students 9∼12 (who haven't mastered concept 1) cannot pass test items testing concept 2. Therefore, $A[s,2]=0$ for $s=9\sim12$. However, mathematically, setting $A[s,2]=1$ for $s=9\sim12$ still satisfies $\overline{R}=\overline{A}\circ Q^{\mathrm{T}}$. See the entries 0^* in matrix A in Example 4.*

In general, we can prove the following.

Lemma 6 *Suppose that column $Q[:,i]$ dominates column $Q[:,j]$. Then $[A[s,i]=0]\Rightarrow[A[s,j]=0]$.*

The input to our algorithms is just $\overline{R}$, and the complemented knowledge state matrix $\overline{A}$ is an output. Algorithm

Table 7 The attribute picked in each round of `Pick-Largest`

Round	1	2	3	4	5
$\max_i\{\delta_i\}$	32	30	19	9	0
$\operatorname{argmax}_i\{\delta_i\}$	1	4	3	2	5–11

`Pick-Largest` computes the values of δ_i in each round, whose maxima are shown in Table 7.

Algorithm `Remove-Smallest` removes attributes in the increasing order of σ_i, provided the product remains the same, i.e., $\overline{R}$. For this example, both algorithms decompose $\overline{R}$ into factor matrices with the common dimension ($k=4$), which equals the dimension of the original factors [43].

7 Conclusions and discussions

We have presented two heuristic algorithms to find an exact decomposition $M=U\circ V$ such that U consists of a subset of the columns of M. *Exact* BMD is aesthetically pleasing and intellectually satisfying, and we believe that it will find useful applications in the future. In the present day data mining applications, however, it may not be necessary or very important. So we also showed that our algorithms can be used for approximate BMD, namely to find a product $U\circ V$ that covers most of the 1's and no 0's in M. This is sometimes called "from-below" approximation [4].

We ran our algorithms on several real datasets, which are often used as benchmarks. On these particular datasets, our algorithms perform rather well, compared with the known algorithms proposed in [4,5,13,49]. These results are despite the column-use condition that we impose, but the others do not. We think this fact is rather note-worthy. If this is generally true for large databases, it has a great potential for practical data mining. Clearly, more extensive tests are called for to arrive at any definite conclusions. Ene et al. [9] also report some unexpected, favorable properties of real datasets, which help role mining. It would be interesting to explore and understand how they are caused. Incidentally, when the column-use condition is imposed, it seems that the idea of concept lattice [12] is not particularly useful.

We have made an interesting observation that the sizes of the largest, second largest tiles, etc., picked by `Pick-Largest` follow Zipf's distribution rather well.

Although we have concentrated on the elimination of column dominance, it is possible that a given matrix M has more row dominance than column dominance. In any case, it would be worthwhile to apply our algorithms to both M and M^{T}, and pick the result with the smaller factor matrix dimensions. There may be situations where a decomposition of $M=A\circ B$ is already known, but it is desired to reduce the number of attributes (columns) in A. In such a case, we can apply our algorithms to decompose A as $A = U \circ V$. We then have $M = U \circ (V \circ B)$ such that U consists of a subset of the columns of A.

From Proposition 3, there is a lot of parallelism in matrix product computation. This implies that if the given matrix M is very large, our algorithms are amenable to the *map-reduce* technique [39].

Finally, as mentioned before, we have not examined the approximation ratio of our heuristic algorithms relative to the optimum. We leave it as future work.

Acknowledgments We thank Martin Trnecka of Palacký University, Czech Republic, for providing us with the testbed datasets tailored as inputs to MATLAB. We also thank the reviews for a number of useful critiques.

References

1. Amilhastre, J., Vilarem, M., Janssen, P.: Complexity of minimum biclique cover and minimum biclique decomposition for bipartite domino-free graphs. Discrete Appl. Math. **86**, 125–144 (1998)
2. Barnes, T.: Novel derivation and application of skill matrices: the q-matrix method. In: Romero, C., Ventura, S., Pechenizkiy, M., Baker, R. (eds.) Handbook on Educational Data Mining, Chap. 11, pp. 159–172. CRC Press, Florida (2010)
3. Berry, M., Browne, M., Langville, A., Pauca, V., Plemmons, R.: Algorithms and applications for approximate nonnegative matrix factorization. Comput. Stat. Data Anal. **52**(1), 155–173 (2007)
4. Bělohlávek, R., Trnečka, M.: From-below approximations in boolean matrix factorization: geometry and new algorithm. J. Comput. Syst. Sci. **81**(8), 1678–1697 (2015)
5. Bělohlávek, R., Vychodil, V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. J. Comput. Syst. Sci. **76**(1), 3–20 (2010)
6. Doherty, F., Lundgren, J., Siewert, D.: Biclique covers and partitions of bipartite graphs and digraphs and related matrix ranks of 0, 1-matrices. Congr. Numerantium **136**(2), 73–96 (1999)
7. Drineas, P., Kannan, R., Mahoney, M.: Fast Monte Carlo algorithms for matrices III: computing a compressed approximate matrix decomposition. SIAM J. Comput. **36**(1), 184–206 (2006)
8. Drineas, P., Mahoney, M., Muthukrishnan, S.: Relative-error CUR matrix decompositions. SIAM J. Matrix Anal. Appl. **30**(2), 844–881 (2008)
9. Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.: Fast exact and heuristic methods for role minimization problems. In: Proceedings ACM Symposium on Access Control Models and Technologies, pp. 1–10 (2008)
10. Feige, U.: A threshold of ln n for approximating set cover. J. ACM **45**(4), 634–652 (1998)
11. Franzblau, D., Kleitman, D.: An algorithm for covering polygons with rectangles. Inform. Control **63**, 164–189 (1984)
12. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
13. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Discovery Science. No. 3245 in LNCS, pp. 278–289. Springer (2004)
14. Golub, G., Van Loan, C.: Matrix Computations. Johns Hopkins University Press, Baltimore (1996)
15. Gregory, D., Pullman, N.: Semiring rank: Boolean rank and nonnegative rank factorizations. J. Combin. Inform. Syst. Sci. **8**, 223–233 (1983)
16. Grünwald, P.: The Minimum Description Length Principle. MIT Press, Cambridge (2007)
17. Hochbaum, D.: Approximating clique and biclique problems. J. Algorithms **29**(1), 174–200 (1998)
18. Hyvönen, S., Miettinen, P., Terzi, E.: Interpretable nonnegative matrix decompositions. In: Proceedings 14th ACM International Conference on Knowledge Discovery & Data Mining (KDD), pp. 345–353 (2008)
19. Keprt, A., Snášel, V.: Binary factor analysis with help of formal concepts. In: Proceedings CEUR Workshop, vol. 110, pp. 90–101 (2004)
20. Kim, K.: Boolean Matrix Theory and Applications. M. Dekker, New York (1982)
21. Koedinger, K., McLaughlin, E., Stamper, J.: Automated student model improvement. In: Proceedings 5th International Conference on Educational Data Mining (2012)
22. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, New York (1996)
23. Lan, G., DePuy, G., Whitehouse, G.: An effective and simple heuristic for the set covering problem. Eur. J. Oper. Res. **176**, 1387–1403 (2007)
24. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings 39th International Symposium on Symbolic and Algebraic Computation (ISSAC) (2014)
25. Lee, D., Seung, H.: Learning the parts of objects by non-negative matrix factorization. Nature **401**, 788–791 (1999)
26. Lee, D., Seung, H.: Algorithms for non-negative matrix factorization. Adv. Neural Inf. Process. Syst. **13**, 556–562 (2001)
27. Lichman, M.: UCI machine learning repository. Technical Report, School of Information and CS, University of California, Irvine, CA (2013). http://www.ics.uci.edu/ml
28. Liu, J., Xu, G., Ying, Z.: Data-driven learning of q-matrix. Appl. Psychol. Meas. **36**(7), 548–564 (2012)
29. Lubiw, A.: The Boolean basis problem and how to cover some polygons by rectangles. SIAM J. Discrete Math. **3**(1), 98–115 (1990)
30. Lubiw, A.: A weighted min–max relation for intervals. J. Combin. Theory **53**(2), 151–172 (1991)
31. Miettinen, P.: The boolean column and column–row matrix decompositions. Data Min. Knowl. Discov. **17**, 39–56 (2008)
32. Miettinen, P.: Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms. Ph.D. thesis, University of Helsinki, Helsinki (2009)
33. Miettinen, P.: On finding joint subspace boolean matrix factorizations. In: Proceedings 12th SIAM International Conference on Data Mining (SDM), pp. 954–965 (2012)
34. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE Trans. Knowl. Data Eng. **20**(10), 1348–1362 (2008)
35. Müller, H.: On edge perfectness and classes of bipartite graphs. Discrete Math. **149**, 159–187 (1996)
36. Myllykangas, S., Himberg, J., Bhling, T., Nagy, B., Hollmén, J., Knuutila, S.: DNA copy number amplification profiling of human neoplasms. Oncogene **25**(55), 7324–7332 (2006)
37. Nau, D., Markowsky, G., Woodbury, M., Amos, D.: A mathematical analysis of human leukocyte antigen serology. Math. Biosci. **40**, 243–270 (1978)
38. Orlin, J.: Contentment in graph theory: covering graphs with cliques. Indag. Math. **80**(5), 406–424 (1977)
39. Rajaraman, A., Leskovec, J., Ullman, J.: Mining of Massive Datasets, 2nd edn. Cambridge University Press, New York (2014)
40. Streich, A., Frank, M., Basin, D., Buhmann, J.: Multi-assignment clustering for Boolean data. In: Proceedings International Conference on Machine Learning (ICML), pp. 969–976 (2009)
41. Sun, Y., Ye, S., Inoue, S., Sun, Y.: Alternating recursive method for Q-matrix learning. In: Proceedings 7th International Conference on Educational Data Mining (EDM), pp. 14–20 (2014)

42. Tatsuoka, C.: Data-analytic methods for latent partially ordered classification models. Appl. Stat. (JRSS-C) **51**, 337–350 (2002)
43. Tatsuoka, K.: Cognitive Assessment: An Introduction to the Rule Space Method. Routledge, New York (2009)
44. Umetani, S., Yagiura, M.: Relaxation heuristic for the set covering problem. J. Oper. Res. Soc. Jpn. **50**(4), 350–375 (2007)
45. Vaidya, J.: Boolean matrix decomposition problem: theory, variations and applications to data engineering. In: Proceedings IEEE 28th International Conference on Data Eng, pp. 1222–1224 (2012)
46. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: finding a minimal descriptive set of roles. In: Proceedings ACM Symposium Access Control Models and Technologies, pp. 175–184 (2007)
47. Vavasis, S.: On the complexity of nonnegative matrix factorization. SIAM J. Optim. **20**, 1364–1377 (2010)
48. Williams, V.: Multiplying matrices faster than Coppersmith–Winograd. In: Proceedings 44th ACM Symposium Theory of Computing (STOC), pp. 887–898 (2012)
49. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.: Summarizing transactional databases with overlapped hyperrectangles. Data Min. Knowl. Discov. **23**, 215–251 (2011)
50. Zhang, S., DeCarlo, L., Ying, Z.: Non-identifiability, Equivalence Classes, and Attribute-Specific Classification in Q-Matrix Based Cognitive Diagnosis Models. Technical Report, Columbia University (2013)