Delegation-Relegation for Boolean Matrix Factorization

Florent Avellaneda[1, 2], **Roger Villemaire**[1]

[1] Université du Québec à Montréal (UQAM), Montréal, Canada
[2] Centre de Recherche de l'Institut Universitaire de Gériatrie de Montréal, Montréal, Canada
avellaneda.florent@uqam.ca, villemaire.roger@uqam.ca

Abstract

The Boolean Matrix Factorization (BMF) problem aims to represent a $n \times m$ Boolean matrix as the Boolean product of two matrices of small rank k, where the product is computed using Boolean algebra operations. However, finding a BMF of minimum rank is known to be NP-hard, posing challenges for heuristic algorithms and exact approaches in terms of rank found and computation time, particularly as matrix size or the number of entries equal to 1 grows.

In this paper, we present a new approach to simplifying the matrix to be factorized by reducing the number of 1-entries, which allows to directly recover a Boolean factorization of the original matrix from its simplified version. We introduce two types of simplification: one that performs numerous simplifications without preserving the original rank and another that performs fewer simplifications but guarantees that an optimal BMF on the simplified matrix yields an optimal BMF on the original matrix. Furthermore, our experiments show that our approach outperforms existing exact BMF algorithms.

1 Introduction

The problem of Boolean Matrix Factorization (BMF) is to represent a $n \times m$ matrix $\mathbf{X}$ as the Boolean product of an $n \times k$ and $k \times m$ matrices $\mathbf{A} \circ \mathbf{B}$. The Boolean semiring operator, denoted by $\circ$, represents a matrix multiplication in which the product is computed using the logical "AND" operator and the addition is computed using the logical "OR" operator. By searching for matrices $\mathbf{A}$ and $\mathbf{B}$ of small rank k, the BMF allows us to represent the initial matrix $\mathbf{X}$ in a concise way. BMF is widely used in various fields such as data mining (Wicker, Pfahringer, and Kramer 2012), role mining (Ene et al. 2008), bioinformatics (Liang, Zhu, and Lu 2020), logic synthesis (Ma, Hashemi, and Reda 2022; Hashemi, Tann, and Reda 2018), and network analysis (Kocayusufoglu, Hoang, and Singh 2018), and has been the subject of numerous research studies (Miettinen and Neumann 2020).

In the context of BMF, two types of problems arise. The first problem is, given a matrix $\mathbf{X}$, finding a BMF of minimal rank. The second problem is, given a matrix $\mathbf{X}$ and a rank k, finding a BMF that is as close as possible to the matrix $\mathbf{X}$. In this paper, we only consider exact solutions in which no error is allowed. Exact solutions are a key concern in application areas such as security role mining (Ene et al. 2008) where a spurious authorization would jeopardize a security policy. Moreover, it is well-known that the set basis and biclique covering problems are equivalent to BMF (Miettinen and Neumann 2020). Therefore, this opens up additional domains that would benefit from exact BMF such as network nodes' service that provide functions bundling (Markopoulou and E. Anagnostou 1998), encryption key management (Shu, Lee, and Yannakakis 2006), and frameproof codes (Hajiabolhassan and Moazami 2012).

In this setting, our objective is to devise methods to find exact BMF with minimal or as close to minimal rank as possible. Since the problem of finding the BMF of minimal rank k is known to be NP-hard (Miettinen et al. 2008), most existing algorithms in the literature aim to minimize the rank k without guaranteeing optimality (Asso (Miettinen et al. 2008), GreConD (Belohlavek and Vychodil 2010), Hyper (Xiang et al. 2011), MEBF (Wan et al. 2020), Nassau (Karaev, Miettinen, and Vreeken 2015), PaNDa+ (Lucchese, Orlando, and Perego 2010), Proximus (Koyutürk and Grama 2003), Tiling (Geerts, Goethals, and Mielikäinen 2004)). However, recent work based on constraint solvers proposed methods for finding optimal BMF (Kovacs, Gunluk, and Hauser 2021; Avellaneda and Villemaire 2022). While these approaches can find optimal factorizations for small matrices only, they can still find low-rank factorization for larger matrices by using relaxations and approximations. Therefore, these constraint solver-based methods are more computationally demanding than traditional approaches, but they generally produce smaller rank BMF. A natural question that arises is: is it possible to simplify the matrices before factorization in order to make the constraint solving algorithms run more efficiently?

The idea of simplifying the matrices to be factorized before performing the factorization is not new and has been introduced in a particular family of BMF algorithms based on *formal concept analysis* (Ganter and Wille 2012). A *formal concept* corresponds to a pair of lines and columns (I, J) such that $\forall i \in I, j \in J, X_{i,j} = 1$. Formal concepts are partially ordered and give rise to a lattice called the *concept lattice*. The BMF problem then coincides with the selection

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

of a set of formal concepts called *concept factors*.

Although this representation does not allow missing values in the matrix to be factorized, it is used by an algorithm called GreEss (Belohlavek and Trnecka 2015) to locate so-called "essential entries". The authors furthermore show that a BMF covering all the essential entries can be transformed into a BMF of equivalent rank covering all the entries of the initial matrix. Although the rank of this factorization is not optimal for the initial matrix, the gain related to the simplification of the matrix allows good execution time and factorization of low rank. This approach has been pushed to the extreme in Iteress (Belohlavek, Outrata, and Trnecka 2019) by applying this simplification process iteratively until a fixed point is reached.

We revisit the concept of matrix simplification used in GreEss and Iteress, but from the perspective of row/column inclusion. In contrast to previous work, which only apply to complete 0/1 matrices, we propose a simplification method that incorporates missing values (don't care) and allows for more extensive simplification than Iteress. Furthermore, we introduce a second line/column inclusion notion that additionally guarantees that an optimal BMF on the simplified matrix yields an optimal BMF on the initial matrix.

The BMF problem can be seen as the problem of covering the matrix's 1-entries by the so-called *blocks*. The rationale for our approach is that reducing the number of 1s in a matrix will simplify their covering and should generally lead to BMF speed-ups. In particular, with constraint-based BMF algorithms, where the presence of 1s in the matrix to be factorized generate constraints that are harder to satisfy than missing values or 0s, we show that our simplification approach can significantly reduce the computational time.

The paper is structured as follows. First, to formalize the approach, we present definitions related to BMF and introduce the concept of matrix inclusion (Section 2). Then, in Section 3, we define a simplification operator, which we call "delegation", and show how to obtain a BMF on the original matrix from a BMF on a simplified matrix using the "relegation" operator. Algorithms based on the delegation and relegation operator are then proposed in Section 4. Finally, we report the results of several experiments in Section 5.

2 Notation

We denote by $\mathbf{X}_{m\times n}$ a *Boolean matrix* $\mathbf{X} \in \{0, 1, \emptyset\}^{n\times m}$ with m rows, and n columns. We use $\emptyset$ to represent missing data and if $\mathbf{X} \in \{0, 1\}^{n\times m}$ we say that the matrix $\mathbf{X}$ is *complete*. We also use the notation $\mathbf{X}_{i,j}$ to represent the entry in the i-th row and the j-th column and the notation $\mathbf{X}_{i,:}$ and $\mathbf{X}_{:,j}$, to represent the i-th row and the j-th column of $\mathbf{X}$, respectively. We now formally define the Boolean product of two matrices.

Definition 2.1. *The* Boolean product *of two complete matrices $\mathbf{A}_{m\times k}$ and $\mathbf{B}_{k\times n}$ is a matrix $(\mathbf{A} \circ \mathbf{B})_{m\times n}$ defined by:*

$$(\mathbf{A} \circ \mathbf{B})_{i,j} = \bigvee_{\ell=1}^{k} (\mathbf{A}_{i,\ell} \wedge \mathbf{B}_{\ell,j})$$

This definition is similar to the classical matrix product, where the product is replaced by the logical "AND" and the addition is replaced by the logical "OR".

Note that the Boolean product of two rank-k matrices can also be seen as the union of k rank-1 matrices called *blocks*:

$$(\mathbf{A} \circ \mathbf{B}) = \bigvee_{\ell=1}^{k} (\mathbf{A}_{:,\ell} \circ \mathbf{B}_{\ell,:})$$

where the $\bigvee$ operator applies entrywise.

Therefore, the BMF problem corresponds to finding a set of blocks that cover all the 1s of the matrix and do not cover any 0s. The idea developed in this paper is that blocks that cover certain 1s can be "extended" to cover some other specific 1s. Therefore, the 1s that can be covered by "extending" these blocks can be ignored and removed from the matrix. To define these specific 1s, we introduce the concept of matrix inclusion.

Definition 2.2. *A matrix $\mathbf{X}'_{m\times n}$ is* existentially included *in a matrix $\mathbf{X}_{m\times n}$ (denoted $\mathbf{X}' \leq^{\exists} \mathbf{X}$) if there is no i, j such that $\mathbf{X}'_{i,j} = 1$ and $\mathbf{X}_{i,j} = 0$.*

Intuitively, this definition means that it is possible to fill in the missing values, represented by $\emptyset$, in both matrices $\mathbf{X}$ and $\mathbf{X}'$ such that the inequality $\mathbf{X}'_{i,j} \leq \mathbf{X}_{i,j}$ holds true for all indices i, j.

Definition 2.3. *A matrix $\mathbf{X}'_{m\times n}$ is* universally included *in a matrix $\mathbf{X}_{m\times n}$ (denoted $\mathbf{X}' \leq^{\forall} \mathbf{X}$) if for all i, j, if $\mathbf{X}_{i,j} = 0$, then $\mathbf{X}'_{i,j} = 0$.*

Intuitively, this definition means that regardless of how we fill in the missing values $\emptyset$ in $\mathbf{X}'$, we can always find a way to fill in the missing values $\emptyset$ in $\mathbf{X}$ such that $\mathbf{X}'_{i,j} \leq \mathbf{X}_{i,j}$ holds true for all indices i, j.

Definition 2.4. *A matrix $\mathbf{X}'_{m\times n}$ is* consistent *with a matrix $\mathbf{X}_{m\times n}$ (denoted $\mathbf{X}' \simeq \mathbf{X}$) if $\mathbf{X} \leq^{\exists} \mathbf{X}'$ and $\mathbf{X}' \leq^{\exists} \mathbf{X}$.*

Intuitively, this definition means that it is possible to fill in the missing values $\emptyset$ in both $\mathbf{X}$ and $\mathbf{X}'$ such that for all i, j, we have $\mathbf{X}'_{i,j} = \mathbf{X}_{i,j}$.

Definition 2.5. *Two complete matrices $\mathbf{A}_{m\times k}$ and $\mathbf{B}_{k\times n}$ are a BMF of the matrix $\mathbf{X}_{m\times n}$ if $(\mathbf{A} \circ \mathbf{B}) \simeq \mathbf{X}$.*

3 Transformation Into Sparse Matrices

In this section, we show how to simplify a matrix to be factorized using the "delegation" operator and demonstrate how to obtain a BMF on the original matrix from a BMF on a simplified matrix using the "relegation" operator.

Definition 3.1. *We denote by $\mathbf{X}^{v\downarrow w}$ the* delegation *of the line w to the line v in the matrix $\mathbf{X}$.*

$$\mathbf{X}^{v\downarrow w}_{i,j} = \begin{cases} 0 & \text{if } i = v \text{ and } \mathbf{X}_{w,j} = 0, \\ \emptyset & \text{if } i = w \text{ and } \mathbf{X}_{v,j} = 1, \\ \mathbf{X}_{i,j} & \text{otherwise.} \end{cases}$$

Definition 3.2. *We denote by $\mathbf{A}^{v\uparrow w}$ the* relegation *of the line w from the line v in the matrix $\mathbf{A}$.*

$$\mathbf{A}^{v\uparrow w}_{i,j} = \begin{cases} 1 & \text{if } i = w \text{ and } \mathbf{A}_{v,j} = 1, \\ \mathbf{A}_{i,j} & \text{otherwise.} \end{cases}$$

We now show that if a line v is existentially included in a line w, then we can obtain a BMF of the original matrix $\mathbf{X}$ from a BMF $\mathbf{A} \circ \mathbf{B}$ on the simplified matrix $\mathbf{X}^{v \downarrow w}$ (Theorem 3.5).

The intuition, illustrated in Figure 1, is as follows. A rank-k BMF can be seen as the union of k rank-1 BMF, the *blocks*. Since each 1 present in the matrix to be factorized must be covered by at least one block, we can see that if a line v is included in a line w, then a block covering a 1 on v can also be extended to cover the line w. Thus, all columns where there is a 1 on the line v and the line w can be removed from the line w. However, to ensure that the line v is indeed included in the line w, we must replace every $\emptyset$ from the line v with zeros if the column j is such that $X_{w,j} = 0$.

$$\begin{array}{l} \;\;0 \\ 0\,1\,1\,\emptyset\,\emptyset\,\emptyset\,0\,0\,0 \quad v \\ 1\,\cancel{1}\,\cancel{1}\,1\,\emptyset\,0\,0\,1\,\emptyset \quad w \end{array}$$

Figure 1: Delegation $\mathbf{X}^{v \downarrow w}$ in the case where a line v is included in a line w.

We now prove this result more formally, using two intermediate propositions.

Proposition 3.3. *Let v, w be such that $\mathbf{X}_{v,:} \leq^{\exists} \mathbf{X}_{w,:}$ and $\mathbf{Y}$ be complete. If $\mathbf{Y} \simeq \mathbf{X}^{v \downarrow w}$ then $\mathbf{Y}^{v \uparrow w} \simeq \mathbf{X}$*

Proof. Let $\mathbf{Y} \simeq \mathbf{X}^{v \downarrow w}$. If $\mathbf{Y}^{v \uparrow w}_{i,j}$, $\mathbf{X}_{i,j}$ take different values in $\{0, 1\}$ we either have that $\mathbf{Y}^{v \uparrow w}_{i,j} \neq \mathbf{Y}_{i,j}$ or $\mathbf{X}_{i,j} \neq \mathbf{X}^{v \downarrow w}_{i,j}$.

If $\mathbf{Y}^{v \uparrow w}_{i,j} \neq \mathbf{Y}_{i,j}$ by definition 3.2 we must have $i = w$, $\mathbf{Y}^{v \uparrow w}_{w,j} = 1$ and $\mathbf{Y}_{v,j} = 1$. Since $\mathbf{Y}^{v \uparrow w}_{w,j} \neq \mathbf{X}_{w,j}$, it follows that $\mathbf{X}_{w,j} = 0$ and by definition 3.1 $\mathbf{X}^{v \downarrow w}_{v,j} = 0$, which contradicts $\mathbf{Y} \simeq \mathbf{X}^{v \downarrow w}$.

If $\mathbf{X}_{i,j} \neq \mathbf{X}^{v \downarrow w}_{i,j}$ by definition 3.1 we either have that $i = v$ with $\mathbf{X}^{v \downarrow w}_{v,j} = 0$ and $\mathbf{X}_{w,j} = 0$ or $i = w$, $\mathbf{X}^{v \downarrow w}_{w,j} = \emptyset$ and $\mathbf{X}_{v,j} = 1$.

In the first case, it follows from $\mathbf{X}_{w,j} = 0$ that $\mathbf{X}_{v,j} = 0$ since this entry is in $\{0, 1\}$ by hypothesis and $\mathbf{X}_{v,:} \leq^{\exists} \mathbf{X}_{w,:}$. Now $\mathbf{Y}^{v \uparrow w}_{v,j} = 1$ since it differs from $\mathbf{X}_{v,j}$. Furthermore, $\mathbf{Y}_{v,j}$ is also 1 since in definition 3.2 line v is left unchanged. But now $\mathbf{X}^{v \downarrow w}_{v,j} = 0$ contradicts $\mathbf{Y} \simeq \mathbf{X}^{v \downarrow w}$.

In the final case, $i = w$, $\mathbf{X}^{v \downarrow w}_{w,j} = \emptyset$ and $\mathbf{X}_{v,j} = 1$, we have from $\mathbf{X}_{v,:} \leq^{\exists} \mathbf{X}_{w,:}$ that $\mathbf{X}_{w,j} = 1$ and hence $\mathbf{Y}^{v \uparrow w}_{w,j} = 0$ since these two last values are different and in $\{0, 1\}$. Definition 3.1 yields, from $\mathbf{X}_{w,j} = 1$, that $\mathbf{X}^{v \downarrow w}_{v,j} = \mathbf{X}_{v,j} = 1$. Now since $\mathbf{Y}^{v \uparrow w}_{w,j} = 0$ it follows from definition 3.2 that $\mathbf{Y}_{v,j} \neq 1$ and is hence 0 but then $\mathbf{Y}_{v,j} \neq \mathbf{X}^{v \downarrow w}_{v,j}$ contradicts $\mathbf{Y} \simeq \mathbf{X}^{v \downarrow w}$. □

Proposition 3.4. *Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices. Then:*

$$(\mathbf{A} \circ \mathbf{B})^{v \uparrow w} = \mathbf{A}^{v \uparrow w} \circ \mathbf{B}$$

Proof. By Definition 3.2, the *relegation* operator only modifies line w, it hence follows that for $i \neq w$, $(\mathbf{A} \circ \mathbf{B})^{v \uparrow w}_{i,j} = (\mathbf{A} \circ \mathbf{B})_{i,j}$ and $\mathbf{A}^{v \uparrow w}_{i,\ell} = \mathbf{A}_{i,\ell}$, hence $(\mathbf{A}^{v \uparrow w} \circ \mathbf{B})_{i,j} = (\mathbf{A} \circ \mathbf{B})_{i,j}$. Let us now consider the entries on lines $i = w$.

By Definition 3.2 we have that $(\mathbf{A} \circ \mathbf{B})^{v \uparrow w}_{w,j} = 1$ exactly when either $(\mathbf{A} \circ \mathbf{B})_{w,j} = 1$ or $(\mathbf{A} \circ \mathbf{B})_{v,j} = 1$.

Similarly, $(\mathbf{A})^{v \uparrow w}_{w,\ell} = 1$ holds exactly when either $\mathbf{A}_{w,\ell} = 1$ or $\mathbf{A}_{v,\ell} = 1$. Furthermore, from Definition 2.1 we have that $(\mathbf{A}^{v \uparrow w} \circ \mathbf{B})_{w,j} = 1$ exactly when $\mathbf{A}^{v \uparrow w}_{w,\ell} = 1$ and $\mathbf{B}_{\ell,j} = 1$ for some ℓ. Combining both, $(\mathbf{A}^{v \uparrow w} \circ \mathbf{B})_{w,j} = 1$ exactly when either $\mathbf{A}_{w,\ell} = 1$ and $\mathbf{B}_{\ell,j} = 1$ or $\mathbf{A}_{v,\ell} = 1$ and $\mathbf{B}_{\ell,j} = 1$ holds, for some ℓ. This therefore holds exactly when either $(\mathbf{A} \circ \mathbf{B})_{w,j} = 1$ or $(\mathbf{A} \circ \mathbf{B})_{v,j} = 1$ and $(\mathbf{A} \circ \mathbf{B})^{v \uparrow w}_{w,j} = (\mathbf{A}^{v \uparrow w} \circ \mathbf{B})_{w,j}$ as required. □

Theorem 3.5. *Let v, w be such that $\mathbf{X}_{v,:} \leq^{\exists} \mathbf{X}_{w,:}$. If $(\mathbf{A} \circ \mathbf{B}) \simeq \mathbf{X}^{v \downarrow w}$ then $(\mathbf{A}^{v \uparrow w} \circ \mathbf{B}) \simeq \mathbf{X}$.*

Proof. By Proposition 3.3, if $(\mathbf{A} \circ \mathbf{B}) \simeq \mathbf{X}^{v \downarrow w}$ then $(\mathbf{A} \circ \mathbf{B})^{v \uparrow w} \simeq \mathbf{X}$. Then, by Proposition 3.4, $(\mathbf{A}^{v \uparrow w} \circ \mathbf{B}) \simeq \mathbf{X}$. □

Now, let us show that if the line v is universally included in the line w, then finding an optimal BMF on the original matrix $\mathbf{X}$ is equivalent to finding an optimal BMF on the simplified matrix $\mathbf{X}^{v \downarrow w}$ (Theorem 3.7). The intuition here is that if the line v is universally included in the line w, no $\emptyset$ of the line v will have to be changed to 0 during the delegation operation (see Figure 1). Thus, no assumptions about $\emptyset$ will be made, only reductions of 1s that we know can be covered by relegation. We prove this theorem using an intermediate lemma.

Lemma 3.6. *Let v, w be such that $\mathbf{X}_{v,:} \leq^{\forall} \mathbf{X}_{w,:}$. There exists a k-rank BMF on $\mathbf{X}$ if and only if there exists a k-rank BMF on $\mathbf{X}^{v \downarrow w}$.*

Proof. If $(\mathbf{A} \circ \mathbf{B}) \simeq \mathbf{X}^{v \downarrow w}$ then, since $\mathbf{X}_{v,:} \leq^{\forall} \mathbf{X}_{w,:}$ implies $\mathbf{X}_{v,:} \leq^{\exists} \mathbf{X}_{w,:}$, by Theorem 3.5, we know that $(\mathbf{A}^{v \uparrow w} \circ \mathbf{B}) \simeq \mathbf{X}$.

Conversely, if $(\mathbf{A} \circ \mathbf{B}) \simeq \mathbf{X}$, let us show that $(\mathbf{A} \circ \mathbf{B}) \simeq \mathbf{X}^{v \downarrow w}$. By Definition 2.3 $\mathbf{X}_{v,:} \leq^{\forall} \mathbf{X}_{w,:}$ means that for all j, $\mathbf{X}_{v,j} = 0$ when $\mathbf{X}_{w,j} = 0$. Thus, by applying $\mathbf{X}^{v \downarrow w}$, the only changes will be the replacement of some 1s by $\emptyset$ (Definition 3.1). Thereby, $(\mathbf{A} \circ \mathbf{B}) \simeq \mathbf{X}^{v \downarrow w}$. □

Theorem 3.7. *Let v, w be such that $\mathbf{X}_{v,:} \leq^{\forall} \mathbf{X}_{w,:}$. $(\mathbf{A}^{v \uparrow w} \circ \mathbf{B})$ is an optimal BMF for $\mathbf{X}$ if and only if $(\mathbf{A} \circ \mathbf{B})$ is an optimal BMF for $\mathbf{X}^{v \downarrow w}$.*

Proof. Direct application of Lemma 3.6. □

We now introduce the analogous notions for columns, and extend the previous properties to the case of column inclusion.

Definition 3.8. *We denote $\mathbf{X}^{v \to w}$ the* delegation *of the column w to the column v in the matrix $\mathbf{X}$.*

$$\mathbf{X}^{v\to w}_{i,j} = \begin{cases} 0 & \text{if } j = v \text{ and } \mathbf{X}_{i,w} = 0, \\ \emptyset & \text{if } j = w \text{ and } \mathbf{X}_{i,v} = 1, \\ \mathbf{X}_{i,j} & \text{otherwise.} \end{cases}$$

Definition 3.9. *We denote $\mathbf{B}^{v\leftarrow w}$ the* relegation *of the column w from the column v in the matrix $\mathbf{B}$.*

$$\mathbf{B}^{v\leftarrow w}_{i,j} = \begin{cases} 1 & \text{if } j = w \text{ and } \mathbf{B}_{i,v} = 1, \\ \mathbf{B}_{i,j} & \text{otherwise.} \end{cases}$$

Theorem 3.10. *Let v, w be such that $\mathbf{X}_{:,v} \leq^{\exists} \mathbf{X}_{:,w}$. If $(\mathbf{A} \circ \mathbf{B}) \simeq \mathbf{X}^{v\to w}$ then $(\mathbf{A} \circ \mathbf{B}^{v\leftarrow w}) \simeq \mathbf{X}$.*

Proof. Similar to Theorem 3.5. □

Theorem 3.11. *Let v, w such that $\mathbf{X}_{:,v} \leq^{\forall} \mathbf{X}_{:,w}$. $(\mathbf{A} \circ \mathbf{B}^{v\leftarrow w})$ is an optimal BMF for $\mathbf{X}$ if and only if $(\mathbf{A} \circ \mathbf{B})$ is an optimal BMF for $\mathbf{X}^{v\to w}$.*

Proof. Similar to Theorem 3.7. □

4 Algorithm

In this section, we leverage the theorems from the previous section to build a BMF algorithm. This algorithm (Algorithm 2) consists of simplifying the initial matrix using the delegation operation, applying a BMF algorithm on the simplified matrix, and then transforming this factorization using the relegation operation to obtain a factorization of the initial matrix. In the rest of this paper, we say that a matrix $\mathbf{X}$ is *universally simplified* if we apply this Algorithm 2 to the matrix $\mathbf{X}$ with the operator $\forall$, and *existentially simplified* if we apply this Algorithm 2 to the matrix $\mathbf{X}$ with the operator $\exists$. To avoid adding unnecessary delegation, the algorithm only applies delegation when at least one 1 entry is replaced by an $\emptyset$. Algorithm 1 therefore first performs the line delegation each time it detects that a line is included in a second one and replaces a 1 by a $\emptyset$ in a least one column, then performs the same operation for the column delegation. This process is performed as long as there are lines or columns that can be delegated. The inclusion detection can be performed using either universal or existential inclusion, depending on whether the goal is to preserve the rank of the original matrix or perform more simplifications.

Example 4.1. *Consider the matrix of Figure 2. Line 1 is universally included in lines 4 and 5 and a delegation of these two lines to line 1 can be performed, thereby removing six 1s, and replacing them with the emptyset value as shown in the left matrix of Figure 3. There is no more further universal simplifications between the lines of this resulting matrix. For existential simplification, line 1 is also existentially included in lines 4 and 5 but now we furthermore have that line 2 is existentially included in lines 3 and 5. Performing these existential simplifications yields the further existential inclusion of line 3 in line 4 and of line 4 in line 3. Performing the first of these two last existential simplifications leads to the right matrix in Figure 3. There is then no more existential simplifications to apply.*

Note that the simplified matrix we obtain will depend on the order in which the delegations are made. After performing all the line delegations, the algorithm delegates columns, leading to the matrices in Figure 4. From this example, we see that the existentially simplified matrix has fewer 1-entries (3) than the universally simplified matrix (7), which should yield a simpler BMF problem. However, as seen before, the minimum rank on the universally simplified matrix is guaranteed to be the same as the minimum rank of the initial matrix, while this is not necessarily the case for the existential simplification.

$$\begin{array}{c|c} & \begin{matrix} j_1 & j_2 & j_3 & j_4 & j_5 & j_6 \end{matrix} \\ \begin{matrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \end{matrix} & \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & \emptyset & \emptyset & 1 \\ 1 & 1 & \emptyset & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

Figure 2: Example of a Boolean matrix to factorize.

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & \emptyset & \emptyset & 1 \\ 1 & 1 & \emptyset & 0 & 1 & 1 \\ 0 & 0 & \emptyset & \emptyset & 1 & \emptyset \\ 1 & 1 & \emptyset & \emptyset & 0 & \emptyset \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & \mathbf{0} & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{0} & \emptyset & 0 & 1 & \emptyset \\ 0 & 0 & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & 0 & \emptyset \end{pmatrix}$$

Figure 3: Matrix from Figure 2 after applying $\mathbf{X}^{v\downarrow w}$ for every $\mathbf{X}_{v,:} \leq^{\forall} \mathbf{X}_{w,:}$ at left and for every $\mathbf{X}_{v,:} \leq^{\exists} \mathbf{X}_{w,:}$ at right.

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & \emptyset \\ 1 & \emptyset & 0 & \emptyset & \emptyset & \emptyset \\ 1 & \emptyset & \emptyset & 0 & 1 & \emptyset \\ 0 & 0 & \emptyset & \emptyset & 1 & \emptyset \\ 1 & \emptyset & \emptyset & \emptyset & 0 & \emptyset \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 & \emptyset & 0 & \emptyset \\ 1 & \emptyset & 0 & \mathbf{0} & \mathbf{0} & \emptyset \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 1 & \emptyset \\ 0 & 0 & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & 0 & \emptyset \end{pmatrix}$$

Figure 4: Left matrix from Figure 3 after applying $\mathbf{X}^{v\to w}$ for every $\mathbf{X}_{:,v} \leq^{\forall} \mathbf{X}_{:,w}$ at left and right matrix from Figure 3 for every $\mathbf{X}_{:,v} \leq^{\exists} \mathbf{X}_{:,w}$ at right.

Theorem 4.2. *Algorithm 2 is correct, i.e., it correctly produces a BMF of Matrix $\mathbf{X}$ and this for both values $OP \in \{\exists, \forall\}$.*

Proof. First, Algorithm 1 applies delegation (both on lines and columns) in a greedy fashion by stacking the applied delegations in LIFO $Delegation$. The resulting matrix and delegation LIFO are then affected in $\mathbf{X}'$ and $Delegation$ in line 2 of Algorithm 2. Algorithm 2 then proceeds to relegate in the inverse order of the delegations, which by an iterative use of Theorems 3.5 and 3.10 yields a BMF of the original matrix $\mathbf{X}$. □

Theorem 4.3. *Algorithm 2 with $OP = \forall$ returns an optimal BMF whenever the BMF routine returns optimal BMF.*

Algorithm 1: Simpli^{OP}

Input: Matrix $\mathbf{X}$, $OP \in \{\exists, \forall\}$
$Delegation \leftarrow LIFO()$
repeat
  $noChange \leftarrow true$
  while $\exists v, w, j$ such that $\mathbf{X}_{v,:} \leq^{OP} \mathbf{X}_{w,:}$ and $\mathbf{X}_{v,j} = \mathbf{X}_{w,j} = 1$ **do**
    $\mathbf{X} \leftarrow \mathbf{X}^{v \downarrow w}$
    $Delegation.push(v \downarrow w)$
    $noChange \leftarrow false$
  end while
  while $\exists v, w, i$ such that $\mathbf{X}_{:,v} \leq^{OP} \mathbf{X}_{:,w}$ and $\mathbf{X}_{i,v} = \mathbf{X}_{i,w} = 1$ **do**
    $\mathbf{X} \leftarrow \mathbf{X}^{v \rightarrow w}$
    $Delegation.push(v \rightarrow w)$
    $noChange \leftarrow false$
  end while
until $noChange$ is $true$
return $\mathbf{X}, Delegation$

Algorithm 2: BMF_through_simplified_matrix

Input: Matrix $\mathbf{X}$, $OP \in \{\exists, \forall\}$
$\mathbf{X}', Delegation \leftarrow Simpli^{OP}(\mathbf{X})$
$\mathbf{A}, \mathbf{B} \leftarrow BMF(\mathbf{X}')$
while $Delegation$ is not empty **do**
  if $(v \downarrow w) \leftarrow Delegation.pop()$ **then**
    $\mathbf{A} \leftarrow \mathbf{A}^{v \uparrow w}$
  end if
  if $(v \rightarrow w) \leftarrow Delegation.pop()$ **then**
    $\mathbf{B} \leftarrow \mathbf{B}^{v \leftarrow w}$
  end if
end while
return $\mathbf{A}, \mathbf{B}$

Proof. The analysis proceeds as in the proof of Theorem 4.2 expect that Lemma 3.6 guarantees that the rank of the matrix after a universal simplification remains identical to the rank of the initial matrix. Therefore, an optimal BMF algorithm will find a factorization of this rank for the simplified matrix, and since the relegation operation does not increase the rank of the factorization, it will find an optimal BMF for the initial matrix. □

5 Experimentation

Our algorithms were implemented in C++[1], and we performed the experiments on an Intel® Gold 6148 Skylake processor using a single thread and 32 Go of RAM.

We conducted an evaluation of our methods, Simpli$^\exists$ and Simpli$^\forall$, focusing on two key aspects: the degree of simplification they achieve and their effect on the time savings when performing factorizations on the simplified matrices using existing constraint-based BMF solvers. We first evaluated the performance of our methods on well-established datasets from the literature and then on synthetic datasets.

[1] https://github.com/FlorentAvellaneda/Delegation_BMF

5.1 Classic Medium Datasets

We considered 31 classic datasets from the literature, namely: "Audiology", "Autism", "Balance Scale", "Brest Cancer", "Car Evaluation", "Chess", "Contraceptive Method Choice", "Dermatology", "Firewall", "Solar Flare", "Heart Disease", "Hepatitis", "Iris", "Lymphography", "Mushroom", "Nursery", "Website Phishing", "Soybean", "Student Performance", "Thoracic Surgery", "Tic-Tac-Toe Endgame", "Primary Tumor", "Voting Records", "Wine", "Zoo" from UCI (Kelly, Longjohn, and Nottingham 2023), "Americas-small", "Apj", "Customer" from (Ene et al. 2008), "DNA" from (Myllykangas et al. 2006) and "Paleo" from (Žliobaitė et al. 2023).

To evaluate the degree of simplification provided by our method, we compared the number of 1s present in the initial matrix with the number of 1s remaining after applying our algorithms Simpli$^\exists$ and Simpli$^\forall$. We then compared these results with the only matrix simplification algorithm from the literature, Iteress.

As shown in Table 1, our method Simpli$^\exists$ consistently produces simplified matrices with fewer 1s compared to the simplified matrices obtained by Iteress. In regards to the second algorithm, even though Simpli$^\forall$ performs only simplifications that do not increase the rank of the matrix, it still generally produces matrices with fewer 1s compared to the simplified matrices obtained by Iteress.

To evaluate the impact of these simplifications, we compared the performance of two constraint-based BMF solvers: CG (Kovacs, Gunluk, and Hauser 2021) and OptiBlock (Avellaneda and Villemaire 2022). CG is an algorithm based on a Mixed-Integer Programming (MIP) solver that uses a column generation approach, while OptiBlock uses a MaxSAT solver. We set a time limit of 3 hours and recorded the execution time and rank found for CG in Table 1 and for OptiBlock in Table 2. We present results for both solvers using the original matrix and using the matrix simplified by Iteress, Simpli$^\forall$, and Simpli$_0^\exists$. The approach Simpli$_0^\exists$ consists of replacing empty values with zeros in the simplified matrix obtained by Simpli$^\exists$. While this additional simplification could theoretically lead to a degradation in matrix rank, practical observations indicate that such rank degradation is rare. Furthermore, this step often leads to a reduction in the time required to factorize the simplified matrix.

In Table 1, an asterisk indicates instances where optimality is proven, meaning that CG reports an optimal factorization on a matrix that retains the same rank as the initial matrix. While CG manages to report the optimal BMF for 5 out of 31 datasets using the original matrices, using the Simpli$^\forall$ simplification increases this number to 11 out of 31 datasets. Moreover, the use of Simpli$^\forall$ never degrades the rank of the obtained factorizations and improves it for 6 of the datasets. While finding an optimal BMF when CG is applied to the matrix simplified by Simpli$_0^\exists$ is not guaranted, we observe that the rank of the BMF obtained is the best for 29 out of 31 datasets. Moreover, the computation times are frequently and significantly reduced, often by several orders of magnitude. Note that for some datasets the value "–" is present for Iteress, corresponding to the fact that Iteress is

Dataset	Characteristics		# ones after simplify			BMF with CG : time (rank)			
	Size	# Ones	Iteress	Simpli$^{\forall}$	Simpli$^{\exists}$	Original	Iteress	Simpli$^{\forall}$	Simpli$^{\exists}_{0}$
Advert.	3279×1557	45139	5941	3942	**705**	3h (1556)	3h (1596)	3h (1556)	**3h (704)**
Americ.	3477×1587	105205	49980	396	**187**	3h (1521)	3h (1251)	3h (299)	**2m (187)**
Apj	2044×1164	6841	2946	503	**453**	3h (1148)	3h (1133)	3h (497)	**20m (453)**
Audio	200×310	2333	225	**200**	**200**	3h (200)	2m (200)	**3m (200*)**	2m (200)
Autism	704×155	6832	–	5702	**1776**	3h (154)	–	3h (154)	**50m (154)**
Balance	625×23	3125	**3125**	**3125**	**3125**	**20m (23*)**	20m (23)	**20m (23*)**	20m (23)
Breast	699×90	6516	–	4153	**4149**	**3h (90)**	–	**3h (90)**	**3h (90)**
Car	1728×25	12096	**11071**	**11071**	**11071**	**1h (25*)**	30m (25)	3h (25*)	30m (25)
Chess	3196×39	25582	368	780	**38**	3h (38)	1s (38)	**20m (38*)**	1s (38)
Cmc	1473×71	11009	9992	10179	**9573**	**3h (71)**	**3h (71)**	**3h (71)**	**3h (71)**
Custom.	10961×277	45427	2771	1442	**276**	3h (277)	6m (276)	3h (277)	**3m (276)**
Dermat.	366×194	12482	–	6823	**6164**	3h (194)	–	1h (194)	**3h (187)**
DNA	4590×392	26527	1556	539	**367**	1m (392)	3h (368)	3h (384)	**5m (367)**
Firewa.	365×709	31951	2744	88	**65**	3h (64)	9m (65)	**1h (64*)**	1s (65)
Flare	1066×43	9283	2928	1950	**428**	2m (43)	3h (42)	**1h (42*)**	3h (42)
Heart	270×382	3036	325	1459	**270**	3h (270)	**9m (270)**	3h (270)	**9m (270)**
Hepati.	155×337	1377	–	1027	**154**	3h (154)	–	3h (154)	**10s (154)**
Iris	150×126	750	502	515	**486**	**10m (121*)**	8m (121)	**10m (121*)**	9m (121)
Lymph	148×54	1823	1288	1543	**1283**	3h (52)	20m (53)	**3h (52*)**	40m (53)
Mush.	8124×54	124768	–	1654	**51**	3h (54)	–	2h (51)	**1s (51)**
Nursery	12960×31	110160	61196	59480	**53001**	**3h (30)**	**3h (30)**	**3h (30)**	**3h (30)**
Paleo	501×139	3537	284	1853	**139**	3h (139)	**1s (139)**	3h (139)	**1s (139)**
Phishi.	1353×26	11507	8430	4856	**4466**	3h (26)	3h (26)	3h (26)	**6m (26)**
Soybean	307×100	6315	–	3354	**2909**	1h (99)	–	20m (90)	**3h (88)**
Student	395×176	9254	8488	8517	**8470**	**3h (176)**	**3h (176)**	**3h (176)**	**3h (176)**
Thorac.	470×340	3376	2373	2439	**2310**	**3h (304)**	**3h (304)**	**3h (304)**	**3h (304)**
Tictac.	958×28	8954	**8954**	**8954**	**8954**	**3h (28)**	**3h (28)**	**3h (28)**	**3h (28)**
Tumor	339×44	2136	–	1099	**107**	40m (44*)	–	**20m (44*)**	2h (44)
Votes	435×17	2961	–	381	**17**	**90m (17*)**	–	3h (17*)	1s (17)
Wine	178×1279	2492	816	190	**178**	3h (178)	20s (178)	**4m (178*)**	20s (178)
Zoo	101×28	640	85	108	**25**	3h (25)	1s (25)	**3h (25*)**	1s (25)

Table 1: Characteristics of the datasets used, comparison of the number of 1s after simplification across different algorithms and time required to find a BMF with CG. Bolded values indicate the best performance, and underlined values indicate the second best performance. The asterisk signifies that the tool was able to prove the optimality of the rank.

not able to factorize incomplete matrices.

In Table 2, we notice that no asterisk is present since OptiBlock is not capable of proving the optimality of the found factorizations. Similar to the previous result with CG, we observe that the use of Simpli$^{\forall}$ and Simpli$^{\exists}_{0}$ improves the rank of the BMF found by OptiBlock. If we compare the rank of the BMF and, in the case of equality, the resolution times, OptiBlock applied to matrices simplified with Simpli$^{\exists}_{0}$ gave the best results in 25 out of the 31 datasets.

5.2 Synthetic Datasets

In this second experiment, we reused the protocol used by GreEss (Belohlavek and Trnecka 2015), the ancestor of Iteress. In their protocol, the authors created synthetic matrices of size 1000×500 with four different levels of density (0.1, 0.2, 0.3, and 0.4) and three different levels of rank (20, 30, and 40). Based on the same protocol, we compared our two algorithms Simpli$^{\exists}$ and Simpli$^{\forall}$ to Iteress. Note that we have not included GreEss because Iteress, which uses the same method iteratively, performs more simplifications than GreEss.

As shown in Table 3, the results indicate that Simpli$^{\exists}$ consistently finds the best simplification with lowest number of 1-entries, Simpli$^{\forall}$ generally finds the second-best simplification, and Iteress generally finds a simplification with more 1-entries compared to the other two algorithms. Note that in our experiment, Simpli$^{\exists}$ almost always finds minimal simplifications, i.e., containing only k 1s for matrices of rank k. Indeed, since a k-rank BMF can be viewed as the union of k 1-rank BMF, a matrix of rank k containing k 1s cannot be further simplified and its factorization is trivial.

To quantify how these simplifications, in terms of the number of 1s removed from the original matrix, affect the performance of constraint-based BMF algorithms, we used the UndercoverBMF tool (Avellaneda and Villemaire 2022) with the "--optimal" option to find optimal BMF.

Table 3 shows the time used by this tool to find and guarantee an optimal BMF from the original and the universally

DATA	BMF WITH OPTIBLOCK : TIME (RANK)			
	ORIGINAL	SIMPLI$^{\forall}$	ITERESS	SIMPLI$^{\exists}_{0}$
ADV.	3h (794)	3h (749)	3h (711)	**3h (703)**
AME.	3h (216)	3h (188)	1h (189)	**1h (187)**
APJ	3h (474)	3h (453)	**2h (453)**	**2h (453)**
AUD.	30m (200)	**1m (200)**	**1m (200)**	**1m (200)**
AUT.	3m (154)	3m (154)	–	**1m (154)**
BAL.	**1s (23)**	**1s (23)**	**1s (23)**	**1s (23)**
BRE.	**10s (90)**	1m (90)	–	30s (90)
CAR	**3s (25)**	**3s (25)**	**3s (25)**	**3s (25)**
CHE.	1m (38)	20s (38)	**10s (38)**	**10s (38)**
CMC	40s (71)	40s (71)	**20s (71)**	**20s (71)**
CUS.	3h (282)	2h (277)	**1h (276)**	**1h (276)**
DER.	17m (189)	**17m (184)**	–	6m (192)
DNA	3h (497)	3h (373)	1h (368)	**1h (367)**
FIRE.	2m (64)	**1m (64)**	30s (65)	30s (65)
FLA.	14s (42)	14s (42)	**4s (42)**	**4s (42)**
HEA.	90m (270)	90m (270)	**2m (270)**	**2m (270)**
HEP.	9m (154)	7m (154)	–	**20s (154)**
IRIS	**10s (121)**	10s (122)	10s (122)	20s (122)
LYM.	5s (54)	6s (54)	3s (55)	**3s (54)**
MUS.	20m (51)	3m (51)	–	**1m (51)**
NUR.	**40s (30)**	**40s (30)**	90s (30)	**40s (30)**
PAL.	4m (139)	2m (139)	**30s (139)**	**30s (139)**
PHI.	10s (26)	**5s (26)**	**5s (26)**	**5s (26)**
SOY.	**2m (84)**	1m (85)	–	1m (88)
STU.	6m (176)	6m (176)	5m (177)	**4m (176)**
THO.	**10m (304)**	20m (306)	20m (306)	20m (305)
TIC.	**3s (28)**	**3s (28)**	**3s (28)**	**3s (28)**
TUM.	4s (44)	3s (44)	–	**1s (44)**
VOT.	2s (17)	**1s (17)**	–	**1s (17)**
WIN.	6m (178)	**2m (178)**	**2m (178)**	**2m (178)**
ZOO	**1s (25)**	**1s (25)**	**1s (25)**	**1s (25)**

Table 2: Characteristics of the datasets used, comparison of the number of 1s after simplification across different algorithms and time required to find a BMF with OptiBlock. Bolded values indicate the best performance, and underlined values indicate the second best performance.

simplified datasets. We did not use the existentially simplified dataset since the simplifications made in this case no longer guarantee finding a minimal rank factorization. We observed that the simplification obtained by the universal simplification improved the computation time of this constraint solver by several orders of magnitude. Note that when CG replaces optimal UndercoverBMF, the results remain similar: for universally simplified matrices, CG identifies optimal BMF within seconds for $k = 20$, within minutes for $k = 30$, and within few dozen minutes for $k = 40$. However, after 3 hours, no solution is found for $k = 40$ with a density of 0.4. And when the matrix is not simplified, no proof of optimality is obtained after 3 hours of computation.

This experiment showed that for matrices of low rank, our approach allowed us to find optimal BMF, even if the size of the matrix to be factorized is large.

DENS	NUMBER OF REMAINING 1S			BMF	
	SIMPLI$^{\exists}$	SIMPLI$^{\forall}$	ITERESS	OPT$^{\forall}$	OPT
K=20					
0.1	**20**	**20**	3218	**4s**	5m
0.2	**20**	**20**	1561	**4s**	14m
0.3	**20**	**20**	796	**3s**	20m
0.4	**20**	**20**	343	**2s**	30m
K=30					
0.1	**30**	**30**	1525	**1m**	2h
0.2	**30**	**30**	550	**4m**	1h
0.3	**30**	188	584	**8m**	1h
0.4	**36**	2856	1324	**4m**	2h
K=40					
0.1	**40**	**72**	1011	**11m**	–
0.2	**40**	259	669	**54m**	–
0.3	**40**	512	513	**3h**	–
0.4	**57**	9029	5292	–	–

Table 3: Number of ones present in a synthetic matrix after simplification and time to find an optimal BMF from the original and simplified matrices. Bolded values indicate the best performance, and underlined values indicate the second best performance.

6 Conclusion

This paper has addressed the problem of BMF by exploring the potential of matrix simplification before factorization to reduce the time required by existing constraint-based BMF algorithms.

We introduced two methods to simplify Boolean matrices before factorization by replacing specific 1-entries in the matrix with missing values. The central idea is that these simplifications allow for an easy translation from a BMF on the simplified matrix to a BMF on the original matrix. The first method, existential simplification (Simpli$^{\exists}$), can perform numerous simplifications, but does not ensure that the rank of the simplified matrix matches the rank of the original matrix. The second method, universal simplification (Simpli$^{\forall}$), performs fewer simplifications, but guarantees that an optimal BMF on the simplified matrix results in an optimal BMF on the original matrix.

The benefits of our approach were quantified through experiments. First, our methods effectively reduced the number of 1s in the matrices to be factorized, regardless of whether they come from real or synthetic datasets. Second, a significant reduction in computational time was observed for factorizations on simplified matrices compared to the original matrices. Third, the simplification of matrices not only improved the computation time, but also allowed heuristics to discover lower-rank factorizations. This result can be attributed to the reduction of the search space due to simplification. Finally, our method demonstrated the ability to identify optimal rank factorizations for ranks up to several dozen, even with large matrices.

Acknowledgements

We gratefully acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RGPIN-2023-04468 and RGPIN-2023-04557].

References

Avellaneda, F.; and Villemaire, R. 2022. Undercover Boolean Matrix Factorization with MaxSAT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4): 3672–3681.

Belohlavek, R.; Outrata, J.; and Trnecka, M. 2019. Factorizing Boolean matrices using formal concepts and iterative usage of essential entries. *Information Sciences*, 489: 37–49.

Belohlavek, R.; and Trnecka, M. 2015. From-below approximations in Boolean matrix factorization: Geometry and new algorithm. *Journal of Computer and System Sciences*, 81(8): 1678–1697.

Belohlavek, R.; and Vychodil, V. 2010. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *Journal of Computer and System Sciences*, 76(1): 3–20.

Ene, A.; Horne, W.; Milosavljevic, N.; Rao, P.; Schreiber, R.; and Tarjan, R. E. 2008. Fast exact and heuristic methods for role minimization problems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, 1–10.

Ganter, B.; and Wille, R. 2012. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media.

Geerts, F.; Goethals, B.; and Mielikäinen, T. 2004. Tiling databases. In *International conference on discovery science*, 278–289. Springer.

Hajiabolhassan, H.; and Moazami, F. 2012. Secure frameproof codes through biclique covers. *Discrete Mathematics & Theoretical Computer Science*, Vol. 14 no. 2.

Hashemi, S.; Tann, H.; and Reda, S. 2018. BLASYS: Approximate logic synthesis using Boolean matrix factorization. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 1–6. IEEE.

Karaev, S.; Miettinen, P.; and Vreeken, J. 2015. Getting to know the unknown unknowns: Destructive-noise resistant boolean matrix factorization. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, 325–333. SIAM.

Kelly, M.; Longjohn, R.; and Nottingham, K. 2023. The UCI Machine Learning Repository.

Kocayusufoglu, F.; Hoang, M. X.; and Singh, A. K. 2018. Summarizing network processes with network-constrained Boolean matrix factorization. In *2018 IEEE International Conference on Data Mining (ICDM)*, 237–246. IEEE.

Kovacs, R. A.; Gunluk, O.; and Hauser, R. A. 2021. Binary matrix factorisation via column generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5): 3823–3831.

Koyutürk, M.; and Grama, A. 2003. PROXIMUS: a framework for analyzing very high dimensional discrete-attributed datasets. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 147–156.

Liang, L.; Zhu, K.; and Lu, S. 2020. BEM: mining coregulation patterns in transcriptomics via boolean matrix factorization. *Bioinformatics*, 36(13): 4030–4037.

Lucchese, C.; Orlando, S.; and Perego, R. 2010. Mining top-k patterns from binary datasets in presence of noise. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, 165–176. SIAM.

Ma, J.; Hashemi, S.; and Reda, S. 2022. Approximate Logic Synthesis Using Boolean Matrix Factorization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(1): 15–28.

Markopoulou, A. P.; and E. Anagnostou, M. 1998. Optimal grouping of components in a distributed system. *Computer Communications*, 21(16): 1452–1461.

Miettinen, P.; Mielikäinen, T.; Gionis, A.; Das, G.; and Mannila, H. 2008. The discrete basis problem. *IEEE transactions on knowledge and data engineering*, 20(10): 1348–1362.

Miettinen, P.; and Neumann, S. 2020. Recent Developments in Boolean Matrix Factorization. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 4922–4928. ijcai.org.

Myllykangas, S.; Himberg, J.; Böhling, T.; Nagy, B.; Hollmén, J.; and Knuutila, S. 2006. DNA copy number amplification profiling of human neoplasms. *Oncogene*, 25(55): 7324–7332.

Shu, G.; Lee, D.; and Yannakakis, M. 2006. A note on broadcast encryption key management with applications to large scale emergency alert systems. In *IEEE International Parallel & Distributed Processing Symposium*, 8 pp.–.

Wan, C.; Chang, W.; Zhao, T.; Li, M.; Cao, S.; and Zhang, C. 2020. Fast and efficient boolean matrix factorization by geometric segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 6086–6093.

Wicker, J.; Pfahringer, B.; and Kramer, S. 2012. Multi-label classification using boolean matrix decomposition. In *Proceedings of the 27th annual ACM symposium on applied computing*, 179–186.

Xiang, Y.; Jin, R.; Fuhry, D.; and Dragan, F. F. 2011. Summarizing transactional databases with overlapped hyperrectangles. *Data Mining and Knowledge Discovery*, 23(2): 215–251.

Žliobaitė, I.; Fortelius, M.; Bernor, R. L.; Van den Hoek Ostende, L. W.; Janis, C. M.; Lintulaakso, K.; Säilä, L. K.; Werdelin, L.; Casanovas-Vilar, I.; Croft, D. A.; et al. 2023. The NOW database of fossil mammals. In *Evolution of Cenozoic Land Mammal Faunas and Ecosystems: 25 Years of the NOW Database of Fossil Mammals*, 33–42. Springer.