# Recent Developments in Boolean Matrix Factorization

**Pauli Miettinen**[1*] and **Stefan Neumann**[2†]
[1]University of Eastern Finland, Kuopio, Finland
[2]University of Vienna, Vienna, Austria
pauli.miettinen@uef.fi, stefan.neumann@univie.ac.at

## Abstract

The goal of Boolean Matrix Factorization (BMF) is to approximate a given binary matrix as the product of two low-rank *binary* factor matrices, where the product of the factor matrices is computed under the Boolean algebra. While the problem is computationally hard, it is also attractive because the binary nature of the factor matrices makes them highly interpretable. In the last decade, BMF has received a considerable amount of attention in the data mining and formal concept analysis communities and, more recently, the machine learning and the theory communities also started studying BMF. In this survey, we give a concise summary of the efforts of all of these communities and raise some open questions which in our opinion require further investigation.

## 1 Introduction

Boolean matrix factorization (BMF) is a variant of the standard matrix factorization problem in the Boolean semiring: given a binary matrix, the task is to find two smaller *binary* matrices so that their product, taken over the Boolean semiring, is as close to the original matrix as possible. Because the matrix product is not done over a field but over a semiring, many standard matrix factorization techniques fail to work. Indeed, finding the best Boolean factorization is computationally hard.

The computational hardness of the problem has not prevented people from studying it. In psychometrics, some of the first algorithms appeared in the 1980's (see Bělohlávek and Trnecka [2018]). Even before that, mathematicians studying combinatorics had studied the "Boolean linear algebra" [Kim, 1982; Monson *et al.*, 1995]. More recently, Miettinen *et al.* [2006] introduced the problem to the data mining community, Vaidya *et al.* [2007] to the access control community, Bělohlávek and Vychodil [2010] to formal concept analysts, and van den Broeck and Darwiche [2013] to lifted inference community, to name but a few examples. Even more recently, Ravanbakhsh *et al.* [2016] studied the problem in the framework of machine learning, increasing the interest of that community, and Chandran *et al.* [2016]; Ban *et al.* [2019]; Fomin *et al.* [2019] (re-)launched the interest in the problem in the theory community.

The various fields in which BMF is used are connected by the desire to "keep the data binary". This can be because the factorization is used as a pre-processing step, and the subsequent methods require binary input, or because binary matrices are more interpretable in the application domain. In the latter case, Boolean algebra is often preferred over the field $GF[2]$, as the XOR-operation can be harder to interpret.

The large number of areas where BMF or related problems are studied has given raise to many interesting approaches for solving this computationally hard problem. Unfortunately, it has also led to numerous re-inventions and repeated effort. Given the recent interest towards BMF in machine learning and AI communities, we believe that a timely survey will help to guide the research to novel directions and avoid re-inventions of older ideas.

To that end, we present a concise survey of the recent theoretical results on the hardness of the problem in Section 4 and a survey on different types of algorithms proposed for the problem in Section 5. We will provide some lessons we have learned when developing and using some of these algorithms, as well as the main research directions and open problems we find most interesting. Before going further, however, we will provide the formal definitions of BMF in Section 2 and some applications thereof in Section 3.

## 2 Definitions and Related Formulations

In this section we will explain the basic definitions for Boolean matrix factorization (BMF) and related concepts. We will also cover the related formulations of the problem using bipartite graphs and set systems. These formulations are all equivalent, but they make connections to different problems more clear.

### 2.1 Boolean Matrix Factorization

The goal of matrix decompositions is to decompose (or factorize) a given input matrix into two (or more) smaller *factor matrices* whose product is close to the original matrix. Thus, to define BMF, we first define the corresponding product.

[*] Contact Author

[†] The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement No. 340506, and the Doctoral Programme "Vienna Graduate School on Computational Optimization" which is funded by the Austrian Science Fund (FWF, project no. W1260-N35).

**Definition 1.** Given Boolean matrices $\boldsymbol{B} \in \{0,1\}^{m\times k}$ and $\boldsymbol{C} \in \{0,1\}^{k\times n}$, the *Boolean product* of $\boldsymbol{B}$ and $\boldsymbol{C}$ is denoted $\boldsymbol{B} \circ \boldsymbol{C} \in \{0,1\}^{m\times n}$ and it is defined element-wise as

$$(\boldsymbol{B} \circ \boldsymbol{C})_{ij} = \bigvee_{\ell=1}^{k} \boldsymbol{B}_{i\ell}\boldsymbol{C}_{\ell j} \ . \tag{1}$$

The Boolean matrix product is like the standard matrix product of two binary-valued matrices, but the algebra is the Boolean semi-ring (the summation is defined as $1+1=1$).

The *Boolean rank* of matrix $\boldsymbol{A} \in \{0,1\}^{m\times n}$ is the least integer $k$ such that there exists matrices $\boldsymbol{B} \in \{0,1\}^{m\times k}$ and $\boldsymbol{C} \in \{0,1\}^{k\times n}$ with $\boldsymbol{A} = \boldsymbol{B} \circ \boldsymbol{C}$. In practice, the Boolean rank is rarely that interesting; like normal matrix rank, it is not robust to noise, and many real-world matrices have almost-full rank (though exceptions exist, see Ene *et al.* [2008]).

**Problem 1.** Given a matrix $\boldsymbol{A} \in \{0,1\}^{m\times n}$ and integer $k$, the goal of *Boolean matrix factorization* (BMF) is to find matrices $\boldsymbol{B} \in \{0,1\}^{m\times k}$ and $\boldsymbol{C} \in \{0,1\}^{k\times n}$ such that they minimize

$$\|\boldsymbol{A} - \boldsymbol{B} \circ \boldsymbol{C}\|_F^2 = \sum_{i,j} \boldsymbol{A}_{ij} \oplus (\boldsymbol{B} \circ \boldsymbol{C})_{ij} \ . \tag{2}$$

In (2), subtraction, Frobenius norm $\|\cdot\|_F$, and the summation are taken over the standard algebra while $\oplus$ stands for the element-wise XOR-operation. It might seem curious that the error is defined over different algebra than the matrix product – after all, the whole point of BMF is that it uses the Boolean matrix product – but it should be noted that the Boolean algebra does not allow for subtraction. On the other hand, as all elements of $\boldsymbol{A}$ and $\boldsymbol{B} \circ \boldsymbol{C}$ are binary, the (squared) Frobenius norm simplifies into the Hamming distance between $\boldsymbol{A}$ and $\boldsymbol{B} \circ \boldsymbol{C}$, i.e. it is simply the number of entries with $\boldsymbol{A}_{ij} \neq (\boldsymbol{B} \circ \boldsymbol{C})_{ij}$.

### 2.2 Covering by Bicliques

Any rectangular binary matrix $\boldsymbol{A} \in \{0,1\}^{m\times n}$ can be viewed as the *bi-adjacency matrix* of a bipartite graph $G = (U \cup V, E)$. The rows of $\boldsymbol{A}$ correspond to vertices in $U$, columns of $\boldsymbol{A}$ correspond to vertices in $V$, and $\boldsymbol{A}_{ij} = 1$ if $\{i,j\} \in E$. Vice versa, any (unweighted, undirected) bipartite graph $G$ can be identified with its bi-adjacency matrix $\boldsymbol{A}(G)$.

For bipartite graphs, BMF is equivalent to *biclique cover*.

**Problem 2.** Given a bipartite graph $G = (U \cup V, E)$ and an integer $k$, the goal of *biclique cover* is to find $k$ *bicliques* $C_1, \ldots, C_k$, such that they minimize

$$\left| E \oplus \bigcup_{\ell=1}^{k} E(C_\ell) \right| \ . \tag{3}$$

A *biclique* is a complete bipartite subgraph $C_\ell = (U_\ell \cup V_\ell, E(C_\ell))$, where $U_\ell \subseteq U$, $V_\ell \subseteq V$ and $E(C_\ell) = U_\ell \times V_\ell$, i.e. $C_\ell$ contains edges between all pairs of vertices in $U_\ell$ and $V_\ell$. Note that Problem 2 does not require that the edges of $C_\ell$ are a subset of edges of $G$. Abusing the notation, in (3) the symbol $\oplus$ is denotes the symmetric difference between sets.

The connection between Problems 1 and 2 can observed as follows: Let $\boldsymbol{b}_\ell$ be the $\ell$th column of $\boldsymbol{B}$ and $\boldsymbol{c}^\ell$ be the $\ell$th row of $\boldsymbol{C}$. Then we can write $\boldsymbol{B} \circ \boldsymbol{C}$ as the OR of outer products of $\boldsymbol{b}_\ell$ and $\boldsymbol{c}^\ell$, i.e. $\boldsymbol{B} \circ \boldsymbol{C} = \bigvee_{\ell=1}^{k} \boldsymbol{b}_\ell \boldsymbol{c}^\ell$. Each of these rank-one matrices (called *components*) $\boldsymbol{b}_\ell \boldsymbol{c}^\ell$ defines a rectangle of 1s, i.e. $(\boldsymbol{b}_\ell \boldsymbol{c}^\ell)_{ij} = 1$ iff $(\boldsymbol{b}_\ell)_i = (\boldsymbol{c}^\ell)_j = 1$. Hence, the bi-adjacency matrix corresponding to $\boldsymbol{b}_\ell \boldsymbol{c}^\ell$ defines a biclique $C_\ell = (U_\ell, V_\ell)$ with $U_\ell = \{i : (\boldsymbol{b}_\ell)_i = 1\}$ and $V_\ell = \{j : (\boldsymbol{c}^\ell)_j = 1\}$. Then the bi-adjacency matrix of the bipartite graph with edges $\bigcup_{\ell=1}^{k} E(C_\ell)$ is $\bigvee_{\ell=1}^{k} \boldsymbol{B}_{i\ell}\boldsymbol{C}_{\ell j} = \boldsymbol{B} \circ \boldsymbol{C}$.

### 2.3 Set Systems

A binary matrix $\boldsymbol{A} \in \{0,1\}^{m\times n}$ can also be identified as a *set system*: the columns correspond to $n$ items in universe $U$ and the rows are incidence vectors of $m$ sets in a collection of sets $\mathcal{S} = \{S_i \subseteq U : i = 1, \ldots, m\}$. Then the following problem is equivalent to BMF and will be called *set basis*.

**Problem 3.** Given a set system $(U, \mathcal{S})$ and an integer $k$, the goal of *set basis* is to find a collection of $k$ sets $\mathcal{C} = \{C_i \subseteq U : \ell = 1, \ldots, k\}$ such that for every set $S_i \in \mathcal{S}$ there exists a subcollection of $\mathcal{B}_i \subseteq \mathcal{C}$ that jointly minimize

$$\sum_{i=1}^{m} \left| S_i \oplus \bigcup_{C \in \mathcal{B}_i} C \right| \ . \tag{4}$$

In this formulation, the rows of $\boldsymbol{A}$ correspond to the sets $S_i$ with $S_i = \{j : \boldsymbol{A}_{ij} = 1\}$. The matrix $\boldsymbol{C}$ encodes the incidence vectors of the sets $C_\ell \in \mathcal{C}$, i.e. $\boldsymbol{C}_{\ell i} = 1$ iff $i \in C_\ell$. The choice of which sets to include in which subcollection $\mathcal{B}_i$ is given by the rows of $\boldsymbol{B}$, i.e. $\boldsymbol{B}_{i\ell} = 1$ iff $C_\ell \in \mathcal{B}_i$. Thus, Problem 3 is equivalent to Problem 1.

## 3 Applications

Let us now discuss applications of BMF. Wicker *et al.* [2012] use BMF to pre-process multi-label classification data. They decompose the binary label matrix and use the columns of the left factor matrix $\boldsymbol{B}$ as sort-of "super-labels." For a recent review of this and other approaches for dimensionality reduction for multi-label classification, see Siblini *et al.* [2019].

Somewhat similarly, van den Broeck and Darwiche [2013] use BMF to speed up lifted inference in Boolean data. They show that the inference is efficient if the data has low Boolean rank, and to transform the data, they compute low-rank Boolean decomposition before doing the inference.

Vaidya *et al.* [2007] propose using BMF for *role mining*. In role mining, we are given a binary matrix that represents users and permissions, and the goal is to assign users to (possibly overlapping) roles and to give these roles necessary permissions. Hence, if $\boldsymbol{A}$ is the user–permission matrix, $\boldsymbol{B}$ assigns users to $k$ different roles and $\boldsymbol{C}$ assigns the permissions to these roles. Ene *et al.* [2008] showed that many real-world matrices in this domain actually have low Boolean rank.

BMF has many more applications and due to lack of space, we can only name a few. In bioinformatics, BMF is used for studying transcriptomic data [Liang *et al.*, 2020] and for identifying functional interactions in brain networks [Haddad *et al.*, 2018]. BMF was also used for adding missing attribute information to products in the Amazon catalog [Rukat *et al.*, 2017b]. Further applications include approximate logic synthesis [Hashemi *et al.*, 2018] and discovery of patterns in network data [Kocayusufoglu *et al.*, 2018].

# 4 Theoretical Results

In this section we summarize computational lower bounds and upper bounds for computing BMF.

## 4.1 Computational Complexity

Let us review the computational complexity of Problem 1.

Orlin [1977] showed that, given a bipartite graph $G$ and a parameter $k$, it is NP-complete to decide whether the edges of $G$ can be covered with $k$ bicliques. Via the discussion from Section 2.2 this implies that it is NP-complete to decide whether there exist Boolean matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ of rank $k$ such that $\|\boldsymbol{A} - \boldsymbol{B} \circ \boldsymbol{C}\|_F^2 = 0$. Hence, Problem 1 is NP-complete.

Chandran *et al.* [2016] showed that under a standard assumption in computational complexity theory, one cannot distinguish between the case when the objective function in Problem 1 is 0 or at least 1 in time $2^{2^{o(k)}}(mn)^{O(1)}$. This implies that either the running time of an algorithm must be $2^{2^{\Omega(k)}}$ or it must be $(mn)^{\omega(1)}$, i.e. the algorithm's running time is either doubly exponential in $k$ or superpolynomial in the size of the input matrix.

In fact, the above results hold for any $\alpha$-approximate solution. Here, we say that matrices $\boldsymbol{B}' \in \{0,1\}^{m \times k}$ and $\boldsymbol{C}' \in \{0,1\}^{k \times n}$ provide an $\alpha$-*approximation* of the optimal objective function value if

$$\|\boldsymbol{A} - \boldsymbol{B}' \circ \boldsymbol{C}'\|_F^2 \leq \alpha \cdot \min_{\boldsymbol{B},\boldsymbol{C}} \|\boldsymbol{A} - \boldsymbol{B} \circ \boldsymbol{C}\|_F^2 \ .$$

Observe that any $\alpha$-approximation must be able to distinguish between whether $\min_{\boldsymbol{B},\boldsymbol{C}} \|\boldsymbol{A} - \boldsymbol{B} \circ \boldsymbol{C}\|_F^2 = 0$ or $\min_{\boldsymbol{B},\boldsymbol{C}} \|\boldsymbol{A} - \boldsymbol{B} \circ \boldsymbol{C}\|_F^2 > 0$. Hence, the above results imply that for any $\alpha \geq 1$, computing an $\alpha$-approximation for Problem 1 is NP-complete and cannot be solved in time $2^{2^{o(k)}}(mn)^{O(1)}$ under a standard complexity assumption.

## 4.2 Theoretical Algorithms

Let us now look at algorithms with theoretical guarantees.

Gramm *et al.* [2008] and Fomin *et al.* [2020] showed that an *exact* solution for BMF can be computed in time $2^{2^{O(k)}}(mn)^{O(1)}$, thus matching the above lower bound.

Furthermore, Fomin *et al.* [2019] provided an algorithm which computes a $(1+\varepsilon)$-approximate BMF in time $2^{2^{O(k)}/\varepsilon^2 \cdot \lg^2(1/\varepsilon)} \cdot mn$. Note that when $k$ and $\varepsilon$ are constants, then the running time of this algorithm becomes $O(mn)$, i.e. the algorithm runs in time linear in the size of the $m \times n$ input matrix $\boldsymbol{A}$. This is unlike the exact algorithms, where the exponent of the $mn$-term is strictly larger than 1, i.e. they require superlinear running time even when $k$ is a fixed constant. Ban *et al.* [2019] obtained a similar algorithm with slightly worse running time, but their algorithm extends to any finite field.

Bhattacharya *et al.* [2019] extended ideas of Fomin *et al.* [2019] and Ban *et al.* [2019] to obtain a 4-pass streaming algorithm which computes a $(1+\varepsilon)$-approximate BMF. Their algorithm never stores more than $2^{\tilde{O}(2^k/\varepsilon^2)} \cdot (\lg n)^{2k}$ rows of the matrix and it has running time $2^{\tilde{O}(2^k/\varepsilon^2)} \cdot (\lg n)^{2k} \cdot mn$. Note that when $k$ is a fixed constant, then the algorithm only stores a polylogarithmic number of rows.

## 4.3 Discussion

From a theoretical perspective, BMF is relatively well-understood. Indeed, note that the lower bound results by Chandran *et al.* [2016] and the running times of the algorithms by Fomin *et al.* [2019] and Ban *et al.* [2019] match up to lower order terms. Hence, there is no hope in removing the $2^{2^{O(k)}}$-terms in the running times of the algorithms.

Unfortunately, all of the above algorithms are rather impractical. First of all, the $2^{2^{O(k)}}$-term in the running time grows extremely fast (even for $k = 7$, $2^{2^7} > 3 \cdot 10^{38}$). Secondly, the algorithms have in common that they use several exhaustive enumeration procedures which are extremely slow in practice.

There are two main open questions in this area. First, can one improve upon the results by Bhattacharya *et al.* [2019] by providing a streaming algorithm which performs less than 4 passes and returns a BMF. Second, it would be interesting whether there are practical algorithms for which one can obtain similar running time and approximation guarantees to the ones mentioned above.

# 5 Algorithms

In this section, we discuss several practical algorithms for computing BMF. From a high-level point of view, the algorithms can be placed into two categories: Those based on combinatorial optimization and those based on continuous optimization. We will now discuss each type of these algorithms and conclude the section with some open questions.

## 5.1 Combinatorial Optimization

Some of the earliest algorithms for BMF come from psychometrics community. One example is the 8M algorithm in the BMDP statistical software (see Bělohlávek and Trnecka [2018] for a detailed description). The 8M algorithm follows a simple local update process: it starts by building initial matrices $\boldsymbol{B}$ and $\boldsymbol{C}$, and then iteratively replaces the initial columns of $\boldsymbol{B}$ and rows of $\boldsymbol{C}$ with (locally) better ones.

The iterative update approach is also used by Bělohlávek and Trnecka [2018] in their GRECOND+ algorithm. GRECOND+ takes carefully into account the anti-monotonicity of Boolean algebra: if one component $\boldsymbol{b}_\ell \boldsymbol{c}^\ell$ introduces a 1 in location $(i,j)$ where $\boldsymbol{A}_{ij} = 0$, this error is irrecoverable by the other components. Hence, GRECOND+ starts by generating a decomposition that does not over-cover (i.e. they generate a tiling) and then the components are iteratively updated to commit over-covering when that decreases the error. After one component is updated, other components are examined if they can be edited – or removed entirely – to improve the error.

The GRECOND+ algorithm belongs to a collection of BMF algorithms based on *formal concept analysis* [Ganter and Wille, 1999] (see also Bělohlávek and Vychodil [2010]; Bělohlávek and Trnecka [2015]). A formal concept, in the BMF framework, is a pair of row and column index sets $(I,J)$ such that $\boldsymbol{A}_{ij} = 1$ for all $i \in I$ and all $j \in J$ and such that no new index can be added to neither $I$ or $J$. Such a pair of indices corresponds to components $\boldsymbol{b}_\ell \boldsymbol{c}^\ell$ that do not over-cover (they also correspond to *closed itemsets* in frequent itemset mining). Most formal-concept based algorithms find non-overcovering factorizations as a consequence.

The tiling approach also motivated the PaNDa$^+$ algorithm [Lucchese *et al.*, 2013]. However, instead of finding strictly non-overcovering initial components, as GreConD+ does, PaNDa$^+$ finds *dense* initial components, that is, initial components that overcover only strictly limited amount of elements. In the second phase of the algorithm, these dense cores are extended, again provided that the extension does not increase the error. An important feature of PaNDa$^+$ is that the number of components, $k$, is not predefined. Rather, PaNDa$^+$ selects the number of components automatically, based on user-selected criteria for balancing the decrease in error and increase in the number of components.

Yet another approach is presented by the Asso algorithm [Miettinen *et al.*, 2008]. Unlike the previously-described algorithms, Asso starts by generating a large collection of candidate columns for $\boldsymbol{B}$. There will be a candidate column for every row $\boldsymbol{a}^i$ of $\boldsymbol{A}$. A candidate column is generated by comparing the association confidence from $\boldsymbol{a}^i$ to $\boldsymbol{a}^j$, for any $j = 1, \ldots, m$, to a user-supplied threshold $\tau$. The candidate column will have a 1 in the $j$th entry if the confidence, computed as $\boldsymbol{a}^i(\boldsymbol{a}^j)^T/(\boldsymbol{a}^i(\boldsymbol{a}^i)^T)$, is above the threshold $\tau$.

Asso then greedily selects the candidates one-by-one to be added to $\boldsymbol{B}$ by computing how much error each candidate would reduce. In order to compute the error, Asso computes for each column $\boldsymbol{a}$ of $\boldsymbol{A}$ and for each candidate column $\boldsymbol{d}$, how much the reconstruction error of $\boldsymbol{a}$ would change if $\boldsymbol{d}$ would be used to cover the column, that is, it computes $\|\boldsymbol{a} - \boldsymbol{B} \circ \boldsymbol{c}\|_F^2 - \|\boldsymbol{a} - [\boldsymbol{B}\ \boldsymbol{d}] \circ [\begin{smallmatrix}\boldsymbol{c}\\1\end{smallmatrix}]\|_F^2$, where $\boldsymbol{B}$ is the current left factor matrix and $\boldsymbol{c}$ is the column of the right factor matrix corresponding to $\boldsymbol{a}$. The candidate is used in every column of $\boldsymbol{A}$ where it reduces the error, and the candidate that reduces the error the most over all columns of $\boldsymbol{A}$ is selected. The corresponding row of $\boldsymbol{C}$ is built based on the greedy computation: if the selected $\boldsymbol{d}$ reduces the error on column $j$, the corresponding column in $\boldsymbol{C}$ will be set to 1.

As mentioned, PaNDa$^+$ uses some measure to balance the reduction in error and the size of the factorization. There are different ways to do this, but one of the popular and principled ways for solving the *model order selection* problem is to use the *minimum description length* (MDL) principle [Rissanen, 1978]. Miettinen and Vreeken [2014] proposed to use two-part MDL for selecting the number of components in BMF. Two-part MDL computes the encoding length of the data $L(\boldsymbol{A})$ as the length of the model (or hypothesis) $L(\mathcal{M})$ plus the length of the data given the model $L(\boldsymbol{A} \mid \mathcal{M})$. In case of BMF, the model consists of the factor matrices, while the data given the model is the error the factorization causes. That is,

$$L(\boldsymbol{A}) = L(\boldsymbol{B}, \boldsymbol{C}) + L\big(\boldsymbol{A} \oplus (\boldsymbol{B} \circ \boldsymbol{C})\big) \ . \tag{5}$$

The length of encoding a binary matrix can be computed using standard methods for binary string encoding. Miettinen and Vreeken [2014] propose the use of so-called *data-to-model* encodings and differentiating over- and under-covering noise.

Miettinen and Vreeken [2014] use a variation of Asso to find the decomposition that minimizes (5). Also PaNDa$^+$ can be used to find such decomposition. Karaev *et al.* [2015] also try to minimize (5) and their algorithm, called nassau, shares similarities with many of the above algorithms: it starts by finding a set of seed columns (akin to candidates in Asso). Unlike Asso, though, nassau uses random walks in the bipartite graph corresponding to $\boldsymbol{A}$ to find the seeds. In short, they consider the probability of a random walk starting from a vertex $u \in U$ to be in another vertex $u' \in U$. The seed column corresponding to vertex $u \in U$ has 1-entries in the rows corresponding to the vertices $u'$ where the random walk would be with a sufficiently high probability. The seed columns are then turned into rank-1 components using a similar greedy strategy as used by Asso. After a new rank-1 component is added, the existing ones are iteratively updated, similarly to 8M, potentially dropping some if they become redundant.

Many of the above algorithms first build some initial solution or candidate set, and then refine it. The BMaD framework [Tyukin *et al.*, 2014] formalizes this process and allows the user to define the approach for candidate generation, the approach for selecting the candidates, and the approach for building the factorization from the candidates.

## 5.2 Continuous Optimization

Next, let us look at continuous optimization algorithms.

Hess *et al.* [2017] propose to use a proximal alternating linearized minimization technique for finding the MDL-optimizing BMF. The idea is to replace BMF with a surrogate continuous optimization problem with a two-part goal: minimizing the error and making the factor matrices binary. This approach allows alternating between gradient descent optimization of the factor matrices and computing the proximal function. Hess *et al.* [2018] use a similar technique for finding Boolean decompositions that limit the *false discovery rate*, i.e. the probability that the found patterns arise from noise.

Next, we discuss a sequence of works which try to find factor matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ with maximum likelihood under the given input matrix $\boldsymbol{A}$. This includes works of Ravanbakhsh *et al.* [2016]; Rukat *et al.* [2017a]; Rukat *et al.* [2018]; Liang and Lu [2019]; Frolov *et al.* [2014]. In order to compute the likelihoods of the factor matrices, they make some prior assumptions on the data-generating process. We will only discuss the model of Ravanbakhsh *et al.* [2016] in detail.

From a high-level point of view, Ravanbakhsh *et al.* [2016] assume that the input matrix $\boldsymbol{A}$ for Problem 1 is generated as follows. First, two ground-truth factor matrices $\boldsymbol{B} \in \{0,1\}^{m \times k}$ and $\boldsymbol{C} \in \{0,1\}^{k \times n}$ are generated randomly according to a certain distribution. This gives a matrix $\boldsymbol{Z} = \boldsymbol{B} \circ \boldsymbol{C}$ of Boolean rank $k$. Now the matrix $\boldsymbol{A}$ is generated from $\boldsymbol{Z}$ by setting $\boldsymbol{A} = \boldsymbol{Z}$ and then flipping the bit in each entry $\boldsymbol{A}_{ij}$ according to some distribution.

Let us now look consider this model in more detail. For all entries of the factor matrices $\boldsymbol{B} \in \{0,1\}^{m \times k}$ and $\boldsymbol{C} \in \{0,1\}^{k \times n}$ there exist priors $p_{ir}^{\boldsymbol{B}}(\cdot)$ and $p_{rj}^{\boldsymbol{C}}(\cdot)$ for all $i = 1, \ldots, m$, $r = 1, \ldots, k$ and $j = 1, \ldots, n$. In particular, it is assumed that $\boldsymbol{B}_{ir} = 1$ ($\boldsymbol{C}_{rj} = 1$) with probability $p_{ir}^{\boldsymbol{B}}(\boldsymbol{B}_{ir})$ ($p_{rj}^{\boldsymbol{C}}(\boldsymbol{C}_{rj})$) and that for both factor matrices the prior takes the following separable product form:

$$p^{\boldsymbol{B}}(\boldsymbol{B}) = \prod_{i,r} p_{ir}^{\boldsymbol{B}}(\boldsymbol{B}_{ir}) \ , \qquad p^{\boldsymbol{C}}(\boldsymbol{C}) = \prod_{r,j} p_{rj}^{\boldsymbol{C}}(\boldsymbol{C}_{rj}) \ .$$

Next, given $\boldsymbol{B}$ and $\boldsymbol{C}$ as above, consider the product matrix $\boldsymbol{Z} = \boldsymbol{B} \circ \boldsymbol{C} \in \{0,1\}^{m \times n}$ and observe that $\boldsymbol{Z}$ only depends on the randomness of the entries in $\boldsymbol{B}$ and $\boldsymbol{C}$. It is assumed that for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$ there exists another prior $p_{ij}^{\boldsymbol{A}}(\boldsymbol{A}_{ij} \mid \boldsymbol{Z}_{ij})$, i.e. the probability of flipping the bit in entry $\boldsymbol{A}_{ij}$ can depend on $i$, $j$ and $\boldsymbol{Z}_{ij}$. It is further assumed that for the matrix $\boldsymbol{A}$ it holds that

$$p^{\boldsymbol{A}}(\boldsymbol{A} \mid \boldsymbol{Z}) = \prod_{i,j} p_{ij}^{\boldsymbol{A}}(\boldsymbol{A}_{ij} \mid \boldsymbol{Z}_{ij}) \ .$$

Recall that in Problem 1 the input is the matrix $\boldsymbol{A}$ and we are actually interested in computing the (unknown) factor matrices $\boldsymbol{B}$ and $\boldsymbol{C}$. The approach of Ravanbakhsh *et al.* [2016] is to solve the maximum a posteriori inference problem

$$\arg\max_{\boldsymbol{B},\boldsymbol{C}} p(\boldsymbol{B}, \boldsymbol{C} \mid \boldsymbol{A}) \ .$$

To solve the above inference problem, they introduce a graphical model for the posterior and use belief propagation to find the matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ which maximize the posterior.

Frolov *et al.* [2014] and Liang and Lu [2019] used a similar generative model and expectation maximization instead of belief propagation. Rukat *et al.* [2017a] used a Metropolised Gibbs sampler for the posterior inference. Later, Rukat *et al.* [2018] extended their algorithm for Boolean tensors.

### 5.3 Open Problems and Lessons Learned

The algorithms based on combinatorial optimization use various heuristics and iterative update approaches to avoid the problem that their optimization landscape is not at all smooth. Indeed, most iterative update algorithms will converge in just few rounds (see, e.g. the experiments by Miettinen [2009]). No currently-published algorithm combines all of the approaches (many candidates, preference for under-covering, iterative local updates, etc.). Whether a combination of these approaches would improve the results will remain as a topic of further research, although our experience indicates that combining multiple approaches has diminishing returns in most cases.

At the same time it should also be noticed that while it is easy to build examples where greedy heuristics will fail spectacularly, they often tend to work very well in real-world applications. In case of BMF, the experiments by Miettinen [2009] seem to confirm this at least when it comes to the problem of selecting the candidates to build the factorization.

When the data is very sparse, some algorithms based on continuous optimization outperform all combinatorial algorithms we are aware of. Specifically, the experiments by Ravanbakhsh *et al.* [2016] show that on synthetic data, their algorithm recovers planted structures close to what is possible (they show that the reconstruction error of their algorithm is close to an information theoretic lower bound by Davenport *et al.* [2014]). This is also supported by the experiments by Neumann [2018].

A major open question is to make existing algorithms more scalable. Indeed, existing algorithms do not easily scale to matrices with tens or hundreds of thousands rows and columns. For the combinatorial optimization algorithms, a major bottleneck is that they need to generate a lot of candidate factor matrices and, hence, need to cover the input matrix many times. For the continuous optimization algorithms, a major bottleneck is that usually they maintain dense real-valued versions of the factor matrices; this makes them slow and memory-intensive. However, these methods may benefit from the recent GPU development and methods such as stochastic gradient descent; indeed, such techniques are used by Hess *et al.* [2017]. In practice, the scalability problem is often bypassed by pruning all rows and columns from the input matrix which have less than a user-specified threshold of non-zero entries.

None of the above algorithms comes with provable guarantees to return a solution within a certain approximation ratio. While some of them will converge to an optimal solution, it is not clear how many iterations will be necessary.

## 6 Related Work

BMF has been extended (and constrained) in various forms to work better in different applications.

Tatti and Miettinen [2019] aim at finding BMF where the rows and columns of the data and factor matrices can be permuted such that the 1s in the columns of $\boldsymbol{B}$ and in the rows of $\boldsymbol{C}$ all appear consecutively. This is used for sorting vertices when graphs are drawn as edge bundles.

Many datasets are multi-view: they present different data about the same entities. Sometimes such data can be expressed as two (or more) matrices, and they can be jointly decomposed. Miettinen [2012] solves *joint BMF*: given $\boldsymbol{A}$ and $\boldsymbol{B}$ whose rows represent the same entities, find $\boldsymbol{W}$, $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{H}$, and $\boldsymbol{L}$ such that $\boldsymbol{A} \approx [\boldsymbol{W}\ \boldsymbol{U}] \circ \boldsymbol{H}$ and $\boldsymbol{B} \approx [\boldsymbol{W}\ \boldsymbol{V}] \circ \boldsymbol{L}$. Similarly, Hess and Morik [2017] find decompositions of type $\boldsymbol{A} \approx (\boldsymbol{W} \vee \boldsymbol{U}) \circ \boldsymbol{H}$ and $\boldsymbol{B} \approx (\boldsymbol{W} \vee \boldsymbol{V}) \circ \boldsymbol{L}$ (note that the dimensions of the matrices are not the same in these two definitions).

One of the most actively studied extension of BMF is Boolean tensor decompositions (see, e.g. Miettinen [2011]; Erdős and Miettinen [2013]; Rukat *et al.* [2018]). Briefly, tensors extend the matrix (2-way) data to multi-way data. For example, a Boolean rank-$r$ CP decomposition of a three-way tensor $\boldsymbol{\mathcal{A}}$ would decompose it into three factor matrices, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$, minimizing

$$\sum_{i,j,k} \left| \boldsymbol{\mathcal{A}}_{ijk} - \bigvee_{\ell=1}^{r} \boldsymbol{B}_{i\ell} \boldsymbol{C}_{j\ell} \boldsymbol{D}_{k\ell} \right| \ .$$

While BMF can be seen as covering a bipartite graph with (potentially overlapping) bicliques, many algorithms have been proposed for the problem of covering a bipartite graph with *disjoint* bicliques. These problems are often studied under names such as *biclustering* [Lim *et al.*, 2015], *co-clustering* [Dhillon, 2001] or *bipartite graph partitioning* [Zha *et al.*, 2001]. Kumar *et al.* [2019] provide a constant factor approximation algorithm for covering a bipartite graph with disjoint bicliques with running time $2^{O(k^2)}(mn)^{O(1)}$.

Covering *random* bipartite graphs with disjoint bicliques has recently received some attention [Lim *et al.*, 2015; Xu *et al.*, 2014; Zhou and Amini, 2018; Zhou and Amini, 2019; Ndaoud *et al.*, 2019]. In these *bipartite stochastic block models* one assumes that the input bipartite graph was randomly generated based on some hidden bicliques. The goal is to state conditions under which the hidden bicliques can be recovered when the input only consists of the random bipartite graph. These random graph models are similar to the data-generating models mentioned in Section 5.2 (assumptions on the hidden bicliques correspond to priors on factor matrices). Neumann [2018] showed that in such random bipartite graphs a simple two-step algorithm can recover even very tiny clusters, which is not possible in general graphs.

## 7 Future Directions and Conclusion

In this survey, we have summarized the current state of the art for BMF algorithms including several recent developments based on results from the machine learning and theoretical computer science community.

From a practical point of view, perhaps the most important open problem is to make the existing algorithms more scalable.

From a theoretical point of view, the lower bounds from Section 4.1 essentially rule out any *practical* algorithms which provably approximate BMF. Nonetheless, we have discussed in Section 5.3 that there are algorithms which appear to recover very high quality results in practice. Hence, we believe that one should try to analyze BMF algorithms not using the current worst-case analysis approach, but using other complexity measures. In our opinion, one promising approach would be to generalize the results for bipartite stochastic block models we mentioned in Section 6 from disjoint biclique covers to overlapping biclique covers. This would imply practical algorithms for BMF with provable quality guarantees. Since some of the bipartite stochastic block model algorithms work well in practice and their running times do not carry large overheads in some parameters as the ones from Section 4.2, we believe that there is hope to obtain such results.

## References

[Ban *et al.*, 2019] Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P. Woodruff. A PTAS for $\ell_p$-low rank approximation. In *SODA*, pages 747–766, 2019.

[Bělohlávek and Trnecka, 2015] Radim Bělohlávek and Martin Trnecka. From-below approximations in Boolean matrix factorization: Geometry and new algorithm. *J. Comput. Syst. Sci.*, 81(8):1678–1697, 2015.

[Bělohlávek and Trnecka, 2018] Radim Bělohlávek and Martin Trnecka. A new algorithm for Boolean matrix factorization which admits overcovering. *Discrete Appl. Math.*, 249:36–52, 2018.

[Bělohlávek and Vychodil, 2010] Radim Bělohlávek and Vilém Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. Syst. Sci.*, 76(1):3–20, 2010.

[Bhattacharya *et al.*, 2019] Anup Bhattacharya, Dishant Goyal, Ragesh Jaiswal, and Amit Kumar. Streaming PTAS for binary $\ell_0$–low rank approximation. *CoRR*, abs/1909.11744, 2019.

[Chandran *et al.*, 2016] L. Sunil Chandran, Davis Issac, and Andreas Karrenbauer. On the parameterized complexity of biclique cover and partition. In *IPEC*, pages 11:1–11:13, 2016.

[Davenport *et al.*, 2014] Mark A. Davenport, Yaniv Plan, Ewout van den Berg, and Mary Wootters. 1-bit matrix completion. *Inf. Inference*, 3(3):189–223, 2014.

[Dhillon, 2001] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.

[Ene *et al.*, 2008] Alina Ene, William Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert E Tarjan. Fast exact and heuristic methods for role minimization problems. In *SACMAT*, pages 1–10, 2008.

[Erdős and Miettinen, 2013] Dóra Erdős and Pauli Miettinen. Walk'n'Merge: A scalable algorithm for Boolean tensor factorization. In *ICDM*, pages 1037–1042, 2013.

[Fomin *et al.*, 2019] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Approximation schemes for low-rank binary matrix approximation problems. *ACM Trans. Algorithms*, 16(1):12:1–12:39, 2019.

[Fomin *et al.*, 2020] Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. Parameterized low-rank binary matrix approximation. *Data Min. Knowl. Discov.*, 34(2):478–532, 2020.

[Frolov *et al.*, 2014] Alexander A. Frolov, Dušan Húsek, and Pavel Y. Polyakov. Two expectation-maximization algorithms for Boolean factor analysis. *Neurocomputing*, 130:83–97, 2014.

[Ganter and Wille, 1999] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin, 1999.

[Gramm *et al.*, 2008] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM J. Exp. Algorithmics*, 13, 2008.

[Haddad *et al.*, 2018] Ali Haddad, Foroogh Shamsi, Li Zhu, and Laleh Najafizadeh. Identifying dynamics of brain function via Boolean matrix factorization. In *ACSSC*, pages 661–665, 2018.

[Hashemi *et al.*, 2018] Soheil Hashemi, Hokchhay Tann, and Sherief Reda. BLASYS: approximate logic synthesis using Boolean matrix factorization. In *DAC*, pages 55:1–55:6, 2018.

[Hess and Morik, 2017] Sibylle Hess and Katharina Morik. C-SALT: Mining class-specific alterations in Boolean matrix factorization. In *ECMLPKDD*, pages 547–563, 2017.

[Hess *et al.*, 2017] Sibylle Hess, Katharina Morik, and Nico Piatkowski. The PRIMPING routine—Tiling through proximal alternating linearized minimization. *Data Min. Knowl. Discov.*, 31(4):1090–1131, 2017.

[Hess *et al.*, 2018] Sibylle Hess, Nico Piatkowski, and Katharina Morik. The trustworthy pal: Controlling the false discovery rate in Boolean matrix factorization. In *SDM*, pages 405–413, 2018.

[Karaev *et al.*, 2015] Sanjar Karaev, Pauli Miettinen, and Jilles Vreeken. Getting to know the unknown unknowns: Destructive-noise resistant Boolean matrix factorization. In *SDM*, pages 325–333, 2015.

[Kim, 1982] Ki Hang Kim. *Boolean matrix theory and applications*. Marcel Dekker, New York, 1982.

[Kocayusufoglu *et al.*, 2018] Furkan Kocayusufoglu, Minh X. Hoang, and Ambuj K. Singh. Summarizing network processes with network-constrained Boolean matrix factorization. In *ICDM*, pages 237–246, 2018.

[Kumar *et al.*, 2019] Ravi Kumar, Rina Panigrahy, Ali Rahimi, and David P. Woodruff. Faster algorithms for binary matrix factorization. In *ICML*, pages 3551–3559, 2019.

[Liang and Lu, 2019] Lifan Liang and Songjian Lu. Noisy and incomplete boolean matrix factorization via expectation maximization. *CoRR*, abs/1905.12766, 2019.

[Liang *et al.*, 2020] Lifan Liang, Kunju Zhu, and Songjian Lu. BEM: Mining coregulation patterns in transcriptomics via Boolean matrix factorization. *Bioinformatics*, 2020.

[Lim *et al.*, 2015] Shiau Hong Lim, Yudong Chen, and Huan Xu. A convex optimization framework for bi-clustering. In *ICML*, pages 1679–1688, 2015.

[Lucchese *et al.*, 2013] Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. A unifying framework for mining approximate top-$k$ binary patterns. *IEEE Trans. Knowl. Data Eng.*, 26(12):2900–2913, 2013.

[Miettinen and Vreeken, 2014] Pauli Miettinen and Jilles Vreeken. MDL4BMF: Minimum description length for Boolean matrix factorization. *ACM Trans. Knowl. Discov. Data*, 8(4), 2014.

[Miettinen *et al.*, 2006] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. In *PKDD*, page 335–346, 2006.

[Miettinen *et al.*, 2008] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE Trans. Knowl. Data Eng.*, 20(10):1348–1362, 2008.

[Miettinen, 2009] Pauli Miettinen. *Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms*. PhD thesis, University of Helsinki, 2009.

[Miettinen, 2011] Pauli Miettinen. Boolean tensor factorizations. In *ICDM*, pages 447–456, 2011.

[Miettinen, 2012] Pauli Miettinen. On finding joint subspace Boolean matrix factorizations. In *SDM*, pages 954–965, 2012.

[Monson *et al.*, 1995] Sylvia D. Monson, Norman J. Pullman, and Rolf Rees. A survey of clique and biclique coverings and factorizations of (0,1)-matrices. *Bull. ICA*, 14:17–86, 1995.

[Ndaoud *et al.*, 2019] Mohamed Ndaoud, Suzanne Sigalla, and Alexandre B. Tsybakov. Improved clustering algorithms for the bipartite stochastic block model. *CoRR*, abs/1911.07987, 2019.

[Neumann, 2018] Stefan Neumann. Bipartite stochastic block models with tiny clusters. In *NeurIPS*, pages 3871–3881, 2018.

[Orlin, 1977] James Orlin. Contentment in graph theory: Covering graphs with cliques. *Indag. Math.*, 80(5):406–424, 1977.

[Ravanbakhsh *et al.*, 2016] Siamak Ravanbakhsh, Barnabás Póczos, and Russell Greiner. Boolean matrix factorization and noisy completion via message passing. In *ICML*, pages 945–954, 2016.

[Rissanen, 1978] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[Rukat *et al.*, 2017a] Tammo Rukat, Christopher C. Holmes, Michalis K. Titsias, and Christopher Yau. Bayesian Boolean matrix factorisation. In *ICML*, pages 2969–2978, 2017.

[Rukat *et al.*, 2017b] Tammo Rukat, Dustin Lange, and Cédric Archambeau. An interpretable latent variable model for attribute applicability in the Amazon catalogue. In *NIPS Symp. Interpret. ML*, 2017.

[Rukat *et al.*, 2018] Tammo Rukat, Christopher C. Holmes, and Christopher Yau. Probabilistic Boolean tensor decomposition. In *ICML*, pages 4410–4419, 2018.

[Siblini *et al.*, 2019] Wissam Siblini, Pascale Kuntz, and Frank Meyer. A review on dimensionality reduction for multi-label classification. *IEEE Trans. Knowl. Data Eng.*, pages 1–1, 2019.

[Tatti and Miettinen, 2019] Nikolaj Tatti and Pauli Miettinen. Boolean matrix factorization meets consecutive ones property. In *SDM*, pages 729–737, 2019.

[Tyukin *et al.*, 2014] Andrey Tyukin, Stefan Kramer, and Jörg Wicker. BMaD – A Boolean matrix decomposition framework. In *ECML PKDD*, pages 481–484, 2014.

[Vaidya *et al.*, 2007] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: Finding a minimal descriptive set of roles. In *SACMAT*, pages 175–184, 2007.

[van den Broeck and Darwiche, 2013] Guy van den Broeck and Adnan Darwiche. On the complexity and approximation of binary evidence in lifted inference. In *NIPS*, pages 2868–2876, 2013.

[Wicker *et al.*, 2012] Jörg Wicker, Bernhard Pfahringer, and Stefan Kramer. Multi-label classification using Boolean matrix decomposition. In *SAC*, pages 179–186, 2012.

[Xu *et al.*, 2014] Jiaming Xu, Rui Wu, Kai Zhu, Bruce E. Hajek, R. Srikant, and Lei Ying. Jointly clustering rows and columns of binary matrices: Agorithms and trade-offs. In *SIGMETRICS*, pages 29–41, 2014.

[Zha *et al.*, 2001] Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu, and Horst D. Simon. Bipartite graph partitioning and data clustering. In *CIKM*, pages 25–32, 2001.

[Zhou and Amini, 2018] Zhixin Zhou and Arash A. Amini. Optimal bipartite network clustering. *CoRR*, abs/1803.06031, 2018.

[Zhou and Amini, 2019] Zhixin Zhou and Arash A. Amini. Analysis of spectral clustering algorithms for community detection: The general bipartite setting. *J. Mach. Learn. Res.*, 20:47:1–47:47, 2019.