XOR-based Boolean Matrix Decomposition

Jörg Wicker, Yan Cathy Hua, Rayner Rebello
School of Computer Science,
The University of Auckland,
Auckland, New Zealand
j.wicker@auckland.ac.nz,
{yhua219,rreb176}@aucklanduni.ac.nz

Bernhard Pfahringer
Department of Computer Science,
University of Waikato,
Hamilton,New Zealand,
bernhard@cs.waikato.ac.nz

***Abstract*—Boolean matrix factorization (BMF) is a data summarizing and dimension-reduction technique. Existing BMF methods build on matrix properties defined by Boolean algebra, where the addition operator is the logical inclusive OR and the multiplication operator the logical AND. As a consequence, this leads to the lack of an additive inverse in all Boolean matrix operations, which produces an indelible type of approximation error. Previous research adopted various methods to address such an issue and produced reasonably accurate approximation. However, an exact factorization is rarely found in the literature. In this paper, we introduce a new algorithm named XBMAD (XOR-based Boolean Matrix Decomposition) where the addition operator is defined as the exclusive OR (XOR). This change completely removes the error-mitigation issue of OR-based BMF methods, and allows for an exact error-free factorization. An evaluation comparing XBMAD and classic OR-based methods suggested that XBMAD performed equal or in most cases more accurately and faster.**

I. Introduction

Data summarization is an important underpinning of machine learning and data mining. It exists in many forms, such as dimensionality reduction, data compression, and pattern identification [28], [21], [9], [15]. Effective data summarization benefits almost all tasks, including data storage, communication, pre-processing, modelling, and learning.

Among the most common data types, instance-attribute relations are often represented as a matrix. When the focus is on a Boolean relationship, such as the absence or presence of each attribute per instance, the data can be represented as a Boolean matrix. Categorical and continuous data are also often re-coded into Boolean variables during pre-processing. Boolean matrices are therefore a data format encountered frequently in machine learning and data mining tasks.

Boolean matrix factorization (BMF), also commonly known as Boolean matrix decomposition, is a data summarization technique for data that is represented as a Boolean-valued matrix [35]. It has gained increasing importance and popularity along with the rapid upsurge of data scale and data complexity, and has applications in multiple fields such as data mining, machine learning [32], bioinformatics [28], or recommender systems [17].

The aim of BMF is to represent a $n \times m$ Boolean-valued matrix as a product of two ($n \times k$, $k \times m$) factor matrices -which are often smaller. One of the factor matrices can be viewed as a basis matrix describing meaningful features / attributes

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

$$T = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \Rightarrow A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \quad P = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

(a) OR-based factorization

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

$$T = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \Rightarrow A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \quad P = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

(b) XOR-based factorization

Fig. 1: Example factorization and reconstruction using OR (Figure 1a) and XOR (Figure 1b) as the Boolean product. We factorize matrix $\mathbf{T}$ such that $\mathbf{T} \approx \mathbf{P} = \mathbf{A} \circ \mathbf{B}$, where $\mathbf{T} = \mathbf{P}$ when the factorization is exact. Note that the same factor matrices are used, but XOR is able to avoid one error in the reconstruction.

of the dataset and the other describes different observed combinations of these features. Although the problem has been shown to be NP-hard, the past few decades have seen the development of a number of effective heuristic algorithms [28], [21].

So far, most of the well-known BMF algorithms are based on standard Boolean algebra, the characteristics of which pose challenges for tackling certain types of approximation error. The main problem arises from the fact that when there is a 1 in a factor of the product, there is no possibility to change the result back to a 0. Therefore the factor matrices tend to be sparse to compensate for this problem. In this paper, we address this by changing the definition of the Boolean algebra to use the XOR operator for the Boolean product. This change can fix certain errors in the factorization and can achieve an exact factorization (given a high enough factor size) – this differs from most OR-based approaches. An example factorization is given in Figure 1.

There are three main contributions by this paper: (1) We change the Boolean algebra to include an additive inverse, allowing for exact factorizations. (2) We propose an algorithm based on the new Boolean algebra. (3) We evaluate the new algorithm on standard data sets and compare to state-of-the-art approaches.

In the following sections, we provide a summary of the BMF problem, the challenge of using standard Boolean-algebra, and propose a modified algebra and a new algorithm using the exclusive-or (XOR) operand. In the experiment section, we share results of comparing our new algorithm with some OR-based counterparts, followed by discussion.

II. Problem definition

A Boolean matrix is any matrix $\mathbf{T}$ with Boolean-valued entries, i.e., $\mathbf{T}_{ij} \in \{true, false\}$, where the truth values are often represented by 1 and 0 respectively. Boolean Matrix Factorization (BMF) is the problem that given an $n \times m$ Boolean matrix $\mathbf{T}$, find two factor Boolean matrices $\mathbf{A}$ ($n \times k$) and $\mathbf{B}$ ($k \times m$) such that the Boolean matrix product of $\mathbf{A}$ and $\mathbf{B}$ is either equivalent to or an approximation of $\mathbf{T}$, i.e., $\mathbf{T} \approx \mathbf{P} = \mathbf{A} \circ \mathbf{B}$, where $\mathbf{T} = \mathbf{P}$ when the factorization is exact [28].

In the above definition, the dimension k is the Boolean rank, defined as the smallest number of factors k required to solve the DBP problem [21], [4]. The symbol $\circ$ represents the Boolean matrix product defined under Boolean algebra, where addition is the logical inclusive disjunction, and multiplication the logical conjunction [21]. Thus, the Boolean matrix multiplication is defined as

$$(\mathbf{A} \circ \mathbf{B})_{ij} = \bigvee_{l=1}^{k} (\mathbf{A}_{il} \wedge \mathbf{B}_{lj}) \tag{1}$$

and the Boolean matrix addition is

$$(\mathbf{A} \vee \mathbf{B})_{ij} = \mathbf{A}_{ij} \vee \mathbf{B}_{ij} \tag{2}$$

One common interpretation of the factor matrices obtained from BMF is to view the original matrix $\mathbf{T}$ as the object-attribute matrix with rows as objects and columns as different attributes. The corresponding factor matrices $\mathbf{A}$ and $\mathbf{B}$ are then called usage and basis vector matrices [21]. In this sense, adopting the matrix-left-multiplication view, each row of $\mathbf{B}$ represents a basis factor, i.e., a particular combination of attributes with observed association above a certain threshold; and the i-th row of $\mathbf{P}$ is a Boolean vector/matrix sum of the rows in $\mathbf{B}$ where corresponding entries in the i-th row in $\mathbf{A}$ are $1s$ [21], [4].

The BMF problem has been shown to be NP-hard due to the NP-completeness of the underpinning set-basis problem [4], [11], and thus approximate factorization is most common. The accuracy of the approximation is often measured in one of two ways: by an error computed via a distance function involving some type of matrix norm [21], [13], or by a coverage index c indicating the extent to which the values (often $1s$) in the original matrix are covered in the approximation [4], [5]. Overall, past research often measures factorization quality using one or both of the following criteria [4]: 1) for a fixed Boolean rank k, a factorization that yielded smaller error or greater coverage is considered better; and 2) within an acceptable error/coverage threshold, a factorization that requires a smaller k is considered better. Some authors [4] further suggested focusing on the marginal effect of the first few factors, as a good factorization should reach high coverage / low error with very small k; and within the first small number of factors, each increase in k results in a steep improvement in error/coverage.

III. Related work

As a sub-topic of matrix factorization, BMF attracts research interest for several reasons: First, Boolean-valued data are naturally used when the analytical focus is on a yes/no relation between subjects and attributes, such as user membership and class enrolment [21]. Second, although Boolean-valued matrices can be factorized using real-valued factorization methods such as singular value decomposition (SVD) and non-negative matrix factorization (NMF), the output factor matrices are harder to make sense of due to the decimal and negative values. On the other hand, BMF provides Boolean-valued factor matrices that are easier to interpret [21].

Recently, BMF research has drawn increasing research interest within the data mining community [13], [28]. Miettinen *et al.* [21], Belohlavek *et al.* [4], and Sun *et al.* [28] provide reasonably structured reviews of representative BMF studies, whereas the remainder of this section will focus on how some past work addresses the Boolean matrix addition problem introduced below.

The matrix operation under Boolean algebra has one major distinction from that of real-valued matrices: With inclusive OR ($\vee$) as addition, where $1 \vee 1 = 1$ and $1 \vee 0 = 1$, there is no additive inverse $\bar{1}$ of 1 such that $1 \vee \bar{1} = 0$. Consequently, for any given real-valued $n \times m$ matrices $\mathbf{T}$ and X, there always exists an $n \times m$ matrix Y such that $\mathbf{T} = X + Y$, but with Boolean matrices, this is not always the case. For example, let two Boolean-valued matrices $\mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ and $\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. No Boolean value in $\mathbf{Y}_{00}$ can "cancel out" the 1 at $\mathbf{X}_{00}$ to make $\mathbf{T} = \mathbf{X} \vee \mathbf{Y}$ hold. Thus, in BMF where $\mathbf{P} = \mathbf{A} \circ \mathbf{B}$, if the original matrix has entry $\mathbf{T}_{ij} = 0$, but $\mathbf{P}_{ij} = 1$, there is no way to rectify this error with the standard definition of OR.

Prior studies made various attempts to address the above-mentioned problem [21], [4], [28], [5], [6], [14]. One example is the Asso algorithm by Miettinen *et al.* [21], which is standard baseline method in BMF [13], [28]. Asso performs a greedy search; it is initialized using an association matrix generated from the input matrix with a confidence threshold τ. The search process is allowed to commit both 1-to-0 and 0-to-1 errors (where the original matrix's entry $\mathbf{T}_{ij} = 1$ but the approximation $\mathbf{P}_{ij} = 0$, and vice versa); moreover, the search is steered by a cover function, which rewards accurate covering of $1s$ and punishes the 0-to-1 error in order to address the problem stated above [21], [5].

Some BMF methods based on the minimum-tiling problem [12] make a more extreme attempt to address the matrix addition problem by completely avoiding the 0-to-1 type error.

One such method is the "from-below" approximation [5], [6], which aims at covering as many existing $1s$ as possible without introducing additional 1s that the original matrix does not have. This improves the interpretability of the results: as the output basis factors ($\mathbf{B}\_{\cdot j}$, the j-th row in matrix $\mathbf{B}$ in $\mathbf{T} \approx \mathbf{A} \circ \mathbf{B}$) represent a subset of the original matrix features, $1s$ in the corresponding entries in the other factor matrix (in this case, $\mathbf{A}$) indicate that the subject truly has every feature described in $\mathbf{B}\_{\cdot j}$ [5]. Similarly, Sun *et al.* [28] also poses the from-below restriction on the 0-to-1 error, with an additional "column-use" condition that the columns of the usage matrix $\mathbf{A}$ should be subsets of those of the original matrix $\mathbf{T}$ in order to reduce candidate search space.

On another track, methods such as Asso-XOR-DtM [22] and C-Salt [13] merged tiling-based Boolean factorization with the numerical optimization principle Minimum Description Length (MDL). Since MDL requires exact reconstruction of the input matrix [22], the authors introduced the concept of an error matrix to accompany the approximation for a lossless reconstruction and used different encodings to "fix" the 0-to-1 errors in the approximation. In C-Salt, the error matrix N is a non-Boolean matrix defined as $\mathbf{N}_{ij} \in \{-1, 0, 1\}$ such that $\mathbf{T} = \mathbf{A} \circ \mathbf{B} + \mathbf{N}$, where $\circ$ still represents the Boolean matrix product, but $+$ is the addition defined over real numbers [13]. In Asso-XOR-DtM [22], Miettinen and Vreeken explicitly highlighted the non-inversibility issue of OR-based Boolean addition and the role of 0-to-1 error. Their solution was to use the exclusive-OR (XOR, $\oplus$) to combine their Boolean-valued error matrix with the approximation, such that $\mathbf{N} = \mathbf{T} \oplus (\mathbf{A} \circ \mathbf{B})$. The heuristics for computing MDL cost is then guided by the goal of minimizing the number of 1's in N. Their experiment results suggest favorable model order and accuracy of the XOR-encoding of error matrix, however, the factorization results are still approximate [22].

Similarly, Kumar *et al.* recently proposed algorithms for fast Boolean matrix factorization [18]. They propose an algorithm that is applied to both traditional OR-based Boolean matrix factorization, and then extended to use $GF(2)$, i.e. XOR instead of the OR. A related problem is the low-rank matrix approximation problem, which is a step in BMF. This was addressed using XOR by Fomin *et al.* [10]. XOR has also been used in independent component analysis [34], [25], [24].

The above review highlights the BMF literature's awareness and some recent attempts in addressing the shortcomings brought by the OR-based Boolean matrix products. In the more recent studies mentioned, approximation errors were explicitly recorded in matrix form [22], yet such an error matrix serves limited use in the factorization process.

IV. Boolean Matrix Factorization

In Section II, we introduced BMF as the task of factorizing a Boolean matrix $\mathbf{T}$ of dimension $n \times m$ into two factor matrices, $\mathbf{A}$ and $\mathbf{B}$ with dimensions $n \times k$ and $k \times m$ respectively. When these factor matrices are multiplied under some definition of a Boolean matrix product returns $\mathbf{R}$ – an approximate or lossless reconstruction of $\mathbf{T}$. Such a factorization would be considered to be a rank-k factorization, generally, the higher the rank the closer $\mathbf{R}$ will match $\mathbf{T}$. The definition of Boolean matrix multiplication in the literature is given in Equation 1 and is analogous to normal matrix multiplication.

This definition of a Boolean matrix product known as OR-Product as the $\vee$-operator is the analog to summation under normal matrix multiplication. Algorithms to solve BMF problems for a fixed k search the space of factorizations for a good approximation. The primary metric to measure the effectiveness is the reconstruction error of two Boolean matrices $\epsilon(\mathbf{R}, \mathbf{T})$ - the number of bits in which $\mathbf{R}$ and $\mathbf{T}$ differ, given by the formula below.

$$\delta(\mathbf{R}, \mathbf{T})_{ij} = \begin{cases} 1 & \text{if } \mathbf{R}_{ij} \neq \mathbf{T}_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\epsilon(\mathbf{R}, \mathbf{T}) = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} \delta(\mathbf{R}, \mathbf{T})_{ij} \tag{4}$$

A. Hillclimbing

As mentioned above, BMF is NP-hard, i.e. problems of non-trivial sizes can not be practically solved to an optimal factorization (with lowest reconstruction error) for a fixed k. Heuristics are employed to give a "good enough" factorization and make problems tractable. The most straightforward heuristic approach is a hillclimbing approach. The hillclimber explores the neighbourhood of factor pairs - solutions to the BMF - $\mathbf{A}, \mathbf{B}$ that differ by exactly one bit. More formally, the neighbourhood rule for a solution is given below.

$$\begin{aligned} N(\mathbf{A}, \mathbf{B}) = \{ & (\mathbf{A}', \mathbf{B}') | \\ & \epsilon(\mathbf{A}', \mathbf{A}) \times (n \times k) + \epsilon(\mathbf{B}', \mathbf{B}) \times (m \times k) = 1 \} \end{aligned} \tag{5}$$

There are exactly $k(n + m)$ neighbouring solutions for every solution. Hillclimbing greedily searches through these neighbourhoods keeping track of a single solution with the least reconstruction error ϵ. This is repeated until we reach a neighborhood with no improvement possible – a locally optimal solution with respect to the neighbourhood rule defined above. Pseudo-code describing hillclimbing is given in Algorithm 1.

Algorithm 1: Hillclimbing algorithm NextDescent

```
Input: $\mathbf{T}, k$
Output: $\mathbf{A}, \mathbf{B}$
1  $\mathbf{A}, \mathbf{B} \leftarrow$ Initialise$(\mathbf{T}, k)$
2  incumbentError $\leftarrow \epsilon(\mathbf{A} \circ \mathbf{B}, \mathbf{T})$
3  improved $\leftarrow true$
4  while improved do
5      improved $\leftarrow false$
6      for $\mathbf{A}', \mathbf{B}' \in N(\boldsymbol{A}, \boldsymbol{B})$ do
7          if $\epsilon(\mathbf{A}' \circ \mathbf{B}', \mathbf{T}') <$ incumbentError then
8              $\mathbf{A}, \mathbf{B} \leftarrow \mathbf{A}', \mathbf{B}'$
9              incumbentError $\leftarrow \epsilon(\mathbf{A}' \circ \mathbf{B}', \mathbf{T})$
10             improved $\leftarrow true$
```

The procedure is called NEXTDESCENT referring to the fact that the next neighbouring solution which gives a lower reconstruction error ϵ is chosen immediately. There is the option of exhausting the entire neighbourhood for a single solution until the neighbouring solution with the least error is chosen but this is often computationally prohibitive. The exact locally optimal solution found depends on the initialization. Often, this initialization is non-deterministic so there are different local optima every time the algorithm completes a restart.

V. XOR-BASED BOOLEAN MATRIX DECOMPOSITION

The key idea of switching to using XOR instead of OR in BMF is that the XOR operation allows fixing errors of a factorization. Consider a reconstructed matrix and its errors in the reconstruction. A residual matrix with 1s at the positions of all reconstruction errors can be used an XOR combination with the reconstructed matrix. The result would be the original matrix.

This property is used in our algorithm – XBMAD (XOR-based Boolean Matrix Decomposition), where the main idea is to partition the BMF problem into finding good rank 1-factor matrices and using the XOR operation to iteratively "repair" the factorizations. This approach is similar to other machine learning areas, e.g. Ripple Down Rules [26], where initial models are fixed and refined by using improvements on the residual, i.e. the parts of the data that cannot be perfectly modelled by the initial model.

A. XOR *Reformulation*

To formulate the XBMAD algorithm, we will first define XOR-based BMF. Let $\mathbf{E}$ be the error or residual matrix of an approximation $\mathbf{P}$. The entries of $\mathbf{E}$ are 1 if the approximation in $\mathbf{P}$ differs from the target $\mathbf{T}$ in that position, 0 otherwise. This is equivalent to the element-wise XOR of $\mathbf{R}$ with $\mathbf{T}$.

$$\mathbf{E}_{ij} = \begin{cases} 1 & \text{if } \mathbf{T}_{ij} \neq \mathbf{R}_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$= \mathbf{R}_{ij} \veebar \mathbf{T}_{ij} \tag{7}$$

The target matrix $\mathbf{T}$ can be re-obtained by element-wise XOR.

$$\mathbf{T}_{ij} = \mathbf{R}_{ij} \veebar \mathbf{E}_{ij} \tag{8}$$

Therefore, any target matrix can be exactly represented by the element-wise XOR (denoted by $\oplus$) of an approximated matrix with the error matrix associated with that approximate factorization. The error matrix, by virtue of being a Boolean matrix, can also be factorized into two other factor matrices. For example, if some $\mathbf{T}$ can be factorized into $\mathbf{A^1}, \mathbf{B^1}$ with an error matrix $\mathbf{E^1}$ associated with it. $\mathbf{T} = (\mathbf{A^1} \circ \mathbf{B^1}) \oplus \mathbf{E^1}$. Suppose $\mathbf{E^1}$ has a factorization with factor matrices $\mathbf{A^2}, \mathbf{B^2}$, $\mathbf{T}$ can be written as the following.

$$\mathbf{T} = (\mathbf{A^1} \circ \mathbf{B^1}) \oplus (\mathbf{A^2} \circ \mathbf{B^2}) \oplus \mathbf{E^3} \tag{9}$$

$$\implies \mathbf{T} \approx (\mathbf{A^1} \circ \mathbf{B^1}) \oplus (\mathbf{A^2} \circ \mathbf{B^2}) \tag{10}$$

This process of continuously factorizing the error matrix can be repeated any number of times. So the two factorization example above scales without loss of generality for an arbitrary amount of factorization. Under OR-product this means that the factor pairs for the two separate factorizations must be kept. This motivates an alternate definition of the Boolean matrix product, XOR-Product where the XOR is used as the analog for addition. The explicit formulation is shown below:

$$(\mathbf{A} \otimes \mathbf{B})_{ij} = \underline{\bigvee}_{l=1}^{k} (\mathbf{A}_{il} \wedge \mathbf{B}_{lj}) \tag{11}$$

If the factorizations were solved under XOR-factorizations then the approximation, i.e. $\mathbf{T} \approx (A^1 \otimes B^1) \oplus (A^2 \otimes B^2)$. The factors can be concatenated. The proof is as follows:

$$\mathbf{R}_{ij} = \left[(\mathbf{A^1} \otimes \mathbf{B^1}) \oplus (\mathbf{A^2} \otimes \mathbf{B^2})\right]_{ij} \tag{12}$$

$$= \left[\underline{\bigvee}_{l_1=1}^{k_1} \left(\mathbf{A^1}_{il_1} \wedge \mathbf{B^1}_{l_1 j}\right)\right] \veebar \left[\underline{\bigvee}_{l_2=k_1+1}^{k_1+k_2} \left(\mathbf{A^2}_{il_2} \wedge \mathbf{B^2}_{l_2 j}\right)\right] \tag{13}$$

Relabelling $l_2 \in \{k_1+1, ..., k_1+k_2\}$

$$= \left[\underline{\bigvee}_{l_1=1}^{k_1} \left(\mathbf{A^1}_{il_1} \wedge \mathbf{B^1}_{l_1 j}\right)\right] \veebar \left[\underline{\bigvee}_{l_2=k_1+1}^{k_1+k_2} \left(\mathbf{A^2}_{il_2} \wedge \mathbf{B^2}_{l_2 j}\right)\right] \tag{14}$$

$$= \underbrace{\left(\mathbf{A^1}_{i1} \wedge \mathbf{B^1}_{1j}\right) \veebar \ldots \left(\mathbf{A^1}_{il_1} \wedge \mathbf{B^1}_{l_1 j}\right)}_{l_1 \text{terms}}$$

$$\underbrace{\veebar \left(\mathbf{A^2}_{i(l_1+1)} \wedge \mathbf{B^2}_{(l_1+1)j}\right)}_{l_2 \text{ terms}} \tag{15}$$

$$\underbrace{\veebar \ldots \left(\mathbf{A^1}_{i(l_1+l_2)} \wedge \mathbf{B^1}_{(l_1+l_2)j}\right)}_{l_2 \text{ terms}} \tag{16}$$

$$= \left[\mathbf{A^1} \mathbf{A^2}\right] \otimes \begin{bmatrix} \mathbf{B^1} \\ \mathbf{B^2} \end{bmatrix} \tag{17}$$

B. XBMAD *Algorithm*

The XBMAD (XOR-based Boolean Matrix Decomposition) algorithm recursively approximates the error matrix with lower rank factorizations, giving a pair of factors associated with each approximation. Using the XOR- definition of a Boolean matrix product, the final factorization can be returned by concatenating the lower rank factorizations into two factor matrices yielding a factorization of rank k. Algorithm 2 gives the XBMAD algorithm.

The initialization of rank-1 factors is done by selecting one random non-empty row of $\mathbf{T}$ in $\mathbf{b}$ and setting the corresponding a to all zeros. This is repeated until the specified k rank is reached or a lossless factorization ($\epsilon = 0$) is achieved. Figure 2 gives an example factorization of a 3×3 matrix.

VI. EXPERIMENTS

To evaluate the performance of the XBMAD algorithm, we compared it to other algorithms proposed in BMF literature on

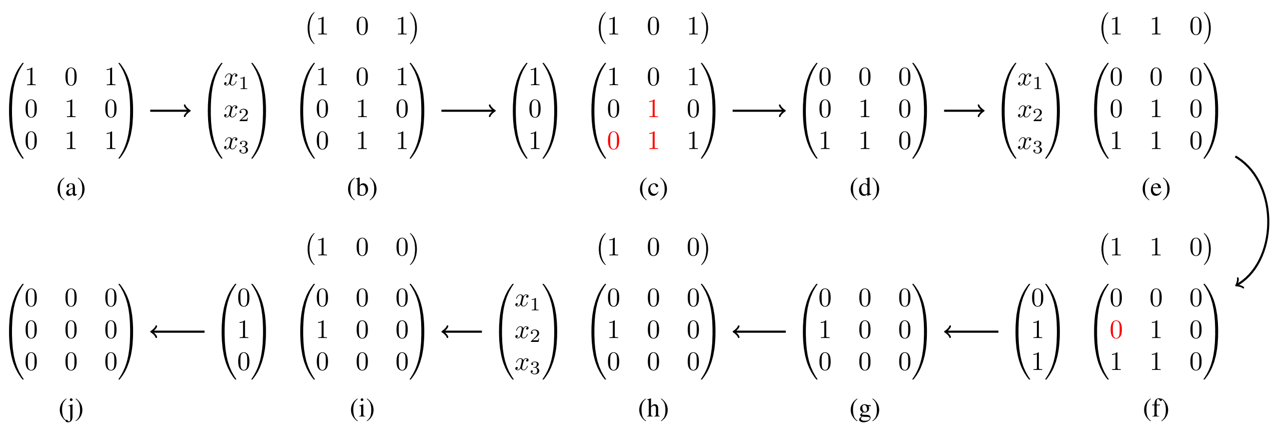


Fig. 2: Example run of the XBMAD algorithm on a 3×3 matrix with $k = 3$. (a) shows the original matrix, in (b), the first factor is initialized by choosing a non-empty row of the matrix. In (c) then the second factor is optimized to minimize the error, leaving only the marked entries wrongly classified. (d) gives the residual matrix, with 1s where the errors are in the reconstruction. (c) to (g) then repeat the steps (b) to (d) on the residual matrix, resulting in a new residual matrix. The same is repeated in (h) to (j) leaving an empty residual matrix, i.e. a lossless factorization. Finally, the factors are combined (not shown) such that the resulting factorization are matrices are $\begin{pmatrix}1&0&0\\0&1&1\\1&1&0\end{pmatrix}$ and $\begin{pmatrix}1&0&1\\1&1&0\\1&0&0\end{pmatrix}$. Note that each residual matrix ((d), (g), and (j)) can be used to calculate the current reconstruction error by simply calculating the density of each matrix ($1/3$, $1/9$, and 0 respectively).

Algorithm 2: XBMAD

```
Input: $\mathbf{T}, k$
Output: $\mathbf{A}, \mathbf{B}$
1  $\mathbf{A}, \mathbf{B} \leftarrow [\ ], [\ ]$
2  $\mathbf{Q} \leftarrow \mathbf{T}$
3  $notdone \leftarrow true$
4  $l \leftarrow 1$
5  while $notdone$ do
6      $\mathbf{a}, \mathbf{b} \leftarrow$ NEXTDESCENT $(\mathbf{Q}, 1)$
7      $\mathbf{A}, \mathbf{B} \leftarrow$ CONCATENATE $(\mathbf{A}, \mathbf{B}, \mathbf{a}, \mathbf{b})$
8      $\mathbf{Q} \leftarrow (\mathbf{A} \otimes \mathbf{B}) \oplus \mathbf{T}$
9      $l \leftarrow l + 1;$
10     $notdone \leftarrow \epsilon(\mathbf{A} \otimes \mathbf{B}, \mathbf{T}) > 0 \wedge l < k$
```

TABLE I: Datasets used in the experiments with their dimensions. Density gives the fraction of 1s in the matrix.

Dataset	Height	Width	Density	
Audiology	200	317	0.060	[3]
Breast	699	91	0.102	[33]
Car	1728	25	0.280	[7]
Digits	2000	240	0.608	[31]
Chess (krvskp)	3196	75	0.400	[29]
Mushroom	8124	119	0.176	[27]
Nursery	12960	28	0.268	[23]
Phishing	1353	27	0.338	[1]
Soybean	307	100	0.215	[20]
Student	649	177	0.186	[8]
Tic-tac-toe	958	28	0.345	[19]
Zoo	101	28	0.305	[16]

several datasets. Our implementation of XBMAD is in Java, it is publicly available as part of the BMaD library [30]. The implementations of Asso or similar algorithms proposed by Miettinen are in the BMaD [30] library which is written in Java. The implementations of the other algorithms evaluated were provided by their respective authors.

A. Experiment Design

For every algorithm evaluated, we factorize each matrix $\mathbf{T}$ of size $n \times m$ to a targeted rank $k \in \{k \mid k = 2^a\ \forall a \in \mathbb{N} \wedge k < \min(n, m)\}$ and record all the statistics of the binary reconstruction R produced. Since every algorithm that was tested relies on some randomization, we ran a total of 100 runs of every dataset and k value pair. A limit of 30 minutes was set for the runtime. Note that the reconstruction matrix was produced using XOR-based Boolean algebra in the cases of XOR-based algorithms (XBMAD and Hillclimbing$_{XOR}$), and OR-based Boolean algebra for OR-based algorithms. In cases where algorithms automatically optimize the rank internally, we set the parameters to choose a fixed rank for our experiments to compare the performance on a given size of the factor matrices.

B. Datasets

We conducted tests using 12 real-world datasets of various sizes from the UCI Machine Learning repository[1]. Details on the datasets are shown in Table I. We treat missing values as false bits as it is not the focus of this paper.

C. Evaluation Metrics

In our experiments, the quality of factorization for a target matrix $\mathbf{T}$ is measured by comparing $\mathbf{T}$ to the reconstructed

[1]see http://archive.ics.uci.edu/ml

matrix R. The reconstruction is the primary evaluation metric given by $\epsilon(T, R)$ (see Equation 4). In addition, we used the well-known metrics recall, precision, and F1-Measure. Recall gives the fraction of all true positives, i.e. 1s in the reconstructed matrix that are also 1s in the original matrix, over all 1s in the original matrix:

$$recall = \frac{TP}{TP + FN} \tag{18}$$

Precision is then the fraction of all true positives, i.e. 1s in the reconstructed matrix that are also 1s in the original matrix, over all 1s in the reconstructed matrix:

$$precision = \frac{TP}{TP + FP} \tag{19}$$

And F1-Measure gives the harmonic mean of the two:

$$F1 = \frac{2 \times recall \times precision}{recall + precision} \tag{20}$$

In addition, we also measured the runtime of the test algorithms to provide an approximate indication of their performances and scalability despite their implementation differences.

Note that both algorithms Primp and Panpal use the CUDA library so they were tested in a different environment than the other algorithms.

D. Algorithms

Overall, we compared XBMaD to eight state-of-the-art matrix factorization algorithms:

a) Hillclimbing: Hillclimbing uses the NextDescent Algorithm 1. This is the basis of XBMaD, used to find low-rank factorizations. We tested two configurations: One using the OR and the other using Xor definition of a Boolean product. We refer to them as Hillclimbing$_{OR}$ and Hillclimbing$_{XOR}$ respectively.

b) FastStep: This algorithm by Araujo *et al.* [2] uses a thresholding operator and relaxation of those operators to model BMF as a numerical optimization problem. This is solved by a gradient descent approach and uses a scalable method to approximate the gradient for each step. This algorithm yields non-negative factors for its binary reconstruction. In the experiments we used the *approximate gradient* approach and left all the parameters as their defaults.

c) PRIMP and PANPAL: Both approaches are presented by Hess *et al.* in the PALTiling framework [14] which uses BMF to solve the Tiling Problem. This is a numerical optimization method somewhat similar to FastStep with binary factors. We used a rank increment of 1 and set the max rank to the desired k. Since k is set to the max rank, the algorithms usually do not reach the full value of k, generally stopping around a value of 4. A solution that was recommended in the paper is to set the rank increment to max rank, however in our experiments that resulted in a rank 0 (or all zero) reconstruction. So we decided on using a rank-increment of 1.

d) DBP, Asso, and Loc-Iter: These are standard algorithms described above and in previous publications. Asso is based on a greedy search and initialized using association rules, all algorithms are implemented in the BMaD library [21], [30].

VII. Results

The results calculating the F1-Measure of the factorizations are given in Figure 3. We give the mean of the 100 runs. Due to space limitations, we display only a selection of representative figures of the result. The full results and figures are provided in the supplements [2]. The given plots are representatives of the overall results and similar behaviour can be found in the remaining results. Note that FastStep did not finish in time for all the experiments and therefore is missing entries in the plots.

We can see that XBMaD in most cases, outperforms other algorithms. Just in the case of higher density datasets, such as "digits", the simple OR-based hillclimbing achieves a comparable performance. This is due to the fact that any OR-based factorization results in a higher number of 1s – once a product (AND operation) produces a 1 in a column-row pair, the result becomes 1. In a denser matrix, this is less likely to cause errors compared to a sparser matrix. This applies to all OR-based matrices and explains the comparable worse performance of XBMaD in these data sets, i.e. it does not perform worse than in other data sets, the other algorithms just "catch up" with its performance. Additionally, this shows that there are easier structures that can be represented by simple OR-based algebra.

For most algorithms the performance on the krvskp dataset stagnates quickly with increasing k. However, XBMaD keeps improving and reaches a lossless factorization around $k = 32$. This shows the advantage of XBMaD in that it can always achieve a lossless reconstruction given a high enough k.

To further understand the differences found in different algorithms' performance, we give the precision of the reconstructions in Figure 4. These results show how well an algorithm is able to reconstruct the 1s in the original matrix compared to all 1s in the reconstructed matrix. As expected, OR-based algorithms tend to produce more 1s in the reconstructed matrix and therefore match the 1s in the original matrix more easily.

Putting the results on precision in relation to the results on F1-measure, we can see that many OR-based algorithms catch up with the performance of Xor-based algorithms. This is caused by the OR-based algorithms tending to predict more 1s than Xor-based algorithms, making it easier to match the 1s in the original matrix. If there are two matching 1s in each of a factor, no matter what is in the remaining rows or columns. While the overall number of 1s increases when using OR-based algorithms, and therefore also the number of FPs, they seem to match enough of the TPs at the same time to raise the overall precision.

[2]Supplementary material is available in the BMaD repository at https://github.com/joergwicker/bmad/tree/master/xbmad-supplementary

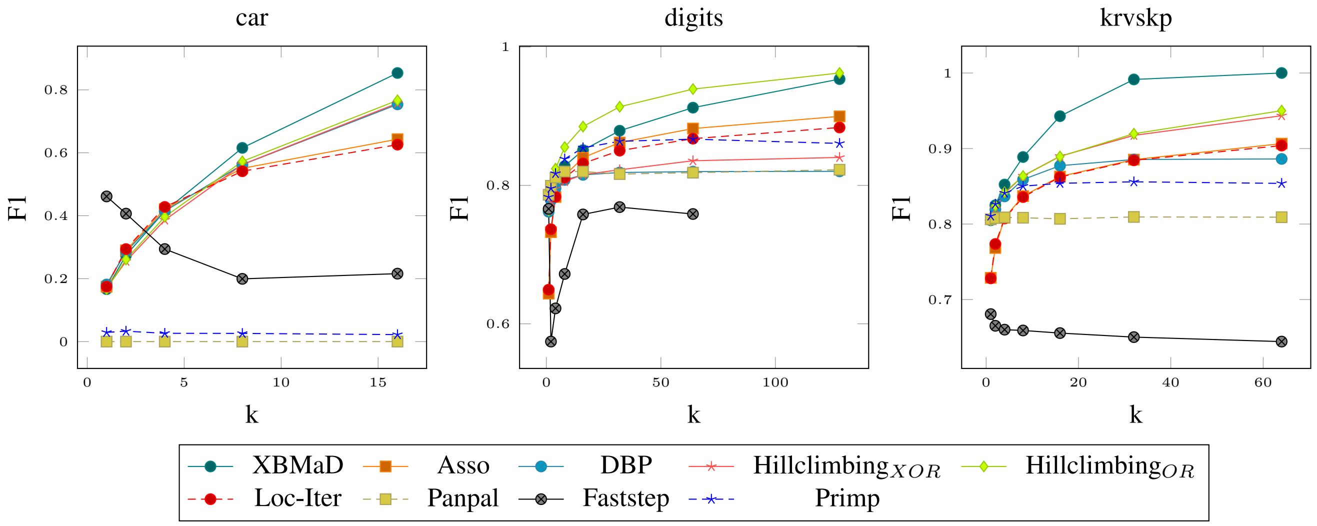


Fig. 3: Results of F1-measure on selected representative datasets (full results are given in the supplement). Choosing XBMAD generally outperforms the other algorithms. In case of the digits dataset, the naive Hillclimbing approach using OR outperforms the other algorithms, which indicates that there seem to be simple structures that can be easily presented using OR and relatively easy to be found in the matrix.

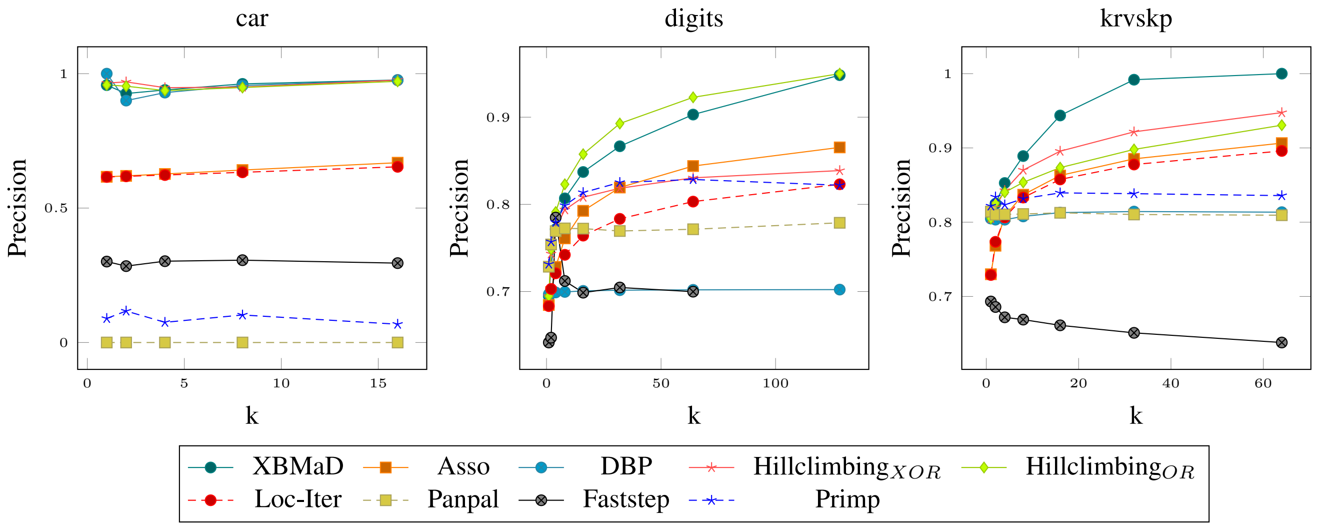


Fig. 4: Results of precision on selected representative datasets. Again, the results show that in the case of the digits data set, the performance of XBMAD can be reached by OR-based algorithms. Overall these results prove the assumption that OR-based algorithms produce more 1s and therefore match the 1s in the original matrix more easily.

This behaviour becomes most obvious in the car data set (or similar data sets in the supplement). Numerous OR-based algorithms are able to achieve high precision. Notably, DBP and the OR-based Hillclimbing algorithm achieve the highest precision over all values of k.

Figure 5 shows the results of the evaluation based on recall. Overall, XBMAD tends to achieve higher recall and outperform OR-based approaches in many cases. Note that in the case of the car dataset, XBMAD needs a relatively low value of k to achieve an almost lossless decomposition with regard to recall, while other approaches need a higher value of k to achieve similar results.

The results of the car dataset shows that XBMAD in comparison corrects more errors (1s) in the residual matrix, archives higher recall, and results in more 1s matched in both factor matrices. With lower k, other algorithms tend to achieve a better recall, but they seem to reach a limit rather early, shown by the curves flat and stabilizing while the recall of XBMAD keeps improving.

Results from the digits dataset shows that with a small k, XBMAD can already achieve a high recall. This suggests that with the right structure, XBMAD can quickly “fix” the errors

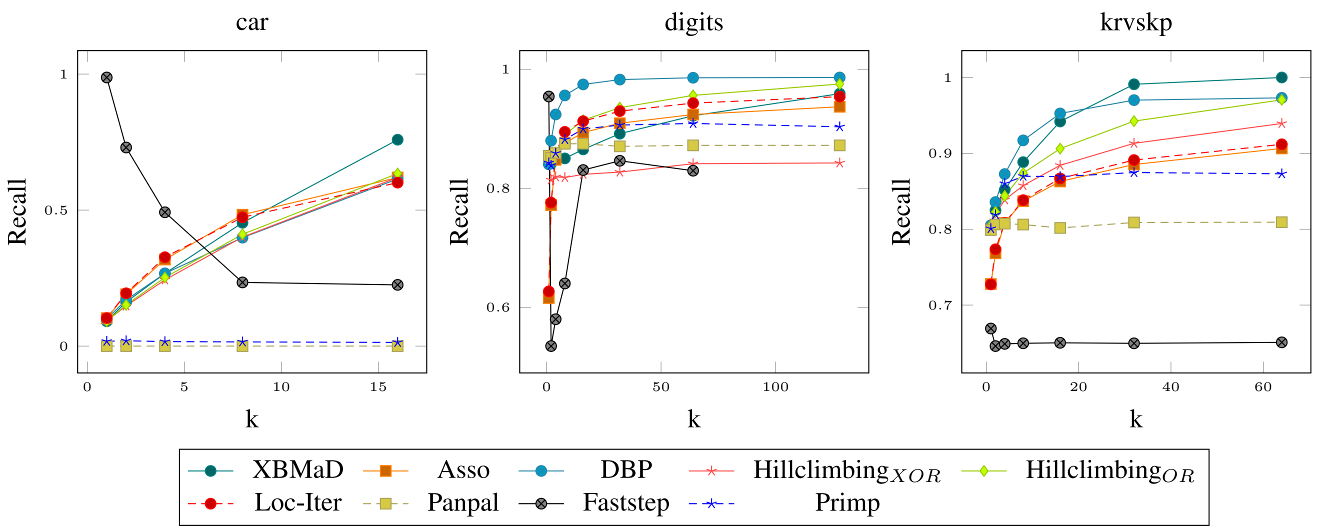


Fig. 5: Results of Recall, i.e. the fraction of 1s in both the reconstructed matrix and the original matrix over all 1s in the original matrix. The results show that this seems to be an easier task for XOR-based algorithms. XBMAD faster reaches a near-perfect factorization in digits and achieves better results on the car dataset on higher k compared to other algorithms.

in the residual matrix. On the other hand, since OR-based algorithms tend to generate approximated matrices with higher density, they are also more prone to generate mismatched 1s as the result suggests.

Figure 6 shows the reconstruction error on the selected datasets. The lower the error, the better the performance. Overall, the reconstruction error exhibits similar patterns to the F1 measure across all datasets. The explanation of the observed behavior is similar to that for F1 mentioned above. F1-measure averages recall and precision, thus measuring a value very similar to what the reconstruction error gives. Note again the low reconstruction error of OR-based Hillclimbing on the denser digits data set.

Finally, Figure 7 shows the runtime behaviour of the algorithms. XBMAD exhibits near linear runtime, as every increase in k results in the same operation to be repeated. In higher ks, the runtime of each step should reduce given that the density of the residual matrices will reduce, resulting in lower complexity for the 1-factor factorization. However, due to the overall low runtime of the approach, this cannot be seen in the figures. Compared to other approaches, XBMAD's runtime is very low.

To statistically evaluate the results, we carried out a Nemenyi-Friedman test and plotted the results in Figure 8. The results' overall rankings show that XBMAD performs similar to XOR-based Hillclimbing and are together significantly better than other algorithms. However, F1-measure shows that while Hillclimbing does not significantly outperform DBP, XBMAD does.

VIII. Conclusion

In this paper, we introduced XBMAD, an XOR-based Boolean matrix factorization algorithm. To do this, we adapted the Boolean algebra to use XOR instead of OR, including an additive inverse that is missing when using an OR-based algebra. This use of XOR guarantees a lossless factorization for a high enough value of k. The evaluation shows that XOR-based algorithms outperform OR-based algorithms in both performance and runtime. Overall, this allows better factorizations at a lower k compared to state-of-the-art approaches.

While the performance of the new approach already proved its usefulness for many applications in machine learning and data mining, there are a number of research questions that arise from this work. First of all, in OR-based BMF, the factors can be interpreted, e.g. for using it in graph clustering. This is due to the fact that the factors represent a compressed version of the original matrix. However, the representation changes when using an XOR-based BMF. Each row/column in the factors represents another residual matrix, i.e. a function to fix errors in another row/column. Therefore it does not translate to the traditional interpretation of a BMF. Future research will address this and aim to find an interpretation of the factors.

Another open problem is the forward-only approach of the algorithm. In each iteration, another 1-factor is added, without going back to previous factors and combining them or reducing factors that might cancel each other out. How this can be done is not trivial, simple approaches that relied on straight-forward techniques did not show improvements. However, this has the potential to further reduce the size of k and achieve smaller factorizations.

Finally, initial experiments showed that the choice of the initialization has a strong impact on the performance of the overall decomposition. Using a row or column of the matrix showed to achieve good results, but this might still be improved by using different candidates, especially in the later factors where the residual matrices become more and more sparse.

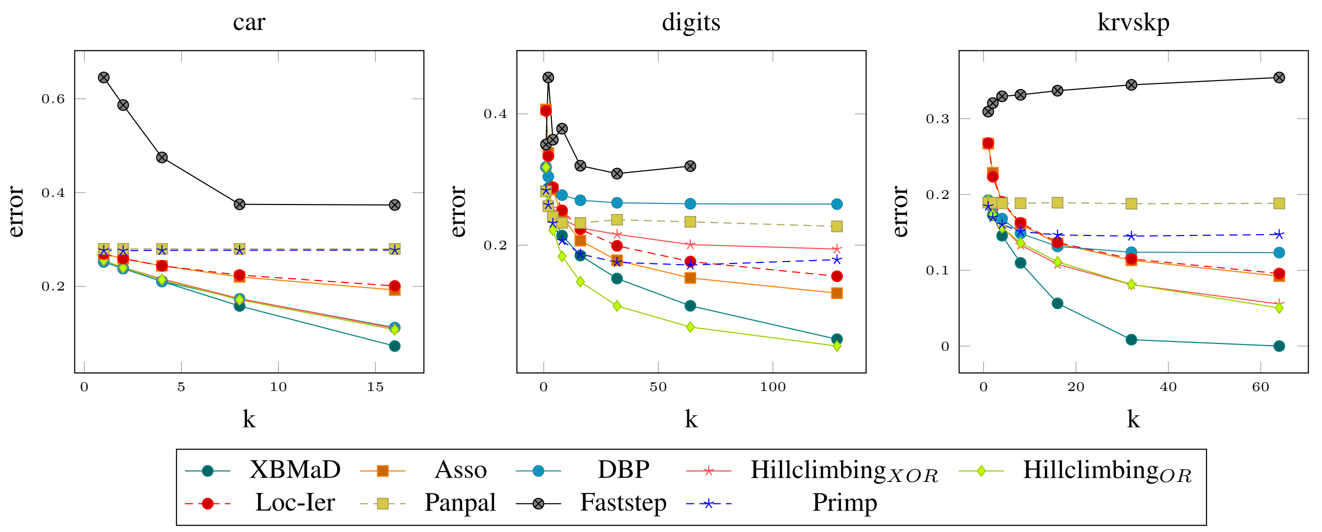


Fig. 6: Reconstruction error on selected datasets (full results in the supplement). The reconstruction error behaves similar to F1 Measure, achieving an almost identical ranking of the algorithms.

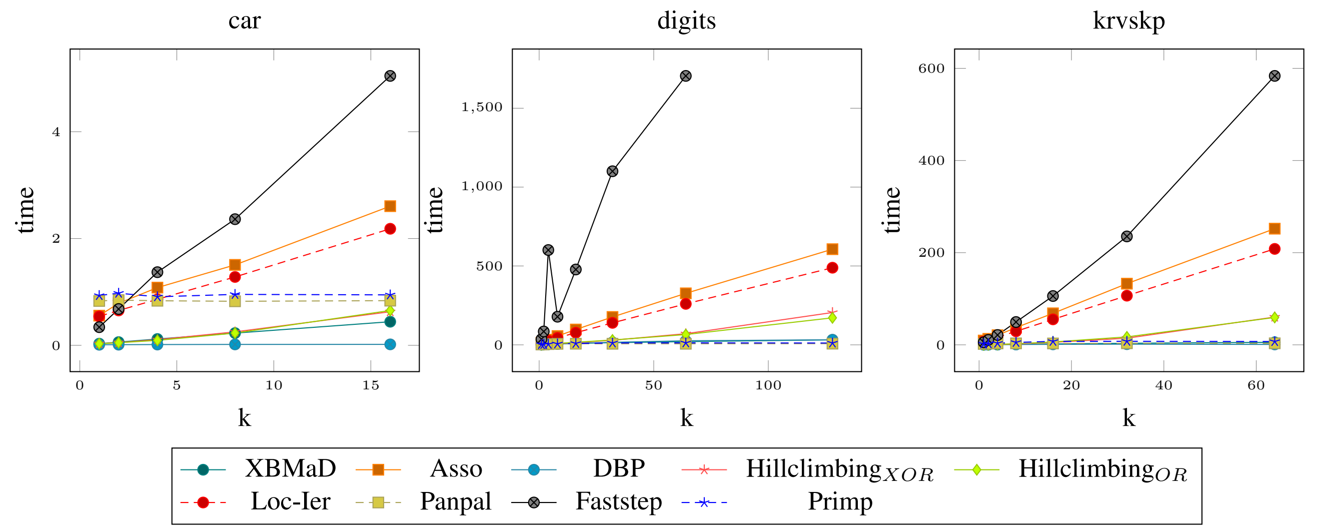


Fig. 7: Runtime of all algorithms on selected datasets (full results in the supplement). The advantage of incrementally increasing the size of the factor matrices and dividing the problem into small sub-problems has a strong impact on the low runtime of all XBMaD variants.

References

[1] Neda Abdelhamid, Aladdin Ayesh, and Fadi Thabtah. Phishing detection based associative classification data mining. *Expert Systems with Applications*, 41(13):5948–5959, 2014.

[2] Miguel Araujo, Pedro Ribeiro, and Christos Faloutsos. Faststep: Scalable boolean matrix decomposition. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 461–473, 2016.

[3] E. Ray Bareiss, Bruce W. Porter, and Craig C. Wier. PROTOS: An exemplar-based learning apprentice. In Yves Kodratoff and Ryszard S. Michalski, editors, *Machine Learning*, pages 112 – 127. Morgan Kaufmann, San Francisco (CA), 1990.

[4] Radim Belohlavek, Jan Outrata, and Martin Trnecka. Toward quality assessment of boolean matrix factorizations. *Information Sciences*, 459:71 – 85, 2018.

[5] Radim Belohlavek and Martin Trnecka. From-below approximations in boolean matrix factorization: Geometry and new algorithm. *Journal of Computer and System Sciences*, 81(8):1678 – 1697, 2015.

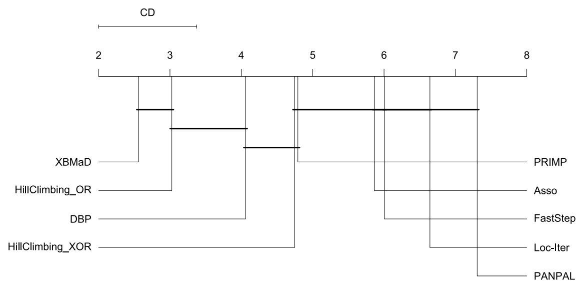
[6] Radim Belohlavek and Vilem Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *Journal of Computer and System Sciences*, 76(1):3 – 20, 2010.

[7] Marko Bohanec and Vladislav Rajkovič. Knowledge acquisition and explanation for multi-attribute decision making. In *8th International Workshop on Expert Systems and their Applications*, pages 59–78, 1988.

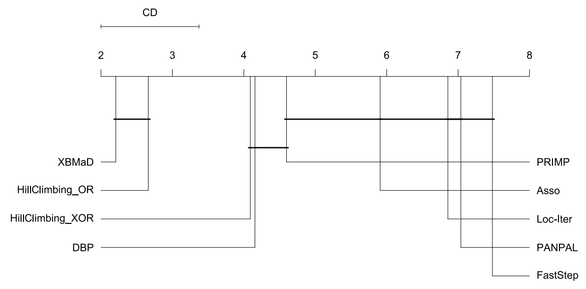
[8] Paulo Cortez and Alice Maria Gonçalves Silva. Using data mining to predict secondary school student performance. In *Proceedings of 5th Annual Future Business Technology Conference*, pages 5–12. EUROSIS-ETI, 2008.

[9] Ahmed Elgohary, Matthias Boehm, Peter J. Haas, Frederick R. Reiss, and Berthold Reinwald. Compressed linear algebra for declarative large-scale machine learning. *Communications of the ACM*, 62(5):83–91, 2019.

[10] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Approximation schemes for low-rank binary matrix approximation problems. *arXiv preprint*, arXiv:1807.07156, 2018.



(a) F1-measure



(b) Reconstruction error

Fig. 8: Nemenyi-Friedman test on the results. Both F1-measure and reconstruction error show a similar well performance of XBMaD and OR-based Hillclimbing, with both of them performing better than other classifiers.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[12] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In Einoshin Suzuki and Setsuo Arikawa, editors, *Discovery Science*, pages 278–289, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[13] Sibylle Hess and Katharina Morik. C-salt: Mining class-specific alterations in boolean matrix factorization. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 547–563, Cham, 2017. Springer International Publishing.

[14] Sibylle Hess, Katharina Morik, and Nico Piatkowski. The primping routine-tiling through proximal alternating linearized minimization. *Data Mining & Knowledge Discovery*, 31(4):1090 – 1131, 2017.

[15] Zachary G. Ives. Technical perspective: Compressing matrices for large-scale machine learning. *Communications of the ACM*, 62(5):82–82, 2019.

[16] Yuan Jiang and Zhi-Hua Zhou. Editing training data for knn classifiers with neural network ensemble. In *ISNN (1)*, page 356, 2004.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 8:30–37, 2009.

[18] Ravi Kumar, Rina Panigrahy, Ali Rahimi, and David Woodruff. Faster algorithms for binary matrix factorization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3551–3559, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[19] Christopher J Matheus and Larry A Rendell. Constructive induction on decision trees. In *International Joint Conference on Artificial Intelligence*, pages 645–650, 1989.

[20] Ryszard Michalski. Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of development an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2):125–161, 1980.

[21] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering*, 20(10):1348–1362, 2008.

[22] Pauli Miettinen and Jilles Vreeken. Mdl4bmf: Minimum description length for boolean matrix factorization. *ACM Trans. Knowl. Discov. Data*, 8(4):18:1–18:31, 2014.

[23] Manuel Olave, Vladislav Rajkovič, and Marko Bohanec. An application for admission in public school systems. In *Expert Systems in Public Administration,*, pages 145–160, 1989.

[24] Amichai Painsky, Saharon Rosset, and Meir Feder. Generalized independent component analysis over finite alphabets. *IEEE Transactions on Information Theory*, 62(2):1038–1053, Feb 2016.

[25] Amichai Painsky, Saharon Rosset, and Meir Feder. Linear independent component analysis over finite fields: Algorithms and bounds. *IEEE Transactions on Signal Processing*, 66(22):5875–5886, Nov 2018.

[26] Debbie Richards. Two decades of ripple down rules research. *The Knowledge Engineering Review*, 24(2):159–184, 2009.

[27] Jeffrey Schlimmer. *Concept acquisition through representational adjustment*. PhD thesis, Department of Information and Computer Science, University of California, 1987.

[28] Yuan Sun, Shiwei Ye, Yi Sun, and Tsunehiko Kameda. Exact and approximate boolean matrix decomposition with column-use condition. *International Journal of Data Science and Analytics*, 1(3):199–214, 2016.

[29] Ken Thompson. Retrograde analysis of certain endgames. *Journal of the International Computer Chess Association*, 9:131–139, 1986.

[30] Andrey Tyukin, Stefan Kramer, and Jörg Wicker. BMaD – a Boolean matrix decomposition framework. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8726 of *Lecture Notes in Computer Science*, pages 481–484. Springer Berlin Heidelberg, 2014.

[31] M P. W van Breukelen, D M. J Tax, and J E den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, vol. 34:381–386, 1998.

[32] Jörg Wicker, Bernhard Pfahringer, and Stefan Kramer. Multi-label classification using Boolean matrix decomposition. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 179–186. ACM, 2012.

[33] W Wolberg and O Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology,. In *Proceedings of the National Academy of Sciences*, pages 9193–9196, Dec 1990.

[34] Arie Yeredor. Ica in boolean xor mixtures. In Mike E. Davies, Christopher J. James, Samer A. Abdallah, and Mark D. Plumbley, editors, *Independent Component Analysis and Signal Separation*, pages 827–835, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[35] Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang. Binary matrix factorization with applications. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 391–400. IEEE, 2007.