2023 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2023, ROME, ITALY

ALGORITHMS FOR BOOLEAN MATRIX FACTORIZATION USING INTEGER PROGRAMMING

Christos Kolomvakis, Arnaud Vandaele, Nicolas Gillis

Department of Mathematics and Operational Research
University of Mons, Rue de Houdain 9, 7000 Mons

ABSTRACT

Boolean matrix factorization (BMF) approximates a given binary input matrix as the product of two smaller binary factors. As opposed to binary matrix factorization which uses standard arithmetic, BMF uses the Boolean OR and Boolean AND operations to perform matrix products, which leads to lower reconstruction errors. BMF is an NP-hard problem. In this paper, we first propose an alternating optimization (AO) strategy that solves the subproblem in one factor matrix in BMF using an integer program (IP). We also provide two ways to initialize the factors within AO. Then, we show how several solutions of BMF can be combined optimally using another IP. This allows us to come up with a new algorithm: it generates several solutions using AO and then combines them in an optimal way. Experiments show that our algorithms outperform the state of the art on medium-scale problems.

***Index Terms*—** alternating optimization, Boolean matrix factorization, integer programming

1. INTRODUCTION

Low-rank matrix approximations (LRMAs) are popular methods in machine learning, and have successfully been applied in a wide variety of applications such as document classification, community detection, hyperspectral unmixing and recommender systems, to cite a few; see, e.g., [1, 2, 3]. LRMAs perform dimensionality reduction by approximating an input data matrix as the product two factors of smaller sizes. Depending on the problem at hand, different matrix models can be considered. Examples include principal component analysis (PCA) and its variants such as sparse [4] and robust [5] PCA, and nonnegative matrix factorization (NMF) [6]. If the input matrix has elements in $\{0, 1\}$, then it makes sense to impose the factors to have elements in $\{0, 1\}$ as well, leading to binary matrix factorization (bMF) and Boolean matrix factorizations (BMF) [7, 8]. However, due to bMF using the standard addition and multiplication, the approximation will typically produce elements that are not in $\{0, 1\}$. BMF uses the Boolean OR and AND operations to guarantee the approximation to be binary, which improves the interpretability of the model. BMF is a difficult combinatorial problem, and many works have been dedicated to the computation of BMFs [9, 10, 11, 12, 13]; see also [14] for a recent survey.

This paper proposes new ways to compute BMFs, and is organized as follows. In Section 2, we formally define BMF. In Section 3, we describe our proposed alternating optimization (AO) algorithm for BMF where the subproblems are quadratic integer programs (IPs). We also provide two initialization strategies for the AO algorithm. In Section 4, we provide an IP formulation to optimally combine several BMF solutions. In Section 5, we show that our proposed algorithms outperform the state of the art on four medium-scale real-world data sets.

2. BOOLEAN MATRIX FACTORIZATION (BMF)

Let us first define the matrix Boolean product.

Definition 1 (Boolean product)**.** *Given two Boolean matrices, $\mathbf{W} \in \{0,1\}^{m\times r}$ and $\mathbf{H} \in \{0,1\}^{r\times n}$, their Boolean product is denoted $\mathbf{W} \circ \mathbf{H} \in \{0,1\}^{m\times n}$ and is defined for all i, j as*

$$(\mathbf{W} \circ \mathbf{H})_{ij} = \bigvee_{k=1}^{r} \mathbf{W}_{ik}\mathbf{H}_{kj}, \tag{1}$$

where $\vee$ is the logical OR operation (that is, $0\vee 0 = 0$, $1\vee 0 = 1$, and $1\vee 1 = 1$). Interestingly, $\mathbf{W}\circ\mathbf{H} = \min(1, \mathbf{WH})$ where $\mathbf{WH}$ is the usual matrix product between $\mathbf{W}$ and $\mathbf{H}$.

We can now define the BMF problem.

Definition 2 (BMF)**.** *Given a Boolean matrix $\mathbf{X} \in \{0,1\}^{m\times n}$ and a factorization rank r, BMF aims to find matrices $\mathbf{W} \in \{0,1\}^{m\times r}$ and $\mathbf{H} \in \{0,1\}^{r\times n}$ that solve*

$$\min_{\mathbf{W}\in\{0,1\}^{m\times r},\mathbf{H}\in\{0,1\}^{r\times n}} \|\mathbf{X} - \mathbf{W} \circ \mathbf{H}\|_F^2. \tag{2}$$

In [7], it is proven that not only solving (2), but also approximating (2), is NP-hard.

BMF allows one to find subset of columns and rows of $\mathbf{X}$ that are highly correlated, since the entries equal to one in

Emails: {firstname.lastname}@umons.ac.be

979-8-3503-2411-2/23/$31.00 ©2023 IEEE

each binary rank-one factor $\mathbf{W}(:,k)\mathbf{H}(k,:)$ correspond to a rectangular submatrix of $\mathbf{X}$ that should contain many entries equal to one. Applications of BMF include role mining [15] and bioinformatics [16, 17]; see also the recent survey [14].

3. ALTERNATING OPTIMIZATION (AO) FOR BMF

Most algorithms for LRMAs rely on iterative block coordinate descent methods: the subproblem in $\mathbf{H}$ is solved for $\mathbf{W}$ fixed, and vice versa. The reason is that these subproblems are typically convex. For BMF, this is of course not the case. However, the advances in IP solvers, such as Gurobi [18], allows one to tackle medium-scale problems efficiently.

3.1. IP formulation for BMF subproblems

Assuming $\mathbf{W}$ is fixed in (2), we would like to solve the following Boolean least squares (BoolLS) problem in $\mathbf{H}$, that is, solve

$$\min_{\mathbf{H}\in\{0,1\}^{r\times n}} \|\mathbf{X} - \min(1, \mathbf{WH})\|_F^2.$$

Because of the nonlinearity in the objective, this cannot be solved directly with standard IP solvers. Note that the problem in each column of $\mathbf{H}$ is independent:

$$\min_{\mathbf{H}(:,j)\in\{0,1\}^{r\times n}} \|\mathbf{X}(:,j) - \min(1, \mathbf{WH}(:,j))\|_F^2. \quad (3)$$

For simplicity, let $\mathbf{h} = \mathbf{H}(:,j)$ and $\mathbf{x} = \mathbf{X}(:,j)$. Given $\mathbf{W}$ and $\mathbf{x}$, we need to solve $\min_{\mathbf{h}\in\{0,1\}^r} \|\mathbf{x} - \min(1, \mathbf{Wh})\|_F^2$. Introducing the variable $\mathbf{z} = \min(1, \mathbf{Wh})$, (3) can be reformulated as follows:

$$\min_{\mathbf{h}\in\{0,1\}^r, \mathbf{z}\in\{0,1\}^m} \|\mathbf{x} - \mathbf{z}\|_F^2 \quad \text{s.t.} \quad \frac{\mathbf{Wh}}{r} \leq \mathbf{z} \leq \mathbf{Wh}. \quad (4)$$

In fact, for $\mathbf{W}$, $\mathbf{h}$ and $\mathbf{z}$ binary, $\frac{\mathbf{Wh}}{r} \leq \mathbf{z} \leq \mathbf{Wh}$ if and only if $\mathbf{z} = \min(1, \mathbf{Wh})$, since $\mathbf{Wh} \in \{0, 1, \ldots, r\}^m$. Now (4) is a convex quadratic optimization problem with linear constraints over binary variables. Such problems can be solved with commercial software, and we make use of Gurobi [18].

To give an idea of the computational time required for Gurobi to solve (4), let us perform the following experiment for various values of m and r. The setting is as follows: we generate the entries of $\mathbf{W}$ and $\mathbf{h}$ using the uniform distribution in $[0, 1]$, and then threshold all elements to convert them to binary. For all ranks tested, apart from $r = 50$, if an element is larger than 0.7, it is converted to 1, otherwise we convert it to 0. For $r = 50$, the threshold is set to 0.8. We chose relatively sparse $\mathbf{W}$ and $\mathbf{h}$ to make sure $\min(1, \mathbf{Wh})$ is not the all-one vector (in fact, we regenerate $\mathbf{W}$ and $\mathbf{h}$ if $\min(1, \mathbf{Wh})$ is the all-one or all-zero vector). Then we set $\mathbf{x} = \min(1, \mathbf{Wh})$, and 10% of the entries of $\mathbf{x}$ are flipped randomly[1]. Table 1 reports the results.

[1]The noiseless BoolLS problem is much easier to solve since $\mathbf{h}_k = 1$ if and only if the support of the kth column of $\mathbf{W}$ is contained in that of $\mathbf{x}$, that is, $\mathbf{W}(:,k) \leq \mathbf{x}$.

$r\backslash m$	100	1000	5000	10000
2	0.002	0.02	0.47	1.8
5	0.003	0.03	0.47	1.87
10	0.005	0.04	0.56	2.2
20	0.012	0.12	2.4	12.0
50	0.049	2.52	38.0	235

Table 1. Average execution time in seconds over 30 trials of Gurobi to solve noisy BoolLS problems (4) for various values of m and r.

The results are encouraging since solving (4) can be done exactly with Gurobi in a reasonable amount of time, even for relatively large problems, e.g., it takes on average Gurobi 12 seconds to solve this problem with $m = 10^4$ and $r = 20$. Note that one could also use a timelimit for Gurobi, so that Gurobi would return the best solution found within the allotted time (often IP solvers take much more time to guarantee global optimality rather than finding the optimal solution).

3.2. AO for BMF

We can now solve BMF via AO over the factors $\mathbf{W}$ and $\mathbf{H}$ alternatively. Since $\|\mathbf{X} - \min(1, \mathbf{WH})\|_F^2 = \|\mathbf{X}^\top - \min(1, \mathbf{H}^\top\mathbf{W}^\top)\|_F^2$, the problem in $\mathbf{W}$ for $\mathbf{H}$ fixed has the same form. We update $\mathbf{H}$ in a column-by-column fashion by solving the independent BoolLS of the form (3), and similarly for $\mathbf{W}$ row by row. Algorithm 1 summarizes the AO strategy. We have added a safety procedure within AO (steps 4-8): it may happen that some rows of H are set to zero (for example, if $\mathbf{W}$ is not well initialized). In that case, we reinitialize these rows as the rows of the residual $\mathbf{R} = \max\left(0, \mathbf{X} - \max(1, \mathbf{W}_i\mathbf{H}_i)\right)$ whose entries have the largest sum. This guarantees that the error will decrease afer the update of $\mathbf{W}$.

Typically, AO needs a very small number of iterations to converge, given the combinatorial nature of the problem. As we will report in Section 5, among more than 6000 runs on 4 datasets with 3 different ranks, on average 3.7 iterations are needed, with a maximum of 14.

3.3. Initialization of AO

In this section, we provide two initialization strategies for AO-BMF, that is, Algorithm 1.

Randomly selecting columns or rows of $\mathbf{X}$ AO-BMF only requires $\mathbf{W}$ to be initialized. By symmetry, it could also be initialized only with $\mathbf{H}$, starting the AO algorithm by optimizing over $\mathbf{W}$. A simple, fast and meaningful strategy to initialize AO-BMF is to initialize $\mathbf{W}$ (resp. $\mathbf{H}$) with a subset of the columns (resp. rows) of $\mathbf{X}$, that is, set $\mathbf{W} = \mathbf{X}(:, \mathcal{K})$ (resp. $\mathbf{H} = \mathbf{X}(\mathcal{K}, :)$) where $\mathcal{K}$ is a randomly selected set of r indices of the columns (resp. rows) of $\mathbf{X}$.

Algorithm 1 AO algorithm for BMF - `AO-BMF`

Input: Input matrix $\mathbf{X} \in \{0,1\}^{m\times n}$, initial factor matrix $\mathbf{W}_0 \in \{0,1\}^{m\times r}$, maximum number of iterations maxiter.

Output: $\mathbf{W} \in \{0,1\}^{m\times r}$ and $\mathbf{H} \in \{0,1\}^{r\times n}$ such that $X \approx \min(1, WH)$.

$i = 1$, $e(0) = \|\mathbf{X}\|_F^2$, $e(1) = \|\mathbf{X}\|_F^2 - 1$.
while $e(i) < e(i-1)$ and $i \leq$ maxiter **do**
  $\mathbf{H}_i = \texttt{BoolLS}(\mathbf{X}, \mathbf{W}_{i-1})$.
  $\mathcal{K} = \{k \mid \mathbf{H}_i(k,:) = 0\}$.
  if $\mathcal{K} \neq \emptyset$ **then**
    $\mathbf{R} = \max\big(0, \mathbf{X} - \max(1, \mathbf{W}_i\mathbf{H}_i)\big)$.
    $\mathbf{H}_i(\mathcal{K},:) = \mathbf{R}(\mathcal{I},:)$, where $\mathcal{I}$ contains the indices of the $|\mathcal{K}|$ rows of $\mathbf{R}$ with largest sum.
  end if
  $\mathbf{W}_i = \texttt{BoolLS}(\mathbf{X}^\top, \mathbf{H}_i^\top)^\top$.
  $i = i + 1$.
  $e(i) = \|\mathbf{X} - \min(1, \mathbf{W}_i\mathbf{H}_i)\|_F^2$.
end while
return $(\mathbf{W}, \mathbf{H}) = (\mathbf{W}_i, \mathbf{H}_i)$.

NMF-based initialization The second initialization we propose relies on NMF. NMF approximates $\mathbf{X}$ with $\mathbf{WH}$ where $\mathbf{W}$ and $\mathbf{H}$ are nonnegative. We use an NMF algorithm from `https://gitlab.com/ngillis/nmfbook/` which itself initializes the entries of $\mathbf{W}$ and $\mathbf{H}$ using the uniform distribution in $[0, 1]$. Once an NMF solution is computed, we binarize it using the following two steps:

- Normalize the columns of $\mathbf{W}$ and rows of $\mathbf{H}$ such that $\max(\mathbf{W}(:,k)) = \max(\mathbf{H}(k,:))$ for all k, using the scaling degree of freedom in NMF, that is, $\mathbf{W}(:,k)\mathbf{H}(k,:) = (\alpha\mathbf{W}(:,k))(\alpha^{-1}\mathbf{H}(k,:))$ for $\alpha > 0$.
- Set the entries of $\mathbf{W}$ and $\mathbf{H}$ to 0 or 1 using a given threshold δ which is generated uniformly at random in the interval $[0.3, 0.7]$. An alternative would be to use a grid search approach to determine δ, similar to [8, 19]. However, we observed that selecting δ randomly performs better on average.

4. COMBINING MULTIPLE BMF SOLUTIONS

Algorithm 1, AO-BMF, is able to relatively quickly generate locally optimal solutions for (2), in the sense that they cannot be improved by optimizing $\mathbf{W}$ or $\mathbf{H}$ alone. A first natural approach to generate good solutions to BMF is using multiple initializations, and keeping the best solution. We will refer to this strategy as multistart AO (MS-AO). Generating more than one solution using several initializations is typically key as it allows to better explore the search space.

However, it is possible to combine a set of solutions in a more effective way. Assume we have generated p rank-r BMFs: $\mathbf{W}_1\mathbf{H}_1, \ldots, \mathbf{W}_p\mathbf{H}_p$. This gives rp binary rank-one factors, namely $\mathbf{W}_\ell(:,k)\mathbf{H}_\ell(k,:)$ for $k = 1, \ldots, r$ and $\ell = 1, \ldots, p$. Let us denote these rank-one binary matrices $\mathbf{R}_i$ for $i = 1, \ldots, N$ with[2] $N = rp$. To generate a better rank-r BMF, we can pick r rank-one binary factors among the $\mathbf{R}_i$'s, by solving the following combinatorial problem:

$$\min_{\mathbf{y}\in\{0,1\}^N} \left\|\mathbf{X} - \min\big(1, \sum_i \mathbf{y}_i\mathbf{R}_i\big)\right\|_F^2 \text{ such that } \sum_i y_i = r.$$

The variable $y \in \{0,1\}^N$ encodes the r selected rank-one factors, that is, $y_i = 1$ if $\mathbf{R}_i$ is selected in the BMF. As for BoolLS, we can reformulate this problem as a quadratic IP:

$$\min_{\mathbf{y}\in\{0,1\}^N, \mathbf{Z}\in\{0,1\}^{m\times n}} \|\mathbf{X} - \mathbf{Z}\|_F^2 \tag{5}$$
$$\text{such that } \sum_{i=1}^N y_i = r, \frac{\sum_{i=1}^N y_i\mathbf{R}_i}{r} \leq \mathbf{Z} \leq \sum_{i=1}^N y_i\mathbf{R}_i.$$

Denoting y_i^* the optimal solution of (5), the rank-r BMF obtained, $\sum_i y_i^*\mathbf{R}_i$, is guaranteed to be at least as good as all the solutions $\{\mathbf{W}_i\mathbf{H}_i\}_{i=1}^p$, since they are feasible solutions of (5).

Gurobi can solve medium-scale problem of the form (5) in a reasonable amount of time. Table 2 reports the time to solve (5) for $m = n = 101$, $r = 5$, and $N = 2^k\ 50$ for $k = 0, 1, \ldots, 4$. For example, to combine 160 rank-5 solutions (hence 800 rank-one factors) of a 101-by-101 matrix, it takes about 30 seconds.

N	50	100	200	400	800
time (s.)	4.0	4.3	7.6	14.2	30.6

Table 2. Run times to combine N rank-one factors for a 101-by-101 matrix with $r = 5$ (namely, the apb data set, see Section 5).

In practice, if N is too large, we do not need to take into account all rank-one factors, and can only consider rank-one factors corresponding to the best BMFs.

Finally, once a solution combining several BMFs is computed, we will further improve it using AO.

We will refer to this algorithm, namely generating p solutions with AO-BMF, then combining them solving (5), and then applying AO to that solution as MS-Comb-AO.

5. NUMERICAL EXPERIMENTS

All experiments are performed with a 12th Gen Intel(R) Core(TM) i9-12900H 2.50 GHz, 32GB RAM, on MATLAB R2019b. The code and data sets are available on `https://gitlab.com/ngillis/BooleanMF`

[2] In practice, we delete duplicated rank-one factors so that $N \ll rp$.

Data sets We will perform experiments on four real data sets used in [20], and which come from [21, 22]; see Table 3.

	zoo	heart	lymp	apb
$m \times n$	101×17	242×22	148×44	105×105

Table 3. Four binary real-world data sets.

As in [20], we use $r = 2, 5, 10$ for all data sets. In [20], authors proposed two non-trivial IP-based approaches for BMF that perform well against the state of the art (they used a 20 minutes time limit for their method), namely against a greedy scheme [20], ASSO and ASSO++ [7], a penalty formulation from [8], and an NMF-based heuristic. Table 4 reports the best result of all these methods for the four data sets. From now on, we will report the result of a BMF by comparing it with the best solution in Table 4.

	$r = 2$	$r = 5$	$r = 10$
zoo	271	126	39
heart	1185	737	419
lymp	1184	982	728
apb	776	684	573

Table 4. Objective function $\|\mathbf{X} - \min(1, \mathbf{WH})\|_F^2$ of the best solution found by various algorithms in [20, Table 4, page 20].

Results from two other papers We have also run two recent algorithms, from [11] and [13], on these data sets. The method in [11] is based on a continuous reformulation of BMF and uses alternating proximal projected gradient method to solve it. We have used the parameters of the algorithm recommended by the authors. Table 5 reports the results. Out of curiosity, we initialized AO with the best solutions found. We observe in Table 5 that AO is able to significantly improve these solutions, showing that the continuous reformulations of [11] is not able to generate locally optimal solutions.

	$r = 2$	$r = 5$	$r = 10$
zoo	+11 | 0	+5 | -1	+18 | +5
heart	+65 | +18	+29 | -1	+33 | +26
lymp	+64 | +25	+88 | -10	+203 | +60
apb	+72 | +30	+59 | +43	+33 | +15

Table 5. Best result obtained with the method in [11] (left), and result of this best solution improved by AO (right). The numbers indicate the difference compared with the best values in Table 4. A negative value means an improvement, a positive value means a worse solution.

Next, we show the results for the method in [13]. Note that this paper provides a rather general approach, allowing for other Boolean operations between the factors to approximate the entries of $\mathbf{X}$. Their approach is also based on some continuous reformulations, and the use of gradient and descent methods. Table 6 reports their result, using 15 trials where each trial uses 10 random initializations, each optimized with 2000 iterations. We also use AO to improve the best solutions found by this method, and observe a similar behavior as for the method from [11].

	$r = 2$	$r = 5$	$r = 10$
zoo	+2 | 0	+7 | +4	+10 | +4
heart	+165 | +9	+33 | +4	+86 | +57
lymp	+214 | -3	+49 | -16	+123 | +5
apb	+48 | 0	+51 | +23	+112 | +43

Table 6. Best result obtained with the method in [13] (left), and result of this best solution improved by AO (right). The numbers indicate the difference compared with the best values in Table 4.

In summary, on these data sets, we observe that the algorithms from [11, 13] provide solutions which are much worse than the best solutions provided in [20], and that AO can considerably improve the solutions of these two methods, sometimes obtaining better solution than the best from [20] (e.g., for the lymp data set with $r = 5$).

Results of our proposed algorithms Let us provide the results for our two algorithms.

[1.] Multiple starts of AO-BMF (MS-AO). We generate as many BMFs as possible with AO-BMF within the time T, and return the best solution. The initialization of AO-BMF is chosen alternatively as one of the two strategies (NMF-based or random columns/rows of $\mathbf{X}$). Table 7 reports the results for T equal to 30 seconds and 5 minutes.

	$r = 2$	$r = 5$	$r = 10$
zoo	0 | 0	-1 | -1	+3 | 0
heart	+2 | +2	-1 | -1	0 | 0
lymp	-10 | -10	-25 | -32	-15 | -34
apb	0 | 0	+6 | -6	+4 | +2

Table 7. Results of the MS-AO strategy for 30 seconds (left) and 5 minutes (right). The numbers indicate the difference compared with the best values in Table 4.

Quite surprisingly, MS-AO is already able to perform on par or improve upon the state of the art. With a 30 seconds time limit, it does on 8 out of 12 cases: 5/12 cases with improvements, sometimes significant as for the lymp data set,

and 3/12 cases with the same objective. However, for 4 data sets, it is not able to achieve the best solution reported in Table 4, although the generated solutions are only slightly worse (+6 at most). With a 5 minutes time limit, it performs better or on par on 10 out of the 12 cases, and it produces a slightly worse solution in two cases (+2).

To give an idea of the performance of Gurobi on these medium-scale problems, Table 8 reports the number of BMFs generated within 5 minutes for each data set, as well as the average number of iterations required for AO to converge. The average number of iterations for AO to converge is 3.7, and the largest number of iterations AO needed among these 6165 BMFs is 14.

	$r=2$	$r=5$	$r=10$
zoo	844 | 3.3	703 | 3.2	596 | 3.3
heart	503 | 3.0	280 | 3.5	484 | 3.5
lymp	558 | 3.3	568 | 4.2	269 | 4.8
apb	591 | 3.7	381 | 4.2	388 | 4.8

Table 8. Number of BMFs generated via AO within 5 minutes (left), and average number of iterations required for AO to converge (right).

[2.] Multiple starts, combination and AO (MS-Comb-AO). As explained in Section 4, we generate as many BMFs as possible with AO-BMF (as for AO-MS) within time $3T/4$, and then combine them by solving (5) with a time limit of $T/4$. We used the same random seed so that the solutions generated are the same as for AO-MS, except that less solutions are generated, since only 3/4 of the total time is spent for that. Table 9 reports the results.

	$r=2$	$r=5$	$r=10$
zoo	0 | 0	-1 | -1	0 | 0
heart	+2 | 0	-1 | -1	0 | 0
lymp	-10 | -10	-25 | -32	-15 | -34
apb	0 | 0	-1 | -6	+2 | -7

Table 9. Results of the MS-Comb-AO algorithm for 30 seconds (left) and 5 minutes (right). The numbers indicate the difference compared with the best values in Table 4.

For some data set, AO is already able to generate very good solutions, and hence solving (5) is not useful, e.g., for the lymp data set. However, for most cases, this combination is beneficial, sometimes significantly. In particular, with the timelimit of 30 seconds on the apb dataset for $r = 5$, the best solution found by AO-MS has error 689 (+6) while the combination leads to an error of 682 (-1); for the zoo data set with $r = 10$, it goes from 42 (+3) to 39 (0). A similar behavior is observed for 5 minutes: for the apb data set with $r = 10$ from +2 to -7, and the heart data set from +2 to 0.

To conclude, MS-Comb-AO with a 5-minute timelimit is able to either perform on par with the state of the art (we suspect that for these data sets, the corresponding solutions are optimal), or outperform it, sometimes significantly (in particular for the lymp data set).

Experiment on facial images As a last experiment, let us apply AO-BMF to a larger data set, the well-known CBCL facial images. It was used in the seminal paper of Lee and Seung to show that NMF is able to extract facial features [6]. Let us apply AO-BMF on this very same data set. Each column of the data matrix $\mathbf{X} \in \mathbb{R}^{361\times 2429}$ contains a vectorized facial image of size 19×19, and is not binary but satisfies $\mathbf{X}(i,j) \in [0,1]$ for all (i,j). Quite interestingly, AO-BMF can be applied to any input matrix $\mathbf{X}$, even if it is not binary. We use the NMF-based initialization for AO-BMF with $r = 20$, and it converges in 15 iterations within about 1 hour. AO-BMF provides a binary approximation of $\mathbf{X}$ with relative error $\frac{\|\mathbf{X}-\min(1,\mathbf{WH})\|_F}{\|\mathbf{X}\|_F} = 55.94\%$. Figure 1 displays the meaningful binary facial features extracted by AO-BMF as the columns of W.

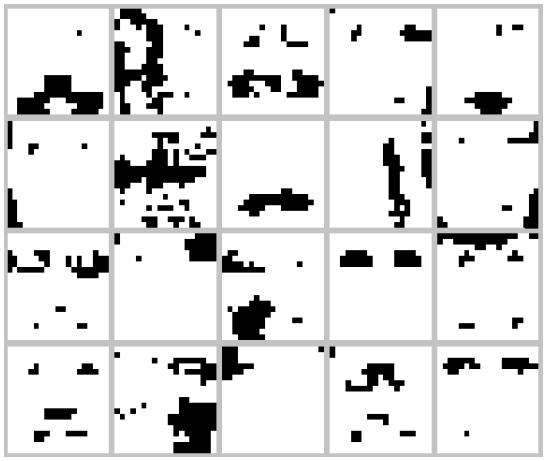

Fig. 1. Facial features extracted by AO-BMF on the CBCL data set.

6. CONCLUSION AND FURTHER WORK

In this paper, we have designed an alternating optimization (AO) strategy to tackle Boolean matrix factorization (BMF) using interger programming (IP). We have also shown how to combine several solutions in an optimal way, using IP as well. We finally showed how these two strategies are able to outperform the state of the art on 4 real-world medium-scale data sets.

Further work includes the adaptation of our algorithms for large-scale data sets. A simple, yet possibly effective ap-

proach, is to use a time limit for Gurobi to tackle the IP subproblems. Typically, Gurobi is able to produce quickly high quality solutions. The design of fast heuristic algorithms to solve the Boolean least squares problem (4) would also be useful, since we have validated the effectiveness of AO for BMF. Another line of work, that we are currently exploring, is to design more sophisticated combination strategies. This has allowed us to generate even better solutions for three cases:

- for lymp with $r = 5$, a BMF with error 939 (-43),
- for lymp with $r = 10$, a BMF with error 680 (-48).
- for apb with $r = 5$, a BMF with error 677 (-7).

We will present these results in an extended version of this paper.

Acknowledgments We thank Sebastian Miron for sending us his BMF code [13], and Sebastian Dalleiger for providing us with good parameters for his BMF algorithm [11]. The authors acknowledge the support by the F.R.S.-FNRS and the FWO (EOS, O005318F-RG47), by the European Research Council (ERC, consolidator grant no 101085607), and by the Francqui Foundation.

7. REFERENCES

[1] I. Markovsky, *Low rank approximation: algorithms, implementation, applications*, Springer, 2012.

[2] M. Udell, C. Horn, R. Zadeh, S. Boyd, et al., "Generalized low rank models," *Foundations and Trends® in Machine Learning*, vol. 9, no. 1, pp. 1–118, 2016.

[3] N. Gillis, *Nonnegative Matrix Factorization*, SIAM, Philadelphia, PA, 2020.

[4] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[5] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 1–37, 2011.

[6] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[7] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila, "The discrete basis problem," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 10, pp. 1348–1362, 2008.

[8] Z. Zhang, T. Li, C. Ding, and X. Zhang, "Binary matrix factorization with applications," in *IEEE Int. Conf. on Data Mining*, 2007, pp. 391–400.

[9] T. Rukat, C. C. Holmes, M. K. Titsias, and C. Yau, "Bayesian Boolean matrix factorisation," in *International Conference on Machine Learning*, 2017.

[10] S. Miron, M. Diop, A. Larue, E. Robin, and D. Brie, "Boolean decomposition of binary matrices using a post-nonlinear mixture approach," *Signal Processing*, vol. 178, pp. 107809, 2021.

[11] S. Dalleiger and J. Vreeken, "Efficiently factorizing boolean matrices using proximal gradient descent," *Advances in Neural Information Processing Systems*, 2022.

[12] M. Araujo, P. Ribeiro, and C. Faloutsos, "Faststep: Scalable boolean matrix decomposition," in *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2016.

[13] R. Cabral Farias and S. Miron, "A generalized approach for boolean matrix factorization," *Signal Processing*, vol. 206, pp. 108887, 2023.

[14] P. Miettinen and S. Neumann, "Recent developments in Boolean matrix factorization," in *International Joint Conference on Artificial Intelligence*, 2021.

[15] H. Lu, J. Vaidya, and V. Atluri, "Optimal boolean matrix decomposition: Application to role engineering," in *International Conference on Data Engineering*, 2008.

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

[17] A. Haddad, F. Shamsi, L. Zhu, and L. Najafizadeh, "Identifying dynamics of brain function via Boolean matrix factorization," in *Asilomar Conference on Signals, Systems, and Computers*, 2018.

[18] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023, `https://www.gurobi.com`.

[19] D. P. Truong, E. Skau, D. Desantis, and B. Alexandrov, "Boolean matrix factorization via nonnegative auxiliary optimization," *IEEE Access*, vol. 9, pp. 117169–117177, 2021.

[20] O. Günlük, R. A. Hauser, and R. A. Kovács, "Binary matrix factorisation and completion via integer programming," *arXiv preprint arXiv:2106.13434*, 2021.

[21] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[22] V. Krebs, "A network of books about recent us politics sold by the online bookseller amazon. com," *Unpublished http://www. orgnet. com*, 2008.