Binary Matrix Factorisation via Column Generation

Réka Á. Kovács,[1] **Oktay Günlük,** [2] **Raphael A. Hauser** [1]
[1] University of Oxford & The Alan Turing Institute
[2] Cornell University
reka.kovacs@maths.ox.ac.uk, ong5@cornell.edu, hauser@maths.ox.ac.uk

Abstract

Identifying discrete patterns in binary data is an important dimensionality reduction tool in machine learning and data mining. In this paper, we consider the problem of low-rank binary matrix factorisation (BMF) under Boolean arithmetic. Due to the hardness of this problem, most previous attempts rely on heuristic techniques. We formulate the problem as a mixed integer linear program and use a large scale optimisation technique of column generation to solve it without the need of heuristic pattern mining. Our approach focuses on accuracy and on the provision of optimality guarantees. Experimental results on real world datasets demonstrate that our proposed method is effective at producing highly accurate factorisations and improves on the previously available best known results for 15 out of 24 problem instances.

1 Introduction

Low-rank matrix approximation is an essential tool for dimensionality reduction in machine learning. For a given $n \times m$ data matrix X whose rows correspond to n observations or items, columns to m features and a fixed positive integer k, computing an optimal rank-k approximation consists of approximately factorising X into two matrices A, B of dimension $n \times k$ and $k \times m$ respectively, so that the discrepancy between X and its rank-k approximate $A \cdot B$ is minimum. The rank-k matrix $A \cdot B$ describes X using only k derived features: the rows of B specify how the original features relate to the k derived features, while the rows of A provide weights how each observation can be (approximately) expressed as a linear combination of the k derived features.

Many practical datasets contain observations on categorical features and while classical methods such as singular value decomposition (SVD) (Golub and Van Loan 1989) and non-negative matrix factorisation (NMF) (Lee and Seung 1999) can be used to obtain low-rank approximates for real valued datasets, for a binary input matrix X they cannot guarantee factor matrices A, B and their product to be binary. Binary matrix factorisation (BMF) is an approach to compute low-rank matrix approximations of binary matrices ensuring that the factor matrices are binary as well (Miettinen 2012). More precisely, for a given binary matrix $X \in \{0,1\}^{n \times m}$ and a fixed positive integer k, the rank-k BMF problem (k-BMF) asks to find two matrices $A \in \{0,1\}^{n \times k}$ and $B \in \{0,1\}^{k \times m}$ such that the product of A and B is a binary matrix denoted by Z, and the distance between X and Z is minimum in the squared Frobenius norm. Many variants of k-BMF exist, depending on what arithmetic is used when the product of matrices A and B is computed. We focus on a variant where the Boolean arithmetic is used:

$$X = A \circ B \iff x_{ij} = \bigvee_{\ell=1}^{k} a_{i\ell} \wedge b_{\ell j},$$

so that 1s and 0s are interpreted as True and False, addition corresponds to logical disjunction ($\vee$) and multiplication to conjunction ($\wedge$). Apart from the arithmetic of the Boolean semi-ring, other choices include standard arithmetic over the integers or modulo 2 arithmetic over the binary field. We focus on the Boolean case, in which the property of Boolean non-linearity, 1 + 1 = 1 holds because many natural processes follow this rule. For instance, when diagnosing patients with a certain condition, it is only the presence or absence of a characteristic symptom which is important, and the frequency of the symptom does not change the diagnosis. As an example, consider the matrix (inspired by (Miettinen et al. 2008))

$$X = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

where rows correspond to patients and columns to symptoms, $x_{ij} = 1$ indicating patient i presents symptom j. Let

$$X = A \circ B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

denote the rank-2 BMF of X. Factor B reveals that 2 underlying diseases cause the observed symptoms, α causing symptoms 1 and 2, and β causing 2 and 3. Factor A reveals that patient 1 has disease α, patient 3 has β and patient 2 has both. In contrast, the best rank-2 real approximation

$$X \approx \begin{bmatrix} 1.21 & 0.71 \\ 1.21 & 0.00 \\ 1.21 & -0.71 \end{bmatrix} \begin{bmatrix} 0.00 & 0.71 & 0.50 \\ 0.71 & 0.00 & -0.71 \end{bmatrix}$$

fails to reveal a clear interpretation, and the best rank-2 NMF

$$X \approx \begin{bmatrix} 1.36 & 0.09 \\ 1.05 & 1.02 \\ 0.13 & 1.34 \end{bmatrix} \begin{bmatrix} 0.80 & 0.58 & 0.01 \\ 0.00 & 0.57 & 0.81 \end{bmatrix}$$

of X suggests that symptom 2 presents with lower intensity in both α and β, an erroneous conclusion (caused by patient 2) that could not have been learned from data X which is of "on/off" type. BMF-derived features are particularly natural to interpret in biclustering gene expression datasets (Zhang et al. 2007), role based access control (Lu, Vaidya, and Atluri 2008, 2014) and market basket data clustering (Li 2005).

1.1 Complexity and Related Work

The Boolean rank of a binary matrix $X \in \{0,1\}^{n\times m}$ is defined to be the smallest integer r for which there exist matrices $A \in \{0,1\}^{n\times r}$ and $B \in \{0,1\}^{r\times m}$ such that $X = A \circ B$, where $\circ$ denotes Boolean matrix multiplication defined as $x_{ij} = \bigvee_{\ell=1}^{r} a_{i\ell} \wedge b_{\ell j}$ for all $i \in \{1,\dots,n\} := [n], j \in [m]$ (Kim 1982). This is equivalent to $x_{ij} = \min\{1, \sum_{\ell=1}^{r} a_{i\ell} b_{\ell j}\}$ using standard arithmetic. Equivalently, the Boolean rank of X is the minimum value of r for which it is possible to factor X into the Boolean combination of r rank-1 binary matrices $X = \bigvee_{\ell=1}^{r} \boldsymbol{a}_\ell \boldsymbol{b}_\ell^\top$ for $\boldsymbol{a}_\ell \in \{0,1\}^n, \boldsymbol{b}_\ell \in \{0,1\}^m$. Interpreting X as the node-node incidence matrix of a bipartite graph G with n vertices on the left and m vertices on the right, the problem of computing the Boolean rank of X is in one-to-one correspondence with finding a minimum edge covering of G by complete bipartite subgraphs (bicliques)(Monson, Pullman, and Rees 1995). Since the biclique cover problem is NP-hard (Orlin 1977) and hard to approximate (Simon 1990), computing the Boolean rank is hard as well. Finding an optimal k-BMF of X has a graphic interpretation of minimizing the number of errors in an approximate covering of G by k bicliques. Even the computation of 1-BMF is hard (Gillis and Vavasis 2018), and can be stated in graphic form as finding a maximum weight biclique of $K_{n,m}$ with edge weights 1 for $(i,j) : x_{ij} = 1$ and -1 for $(i,j) : x_{ij} = 0$.

Many heuristic attempts have been made to approximately compute BMFs by focusing on recursively partitioning the given matrix $X \in \{0,1\}^{n\times m}$ and computing a 1-BMF at each step. The first such recursive method called Proximus (Koyutürk, Grama, and Ramakrishnan 2002) is used to compute BMF under standard arithmetic over the integers. For 1-BMF Proximus uses an alternating iterative heuristic applied to a random starting point which is based on the observation that if $\boldsymbol{a} \in \{0,1\}^n$ is given, then a vector $\boldsymbol{b} \in \{0,1\}^m$ that minimizes the distance between X and $\boldsymbol{a}\boldsymbol{b}^\top$ can be computed in $\mathcal{O}(nm)$ time. Since the introduction of Proximus, much research focused on computing efficient and accurate 1-BMF. (Shen, Ji, and Ye 2009) propose an integer program (IP) for 1-BMF and several linear programming (LP) relaxations of it, one of which leads to a 2-approximation. (Shi, Wang, and Shi 2014) provide a rounding based 2-approximation for 1-BMF by using an observation about the vertices of the polytope corresponding to the LP relaxation of an integer program. In (Beckerleg and Thompson 2020) a modification of the Proximus framework is explored using the approach of (Shen, Ji, and Ye 2009) to compute 1-BMF at each step.

k-BMF under Boolean arithmetic is explicitly introduced in (Miettinen et al. 2006, 2008), along with a heuristic algorithm called ASSO. The core of ASSO is based on an association rule-mining approach to create matrix $B \in \{0,1\}^{k\times m}$ and greedily fix $A \in \{0,1\}^{n\times k}$ with respect to B. The problem of finding an optimal A with respect to fixed B is NP-hard (Miettinen 2008) but can be solved in $\mathcal{O}(2^k kmn)$ time (Miettinen et al. 2008). The association rule-mining approach of (Miettinen et al. 2008) is further improved in (Barahona and Goncalves 2019) by a range of iterative heuristics employing this alternative fixing idea and local search. Another approach based on an alternating style heuristic is explored in (Zhang et al. 2007) to solve a non-linear unconstrained formulation of k-BMF with penalty terms in the objective for non-binary entries. (Wan et al. 2020) proposes another iterative heuristic which at every iteration permutes the rows and columns of X to create a dense submatrix in the upper right corner which is used as a rank-1 component in the k-BMF.

In (Lu, Vaidya, and Atluri 2008, 2014) an exponential size IP for k-BMF is introduced, which uses an explicit enumeration of all possible rows for factor matrix B and corresponding indicator variables. To tackle the exponential explosion, a heuristic row generation using association rule mining and subset enumeration is developed, but no non-heuristic method is considered. An exact linear IP for k-BMF with polynomially many variables and constraints is presented in (Kovacs, Gunluk, and Hauser 2017). This model uses McCormick envelopes (McCormick 1976) to linearise quadratic terms.

1.2 Our Contribution

In this paper, we present a novel IP formulation for k-BMF that overcomes several limitations of earlier approaches. In particular, our formulation does not suffer from permutation symmetry, it does not rely on heuristic pattern mining, and it has a stronger LP relaxation than that of (Kovacs, Gunluk, and Hauser 2017). On the other hand, our new formulation has an exponential number of variables which we tackle using a column generation approach that effectively searches over this exponential space without explicit enumeration, unlike the complete enumeration used for the exponential size model of (Lu, Vaidya, and Atluri 2008, 2014). Our proposed solution method is able to prove optimality for smaller datasets, while for larger datasets it provides solutions with better accuracy than the state-of-the-art heuristic methods. In addition, due to the entry-wise modelling of k-BMF in our approach, we can handle matrices with missing entries and our solutions can be used for binary matrix completion.

The rest of the paper is organised as follows. In Section 2 we briefly discuss the model of (Kovacs, Gunluk, and Hauser 2017) and its limitations. In Section 3 we introduce our integer programming formulation for k-BMF, detail a theoretical framework based on the large scale optimisation technique of column generation for its solution and discuss heuristics for the arising pricing subproblems. Finally, in Section 4 we demonstrate the practical applicability of our approach on several real world datasets.

2 Problem Formulation

Given a binary matrix $X \in \{0,1\}^{n\times m}$, and a fixed integer $k \ll \min(n,m)$ we wish to find two binary matrices $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ to minimise $\|X - A \circ B\|_F^2$, where $\|\cdot\|_F$ denotes the Frobenius norm and $\circ$ stands for Boolean matrix multiplication. Since X and $Z := A \circ B$ are binary matrices, the squared Frobenius and entry-wise ℓ_1 norm coincide and we can expand the objective function

$$\|X - Z\|_F^2 = \sum_{(i,j)\in E} (1 - z_{ij}) + \sum_{(i,j)\notin E} z_{ij}, \qquad (1)$$

where $E := \{(i,j) : x_{ij} = 1\}$ is the index set of the positive entries of X. (Kovacs, Gunluk, and Hauser 2017)

formulate the problem as an exact integer linear program by introducing variables $y_{i\ell j}$ for the product of $a_{i\ell}$ and $b_{\ell j}$ ($(i,\ell,j) \in \mathcal{F} := [n] \times [k] \times [m]$), and using McCormick envelopes (McCormick 1976) to avoid the appearance of a quadratic constraint arising from the product. McCormick envelopes represent the product of two binary variables a and b by a new variable y and four linear inequalities given by $MC(a,b) = \{y \in \mathbb{R} : a + b - 1 \leq y, y \leq a, y \leq b, 0 \leq y\}$. The model of (Kovacs, Gunluk, and Hauser 2017) reads as

$$(\text{IP}_{\text{exact}}) \quad \zeta_{IP} = \min_{a,b,y,z} \sum_{(i,j)\in E} (1 - z_{ij}) \quad + \sum_{(i,j)\notin E} z_{ij} \quad (2)$$

$$\text{s.t. } y_{i\ell j} \leq z_{ij} \leq \sum_{\ell=1}^{k} y_{i\ell j}, \qquad (i,\ell,j) \in \mathcal{F}, \quad (3)$$

$$y_{i\ell j} \in MC(a_{i\ell}, b_{\ell j}), \qquad (i,\ell,j) \in \mathcal{F}, \quad (4)$$

$$a_{i\ell}, b_{\ell j} \in \{0,1\}, \quad z_{ij} \leq 1, \qquad (i,\ell,j) \in \mathcal{F}. \quad (5)$$

The above model is exact in the sense that its optimal solutions correspond to optimal k-BMFs of X. Most general purpose IP solvers use an enumeration framework, which relies on bounds from the LP relaxation of the IP and consequently, it is easier to solve the IP when its LP bound is tighter. For $k = 1$, we have $y_{i1j} = z_{ij}$ for all i, j and the LP relaxation of the model is simply the LP relaxation of the McCormick envelopes which has a rich and well-studied polyhedral structure (Padberg 1989). However, for $k > 1$, IP_{exact}'s LP relaxation (LP_{exact}) only provides a trivial bound.

Lemma 1. *For $k > 1$, LP_{exact} has optimal objective value 0 which is attained by at least $\binom{k}{2}$ solutions.*

For the proof of Lemma 1, see Appendix 5.1. Furthermore, for $k > 1$ the model is highly symmetric, since $AP \circ P^{-1}B$ is an equivalent solution for any permutation matrix P. These properties of the model make it unlikely to be solved to optimality in a reasonable amount of time for a large matrix X, though the symmetries can be partially broken by incorporating constraints $\sum_i a_{i\ell_1} \geq \sum_i a_{i\ell_2}$ for all $\ell_1 < \ell_2$.

Note that constraint (3) implies $\frac{1}{k}\sum_{\ell=1}^{k} y_{i\ell j} \leq z_{ij} \leq \sum_{\ell=1}^{k} y_{i\ell j}$ as a lower and upper bound on each variable z_{ij}. Hence, the objective function may be approximated by

$$\zeta_{IP}(\rho) = \sum_{(i,j)\in E} (1 - z_{ij}) + \rho \sum_{(i,j)\notin E} \sum_{\ell=1}^{k} y_{i\ell j}, \quad (6)$$

where ρ is a parameter of the formulation. By setting $\rho = \frac{1}{k}$ we underestimate the original objective, while setting $\rho = 1$ we overestimate. Using (6) as the objective function reduces the number of variables and constraints in the model. Variables z_{ij} need only be declared for $(i,j) \in E$, and constraint (3) simplifies to $z_{ij} \leq \sum_{\ell=1}^{k} y_{i\ell j}$ for $(i,j) \in E$.

3 A Formulation via Column Generation

The exact model presented in the previous section relies on polynomially many constraints and variables and constitutes the first approach towards obtaining k-BMF with optimality guarantees. However, such a compact IP formulation may be weak in the sense that its LP relaxation is a very coarse approximation to the convex hull of integer feasible points and an IP formulation with exponentially many variables or constraints can have the potential to provide a tighter relaxation(Lübbecke and Desrosiers 2005). Motivated by this fact, we introduce a new formulation with an exponential number of variables and detail a column generation framework for its solution.

Consider enumerating all possible rank-1 binary matrices of size $n \times m$ and let

$$\mathcal{R} = \{\boldsymbol{a}\boldsymbol{b}^\top : \boldsymbol{a} \in \{0,1\}^n, \boldsymbol{b} \in \{0,1\}^m, \boldsymbol{a}, \boldsymbol{b} \neq \boldsymbol{0}\}.$$

The size of $\mathcal{R}$ is $|\mathcal{R}| = (2^n - 1)(2^m - 1)$ as any pair of binary vectors $\boldsymbol{a}, \boldsymbol{b} \neq \boldsymbol{0}$ leads to a unique rank-1 matrix $Y = \boldsymbol{a}\boldsymbol{b}^\top$ with $Y_{ij} = 1$ for $\{(i,j) : a_i = 1, b_j = 1\}$. Define a binary decision variable q_ℓ to denote if the ℓ-th rank-1 binary matrix in $\mathcal{R}$ is included in a rank-k factorisation of X ($q_\ell = 1$), or not ($q_\ell = 0$). Let $\boldsymbol{q} \in \{0,1\}^{|\mathcal{R}|}$ be a vector that has a component q_ℓ for each matrix in $\mathcal{R}$. We form a $\{0,1\}$-matrix M of dimension $nm \times |\mathcal{R}|$ whose rows correspond to entries of an $n \times m$ matrix, columns to rank-1 binary matrices in $\mathcal{R}$ and $M_{(i,j),\ell} = 1$ if the (i,j)-th entry of the ℓ-th rank-1 binary matrix in $\mathcal{R}$ is 1, $M_{(i,j),\ell} = 0$ otherwise. We split M horizontally into two matrices $M_{\boldsymbol{0}}$ and $M_{\boldsymbol{1}}$, so that rows of M corresponding to a positive entry of the given matrix X are in $M_{\boldsymbol{1}}$ and the rest of rows of M in $M_{\boldsymbol{0}}$,

$$M = \begin{bmatrix} M_{\boldsymbol{0}} \\ M_{\boldsymbol{1}} \end{bmatrix} \text{ where } \begin{matrix} M_{\boldsymbol{0}} \in \{0,1\}^{(nm-|E|)\times|\mathcal{R}|}, \\ M_{\boldsymbol{1}} \in \{0,1\}^{|E|\times|\mathcal{R}|}. \end{matrix} \quad (7)$$

The following Master Integer Program over an exponential number of variables is an exact model for k-BMF,

$$(\text{MIP}_{\text{exact}}) \ \zeta_{\text{MIP}} = \min \boldsymbol{1}^\top \boldsymbol{\xi} + \boldsymbol{1}^\top \boldsymbol{\pi} \quad (8)$$

$$\text{s.t. } M_{\boldsymbol{1}} \boldsymbol{q} + \boldsymbol{\xi} \geq \boldsymbol{1} \quad (9)$$

$$M_{\boldsymbol{0}} \boldsymbol{q} \leq k\boldsymbol{\pi} \quad (10)$$

$$\boldsymbol{1}^\top \boldsymbol{q} \leq k \quad (11)$$

$$\boldsymbol{\xi} \geq \boldsymbol{0}, \boldsymbol{\pi} \in \{0,1\}^{nm-|E|}, \quad (12)$$

$$\boldsymbol{q} \in \{0,1\}^{|\mathcal{R}|}. \quad (13)$$

Constraint (11) ensures that at most k rank-1 matrices are active in a factorisation. Variables ξ_{ij} correspond to positive entries of X, and are forced by constraint (9) to take value 1 and increase the objective if the (i,j)-th positive entry of X is not covered. Similarly, variables π_{ij} correspond to zero entries of X and are forced to take value 1 by constraint (10) if the (i,j)-th zero entry of X is erroneously covered in a factorisation. One of the imminent advantages of $\text{MIP}_{\text{exact}}$ is using indicator variables directly for rank-1 matrices instead of the entries of factor matrices A, B, hence no permutation symmetry arises. In addition, for all k not exceeding a certain number that depends on X, the LP relaxation of $\text{MIP}_{\text{exact}}$ ($\text{MLP}_{\text{exact}}$) has strictly positive optimal objective value.

Lemma 2. *Let $i(X)$ be the isolation number of X. For all $k < i(X)$, we have $0 < \zeta_{MLP}$.*

For the definition of isolation number and the proof of Lemma 2 see Appendix 5.2. Similarly to the polynomial size exact model IP_{exact} in the previous section, we consider a

modification of $\text{MIP}_{\text{exact}}$ with an objective that is analogous to the one in Equation (6),

$$(\text{MIP}(\rho))\ z_{\text{MIP}(\rho)} = \min \mathbf{1}^\top \boldsymbol{\xi} + \rho\, \mathbf{1}^\top M_{\mathbf{0}} \boldsymbol{q} \tag{14}$$

$$\text{s.t. (9), (11) hold and} \tag{15}$$

$$\boldsymbol{\xi} \geq \mathbf{0}, \quad \boldsymbol{q} \in \{0,1\}^{|\mathcal{R}|}.$$

The objective of MIP(ρ) simply counts the number of positive entries of X that are not covered by any of the k rank-1 matrices chosen, plus the number of times zero valued entries of X are erroneously covered weighted by parameter ρ. Depending on the selection of parameter ρ, MIP(ρ) provides a lower or upper bound on $\text{MIP}_{\text{exact}}$. We denote the LP relaxation of MIP(ρ) by MLP(ρ).

Lemma 3. *For $\rho = \frac{1}{k}$, the optimal objective values of the LP relaxations MLP_{exact} and MLP($\frac{1}{k}$) coincide.*

For a short proof of Lemma 3 see Appendix 5.3. Combining Lemmas 1, 2 and 3 we obtain the following relations between formulations IP_{exact}, $\text{MIP}_{\text{exact}}$, MIP($\rho$) and their LP relaxations for $k > 1$,

$$z_{\text{MIP}(\frac{1}{k})} \leq \zeta_{IP} = \zeta_{\text{MIP}} \leq z_{\text{MIP}(1)}, \tag{16}$$

$$0 = \zeta_{LP} \leq z_{\text{MLP}(\frac{1}{k})} = \zeta_{\text{MLP}} \leq z_{\text{MLP}(1)}. \tag{17}$$

Let $\boldsymbol{p}$ be the dual variable vector associated to constraints (9) and μ be the dual variable to constraint (11). Then the dual of MLP(ρ) is given by

$$(\text{MDP}(\rho))\ z_{\text{MDP}(\rho)} = \max \mathbf{1}^\top \boldsymbol{p} - k\mu \tag{18}$$

$$\text{s.t. } M_{\mathbf{1}}^\top \boldsymbol{p} - \mu \mathbf{1} \leq \rho\, M_{\mathbf{0}}^\top \mathbf{1}, \tag{19}$$

$$\mu \geq 0,\ \boldsymbol{p} \in [0,1]^{|E|}. \tag{20}$$

Due to the number of variables in the formulation, it is not practical to solve MIP(ρ) or its LP relaxation MLP(ρ) explicitly. *Column generation* (CG) is a technique to solve large LPs by iteratively generating only the variables which have the potential to improve the objective function (Barnhart et al. 1998). The CG procedure is initialised by explicitly solving a Restricted Master LP which has a small subset of the variables in MLP(ρ). The next step is to identify a missing variable with a *negative reduced cost* to be added to this Restricted MLP(ρ). To avoid explicitly considering all missing variables, a *pricing problem* is formulated and solved. The solution of the pricing problem either returns a variable with negative reduced cost and the procedure is iterated; or proves that no such variable exists and hence the solution of the Restricted MLP(ρ) is optimal for the complete formulation MLP(ρ).

We use CG to solve MLP(ρ) by considering a sequence $(t = 1, 2, ...)$ of Restricted MLP(ρ)'s with constraint matrix $M^{(t)}$ being a subset of columns of M, where each column $\boldsymbol{y} \in \{0,1\}^{nm}$ of M corresponds to a flattened rank-1 binary matrix $\boldsymbol{a}\boldsymbol{b}^\top$ according to Equation (7). The constraint matrix of the first Restricted MLP(ρ) may be left empty or can be warm started by identifying a few rank-1 matrices in $\mathcal{R}$, say from a heuristic solution. Upon successful solution of the t-th Restricted MLP(ρ), we obtain a vector of dual variables $[\boldsymbol{p}^*, \mu^*] \geq \mathbf{0}$ optimal for the t-th Restricted MLP(ρ). To identify a missing column of M that has a negative reduced cost, we solve the following pricing problem (PP):

$$(\text{PP})\ \omega(\boldsymbol{p}^*) = \max_{a,b,y} \sum_{(i,j)\in E} p^*_{ij} y_{ij} - \rho \sum_{(i,j)\notin E} y_{ij}$$

$$\text{s.t. } y_{ij} \in MC(a_i, b_j),\ a_i, b_j \in \{0,1\},\ i \in [n], j \in [m].$$

The objective of PP depends on the current dual solution $[\boldsymbol{p}^*, \mu^*]$ and its optimal solution corresponds to a rank-1 binary matrix $\boldsymbol{a}\boldsymbol{b}^\top$ whose corresponding variable q_ℓ in MLP(ρ) has the smallest reduced cost. If $\omega(\boldsymbol{p}^*) \leq \mu^*$, then the dual variables $[\boldsymbol{p}^*, \mu^*]$ of the Restricted MLP(ρ) are feasible for MDP(ρ) and hence the current solution of the Restricted MLP(ρ) is optimal for the full formulation MLP(ρ). If $\omega(\boldsymbol{p}^*) > \mu^*$, then the variable q_ℓ associated with the rank-1 binary matrix $\boldsymbol{a}\boldsymbol{b}^\top$ is added to the Restricted MLP(ρ) and the procedure is iterated. CG optimally terminates if at some iteration we have $\omega(\boldsymbol{p}^*) \leq \mu^*$.

To apply the CG approach above to $\text{MLP}_{\text{exact}}$ only a small modification needs to be made. The Restricted $\text{MLP}_{\text{exact}}$ provides dual variables for constraints (10) which are used in the objective of PP for coefficients of y_{ij} $(i,j) \notin E$. Note however, that CG cannot be used to solve a modification of $\text{MLP}_{\text{exact}}$ in which constraints (10) are replaced by exponentially many constraints $(M_{\mathbf{0}})_\ell\ q_\ell \leq \boldsymbol{\pi}$ for $\ell \in [|\mathcal{R}|]$ where $(M_{\mathbf{0}})_\ell$ denotes the ℓ-th column of $M_{\mathbf{0}}$, see Appendix 5.4.

If the optimal solution of MLP(ρ) is integral, then it is also optimal for MIP(ρ). However, if it is fractional, then it only provides a lower bound on the optimal value of MIP(ρ). In this case we obtain an integer feasible solution by simply adding integrality constraints on the variables of the final Restricted MLP(ρ) and solving it as an integer program. If $\rho = 1$, the optimal solution of this integer program is optimal for MIP(1) if the objective of the Restricted MIP(1) and the ceiling of the Restricted MLP(1) agree. To solve the MIP(ρ) to optimality in all cases, one needs to embed CG into branch-and-bound which we do not do. However, note that even if the CG procedure is terminated prematurely, one can still obtain a lower bound on MLP(ρ) and MIP(ρ) as follows. Let the objective value of any of the Restricted MLP(ρ)'s be

$$z_{\text{RMLP}} = \mathbf{1}^\top \boldsymbol{\xi}^* + \rho \mathbf{1}^\top M_0^* \boldsymbol{q}^* = \mathbf{1}^\top \boldsymbol{p}^* - k\mu^* \tag{21}$$

where $[\boldsymbol{\xi}^*, \boldsymbol{q}^*]$ is the solution of the Restricted MLP(ρ), $[\boldsymbol{p}^*, \mu^*]$ is the solution of the dual of the Restricted MLP(ρ) and $\mathbf{1}^\top M_0^*$ is the objective coefficient of columns in the Restricted MLP(ρ). Assume that we solve PP to optimality and we obtain a column $\boldsymbol{y}$ for which the reduced cost is negative, $\omega(\boldsymbol{p}^*) > \mu^*$. In this case, we can construct a feasible solution to MDP(ρ) by setting $\boldsymbol{p} := \boldsymbol{p}^*$ and $\mu := \omega(\boldsymbol{p}^*)$ and get the following bound on the optimal value $z_{\text{MLP}(\rho)}$ of MLP(ρ),

$$z_{\text{MLP}(\rho)} \geq \mathbf{1}^\top \boldsymbol{p}^* - k\,\omega(\boldsymbol{p}^*) = z_{\text{RMLP}} - k(\omega(\boldsymbol{p}^*) - \mu^*). \tag{22}$$

If we do not have the optimal solution to PP but have an upper bound $\bar{\omega}(\boldsymbol{p}^*)$ on it, $\omega(\boldsymbol{p}^*)$ can be replaced by $\bar{\omega}(\boldsymbol{p}^*)$ in equation (22) and the bound on MLP(ρ) still holds. Furthermore, this lower bound on MLP(ρ) provides a valid lower bound on MIP(ρ). Consequently, our approach always produces a bound on the optimality gap of the final solution which heuristic methods cannot do. We have, however, no a priory (theoretical) bound on this gap.

3.1 The Pricing Problem

The efficiency of the CG procedure described above greatly depends on solving PP efficiently. In standard form PP can be written as a bipartite binary quadratic program (BBQP)

$$\text{(PP)} \quad \omega(\boldsymbol{p}^*) = \max_{\boldsymbol{a}\in\{0,1\}^n, \boldsymbol{b}\in\{0,1\}^m} \boldsymbol{a}^\top H \boldsymbol{b} \qquad (23)$$

for H an $n \times m$ matrix with $h_{ij} = p^*_{ij} \in [0,1]$ for $(i,j) \in E$ and $h_{ij} = -\rho$ for $(i,j) \notin E$. BBQP is NP-hard in general as it includes the maximum edge biclique problem (Peeters 2003), hence for large X it may take too long to solve PP to optimality at each iteration. To speed up computations, the IP formulation of PP may be improved by eliminating redundant constraints. The McCormick envelopes set two lower and two upper bounds on y_{ij}. Due to the objective function it is possible to declare the lower (upper) bounds y_{ij} for only $(i,j) \notin E$ ($(i,j) \in E$) without changing the optimum.

If a heuristic approach to PP provides a solution with negative reduced cost, then it is valid to add this heuristic solution as a column to the next Restricted MLP(ρ). Most heuristic algorithms that are available for BBQP build on the idea that the optimal $\boldsymbol{a} \in \{0,1\}^n$ with respect to a fixed $\boldsymbol{b} \in \{0,1\}^m$ can be computed in $\mathcal{O}(nm)$ time and this procedure can be iterated by alternatively fixing $\boldsymbol{a}$ and $\boldsymbol{b}$. (Karapetyan and Punnen 2013) present several local search heuristics for BBQP along with a simple greedy algorithm. Below we detail this greedy algorithm and introduce some variants of it which we use in the next section to provide a warm start to PP at every iteration of the CG procedure.

The greedy algorithm of (Karapetyan and Punnen 2013) aims to set entries of $\boldsymbol{a}$ and $\boldsymbol{b}$ to 1 which correspond to rows and columns of H with the largest positive weights. In the first phase of the algorithm, the row indices i of H are put in decreasing order according to their sum of positive entries, so $\gamma_i^+ \geq \gamma_{i+1}^+$ where $\gamma_i^+ := \sum_{j=1}^m \max(0, h_{ij})$. Then sequentially according to this ordering, a_i is set to 1 if $\sum_{j=1}^m \max(0, \sum_{\ell=1}^{i-1} a_\ell h_{\ell j}) < \sum_{j=1}^m \max(0, \sum_{\ell=1}^{i-1} a_\ell h_{\ell j} + h_{ij})$ and 0 otherwise. In the second phase, b_j is set to 1 if $(\boldsymbol{a}^\top H)_j > 0$, 0 otherwise. The precise outline of the algorithm is given in Appendix 5.5.

There are many variants of the greedy algorithm one can explore. First, the solution greatly depends on the ordering of i's in the first phase. If for some $i_1 \neq i_2$ we have $\gamma_{i_1}^+ = \gamma_{i_2}^+$, comparing the sum of negative entries of rows i_1 and i_2 can put more "influential" rows of H ahead in the ordering. Let us call this ordering the *revised ordering* and the one which only compares the positive sums as the *original ordering*. Another option is to use a completely *random order* of i's or to apply a small perturbation to sums γ_i^+ to get a *perturbed* version of the revised or original ordering. None of the above ordering strategies clearly dominates the others in all cases but they are fast to compute hence one can evaluate all five ordering strategies (original, revised, original perturbed, revised perturbed, random) and pick the best one. Second, the algorithm as presented above first fixes $\boldsymbol{a}$ and then $\boldsymbol{b}$. Changing the order of fixing $\boldsymbol{a}$ and $\boldsymbol{b}$ can yield a different result hence it is best to try for both H and $H^\top$. In general, it is recommended to start the first phase on the smaller dimension. Third, the solution from the greedy algorithm may be improved by computing the optimal $\boldsymbol{a}$ with respect to fixed $\boldsymbol{b}$. This idea then can be used to fix $\boldsymbol{a}$ and $\boldsymbol{b}$ in an alternating fashion and stop when no changes occur in either.

4 Experiments

The CG approach introduced in the previous section provides a theoretical framework for computing k-BMF with optimality guarantees. In this section we present some experimental results with CG to demonstrate the practical applicability of our approach on eight real world categorical datasets that were downloaded from online repositories (Dua and Graff 2017), (Krebs 2008). Table 1 shows a short summary of the eight datasets used, details on converting categorical columns into binary and missing value treatment can be found in Appendix 5.8. Table 1 also shows the value of the isolation number $i(X)$ for each dataset, which provides a lower bound on the Boolean rank (Monson, Pullman, and Rees 1995).

	zoo	tumor	hepat	heart	lymp	audio	apb	votes
n	101	339	155	242	148	226	105	434
m	17	24	38	22	44	94	105	32
$i(X)$	16	24	30	22	42	90	104	30
%1s	44.3	24.3	47.2	34.4	29.0	11.3	8.0	47.3

Table 1: Summary of binary real world datasets

Since the efficiency of CG greatly depends on the speed of generating columns, let us illustrate the speed-up gained by using heuristic pricing. At each iteration of CG, 30 variants of the greedy heuristic are computed to obtain an initial feasible solution to PP. The 30 variants of the greedy algorithm use the original and revised ordering, their transpose and perturbed version and 22 random orderings. All greedy solutions are improved by the alternating heuristic until no further improvement is found. Under *exact* pricing, the best heuristic solution is used as a warm start and PP is solved to optimality at each iteration using (CPLEX Optimization). In simple heuristic (*heur*) pricing, if the best heuristic solution to PP has negative reduced cost, $\omega_{\text{heur}}(\boldsymbol{p}^*) > \mu^*$, then the heuristic column is added to the next Restricted MLP(ρ). If at some iteration, the best heuristic column does not have negative reduced cost, CPLEX is used to solve PP to optimality for that iteration. The multiple heuristic (*heur_multi*) pricing is a slight modification of the simple heuristic strategy, in which at each iteration all columns with negative reduced cost are added to the next Restricted MLP(ρ).

	zoo			tumor		
k	MIP(1)	KGH17	LVA08	MIP(1)	KGH17	LVA08
2	0.0	0.0	100.0	0.9	40.8	*
5	0.0	59.2	100.0	9.3	98.0	*
10	3.0	95.8	100.0	28.4	100.0	*

Table 2: % optimality gap after 20 mins under objective (6)

k		zoo	tumor	hepat	heart	lymp	audio	apb	votes
2	MLP($\frac{1}{k}$)	206.5	1178.9	978.7	882.9	917.2	1256.5	709	1953
	MLP(1)	272	1409.8	1384	1185	1188.8	1499	776	2926
	MIP(1)	272(271)	1411	1384(1382)	1185	1197(1184)	1499	776	2926
5	MLP($\frac{1}{k}$)	42.8	463.9	333.1	291.0	366.7	654.2	433.5	715.5
	MLP(1)	127	1019.3	1041.1	736	914.0	1159.3	683.0	2135.5
	MIP(1)	127(125)	1029	1228	736	997(991)	1176	684(683)	2277(2274)
10	MLP($\frac{1}{k}$)	4.8	192.8	142.5	102.3	165.1	351.4	166.8	307.9
	MLP(1)	38.8	575.5	734.8	419	653.2	867.2	574.2	1409.5
	MIP(1)	40	579	910	419	737(732)	893	577(572)	1566(1549)

Table 3: Primal objective values of MLP(1), MLP($\frac{1}{k}$), MIP(1) after 20 mins of CG

Figure 1 indicates the differences between pricing strategies when solving MLP(1) via CG for $k = 5, 10$ on the zoo dataset. The primal objective value of MLP(1) (decreasing curve) and the value of the dual bound (increasing curve) computed using the formula in equation (22) are plotted against time. Sharp increases in the dual bound for heuristic pricing strategies correspond to iterations in which CPLEX was used to solve PP, as for the evaluation of the dual bound on MLP(1) we need a strong upper bound on $\omega(\boldsymbol{p}^*)$ which heuristic solutions do not provide. While we observe a tailing off effect (Lübbecke and Desrosiers 2005) on all three curves, both heuristic pricing strategies provide a significant speed-up from exact pricing, with adding multiple columns at each iteration being the fastest.

In Table 2 we present computational results comparing

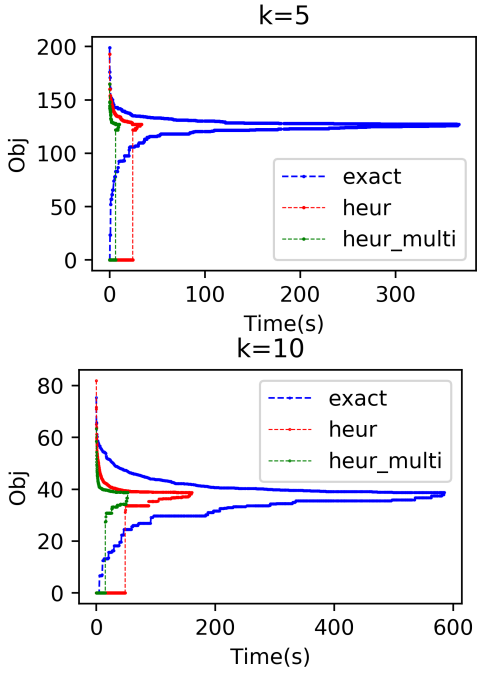

Figure 1: Comparison of pricing strategies for solving MLP(1) on the zoo dataset

the optimality gap ($100 \times \frac{\text{best integer}-\text{best bound}}{\text{best integer}}$) of MIP(1), the compact formulation of (Kovacs, Gunluk, and Hauser 2017) (*KGH17*) and the exponential formulation of (Lu, Vaidya, and Atluri 2008) (*LVA08*) under objective (6) using a 20 mins time budget. See Appendix 5.7 for the precise statement of formulations KGH17 and LVA08 under objective (6). Reading in the full exponential size model LVA08 using 16 GB memory is not possible for datasets other than zoo. Table 2 shows that different formulations and algorithms to solve them make a difference in practice: our novel exponential formulation MIP(1) combined with an effective computational optimization approach (CG) produces solutions with smaller optimality gap than the compact formulation as it scales better and it has a stronger LP relaxation.

In order for CG to terminate with a certificate of optimality, at least one pricing problem has to be solved to optimality. Unfortunately for the larger datasets this cannot be achieved in under 20 mins. Therefore, for datasets other than zoo, we change the multiple heuristic pricing strategy as follows: We impose an overall time limit of 20 mins on the CG process and use the barrier method in CPLEX as the LP solver for the Restricted MLP(ρ) at each iteration. In order to maximise the diversity of columns added at each iteration, we choose at most two columns with negative reduced cost that are closest to being mutually orthogonal. If CPLEX has to be used to improve the heuristic pricing solution, we do not solve PP to optimality but abort CPLEX if a column with negative reduced cost has been found. While these modifications result in a speed-up, they reduce the chance of obtaining a strong dual bound. In case a strong dual bound is desired, we may continue applying CG iterations with exact pricing after the 20 mins of heuristic pricing have run their course.

In our next experiment, we explore the differences between formulations MLP($\frac{1}{k}$), MLP(1) and MIP(1). We warm start CG by identifying a few heuristic columns using the code of (Barahona and Goncalves 2019) and a new fast heuristic (k-*greedy*) which sequentially computes k rank-1 binary matrices via the greedy algorithm for BBQP starting with the coefficient matrix $H = 2X - 1$ and then setting entries of H to zero that correspond to entries of X that are covered. For the precise outline of k-greedy, see Appendix 5.6.

Table 3 shows the primal objective values of MLP(1) and MLP($\frac{1}{k}$) with heuristic pricing using a time limit of 20 mins,

		zoo	tumor	hepat	heart	lymp	audio	apb	votes
k=2	CG	**271**	1411	**1382**	**1185**	1184	**1499**	**776**	**2926**
	IP_{exact}	**271**	**1408**	1391	1187	**1180**	**1499**	**776**	**2926**
	ASSO++	276	1437	1397	1187	1202	1503	**776**	**2926**
	k-greedy	325	1422	1483	1204	1201	**1499**	**776**	2929
	pymf	276	1472	1418	1241	1228	1510	794	2975
	ASSO	367	1465	1724	1251	1352	1505	778	2946
	NMF	291	1626	1596	1254	1366	2253	809	3069
	MEBF	348	1487	1599	1289	1401	1779	812	3268
k=5	CG	**125**	**1029**	**1228**	**736**	**991**	**1176**	**683**	**2272**
	IP_{exact}	133	1055	**1228**	738	1029	1211	690	2293
	ASSO++	133	1055	**1228**	738	1039	1211	694	2302
	k-greedy	233	1055	1306	748	1063	1211	690	2310
	pymf	142	1126	1301	835	1062	1245	730	2517
	ASSO	354	1092	1724	887	1352	1505	719	2503
	NMF	163	1207	1337	995	1158	1565	762	2526
	MEBF	173	1245	1439	929	1245	1672	730	2832
k=10	CG	**40**	**579**	910	**419**	**730**	**893**	**572**	**1527**
	IP_{exact}	41	583	**902**	**419**	805	919	590	1573
	ASSO++	55	583	910	**419**	812	922	591	1573
	k-greedy	184	675	1088	565	819	976	611	1897
	pymf	96	703	1186	581	987	1106	602	2389
	ASSO	354	587	1724	694	1352	1505	661	2503
	NMF	153	826	1337	995	1143	1407	689	2481
	MEBF	122	990	1328	777	1004	1450	662	2460

Table 4: Factorisation errors in $\|\cdot\|_F^2$ for eight methods for k-BMF

and the objective value of MIP(1) solved on the columns generated by MLP(1). If the error measured in $\|\cdot\|_F^2$ differs from the objective of MIP(1), the former is shown in parenthesis. It is interesting to observe that MLP(1) has a tendency to produce near integral solutions and that the objective value of MIP(1) often coincides with the error measured in $\|\cdot\|_F^2$. We note that once a master LP formulation is used to generate columns, any of the MIP models could be used to obtain an integer feasible solution. In experiments, we found that formulation MIP(ρ) is solved much faster than MIP_{exact} and that setting ρ to 1 or 0.95 provides the best integer solutions.

We compare the CG approach against the most widely used k-BMF heuristics and the exact model IP_{exact}.The heuristic algorithms we evaluate include the ASSO algorithm (Miettinen et al. 2006, 2008), the alternating iterative local search algorithm (ASSO++) of (Barahona and Goncalves 2019) which uses ASSO as a starting point, algorithm k-greedy detailed in Appendix 5.6, the penalty objective formulation (pymf) of (Zhang et al. 2007) via the implementation of (Schinnerl 2017) and the permutation-based heuristic (MEBF) (Wan et al. 2020). We also evaluate IP_{exact} with a time limit of 20 mins and provide the heuristic solutions of ASSO++ and k-greedy as a warm start to it. In addition, we compute rank-k NMF and binarise it with a threshold of 0.5. The exact details and parameters used in the computations can be found in Appendix 5.10.[1] Our CG approach (CG) results are obtained by generating columns for 20 mins using formulation MLP(1) with a warm start of initial columns obtained from ASSO++ and k-greedy, then solving MIP(ρ) for ρ set to 1 and 0.95 over the generated columns and picking the best. Table 4 shows the factorisation error in $\|\cdot\|_F^2$ after evaluating the above described methods on all datasets for $k = 2, 5, 10$. The best result for each instance is indicated in boldface. We observe that CG provides the strictly smallest error for 15 out of 24 cases.

[1]Appendix is available at `arxiv.org/abs/2011.04457`.

5 Conclusion

In this paper, we studied the rank-k binary matrix factorisation problem under Boolean arithmetic. We introduced a new integer programming formulation and detailed a method using column generation for its solution. Our experiments indicate that our method using 20 mins time budget is producing more accurate solutions than most heuristics available in the literature and is able to prove optimality for smaller datasets. In certain critical applications such as medicine, spending 20 minutes to obtain a higher accuracy factorisation with a bound on the optimality gap can be easily justified. In addition, solving BMF to near optimality via our proposed method paves the way to more robustly benchmark heuristics for k-BMF. Future directions that could be explored are related to designing more accurate heuristics and faster exact algorithms for the pricing problem. In addition, a full branch-and-price algorithm implementation would be beneficial once the pricing problems are solved more efficiently.

Acknowledgements

During the completion of this work R.Á.K was supported by The Alan Turing Institute and The Office for National Statistics.

References

Barahona, F.; and Goncalves, J. 2019. Local Search Algorithms for Binary Matrix Factorization. https://github.com/IBM/binary-matrix-factorization/blob/master/code. Last accessed on 2020-03-30.

Barnhart, C.; Johnson, E. L.; Nemhauser, G. L.; Savelsbergh, M. W. P.; and Vance, P. H. 1998. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research* 46(3): 316–329.

Beckerleg, M.; and Thompson, A. 2020. A divide-and-conquer algorithm for binary matrix completion. *Linear Algebra and its Applications* 601: 113–133.

Cios, K. J.; and Kurgan, L. A. 2001. UCI Machine Learning Repository: SPECT Heart Data. URL https://archive.ics.uci.edu/ml/datasets/spect+heart. Last accessed on 2020-06-11.

CPLEX Optimization. 2018. *Using the CPLEX Callable Library, V.12.8*. CPLEX Optimization, Inc., Incline Village, NV.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml. Last accessed on 2020-06-11.

Forsyth, R. 1990. UCI Machine Learning Repository: Zoo Data Set. URL http://archive.ics.uci.edu/ml/datasets/Zoo. Last accessed on 2020-06-11.

Gillis, N.; and Vavasis, S. A. 2018. On the Complexity of Robust PCA and l1-Norm Low-Rank Matrix Approximation. *Mathematics of Operations Research* 43(4): 1072–1084.

Golub, G.; and Van Loan, C. 1989. *Matrix Computations*. Baltimore: Johns Hopkins University Press, 2nd edition.

Gong, G. 1988. UCI Machine Learning Repository: Hepatitis Data Set. URL https://archive.ics.uci.edu/ml/datasets/Hepatitis. Last accessed on 2020-06-11.

Karapetyan, D.; and Punnen, A. P. 2013. Heuristic algorithms for the bipartite unconstrained 0-1 quadratic programming problem. arXiv 1210.3684.

Kim, K. 1982. *Boolean Matrix Theory and Applications*. Monographs and textbooks in pure and applied mathematics. Dekker.

Kononenko, I.; and Cestnik, B. 1988a. UCI Mach. Learn. Rep.: Primary Tumor Domain. URL https://archive.ics.uci.edu/ml/datasets/Primary+Tumor. Last accessed on 2020-06-11.

Kononenko, I.; and Cestnik, B. 1988b. UCI Machine Learning Repository: Lymphography Data Set. URL https://archive.ics.uci.edu/ml/datasets/Lymphography. Last accessed on 2020-06-11.

Kovacs, R. A.; Gunluk, O.; and Hauser, R. A. 2017. Low-Rank Boolean Matrix Approximation by Integer Programming. In *NIPS 2017*, Optimization for Machine Learning Workshop. https://opt-ml.org/papers/OPT2017_paper_34.pdf.

Koyutürk, M.; Grama, A.; and Ramakrishnan, N. 2002. Algebraic Techniques for Analysis of Large Discrete-Valued Datasets. In *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '02, 311–324. London, UK, UK: Springer-Verlag.

Krebs, V. 2008. Amazon Political Books. URL http://moreno.ss.uci.edu/data.html#books. Last accessed on 2020-06-11.

Lee, D. D.; and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755): 788–791.

Li, T. 2005. A General Model for Clustering Binary Data. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '05. New York, NY, USA: Association for Computing Machinery.

Lu, H.; Vaidya, J.; and Atluri, V. 2008. Optimal Boolean Matrix Decomposition: Application to Role Engineering. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, 297–306. Washington, DC, USA: IEEE Computer Society.

Lu, H.; Vaidya, J.; and Atluri, V. 2014. An optimization framework for role mining. *Journal of Computer Security* 22(1): 1 – 31.

Lübbecke, M. E.; and Desrosiers, J. 2005. Selected Topics in Column Generation. *Operations Research* 53(6): 1007–1023.

McCormick, G. P. 1976. Computability of Global Solutions to Factorable Nonconvex Programs: Part I – Convex Underestimating Problems. *Math. Program.* 10(1): 147–175.

Miettinen, P. 2008. On the Positive–Negative Partial Set Cover Problem. *Inf. Process. Lett.* 108(4): 219–221.

Miettinen, P. 2012. Binary Matrix Factorisations Tutorial @ ECML PKDD 2012. URL https://people.mpi-inf.mpg.de/~pmiettin/bmf_tutorial/tutorial.pdf. Last accessed on 2021-03-11.

Miettinen, P.; Mielikäinen, T.; Gionis, A.; Das, G.; and Mannila, H. 2006. The Discrete Basis Problem. In *Knowledge Discovery in Databases: PKDD 2006*, 335–346. Berlin, Heidelberg: Springer Berlin Heidelberg.

Miettinen, P.; Mielikäinen, T.; Gionis, A.; Das, G.; and Mannila, H. 2008. The Discrete Basis Problem. *IEEE Trans. on Knowl. and Data Eng.* 20(10): 1348–1362.

Monson, S. D.; Pullman, N. J.; and Rees, R. 1995. A Survey of Clique and Biclique Coverings and Factorizations of (0,1)–Matrices. *Bulletin – Institute of Combinatorics and its Applications* 14: 17–86.

Orlin, J. 1977. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)* 80(5): 406 – 424.

Padberg, M. 1989. The Boolean Quadric Polytope: Some Characteristics, Facets and Relatives. *Math. Program.* 45(1): 139–172.

Peeters, R. 2003. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics* 131(3): 651 – 654.

Quinlan, R. 1992. UCI Machine Learning Repository: Audiology (Standardized) Data Set. URL http://archive.ics.uci.edu/ml/datasets/audiology+(standardized). Last accessed on 2020-06-11.

Schinnerl, C. 2017. PyMF - Python Matrix Factorization Module. URL https://github.com/ChrisSchinnerl/pymf3. Last accessed on 2021-03-11.

Schlimmer, J. 1987. UCI Machine Learning Repository: 1984 US Cong. Voting Records Database. URL https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records. Last accessed on 2020-06-11.

Shen, B.-H.; Ji, S.; and Ye, J. 2009. Mining Discrete Patterns via Binary Matrix Factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, 757–766. New York, NY, USA: ACM.

Shi, Z.; Wang, L.; and Shi, L. 2014. Approximation method to rank-one binary matrix factorization. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, 800–805.

Simon, H. U. 1990. On Approximate Solutions for Combinatorial Optimization Problems. *SIAM J. Discrete Math.* 3: 294–310.

Wan, C.; Chang, W.; Zhao, T.; Li, M.; Cao, S.; and Zhang, C. 2020. Fast and Efficient Boolean Matrix Factorization by Geometric Segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(04): 6086–6093.

Zhang, Z.; Li, T.; Ding, C.; and Zhang, X. 2007. Binary Matrix Factorization with Applications. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, 391–400. USA: IEEE Computer Society.