Binary Matrix Factorisation via Column Generation

Réka Á. Kovács, Oktay Günlük, Raphael A. Hauser

1 University of Oxford & The Alan Turing Institute
2 Cornell University
rekakovacs@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 minimal. 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: $X = A \circ B \iff x_{ij} = \bigvee_{\ell=1}^{k} a_{i\ell} \land b_{\ell j}$, so that 1s and 0s are interpreted as True and False, addition corresponds to logical disjunction ($\lor$) and multiplication to conjunction ($\land$). 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)):

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

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

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

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

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

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

$$X \approx \begin{bmatrix} 1.36 & 0.69 & 0.50 \\ 1.05 & 1.02 & 0.80 \end{bmatrix}$$

of $X$ suggests that symptom 2 presents with lower intensity in both $\alpha$ and $\beta$, 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 $x_{ij} = \bigvee_{k=1}^{r} a_{ik} \land b_{kj}$ for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. This is equivalent to $x_{ij} = \min\{1, \sum_{k=1}^{r} a_{ik} b_{kj}\}$ 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 $X = \bigvee_{k=1}^{r} a_{k} b_{k}^{T}$ for $a_{k} \in \{0, 1\}^{n}$, $b_{k} \in \{0, 1\}^{m}$. 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 as 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 $a \in \{0, 1\}^{n}$ is given, then a vector $b \in \{0, 1\}^{m}$ that minimizes the distance between $X$ and $ab^{T}$ can be computed in $O(nm)$ time. Since the introduction of Proximus, much research focused on computing efficient and accurate 1-BMF. 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 $\|X - A \circ B\|_{F}$, 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 $\ell_{1}$ norm coincide and we can expand the objective function

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

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

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 $\ell_{1}$ norm coincide and we can expand the objective function

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

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 \( y_{ij} \) for the product of \( a_{ij} \) and \( b_{ij} \) \((i, j) \in E\), 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 exact model presented in the previous section relies on 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

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

subject to

\[
y_{ij} \leq z_{ij} \leq \sum_{\ell=1}^{k} y_{i\ell j}, \quad (i, \ell, j) \in E, \quad (3)
\]

\[
y_{ij} \in MC(a_{ij}, b_{ij}), \quad (i, j) \in F, \quad (4)
\]

\[
a_{ij}, b_{ij} \in \{0, 1\}, \quad z_{ij} \leq 1, \quad (i, j) \in F. \quad (5)
\]

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 \( y_{ij} = z_{ij} \) for all \( i, j \) and the 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_{\text{exact}}’s LP relaxation (LP_{\text{exact}}) only provides a trivial bound.

Lemma 1. For \( k > 1 \), LP_{\text{exact}} has optimal objective value 0 which is attained by at least \( \binom{k-1}{k-2} \) 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 \( \sum_{\ell=1}^{k} a_{i\ell j} \geq \sum_{\ell=2}^{k} a_{i\ell j} \) for all \( \ell_1 < \ell_2 \).

Note that constraint (7) implies \( \frac{1}{k} \sum_{\ell=1}^{k} y_{ij} \leq z_{ij} \leq \sum_{\ell=1}^{k} y_{ij} \) as a lower and upper bound on each variable \( z_{ij} \). Hence, the objective function may be approximated by

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

where \( \rho \) is a parameter of the formulation. By setting \( \rho = \frac{1}{k} \) we underestimate the original objective, while setting \( \rho = \frac{1}{k} \) 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

\[
\mathcal{R} = \{ab^\top : a \in \{0, 1\}^n, b \in \{0, 1\}^m, a, b \neq 0\}.
\]

The size of \( \mathcal{R} \) is \( |\mathcal{R}| = (2^n - 1)(2^m - 1) \) as any pair of binary vectors \( a, b \neq 0 \) leads to a unique rank-1 matrix \( Y = ab^\top \) with \( Y_{ij} = 1 \) for \( \{(i, j) : a_i = 1, b_j = 1\} \). Define a binary decision variable \( q_{\ell} \) to denote if the \( \ell \)-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 \( q \in \{0, 1\}^{1|\mathcal{R}|} \) be a vector that has a component \( q_{\ell} \) 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 \( \ell \)-th rank-1 binary matrix in \( \mathcal{R} \) is 1, \( M_{(i,j),\ell} = 0 \) otherwise. We split \( M \) horizontally into two matrices \( M_0 \) and \( M_1 \), so that rows of \( M \) corresponding to a positive entry of the given matrix \( X \) are in \( M_1 \) and the rest of rows of \( M \) in \( M_0 \).

\[
M = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix} \quad \text{where} \quad M_0 \in \{0, 1\}^{(nm - |E|) \times |\mathcal{R}|}, \quad M_1 \in \{0, 1\}^{1 \times |\mathcal{R}|}.
\]

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

\[
(MIP_{\text{exact}}) \quad \zeta_{MIP} = \min 1^\top \xi + 1^\top \pi \quad (8)
\]

subject to

\[
M_0 q + \xi \geq 1 \quad (9)
\]

\[
1^\top q \leq k \quad (10)
\]

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

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

Constraint (11) ensures that at most \( k \) rank-1 matrices are active in a factorisation. Variables \( \xi_{ij} \) 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 \( \pi_{ij} \) 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 MIP_{\text{exact}} is that 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 MIP_{\text{exact}} (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 \( 0 < \zeta_{MIP} \).

For the definition of isolation number and the proof of Lemma 2, see Appendix 5.2. Similarly to the polynomial size exact model IP_{\text{exact}} in the previous section, we consider a
The CG procedure is initialised by explicitly solving a lower or upper bound on MIP\(_\text{exact}\). We denote the LP relaxation of MIP\((\rho)\) by MLP\((\rho)\).

**Lemma 3.** For \(\rho = \frac{1}{2}\), the optimal objective values of the LP relaxations MIP\(_{\text{exact}}\) and MLP\(_{\frac{1}{2}}\) 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\(_{\text{exact}}\), MIP\(_{\text{exact}}\), MIP\((\rho)\) and their LP relaxations for \(k > 1\),

\[
z_{\text{MIP}}(\frac{k}{2}) \leq \z_{\text{LP}}(\frac{k}{2}) \leq \z_{\text{LP}}(1) \leq \z_{\text{MIP}}(1),
\]

(16)

0 = \z_{\text{LP}}(1) = \z_{\text{MLP}}(\frac{1}{2}) = \z_{\text{MLP}}(1) \leq \z_{\text{MIP}}(1).

(17)

Let \(p\) be the dual variable associated to constraints (9) and \(\mu\) be the dual variable to constraint (11). Then the dual of MLP\((\rho)\) is given by

\[
(\text{MDP}(\rho)) \quad z_{\text{MDP}}(\rho) = \max 1^T p - k\mu
\]

(18)

\[
s.t. \quad M_1 p - \mu 1 \leq \rho M_0^T 1,
\]

(19)

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

(20)

Due to the number of variables in the formulation, it is not practical to solve MIP\((\rho)\) or its LP relaxation MLP\((\rho)\) 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\((\rho)\). The next step is to identify a missing variable with a negative reduced cost to be added to this Restricted MLP\((\rho)\). 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\((\rho)\) is optimal for the complete formulation MLP\((\rho)\).

We use CG to solve MLP\((\rho)\) by considering a sequence \((t = 1, 2, \ldots)\) of Restricted MLP\((\rho)\)'s with constraint matrix \(M_1^{(t)}\) being a subset of columns of \(M\), where each column \(y \in \{0, 1\}^{nm}\) of \(M\) corresponds to a flattened rank-1 binary matrix \(ab^\top\) according to Equation (7). The constraint matrix of the first Restricted MLP\((\rho)\) may be left empty or can be warm started by identifying a few rank-1 matrices in \(R\), say from a heuristic solution. Upon successful solution of the \(t\)-th Restricted MLP\((\rho)\), we obtain a vector of dual variables \([p^*, \mu^*] \geq 0\) optimal for the \(t\)-th Restricted MLP\((\rho)\). To identify a missing column of \(M\) that has a negative reduced cost, we solve the following pricing problem (PP):

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

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

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

To apply the CG approach above to MLP\(_{\text{exact}}\) only a small modification needs to be made. The Restricted MLP\(_{\text{exact}}\) provides dual variables for constraints (10) which are used in the objective of PP for coefficients of \(y_{ij}\). Note however, that CG cannot be used to solve a modification of MLP\(_{\text{exact}}\) in which constraints (10) are replaced by exponentially many constraints \([M_0 q_\ell \leq \pi]\) for \(\ell \in \{1, \ldots, ℓ\}\). To solve the Restricted MLP\((\rho)\) 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\((\rho)\) and MLP\(_{\text{exact}}\) as follows. Let the objective value of any of the Restricted MLP\((\rho)\)'s be

\[
z_{\text{RMLP}} = 1^T \xi^* + \rho 1^T M_0^* q^* = 1^T p^* - k\mu^*.
\]

(21)

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

\[
z_{\text{MLP}}(\rho) \geq 1^T p^* - k\omega(p^*) = z_{\text{RMLP}} - k(\omega(p^*) - \mu^*).
\]

(22)

If we do not have the optimal solution to PP but have an upper bound \(\omega(p^*)\) on it, \(\omega(p^*)\) can be replaced by \(\omega(p^*)\) in equation (22) and the bound on MLP\((\rho)\) still holds. Furthermore, this lower bound on MLP\((\rho)\) provides a valid lower bound on MIP\((\rho)\). 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 priori (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)

\[(PP) \quad \omega(p^*) = \max_{a \in \{0,1\}^n, b \in \{0,1\}^m} a^\top H b \]  

for an $n \times m$ matrix with $h_{ij} = p_{ij}^* \in [0, 1]$ for $(i, j) \in E$ and $h_{ij} = -p$ 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($\rho$). Most heuristic algorithms that are available for BBQP build on the idea that the optimal $a \in \{0,1\}^n$ with respect to a fixed $b \in \{0,1\}^m$ can be computed in $O(nm)$ time and this procedure can be iterated by alternatingly fixing $a$ and $b$. (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 $a$ and $b$ 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 $\sum_{j=1}^m \max(0, \sum_{\ell=1}^m a_{i\ell} h_{\ell j}) < \sum_{j=1}^m \max(0, \sum_{\ell=1}^m a_{i\ell} h_{\ell j} + h_{ij})$ and 0 otherwise. In the second phase, $b_j$ is set to 1 if $(a^\top H)_{ij} > 0$, 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 $\gamma_i^- = \gamma_{i_2}^+$, 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 $\gamma_i^+$ 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 $a$ and then $b$. Changing the order of fixing $a$ and $b$ 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 $a$ with respect to fixed $b$. This idea then can be used to fix $a$ and $b$ 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).

| Dataset | n | m | p | q | r |
|---------|---|---|---|---|---|
| zoo     | 101 | 339 | 155 | 242 | 148 |
| tumor   | 17 | 24 | 38 | 22 | 44 |
| hepatic | 22 | 44 | 94 | 105 | 32 |
| lymp    | 105 | 226 | 148 | 226 | 105 |
| audio   | 434 | 434 | 434 | 434 | 434 |
| aph     | 44.3 | 24.3 | 47.2 | 34.4 | 29.0 |
| votes   | 11.3 | 8.0 | 47.3 | 11.3 | 8.0 |

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, $\omega_{\text{heur}}(p^*) > \mu^*$, then the heuristic column is added to the next Restricted MLP($\rho$). If at some iteration, the best heuristic column does not have negative reduced cost, CPLEX is used to solve PP for 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($\rho$).

| 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)
Table 3: Primal objective values of MLP(1), MLP(1/k), MIP(1) after 20 mins of CG

| k   | zoo   | tumor | hepat | heart | lymph | audio | apb | votes |
|-----|-------|-------|-------|-------|-------|-------|-----|-------|
| 2   | MLP(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(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(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) |

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 $\omega(p^*)$ which heuristic solutions do not provide. While we observe a tailing off effect ("ubbecke 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($\rho$) 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(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(1/k) with heuristic pricing using a time limit of 20 mins,
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($\rho$) is solved much faster than MIP exact and that setting $\rho$ 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 Gonçalves 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. 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($\rho$) for $\rho$ 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.

### 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.

---

1 Appendix is available at arxiv.org/abs/2011.04457.

|       | zoo | tumor | hepat | heart | lymp | audio | apb | votes |
|-------|-----|-------|-------|-------|------|-------|-----|-------|
| 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 | 2926  |
| pymf  | 276 | 1472  | 1418  | 1241  | 1228 | 1510  | 776 | 2926  |
| ASSO  | 367 | 1465  | 1724  | 1251  | 1352 | 1505  | 776 | 2926  |
| NMF   | 291 | 1626  | 1596  | 1254  | 1366 | 2253  | 809 | 3069  |
| MEBF  | 348 | 1487  | 1599  | 1289  | 1401 | 1779  | 812 | 3268  |

**Table 4: Factorisation errors in $\|\cdot\|_F^2$ for eight methods for $k$-BMF**
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 2020-03-11.

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 11-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.; Graama, 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/~pmmiettin/bmf_tutorial/tutorial.pdf. Last accessed on 2021-03-11.

Miettinen, P.; Mielikäinen, T.; Gionis, A.; Das, G.; and Maniila, 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 Maniila, 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)](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](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](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.
Appendix

5.1 The strength of the LP relaxation of $IP_{\text{exact}}$

Lemma 1. For $k > 1$, the LP relaxation of $IP_{\text{exact}}$ ($LP_{\text{exact}}$) has optimal objective value 0 which is attained by at least $\binom{k}{2}$ solutions.

Proof. Observe that the objective function of LP$_{\text{exact}}$ satisfies $0 \leq \sum_{(i,j) \in E} (1 - z_{ij}) + \sum_{(i,j) \notin E} z_{ij}$ as constraints (3), the McCormick envelopes and $a_{i\ell}, b_{j\ell} \in [0, 1]$ imply $z_{ij} \in [0, 1]$. Let us construct a feasible solution to LP$_{\text{exact}}$ which attains this bound. For $0 \leq \sum_{i \in [n]} \sum_{\ell \in [k]} a_{i\ell} \leq \frac{k}{2}$ and $b_{j\ell} \leq \frac{k}{2}$ for all $i \in [n], j \in [m], \ell \in [k]$. The McCormick envelopes then are equivalent to $MC(\frac{k}{2}, \frac{k}{2}) = \{y \in \mathbb{R} : \frac{k}{2} + \frac{k}{2} - 1 \leq y, y \leq \frac{k}{2}, y \leq \frac{k}{2}, 0 \leq y \} = [0, \frac{k}{2}]$ hence we may choose the value of $y_{i\ell j} \in MC(\frac{k}{2}, \frac{k}{2})$ depending on the objective coefficient of indices $(i, j)$. For $(i, j) \in E$ the objective function is maximising $z_{ij}$ hence we set $y_{i\ell j} = \frac{k}{2}$ so that the upper bound $\sum_{E} y_{i\ell j} \leq z_{ij}$ becomes greater than equal to 1 and $z_{ij}$ can take value 1. For $(i, j) \notin E$ the objective function is minimising $z_{ij}$ hence we set $y_{i\ell j} = 0$ so that the lower bounds $y_{i\ell j} \leq z_{ij}$ evaluate to 0 and $z_{ij}$ can take value 0. Therefore, the following setting of the variables shows the lower bound of 0 on the objective function is attained,

\begin{align*}
    a_{i\ell} &= \frac{k}{2}, \; i \in [n], \ell \in [k]; \\
    b_{j\ell} &= \frac{k}{2}, \; \ell \in [k], j \in [m]; \\
    y_{i\ell j} &= \frac{k}{2}, \; (i, j) \in E, \ell \in [k]; \\
    y_{i\ell j} &= 0, \; (i, j) \notin E, \ell \in [k]; \\
    z_{ij} &= 1, \; (i, j) \in E; \\
    z_{ij} &= 0, \; (i, j) \notin E.
\end{align*}

Furthermore, for all $(i, j) \in E$ it is enough to set $y_{i\ell_{1} j} = y_{i\ell_{2} j} = \frac{k}{2}$ for only two indices $\ell_{1}, \ell_{2} \in [k]$ since this already achieves the upper bound $z_{ij} \leq 1 = \sum_{E} y_{i\ell j}$. Hence, there is at least $\binom{k}{2}$ different solutions of LP$_{\text{exact}}$ with objective value 0. ☐

5.2 The strength of the LP relaxation of $MIP_{\text{exact}}$

For our proof we will need a definition from the theory of binary matrices.

Definition 2. ([Monson, Pullman, and Rees]1995 Section 2.3) Let $X$ be a binary matrix. A set $S \subseteq E = \{(i, j) : x_{ij} = 1\}$ is said to be an isolated set of ones if whenever $(i_1, j_1), (i_2, j_2)$ are two distinct members of $S$ then

1. $i_1 \neq i_2, j_1 \neq j_2$ and
2. $x_{i_1 j_2} x_{i_2 j_1} = 0$.

The size of the largest cardinality isolated set of ones in $X$ is denoted by $i(X)$ and is called the isolation number of $X$.

Observe that requirement (1) implies that an isolated set of ones can contain the index corresponding to at most one entry in each column and row of $X$. Hence $i(X) \leq \min(n, m)$. Requirement (2) implies that if $(i_1, j_1), (i_2, j_2)$ are members of an isolated set of ones then at least one of the entries $x_{i_1 j_2}, x_{i_2 j_1}$ is zero, hence members of an isolated set of ones cannot be contained in a common rank-1 submatrix of $X$. Therefore if the largest cardinality isolated set of ones has $i(X)$ elements, to cover all 1s of $X$ we need at least $i(X)$ many rank-1 binary matrices in a factorisation of $X$, so $i(X)$ provides a lower bound on the Boolean rank of $X$.

Lemma 2. Let $X$ be a binary matrix with isolation number $i(X)$. Then for rank-$k$ binary matrix factorisation of $X$ with $k < i(X)$, the LP-relaxation of $MIP_{\text{exact}}$ has non-zero optimal objective value.

Proof. Let $k$ be a fixed positive integer and let $X$ be a binary matrix with isolation number $i(X) > k$. For a contradiction, assume that $MIP_{\text{exact}}$ has objective value zero, $\zeta_{\text{MIP}} = 0$.

(1) Now, if $\zeta_{\text{MLP}} = 0$ we must have $M_{\ell} q = 0 \pi$ in constraint (10) which implies that none of the rank-1 binary matrices with $q_{\ell} > 0$ cover zero entries of $X$. In other words, all the rank-1 binary matrices active are submatrices of $X$.

(2) Now, if $\zeta_{\text{MLP}} = 0$ we must also have $\xi = 0$ which implies $M_{\ell} q \geq 1$ in constraint (2) for some $q$ which may be fractional but satisfies $1 \geq q$. Let $S := \{(i_1, j_1), \ldots, (i_r, j_r), \ldots, (i_s, j_s)\} \subseteq E$ be an isolated set of ones in $X$ of cardinality $i(X) = |S|$. Since members of $S$ cannot be contained in a common rank-1 submatrix of $X$, all columns $M_{\ell}$ corresponding to rank-1 binary submatrices of $X$ for entries $(i_r, j_r) \in S$ satisfy

\[ M_{(i_r, j_r), \ell} = 1 \Rightarrow M_{(i, j), \ell} = 0 \forall (i, j) \neq (i_r, j_r), (i, j) \in S. \]

Therefore, we can partition the active rank-1 matrices into $i(X) + 1$ groups $G_r$,

\[ G_r = \{\ell : q_{\ell} > 0 \land M_{(i_r, j_r), \ell} = 1\} \]

for $r = 1, \ldots, i(X)$;

\[ G_{i(X)+1} = \{\ell : q_{\ell} > 0 \land M_{(i_r, j_r), \ell} = 0 \forall (i_r, j_r) \in S\}. \]

While $G_{i(X)+1}$ may be empty, $G_r$‘s for $r = 1, \ldots, i(X)$ are not empty because we know that for all $(i, j) \in E$ we have $\sum_{E} M_{(i, j), \ell} q_{\ell} \geq 1$ and $S \subseteq E$. Hence for all $(i_r, j_r) \in S$ we have $\sum_{E} M_{(i_r, j_r), \ell} q_{\ell} \geq 1$ which implies the contradiction $1 \geq 0 \land i(X) < k$ and therefore $\zeta_{\text{MIP}} > 0$.

Could we replace the condition $k < i(X)$ in Lemma 2 by a requirement that $k$ has to be smaller than the Boolean rank of $X$? The following example shows that we cannot.

Example 2. Let $X = J_4 - I_4$, where $J_4$ is the $4 \times 4$ matrix of all 1s and $I_4$ is the $4 \times 4$ identity matrix. One can verify that the Boolean rank of $X$ is 4 and its isolation number is 3.

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

For $k = 3$, the optimal objective value of $MIP_{\text{exact}}$ is 0 which is attained by a fractional solution in which the following 6 rank-1 binary matrices are active.

\[ q_1 = \frac{1}{2}, \quad q_2 = \frac{1}{2}, \quad q_3 = \frac{1}{2} \]

\begin{tabular}{cccc}
0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0
\end{tabular}
5.3 The relation between the LP relaxations of MIP\textsubscript{exact} and MIP\left(\frac{1}{k}\right)

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

Proof. It is enough to observe that since MLP\textsubscript{exact} is a minimisation problem, \( \pi \) takes the minimal optimal value \( \frac{1}{k} M \alpha q \) in MLP\textsubscript{exact} due to constraint (10) which equals the second term in the objective (14) of MLP\left(\frac{1}{k}\right). \( \square \)

5.4 Column generation applied to the LP relaxation of the strong formulation of MIP\textsubscript{exact}

We obtain a modification of MIP\textsubscript{exact} which we call the “strong formulation” by replacing constraints (10) by exponentially many constraints. The following is the LP relaxation of the strong formulation of MIP\textsubscript{exact}.

\[
\begin{align*}
\text{(MLP\textsubscript{exact} strong)}: \quad & \min 1^\top \xi + 1^\top \pi \\
\text{s.t.} & \quad M \alpha q + \xi \geq 1 \\
& \quad (M_0) q \leq \pi, \quad \ell \in [(2^n - 1)(2^m - 1)] \\
& \quad 1^\top q \leq k \\
& \quad \xi \geq 0, \quad \pi \in [0, 1]^{nm - |E|}, \quad q \in [0, 1]^{|\mathcal{R}|}.
\end{align*}
\]

Lemma 4. Applying the CG approach to MLP\textsubscript{exact strong} cannot be used to generate sensible columns.

Proof. Let us try applying column generation to solve MLP\textsubscript{exact strong} and add a column of all 1s as our first column \( q_1 \). Then at the 1st iteration, for \( q_1 = 1 \) the objective value of the Restricted MLP is \( \zeta_{\text{MLP}}^{(1)} = 0 + (nm - |E|) \) for solution vector \( [\xi^{(1)}, \pi^{(1)}, q^{(1)}] = [0, 1, 1] \). Adding the same column of all 1s at the next iteration and setting \( q_2 = \left[ \frac{1}{2}, \frac{1}{2} \right] \), allows us to keep \( \xi^{(2)} = 0 \) but set \( \pi^{(2)} = \frac{1}{2} \) to get \( \zeta_{\text{MLP}}^{(2)} = 0 + \frac{1}{2} (nm - |E|) \). Therefore continuing adding the same column of all 1s, after \( t \) iterations we have \( \zeta_{\text{MLP}}^{(t)} = 0 + \frac{1}{t} (nm - |E|) \) for solution vector \( [\xi^{(t)}, \pi^{(t)}, q^{(t)}] = [0, \frac{1}{t}, \frac{1}{t}] \). Therefore for \( t \to \infty \) we have \( \zeta_{\text{MLP}} \to 0 \) and we have not generated any other columns but the all 1s. \( \square \)

5.5 The greedy algorithm for bipartite binary quadratic optimisation

For a given \( n \times m \) coefficient matrix \( H \), we aim to find \( a \in \{0, 1\}^n \) and \( b \in \{0, 1\}^m \) so that \( a^\top H b \) is maximised. Let \( \gamma_i^+ \) be the sum of the positive entries of \( H \) for each row \( i \in [n] \), \( \gamma_i^+ = \sum_{j=1}^m \max(0, h_{ij}) \). Reorder the rows of \( H \) according to decreasing values of \( \gamma_i^+ \). Algorithm 1 is a greedy heuristic of (Karapetyan and Punnen 2013) which provides the optimal solution to the bipartite binary quadratic problem if \( \min(n, m) \leq 2 \) and has an approximation ratio of \( 1/(\min(n, m) - 1) \) otherwise.

Algorithm 1: Greedy Algorithm

Phase I. Order \( i \in [n] \) so that \( \gamma_i^+ \geq \gamma_i^{+1} \).

Set \( \alpha = 0_n, s = 0_m \).

for \( i \in [n] \) do

\[ f_0 = \sum_{j=1}^m \max(0, s_j) \]

\[ f_1 = \sum_{j=1}^m \max(0, s_j + h_{ij}) \]

if \( f_0 < f_1 \) then

Set \( a_i = 1, s = s + h_i \)

end

end

Phase II. Set \( b = 0_m \).

for \( j \in [m] \) do

if \( (a^\top H)_j > 0 \) then

Set \( b_j = 1 \)

end

end

5.6 Rank-\( k \) greedy heuristic

For a given \( X \in \{0, 1\}^{n \times m} \) and \( k \in \mathbb{Z}_+ \), according to Equation (1) we may write \( \min \|X - Z\|_F^2 \) as \( |E| - \max \sum_{i=1}^n \sum_{j=1}^m \max(0, z_{ij}) \) where \( h_{ij} \) are entries of \( H := 2X - 1 \). We propose the heuristic in Algorithm 2 to compute \( k \)-BMF by sequentially computing \( k \) rank-1 binary matrices using the greedy algorithm of (Karapetyan and Punnen 2013) given in Algorithm 1.

We remark that this rank-\( k \) greedy algorithm can be used to obtain a heuristic solution to \( k \)-BMF under standard arithmetic as well: simply modify the last line to

\[ H[ab^\top] \rightleftharpoons 1 \rightleftharpoons -K \]

for a large enough positive number \( K \) (say \( K := \sum x_{ij} \)) so that each entry of \( X \) is covered at most once.

Algorithm 2: Greedy algorithm for rank-\( k \) binary matrix factorisation

Input: \( X \in \{0, 1\}^{n \times m}, k \in \mathbb{Z}_+ \).

Set \( H := 2X - 1 \).

for \( \ell \in [k] \) do

\[ a, b = \text{Greedy}(H) \quad // \text{Compute a rank-1 binary matrix via the greedy algorithm} \]

\[ a, b = \text{Alt}(H, a, b) \quad // \text{Improve greedy solution with the alternating heuristic} \]

\[ A_{\ell} := a \]

\[ B_{\ell} := b^\top \]

\[ H[ab^\top] \rightleftharpoons 1 \rightleftharpoons 0 \quad // \text{Set entries of } H \text{ to zero that are covered} \]

end

Output: \( A \in \{0, 1\}^{n \times k}, B \in \{0, 1\}^{k \times m} \).
5.7 Formulations evaluated in Table 2

The formulation KGH17 of (Kovacs, Gunluk, and Hauser 2017) with new objective function (6) for $p = 1$ that was evaluated to get results in column KGH17 of Table 2 reads as

$$\min_{a,b,z} \sum_{(i,j) \in E} (1 - z_{ij}) + \sum_{(i,j) \notin E} \sum_{t=1}^{k} y_{it}\beta_{tj}$$

s.t. $z_{ij} \leq \sum_{t=1}^{k} y_{it}\beta_{tj}, \quad (i, j) \in E,$

$$y_{it}\beta_{tj} \in MC(a_{\ell}, b_{\ell}), \quad i \in [n], \ell \in [k], j \in [m],$$

$$a_{it}, b_{tj} \in \{0, 1\}, \quad (i, j) \in E,$$

$$z_{ij} \in [0, 1], \quad (i, j) \in [k], j \in [m],$$

with $MC(a, b) = \{y \in \mathbb{R} : a + b - 1 \leq y, y \leq a, y, b \leq 0, y \leq a, y, b \leq 0\}$ denoting the McCormick envelopes as defined in Section 2. In the exponential formulation of (Lu, Vaidya, and Atluri 2008) all possible non-zero binary row vectors $B_t \in \{0, 1\}^{1 \times m} (t \in [2^m - 1])$ for factor matrix $B \in \{0, 1\}^{n \times m}$ are explicitly enumerated and treated as fixed input parameters to the formulation. The formulation LVA08 with objective function (6) for $p = 1$ that was evaluated to get results in column LVA08 of Table 2 reads as

$$\min_{\alpha, \delta, \beta} \sum_{(i,j) \in E} (1 - z_{ij}) + \sum_{(i,j) \notin E} \sum_{t=1}^{2^m-1} \alpha_{it}\beta_{tj}$$

s.t. $z_{ij} \leq \sum_{t=1}^{2^m-1} \alpha_{it}\beta_{tj}, \quad (i, j) \in E,$

$$\alpha_{it}, \beta_{tj} \in \{0, 1\}, \quad (i, j) \in E,$$

$$\alpha_{it} \leq \delta_t, \quad i \in [n], t \in [2^m - 1],$$

$$\sum_{t=1}^{2^m-1} \delta_t \leq k,$$

$$z_{ij} \in [0, 1], \quad (i, j) \in E,$$

$$\alpha_{it}, \delta_t \in \{0, 1\}, \quad i \in [n], t \in [2^m - 1].$$

5.8 Datasets

In general if a dataset has a categorical feature $C$ with $N$ discrete options $v_j, (j \in [N])$, we convert feature $C$ into $N$ binary features $B_j (j \in [N])$ so that if the $i$-th sample takes value $v_j$ for $C$ that is $(C)_i = v_j$, then we have value $(B_j)_i = 1$ and $(B_j)_i = 0$ for all $\ell \neq j \in [N]$. This technique of binarisation of categorical columns has been applied in (Kovacs, Gunluk, and Hauser 2017) and (Barahona and Goncalves 2019). The following datasets were used:

- The Zoo dataset (zoo) (Forsyth 1990) describes 101 animals with 16 characteristic features. All but one feature is binary. The categorical column which records the number of legs an animal has, is converted into two new binary columns indicating if the number of legs is less than or equal or greater than four. The size of the resulting fully binary dataset is 101 × 17.

- The Primary Tumor dataset (tumor) (Kononenko and Cestnik 1988a) contains observations on 17 tumour features detected in 339 patients. The features are represented by 13 binary variables and 4 categorical variables with discrete options. The 4 categorical variables are converted into 11 binary variables representing each discrete option. Two missing values in the binary columns are set to value 0. The final dimension of the dataset is 339 × 24.

- The Hepatitis dataset (hepat) (Gong 1988) consists of 155 samples of medical data of patients with hepatitis. The 19 features of the dataset can be used to predict whether a patient with hepatitis will live or die. 6 of the 19 features take numerical values and are converted into 12 binary features corresponding to options: less than or equal to the median value, and greater than the median value. The column that stores the sex of patients is converted into two binary columns corresponding to labels man and female. The remaining 12 columns take values yes and no and are converted into 24 binary columns. The raw dataset contains 167 missing values, and according to the above binarisation if a sample has missing entry for an original feature it will have 0’s in both columns binarised from that original feature. The final binary dataset has dimension $155 \times 38$.

- The SPECT Heart dataset (heart) (Cios and Kurgan 2001) describes cardiac Single Proton Emission Computed Tomography images of 267 patients by 22 binary feature patterns. 25 patients’ images contain none of the features and are dropped from the dataset, hence the final dimension of the dataset is $242 \times 22$.

- The Lymphography dataset (lymp) (Kononenko and Cestnik 1988b) contains data about lymphography examination of 148 patients. 8 features take categorical values and are expanded into 33 binary features representing each categorical value. One column is numerical and we convert it into two binary columns corresponding to options: less than or equal to median value, and larger than median value. The final binary dataset has dimension $148 \times 44$.

- The Audiology Standardized dataset (audio) (Quinlan 1992) contains data about clinical audiology records on 226 patients. The 69 features include patient-reported symptoms, patient history information, and the results of routine tests which are needed for the evaluation and diagnosis of hearing disorders. 9 features that are categorical valued are binarised into 34 new binary variables indicating if a discrete option is selected. The final dimension of the dataset is $226 \times 94$.

- The Amazon Political Books dataset (books) (Krebs 2008) contains binary data about 105 US politics books sold by Amazon.com. Columns correspond to books and rows represent frequent co-purchasing of books by the same buyers. The dataset has dimension $105 \times 105$.

- The 1984 United States Congressional Voting Records dataset (votes) (Schlimmer 1987) includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The 16 categorical variables taking values of “voted for”, “voted against” or “did not vote”, are converted into 32 binary variables. One congressman did not vote for any of the bills and its corresponding row of zero is dropped. The final binary dataset has dimension $434 \times 32$. 

5.9 Data preprocessing

In practice, the input matrix $X \in \{0, 1\}^{n \times m}$ may contain zero rows or columns. Deleting a zero row (column) leads to an equivalent problem whose solution $A$ and $B$ can easily be translated to a solution of the original dimension. In addition, if a row (column) of $X$ is repeated $\alpha_i$ ($\beta_j$) times, it is sufficient to keep only one copy of it, solve the reduced problem and reinsert the relevant copies in the corresponding place. To ensure that the objective function of the reduced problem corresponds to the factorisation error of the original problem, the variable corresponding to the representative row (column) in the reduced problem is multiplied by $\alpha_i$ ($\beta_j$).

5.10 Comparison Methods

The following methods were evaluated for the comparison in Table 4:

- The code for our methods can be found at https://github.com/kovacsrekaagnes/rank_k_BMF.
- For the alternating iterative local search algorithm of (Barahona and Goncalves 2019) (ASSO++) we obtained the code from the author’s github page. The code implements two variants of the algorithm and we report the smaller error solution from two variants of it.
- The greedy algorithm ($k$-greedy) detailed in Appendix 5.6 is evaluated with nine different orderings and the best result is chosen.
- For the method of (Zhang et al. 2007), we used an implementation in the github pymf package by Chris Schinnerl and we ran it for 10000 iterations.
- We evaluated the heuristic method ASSO (Miettinen et al. 2006) which depends on a parameter and we report the best results across nine parameter settings ($\tau \in \{0.1, 0.2, \ldots, 0.9\}$). The code was obtained from the webpage of the author.
- In addition, we computed rank-$k$ non-negative matrix factorisation (NMF) and binarise it by a threshold of 0.5: after an NMF is obtained, values greater than 0.5 are set to 1, otherwise to 0. For the computation of NMF we used sklearn.decomposition module in Python.
- For the MEBF method (Wan et al. 2020) we used the code from the author’s github page. The raw code downloaded contained a bug and did not produce a solution for some instances while for others it produced factorisations whose error in $\| \cdot \|_F$ increased with the factorisation rank $k$. We fixed the code and the results shown in Table 4 correspond to the lowest error for each instance selected across 9 parameter settings $t \in \{0.1, \ldots, 0.9\}$. 