Binary Matrix Factorisation and Completion via Integer Programming

Oktay Günlük*
Cornell University, ong5@cornell.edu

Raphael A. Hauser, Réka Á. Kovács
University of Oxford, The Alan Turing Institute, hauser@maths.ox.ac.uk, reka.kovacs@maths.ox.ac.uk

Binary matrix factorisation is an essential tool for identifying discrete patterns in binary data. In this paper we consider the rank-k binary matrix factorisation problem (k-BMF) under Boolean arithmetic: we are given an $n \times m$ binary matrix $\mathbf{X}$ with possibly missing entries and need to find two binary matrices $\mathbf{A}$ and $\mathbf{B}$ of dimension $n \times k$ and $k \times m$ respectively, which minimise the distance between $\mathbf{X}$ and the Boolean product of $\mathbf{A}$ and $\mathbf{B}$ in the squared Frobenius distance. We present a compact and two exponential size integer programs (IPs) for k-BMF and show that the compact IP has a weak LP relaxation, while the exponential size IPs have a stronger equivalent LP relaxation. We introduce a new objective function, which differs from the traditional squared Frobenius objective in attributing a weight to zero entries of the input matrix that is proportional to the number of times the zero is erroneously covered in a rank-k factorisation. For one of the exponential size IPs we describe a computational approach based on column generation. Experimental results on synthetic and real word datasets suggest that our integer programming approach is competitive against available methods for k-BMF and provides accurate low-error factorisations.

Key words: binary matrix factorisation, binary matrix completion, column generation, integer programming
MSC2000 subject classification: 90C10
OR/MS subject classification: Integer Programming

---

1. Introduction. For a given binary matrix $\mathbf{X} \in \{0,1\}^{n \times m}$ and a fixed positive integer $k \ll \min\{n, m\}$, the rank-k binary matrix factorisation problem (k-BMF) is concerned with finding two matrices $\mathbf{A} \in \{0,1\}^{n \times k}$, $\mathbf{B} \in \{0,1\}^{k \times m}$ such that the product of $\mathbf{A}$ and $\mathbf{B}$ is a binary matrix closest to $\mathbf{X}$ in the squared Frobenius norm. Note that k-BMF requires to compute $\mathbf{A}$ and $\mathbf{B}$ of fixed inner dimension k and the error between the low-rank approximation (product of $\mathbf{A}$ and $\mathbf{B}$) and $\mathbf{X}$ is minimised.

One can define different variants of k-BMF depending on the underlying arithmetic used when computing the product of binary matrices. In this paper we focus on Boolean arithmetic where the product of the binary matrices $\mathbf{A}$ and $\mathbf{B}$ is computed by (i) interpreting 0s as false and 1s as true, and (ii) using logical disjunction ($\vee$) in place of addition and logical conjunction ($\wedge$) in place of multiplication. Observe that Boolean multiplication ($\wedge$) coincides with standard multiplication on

* This work was partially funded by the Office of Naval Research under grant N00014-21-1-2575.

binary input, hence we adopt the notation $a\,b$ in place of $a \wedge b$ in the rest of the paper. We therefore compute the Boolean matrix product of $\mathbf{A}$ and $\mathbf{B}$ as:

$$\mathbf{Z} = \mathbf{A} \circ \mathbf{B} \iff z_{ij} = \bigvee_{\ell} (a_{i\ell}\, b_{\ell j}).$$

Note that Boolean matrix multiplication can be equivalently written as $z_{ij} = \min\{1, \sum_{\ell} a_{i\ell} b_{\ell j}\}$ using standard arithmetic summation. The problem then becomes computing matrices $\mathbf{A}$ and $\mathbf{B}$ whose Boolean product $\mathbf{Z}$ best approximates the input matrix $\mathbf{X}$.

Our motivation for this study comes from data science applications where rows of the matrix $\mathbf{X}$ correspond to data points and columns correspond to categorical features that are either present or absent. Correspondingly, such features are recorded as 1 or 0 rather than being quantified as an intensity level. Categorical data is often collected in questionnaires that contain questions that must be answered with "yes" or "no", with applications in psychometric testing, in risk rating systems in the medical, finance and engineering fields, in marketing surveys, and in processes designed to estimate economic sentiment indicators. A typical problem is to associate data vectors of categorical features with probabilities such as the risk of developing a specific health condition, the probability of default or of other relevant outcomes, and to learn this association by optimizing a parametric model such as logistic regression. In this context low-rank binary matrix factorisation is able to identify hidden structure in the form of patterns of original features that tend to co-occur. Framing the problem in terms of the presence or absence of this smaller number of derived features lowers the dimensionality of the data vectors and reduces the number of parameters required to calibrate a logistic regression model, thereby increasing the ratio of observations to model parameters. This offers the potential of reducing overfitting, particularly in applications where the data points available for learning are limited in number.

As a simple illustration, consider the example of a data matrix $\mathbf{X}$ inspired by [39], whereby rows correspond to patients and columns to symptoms or trauma indicators, with $x_{ij} = 1$ indicating that patient i presents symptom j:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad \mathbf{X} = \mathbf{A} \circ \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \tag{1}$$

In this example the factorization $\mathbf{A} \circ \mathbf{B}$ describes the data $\mathbf{X}$ exactly by use of only 2 derived features, each consisting of a condition described by the combination of original features given in the rows of $\mathbf{B}$, while the rows of $\mathbf{A}$ now identify which of the conditions is present in each patient. For example, patient 1 presents only the first condition, henceforth called α, while patient 2 presents both conditions α and β. If the observations are associated with an outcome, such as patients' death, then we can learn the outcome probability either by applying logistic regression to the original data $\mathbf{X}$, which requires 4 parameters, or to the derived data matrix $\mathbf{A}$, which requires only 3 parameters. The number of patients is the same in both cases, so the ratio of observations to model parameters has increased.

Using Boolean arithmetic allows each data point (rows of $\mathbf{X}$) to be expressed as the union of k possibly overlapping derived features. In contrast, if standard arithmetic is used, overlaps are not possible and each data point is the union of k disjoint derived features. This shows that k-BMF under standard arithmetic always has an error higher or equal to the k-BMF under Boolean arithmetic and in general one needs larger values of k to achieve accurate factorisations using standard arithmetic. For instance, for $\mathbf{X}$ in Equation (1) the following rank-2 factorisation is one of the four optimal solutions with error 1 to 2-BMF under standard arithmetic,

$$\mathbf{X} \approx \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In this factorisation of $\mathbf{X}$, each symptom is caused by exactly one condition, while in real life it is natural to assume that some observed symptoms, such as fever, can be caused by several underlying conditions. The transpose of this factorisation gives another optimal 2-BMF under standard arithmetic, and there are two more optimal factorisations. In general, we observe that BMF under standard arithmetic tends to have more degenerate solutions. Therefore, due to interpretability, smaller error and avoidance of degenerate solutions we chose to work under the Boolean arithmetic.

We note that it is also possible to use classical methods such as singular value decomposition (SVD) [16] or non-negative matrix factorisation (NMF) [30] to obtain low-rank approximations of $\mathbf{X}$ but the resulting factor matrices or their product would typically not be binary, unlike BMF [38]. To demonstrate this we next give the lowest Frobenius norm error rank-2 SVD and NMF approximations of matrix $\mathbf{X}$ in (1), respectively[1]:

$$\mathbf{X} \approx \begin{bmatrix} 0.78 & 0.71 \\ 1.10 & 0.00 \\ 0.78 & -0.71 \end{bmatrix} \begin{bmatrix} 0.78 & 1.10 & 0.78 \\ 0.71 & 0.00 & -0.71 \end{bmatrix}, \qquad \mathbf{X} \approx \begin{bmatrix} 1.05 & 0.00 \\ 0.81 & 0.74 \\ 0.10 & 1.05 \end{bmatrix} \begin{bmatrix} 1.05 & 0.81 & 0.10 \\ 0.00 & 0.74 & 1.05 \end{bmatrix}. \tag{2}$$

Note that neither of these rank-2 approximations provide a clear interpretation. The rank-2 NMF of $\mathbf{X}$ suggests that symptom 2 presents with lower intensity in both conditions α and β, an erroneous conclusion (caused by patient 2) that could not have been learned from data $\mathbf{X}$ which is of "on/off" type.

We note that in addition to the above mentioned applications, BMF-derived features of data have also been shown to be interpretable in biclustering gene expression datasets [52], role based access control [34, 35] and market basket data clustering [31].

1.1. Complexity and related work. The Boolean rank [40, 21] of a binary matrix $\mathbf{X} \in \{0,1\}^{n\times m}$ is defined to be the smallest integer $r = \operatorname{rank}_{\mathbb{B}}(\mathbf{X})$ for which there exist binary matrices $\mathbf{A} \in \{0,1\}^{n\times r}$ and $\mathbf{B} \in \{0,1\}^{r\times m}$ such that $\mathbf{X} = \mathbf{A} \circ \mathbf{B}$. In an equivalent definition, the Boolean rank of $\mathbf{X}$ is the minimum value of r for which it is possible to factor $\mathbf{X}$ into a Boolean combination of r rank-1 binary matrices

$$\mathbf{X} = \bigvee_{\ell=1}^{r} \boldsymbol{a}_\ell \boldsymbol{b}_\ell^\top$$

for $\boldsymbol{a}_\ell \in \{0,1\}^n, \boldsymbol{b}_\ell \in \{0,1\}^m$. Occasionally, the Boolean rank is also referred to as the rectangle cover number, and rank-1 binary matrices $\boldsymbol{a}_\ell \boldsymbol{b}_\ell^\top$ are called rectangle matrices [7, Section 4.10.1]. Much theoretical interest in the Boolean rank is driven by the fact that it provides a lower bound on the nonnegative rank, see for instance [12, 41]. The nonnegative rank of a nonnegative matrix $\mathbf{Y} \in \mathbb{R}_+^{n\times m}$ is the smallest integer $s = \operatorname{rank}_+(\mathbf{Y})$ for which there exist nonnegative matrices $\mathbf{W} \in \mathbb{R}_+^{n\times s}$ and $\mathbf{H} \in \mathbb{R}_+^{s\times m}$ such that $\mathbf{Y} = \mathbf{WH}$, under standard matrix multiplication. It can be readily checked that if $\mathbf{X}$ is the binary matrix which encodes the support of a nonnegative matrix $\mathbf{Y}$, then $\operatorname{rank}_{\mathbb{B}}(\mathbf{X}) \le \operatorname{rank}_+(\mathbf{Y})$. The Boolean rank does not have a clear relationship to the standard real rank. For a detailed treatment of the relationships between different ranks, see [39].

Computing an exact minimum factorisation of a binary matrix $\mathbf{X}$ and by that finding its Boolean rank is called the *exact*-BMF problem. Note that computing an optimal rank-k BMF for a fixed integer k is fundamentally different from exact-BMF. In exact-BMF the rank k is being minimised, while in k-BMF the distance between $\mathbf{X}$ and Boolean rank-k binary matrices which is being minimised. Furthermore, in exact-BMF, it suffices to consider rank-1 binary matrices that correspond to submatrices of $\mathbf{X}$, while in k-BMF any rank-1 binary matrix of the right dimension is a potential

[1] These factorisations were computed by functions `svd` from `numpy.linalg` and `non_negative_factorization` from `sklearn.decomposition`, respectively, and then rounded to two digits which increases the error slightly.

candidate. For instance, it can be readily checked that the optimal rank-1 BMF for the Boolean rank-2 matrix $\mathbf{X}$ shown in Equation (1) is the 3×3 matrix of all 1s, which does not correspond to a submatrix of $\mathbf{X}$. Furthermore, k-BMF is harder than exact-BMF in the sense that by solving logarithmically many k-BMF problems one can obtain an optimal solution to exact-BMF, but not the other way round. These differences between exact and k-BMF warrant for the search of specialised algorithms for k-BMF.

Interpreting $\mathbf{X} \in \{0,1\}^{n \times m}$ as the biadjacency matrix of a bipartite graph $G(\mathbf{X})$ with n vertices on the left and m vertices on the right, the problem of computing the Boolean rank of $\mathbf{X}$ is in one-to-one correspondence with finding a minimum edge covering of $G(\mathbf{X})$ by complete bipartite subgraphs (bicliques)[40]. Since the minimum biclique cover problem is NP-hard [42, Theorem 8.1],[14, Problem GT18], and hard to approximate [50, 4], computing the Boolean rank is hard as well. Finding an optimal rank-k binary factorisation of $\mathbf{X}$ under Boolean arithmetic has a graphic interpretation of minimizing the number of errors in an approximate covering of $G(\mathbf{X})$ by k bicliques which are allowed to overlap. In the rank-1 case the Boolean arithmetic coincides with standard arithmetic and 1-BMF is equivalent to computing a maximum weight biclique on the complete bipartite graph $K_{n,m}$ whose edges that are in $G(\mathbf{X})$ have weight 1 and others weight -1. The maximum edge biclique problem with edge weights in $\{-1, 1\}$ is NP-hard [15], hence even the computation of a rank-1 BMF is computationally challenging.

Due to the hardness results, the majority of methods developed for BMF rely on heuristics. The earliest heuristic for BMF, Proximus [28, 27], computes BMF under standard arithmetic using a recursive partitioning idea and computing 1-BMF at each step. Since Proximus, much research has focused on computing efficient and accurate methods for 1-BMF. [48] proposes an integer program (IP) for 1-BMF and several relaxations of it, one of which leads to a 2-approximation, while [49] provides a rounding based 2-approximation. In [3] an extension of the Proximus framework is explored which uses the formulations from [48] to compute 1-BMF at each step. k-BMF under Boolean arithmetic is explicitly introduced in [38, 39], along with a heuristic called ASSO, which is based on an association rule-mining approach. ASSO is further improved in [1] into an alternating iterative heuristics. Another approach based on an alternating style heuristic is explored in [52] to solve a non-linear unconstrained formulation of k-BMF with penalty terms in the objective for non-binary entries.

In [34, 35] a series of integer programs for k-BMF and exact-BMF are introduced. These IPs have exponentially many variables and constraints and require an explicit enumeration of the 2^m possible binary row vectors for factor matrix $\mathbf{B}$. To tackle the exponential explosion of rows considered, a heuristic row generation using association rule mining and subset enumeration is developed. Another exponential size integer program for exact-BMF is presented in [11]. To solve this exponential size IP the authors use either a precomputed enumeration of maximal rectangles (maximal submatrices of all 1s) or a branch and price method.

1.2. Our contributions. In this paper, we present a comprehensive study on integer programming methods for k-BMF. We examine three integer programs in detail: a compact formulation as introduced briefly in our previous short work [25], the exponential formulation of [34, 35] and another exponential formulation introduced in a preliminary conference version of this paper [26]. We prove several results about the strength of LP-relaxations of the three formulations and their relative comparison that did not appear in our preliminary publications. We note that all three integer programs for k-BMF considered in this paper, as well as any other element-wise models can be naturally applied in the context of rank-k binary matrix completion by simply setting the objective coefficients corresponding to missing entries to 0.

Our compact IP uses McCormick envelopes [37] to linearise the quadratic terms coming from the matrix product leading to polynomially many variables and constraints. We prove that for $k > 2$

the LP relaxation of the compact IP has several fractional vertices with objective value 0, hence provides a weak dual bound. In addition, we argue that our compact IP suffers from permutation symmetry.

Our exponential formulation (introduced in [26]) overcomes several of these limitations and of other previous approaches. In particular, it does not suffer from permutation symmetry and it does not rely on heuristically guided pattern mining. Moreover, it has a stronger LP relaxation than the compact IP. On the other hand, this 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 [34, 35]. Our exponential formulation for k-BMF is related to the exponential IP of [11] for exact-BMF in that they both use an exponential number of variables corresponding to rank-1 binary matrices. However, due to the differences between exact and k-BMF as outlined in the Section 1.1, the two models are not directly comparable. Furthermore, the enumeration method used to precompute all maximal rectangles in [11] is not applicable to our case as we have to consider all rank-1 binary matrices of $\mathbf{X}$'s dimension. To generate rank-1 binary matrices in our column generation method, we explore several greedy heuristics in detail.

In addition, we introduce a new objective function for k-BMF under which the problem becomes computationally easier and we explore the relationship between this new objective function and the original squared Frobenius distance. Finally, we demonstrate that 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, the entry-wise modelling of k-BMF in our formulations naturally extends to handle matrices with missing entries and perform binary matrix completion, we illustrate this way of application experimentally.

The rest of this paper is organised as follows. In Section 2 we detail the three IP formulations for k-BMF and prove several results about their LP-relaxations. In Section 3, we introduce a new objective function and explore its relation to the original squared Frobenius objective. In Section 4 we detail a framework based on the large scale optimisation technique of column generation for the solution of our exponential formulation and discuss heuristics for the arising pricing problems. Finally, in Section 5 we demonstrate the practical applicability of our approach on several artificial and real world datasets.

2. Formulations. Given a binary matrix $\mathbf{X} \in \{0,1\}^{n\times m}$ and a fixed positive integer $k \ll \min(n,m)$ we wish to find two binary matrices $\mathbf{A} \in \{0,1\}^{n\times k}$ and $\mathbf{B} \in \{0,1\}^{k\times m}$ so that $\|\mathbf{X}-\mathbf{Z}\|_F^2$ is minimised, where $\mathbf{Z}$ is the Boolean matrix product of $\mathbf{A}$ and $\mathbf{B}$ and $\|\cdot\|_F$ denotes the Frobenius norm. Let $E = \{(i,j) : x_{ij} = 1\} \subset [n] \times [m]$ denote the index set of nonzero entries of $\mathbf{X}$ where $[n] := \{1,\dots,n\}$. Both $\mathbf{X}$ and $\mathbf{Z}$ being binary matrices, the squared Frobenius and the entry-wise ℓ_1 norm coincide and we may expand the objective function to get a linear expression

$$\|\mathbf{X}-\mathbf{Z}\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} |x_{ij}-z_{ij}| = \sum_{(i,j)\in E}(1-z_{ij}) + \sum_{(i,j)\notin E} z_{ij}. \tag{3}$$

For an incomplete binary matrix $\mathbf{X}$ with missing entries, the above objective is slightly changed to $\sum_{(i,j)\in E}(1-z_{ij}) + \sum_{(i,j)\in \overline{E}} z_{ij}$ where $\overline{E} = \{(i,j) : x_{ij} = 0\}$, to emphasise that $E \cup \overline{E} \neq [n]\times[m]$, and the factorisation error is only measured over known entries. In the following sections we present three different integer programs for k-BMF all with the above derived linear objective function. For each IP, as customary, we consider its linear programming (LP) relaxation under the same objective function. Note that for non-binary $\mathbf{Z}$, the squared Frobenius norm and entry-wise ℓ_1-norm do not coincide. However, we use LP relaxations as a means to lower bound the optimal objective value of the IP which corresponds to the optimal k-BMF approximation error and not to obtain continuous factorisations of $\mathbf{X}$.

2.1. Compact formulation. We start with a formulation that uses a polynomial number of variables and constraints where we denote the McCormick envelope [37] of $a, b \in [0,1]$ by

$$MC(a,b) = \{y \in \mathbb{R} : 0 \le y,\ a+b-1 \le y,\ y \le a,\ y \le b\}. \tag{4}$$

Note that if $a, b \in \{0,1\}$ then $MC(a,b)$ only contains the point $ab \in \{0,1\}$ corresponding to the product of a and b. The following Compact Integer linear Program (CIP) models the entries of matrices $\mathbf{A}, \mathbf{B}, \mathbf{Z}$ directly via binary variables $a_{i\ell}$, $b_{\ell j}$ and z_{ij} respectively (for $i \in [n], \ell \in [k], j \in [m]$) and uses McCormick envelopes to avoid the appearance of quadratic terms that would correspond to the constraints $y_{i\ell j} = a_{i\ell} b_{\ell j}$,

$$\text{(CIP)} \quad \zeta_{\text{CIP}} = \min_{a,b,y,z} \sum_{(i,j)\in E} (1 - z_{ij}) + \sum_{(i,j)\in \overline{E}} z_{ij} \tag{5}$$

$$\text{s.t. } y_{i\ell j} \le z_{ij} \le \sum_{l=1}^{k} y_{ilj} \qquad i \in [n], j \in [m], \ell \in [k], \tag{6}$$

$$y_{i\ell j} \in MC(a_{i\ell}, b_{\ell j}) \qquad i \in [n], j \in [m], \ell \in [k], \tag{7}$$

$$a_{i\ell}, b_{\ell j}, z_{ij} \in \{0,1\} \qquad i \in [n], j \in [m], \ell \in [k]. \tag{8}$$

Constraints (6) encode Boolean matrix multiplication, while a simple modification of the model in which constraints (6) are replaced by $z_{ij} = \sum_{\ell=1}^{k} y_{i\ell j}$ models k-BMF under standard arithmetic. The McCormick envelopes in constraints (7) ensure that for $a_{i\ell}, b_{\ell j} \in \{0,1\}$, $y_{i\ell j}$ are binary variables taking the value $a_{i\ell} b_{\ell j}$. Due to the objective function, constraints (6) and the binary nature of $y_{i\ell j}$, the binary constraints on variables z_{ij} may be relaxed to $z_{ij} \in [0,1]$ without altering optimal solutions of the formulation.

CIP can easily be adapted to give a polynomial size IP for exact-BMF as follows. Let $t = \min\{n, m\}$. As the Boolean rank is bounded by t, we can replace k in CIP by t. Delete variables z_{ij} from the model and in constraints (6) replace z_{ij} by the input values x_{ij}. Introduce indicator variables $d_\ell \in \{0,1\}$ $(\ell \in [t])$ and add the constraints $a_{i\ell} \le d_\ell$ $(i \in [n], \ell \in [t])$ and $b_{\ell j} \le d_\ell$ $(j \in [m], \ell \in [t])$. The objective function $\min_{a,b,y,d} \sum_{\ell \in [t]} d_\ell$ then corresponds to minimising the Boolean rank. Note however that for exact-BMF, solving this polynomial size IP is computationally more challenging then using the methods of [11].

The LP relaxation of CIP (CLP) is obtained by replacing constraints (8) by $a_{i\ell}, b_{\ell,j}, z_{ij} \in [0,1]$. For $k = 1$, we have $z_{ij} = y_{i1j}$ and the feasible region of CIP is the Boolean Quadric Polytope (BQP) over a bipartite graph [43]. The LP relaxation of BQP has half-integral vertices [43], which implies that CLP for $k = 1$ has half-integral vertices as well. One can show that in this case, a simple rounding in which fractional values of CLP are rounded down to 0 gives a 2-approximation to 1-BMF [49]. This however, does not apply for $k > 1$. We next show that CLP for $k > 1$ has an objective function value 0. We note that the statement of a weaker version of the following proposition appeared in our preliminary work [26].

PROPOSITION 1. *Given a binary matrix $\boldsymbol{X} \in \{0,1\}^{n\times m}$, CLP has optimal objective value 0 for $k > 1$. Moreover, for $k > 2$ CLP has at least $k|E| + 1$ vertices with objective value 0.*

Proof. For each $(i,j) \in E$ let $L_{(i,j)} \subseteq [k]$ such that $|L_{(i,j)}| \ge 2$ and consider the point

$$a_{i\ell} = \frac{1}{2} \quad i \in [n], \ell \in [k], \qquad b_{\ell j} = \frac{1}{2} \quad \ell \in [k], j \in [m],$$

$$y_{i\ell j} = \begin{cases} \frac{1}{2} & (i,j) \in E, \ell \in L_{(i,j)} \\ 0 & \text{otherwise,} \end{cases} \qquad z_{ij} = \begin{cases} 1 & (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

For all $(i,j) \in [n] \times [m]$ and $\ell \in [k]$, setting $a_{i\ell} = b_{\ell j} = \frac{1}{2}$ implies that $y_{i\ell j} \in MC(\frac{1}{2}, \frac{1}{2}) = [0, \frac{1}{2}]$ and $\sum_{l=1}^{k} y_{ilj} \geq 1$ holds for all $(i,j) \in E$, hence this point gives a feasible solution to CLP with objective value 0. For $k = 2$, we can only set $L_{(i,j)} = [2]$ for all $(i,j) \in E$, hence the above construction leads to a single unique point. For $k > 2$ however, as the choice of $L_{(i,j)}$'s is arbitrary, there are many feasible points with objective value 0 of this form. As each of these points can differ at only $k|E|$ entries corresponding to entries $y_{i\ell j}$ for $(i,j) \in E$, $\ell \in [k]$, there are at most $k|E|+1$ affinely independent points among them. Next we present $k|E|+1$ affinely independent points of this form. Since the objective value is 0 at these points, they must lie on a face of dimension at least $k|E|$ and this face must have at least $k|E|+1$ vertices of CLP with objective value 0. For each $(i,j)^* \in E$ and $\ell^* \in [k]$, letting $L_{(i,j)} = [k]$ for all $(i,j) \in E \setminus \{(i,j)^*\}$ and $L_{(i,j)^*} = [k] \setminus \{\ell^*\}$ provides $k|E|$ different points of the above form. Each such point has exactly one entry $y_{i\ell j}$ along the indices $(i,j) \in E, \ell \in [k]$ which is zero. Hence the matrix whose columns correspond to these $k|E|$ points has a square submatrix of the form $\frac{1}{2}(\mathbf{J}_{k|E|} - \mathbf{I}_{k|E|})$ corresponding to entries $y_{i\ell j}$ for $(i,j) \in E, \ell \in [k]$, where $\mathbf{J}_t$ is the all ones matrix of size $t \times t$ and $\mathbf{I}_t$ is the identity matrix of size t. Since matrix $\mathbf{J}_t - \mathbf{I}_t$ is nonsingular, the $k|E|$ points are linearly independent. In addition, letting $L_{(i,j)} = [k]$ for all $(i,j) \in E$ gives an additional point for which $y_{i\ell j} = \frac{1}{2}$ for all $(i,j) \in E, \ell \in [k]$, hence the corresponding part of this point is $\frac{1}{2}\mathbf{1}$. Now subtracting $\frac{1}{2}\mathbf{1}$ from the columns of $\frac{1}{2}(\mathbf{J}_{k|E|} - \mathbf{I}_{k|E|})$, we get the nonsingular matrix $-\frac{1}{2}\mathbf{I}_{k|E|}$, hence the $k|E|+1$ above constructed points are affinely independent. □

The above result suggests that unless the factorisation error is 0 i.e. the input matrix is of Boolean rank less than or equal to k, before improving the LP bound of CIP many fractional vertices need to be cut off. To strengthen the formulation of CIP, valid inequalities may be explored. Especially, some of the half-integral points that appear in Proposition 1 may be cut off by odd-cycle inequalities over the bipartite Boolean Quadric Polytope [43, 51]. For each $\ell \in [k]$, every 4-cycle in $K_{m,n}$ gives eight non-dominated odd-cycle inequalities corresponding to an odd subset of edges of the 4-cycle. As an example, two of these inequalities over a 4-cycle with vertices $\{i_1, j_1, i_2, j_2\}$ are,

$$-a_{i_1,\ell} - b_{\ell,j_1} + y_{i_1,\ell,j_1} + y_{i_1,\ell,j_2} + y_{i_2,\ell,j_1} - y_{i_2,\ell,j_2} \leq 0, \tag{9}$$
$$+a_{i_1,\ell} + b_{\ell,j_1} - y_{i_1,\ell,j_1} - y_{i_1,\ell,j_2} - y_{i_2,\ell,j_1} + y_{i_2,\ell,j_2} \leq 1. \tag{10}$$

However, for some matrices, adding all non-dominated odd-cycle inequalities to CLP is not sufficient to cut off all the half-integral points with 0 objective value that appear in Proposition 1.

Example 1. Let $\mathbf{X} = \bar{\mathbf{I}}_4 = \mathbf{J}_4 - \mathbf{I}_4$ (where $\mathbf{J}_t$ is the all 1s matrix and $\mathbf{I}_t$ is the identity matrix.) and $k = 3$. One can show that $\bar{\mathbf{I}}_4$ has Boolean rank 4, so no zero error rank-3 factorisation exists. Yet, the below half-integral 0-objective point, that is one of the points constructed in Proposition 1, is not cut off by any of the odd-cycle inequalities. The point has $a_{i,\ell} = b_{\ell,j} = \frac{1}{2}$ and y's set as

$$\mathbf{Y}_1 = \begin{bmatrix} & \frac{1}{2} & \frac{1}{2} & \\ \frac{1}{2} & & & \frac{1}{2} \\ \frac{1}{2} & & & \frac{1}{2} \\ & \frac{1}{2} & \frac{1}{2} & \end{bmatrix}, \mathbf{Y}_2 = \begin{bmatrix} & \frac{1}{2} & \frac{1}{2} & \\ & \frac{1}{2} & \frac{1}{2} & \\ \frac{1}{2} & & & \frac{1}{2} \\ \frac{1}{2} & & & \frac{1}{2} \end{bmatrix}, \mathbf{Y}_3 = \begin{bmatrix} & \frac{1}{2} & & \frac{1}{2} \\ \frac{1}{2} & & \frac{1}{2} & \\ & \frac{1}{2} & & \frac{1}{2} \\ \frac{1}{2} & & \frac{1}{2} & \end{bmatrix}, \text{where } \mathbf{Y}_\ell = \begin{bmatrix} y_{1,\ell,1} & y_{1,\ell,2} & y_{1,\ell,3} & y_{1,\ell,4} \\ y_{2,\ell,1} & y_{2,\ell,2} & y_{2,\ell,3} & y_{2,\ell,4} \\ y_{3,\ell,1} & y_{3,\ell,2} & y_{3,\ell,3} & y_{3,\ell,4} \\ y_{4,\ell,1} & y_{4,\ell,2} & y_{4,\ell,3} & y_{4,\ell,4} \end{bmatrix}. \tag{11}$$

To see that the above half-integral point is not cut off by any of the odd-cycle inequalities, observe that odd-cycle inequalities can cut off points which have components

$$\begin{bmatrix} y_{i_1,\ell,j_1} & y_{i_1,\ell,j_2} \\ y_{i_2,\ell,j_1} & y_{i_2,\ell,j_2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 \end{bmatrix}, \text{ or } \begin{bmatrix} y_{i_1,\ell,j_1} & y_{i_1,\ell,j_2} \\ y_{i_2,\ell,j_1} & y_{i_2,\ell,j_2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}, \tag{12}$$

or any permutation of these. Since these 2×2 components or their permutation do not appear in $\mathbf{Y}_\ell$ $(\ell = 1, 2, 3)$ in Equation (11), the above half-integral point is a 0-objective point of CLP even after adding all non-dominated odd-cycle inequalities.

In some other cases, adding all the odd-cycle inequalities to CLP does cut off all the half-integral points that appear in Proposition 1, but some other 0-objective points remain.

EXAMPLE 2. Let $\mathbf{X} = \mathbf{I}_4$ and $k = 3$. $\mathbf{I}_4$ clearly has Boolean rank 4, hence no rank-3 BMF can give 0 error. After adding all odd-cycle inequalities to CLP, the below point is still feasible and has objective value 0,

$$a_{i,\ell} = b_{\ell,j} = \frac{1}{3}, \qquad \mathbf{Y}_1 = \mathbf{Y}_2 = \mathbf{Y}_3 = \begin{bmatrix} \frac{1}{3} & & & \\ & \frac{1}{3} & & \\ & & \frac{1}{3} & \\ & & & \frac{1}{3} \end{bmatrix}, \qquad \text{where } \mathbf{Y}_\ell = \begin{bmatrix} y_{1,\ell,1} & y_{1,\ell,2} & y_{1,\ell,3} & y_{1,\ell,4} \\ y_{2,\ell,1} & y_{2,\ell,2} & y_{2,\ell,3} & y_{2,\ell,4} \\ y_{3,\ell,1} & y_{3,\ell,2} & y_{3,\ell,3} & y_{3,\ell,4} \\ y_{4,\ell,1} & y_{4,\ell,2} & y_{4,\ell,3} & y_{4,\ell,4} \end{bmatrix}. \tag{13}$$

Furthermore, for $k > 1$, any feasible rank-k factorisation $\mathbf{A} \circ \mathbf{B}$ and a permutation matrix $\mathbf{P} \in \{0,1\}^{k \times k}$ provide another feasible solution $\mathbf{AP} \circ \mathbf{P}^\top \mathbf{B}$ to CIP with the same objective value. Hence, CIP is highly symmetric for $k > 1$. These properties of CIP make it unlikely to be solved to optimality for $k > 1$ in a reasonable amount of time for a large matrix $\mathbf{X}$, though some symmetries may be broken by enforcing lexicographic ordering of rows of $\mathbf{B}$. In future work, the symmetries in CIP could be tackled by developing a permutation-invariant extended formulation [20]. For small matrices however, CIP constitutes the first approach to get optimal solutions to k-BMF.

2.2. Exponential size formulation I. Any $n \times m$ Boolean rank-k matrix can be equivalently written as the Boolean combination of k rank-1 binary matrices $\bigvee_{\ell=1}^{k} \boldsymbol{a}_\ell \boldsymbol{b}_\ell^\top$ for some $\boldsymbol{a}_\ell \in \{0,1\}^n, \boldsymbol{b}_\ell \in \{0,1\}^m$. This suggest to directly look for k rank-1 binary matrices instead of introducing variables for all entries of factor matrices $\mathbf{A}$ and $\mathbf{B}$. The second integer program we detail for k-BMF relies on this approach by considering an implicit enumeration of rank-1 binary matrices. Let $\mathcal{R}$ denote the set of all rank-1 binary matrices of dimension $n \times m$ and let $\mathcal{R}_{(i,j)}$ denote the subset of rank-1 matrices of $\mathcal{R}$ which have the (i,j)-th entry equal to 1,

$$\mathcal{R} := \{\boldsymbol{a}\boldsymbol{b}^\top : \boldsymbol{a} \in \{0,1\}^n, \boldsymbol{b} \in \{0,1\}^m, \boldsymbol{a}, \boldsymbol{b} \neq \mathbf{0}\} \subset \{0,1\}^{n \times m}, \tag{14}$$

$$\mathcal{R}_{(i,j)} := \{\boldsymbol{a}\boldsymbol{b}^\top \in \mathcal{R} : a_i = b_j = 1\} \subset \mathcal{R} \qquad i \in [n], j \subset [m]. \tag{15}$$

Introducing a binary variable q_r for each rank-1 matrix r in $\mathcal{R}$ and variables z_{ij} corresponding to the known entries of the $\mathbf{X}$, we obtain the following Master Integer linear Program (MIP),

$$(\text{MIP}_\text{F}) \quad \zeta_\text{MIP} = \min_{z,q} \sum_{(i,j)\in E} (1 - z_{ij}) + \sum_{(i,j)\in \overline{E}} z_{ij} \tag{16}$$

$$\text{s.t. } z_{ij} \leq \sum_{r \in \mathcal{R}_{(i,j)}} q_r \qquad (i,j) \in E \tag{17}$$

$$\sum_{r \in \mathcal{R}_{(i,j)}} q_r \leq k\, z_{ij} \qquad (i,j) \in \overline{E} \tag{18}$$

$$\sum_{r \in \mathcal{R}} q_r \leq k \tag{19}$$

$$z_{ij}, q_r \in \{0,1\} \qquad (i,j) \in E \cup \overline{E},\ r \in \mathcal{R} \tag{20}$$

The objective, as before, measures the factorisation error in squared Frobenius norm, and subscript F in MIP_F stands for Frobenius. Constraints (17) and (18) enforce Boolean matrix multiplication: z_{ij} takes value 1 if there is at least one active rank-1 binary matrix that covers entry (i,j), otherwise it takes value 0. Notice, that due to the difference in sign of objective coefficients for variables z_{ij} with $(i,j) \in E$ and $(i,j) \in \overline{E}$ it is enough to declare constraints (17) and (18) for indices $(i,j) \in E$ and $(i,j) \in \overline{E}$ respectively. Constraint (19) ensures that at most k rank-1 binary matrices are active and hence we get a rank-k factorisation of $\mathbf{X}$. Observe that constraints (17) together with q_r being

binary imply that z_{ij} automatically takes binary values for $(i,j) \in E$, and due to the objective function it always takes the value at its upper bound, hence $z_{ij} \in \{0,1\}$ may be replaced by $z_{ij} \leq 1$ for all $(i,j) \in E$ without altering the optimum. In contrast, z_{ij} for $(i,j) \in \overline{E}$ need to be explicitly declared binary as otherwise, if there are some active rank-1 matrices ($q_r > 0$) which cover a zero of $\mathbf{X}$ ($r \in \mathcal{R}_{(i,j)}$, $(i,j) \in \overline{E}$) then variable z_{ij} corresponding to that zero takes the possibly fractional value $\frac{1}{k} \sum_{r \in \mathcal{R}_{(i,j)}} q_r$. One can also consider a *strong formulation* of MIP_F with exponentially many constraints, in which constraints (18) are replaced by $q_r \leq z_{ij}$ for all $r \in \mathcal{R}_{(i,j)}$ and $(i,j) \in \overline{E}$.

The LP relaxation of MIP_F (MLP_F) is obtained by replacing the integrality constraints by $z_{ij}, q_r \in [0,1]$. Unlike CLP, the optimal objective value of MLP_F (ζ_{MLP}) is not always zero. By comparing the rank of the factorisation, k to the *isolation number* of the input matrix $\mathbf{X}$ we can deduce when MLP_F will take non-zero objective value. We next give an extension of the definition of isolation number for binary matrices presented in [18] and [40, Section 2.3]. The isolation number is sometimes also referred to as the fooling set bound [12, 6].

Definition 1. Let $\mathbf{X}$ be a binary matrix with possibly missing entries. A set $S \subseteq E = \{(i,j) : x_{ij} = 1\}$ is said to be an *isolated set* if whenever $(i_1,j_1),(i_2,j_2)$ are two distinct members of S then (a) $i_1 \neq i_2$, $j_1 \neq j_2$ and (b) $(i_1,j_2) \in \overline{E}$ or $(i_2,j_1) \in \overline{E}$ or both, where $\overline{E} = \{(i,j) : x_{ij} = 0\}$. The cardinality of a maximum isolated set of $\mathbf{X}$ is the *isolation number* of $\mathbf{X}$, denoted by $i(\mathbf{X})$.
From the definition it follows that members of an isolated set cannot be covered by a common rank-1 submatrix, and hence the isolation number provides a lower bound on the Boolean rank. The following result (whose weaker preliminary version appeared in [26]) shows that MLP_F must have non-zero objective value whenever k, the rank of the factorisation, is chosen so that it is strictly smaller than the isolation number.

Proposition 2. *Let $\boldsymbol{X}$ have isolation number $i(\boldsymbol{X}) > k$, then $\zeta_{\text{MLP}} \geq \frac{1}{k}(i(\boldsymbol{X}) - k)$.*

Proof. Let S be an isolated set of $\mathbf{X}$ of cardinality $i(\mathbf{X})$. We will establish a feasible solution to the dual of MLP_F (MDP_F) with objective value $\frac{1}{k}(i(\mathbf{X}) - k)$ implying the result.

Let us apply a change of variables $\xi_{ij} = 1 - z_{ij}$ for $(i,j) \in E$ for the ease of avoiding the constant term in the objective function of MLP_F. Then the bound constraints of MLP_F can be written as $\xi_{ij} \geq 0$ for $(i,j) \in E$, $z_{ij} \geq 0$ for $(i,j) \in \overline{E}$ and $q_r \geq 0$, $r \in \mathcal{R}$ as the objective function is minimising both ξ_{ij} and z_{ij} and we have the cardinality constrains on q_r. Associating dual variables $p_{ij} \geq 0$ $(i,j) \in E$ with constraints $\sum_{r \in \mathcal{R}_{i,j}} q_r + \xi_{ij} \geq 1$, $s_{ij} \geq 0$ $(i,j) \in \overline{E}$ with constraints (18) and $\mu \geq 0$ with constraint (19), the Master Dual Program (MDP_F) of MLP_F is

$$
\begin{aligned}
(\text{MDP}_\text{F})\ \zeta_{\text{MDP}} = \max_{p,s,\mu} & \sum_{(i,j) \in E} p_{ij} - k\,\mu && && (21)\\
\text{s.t.} & \sum_{(i,j) \in E \cap \text{supp}(\mathbf{R})} p_{ij} - \sum_{(i,j) \in \overline{E} \cap \text{supp}(\mathbf{R})} s_{ij} \leq \mu && R \in \mathcal{R} && (22)\\
& 0 \leq p_{ij} \leq 1 && (i,j) \in E && (23)\\
& 0 \leq s_{ij} \leq \frac{1}{k} && (i,j) \in \overline{E} && (24)\\
& 0 \leq \mu, && && (25)
\end{aligned}
$$

where $\text{supp}(R) = \{(i,j) : r_{ij} = 1\}$.

Let $s_{ij} = \frac{1}{k}$ for $(i,j) \in \overline{E}$ and let $p_{ij} = \frac{1}{k}$ for $(i,j) \in S$ and $p_{ij} = 0$ for all other $(i,j) \in E \setminus S$. The bound constraints on p_{ij} and s_{ij} are satisfied then. It remains to choose $\mu \geq 0$ such that we satisfy constraint (22) for all rank-1 binary matrices $\mathbf{R} \in \mathcal{R}$. Let $\mathbf{R} \in \mathcal{R}$ be a submatrix of $\mathbf{X}$, so we have $|\overline{E} \cap \text{supp}(\mathbf{R})| = 0$. Then by the definition of isolated sets, $\mathbf{R}$ can contain at most one element from S and hence we have $|\text{supp}(\mathbf{R}) \cap S| \leq 1$. This tells us that for any $\mu \geq \frac{1}{k}$, constraint (22) is satisfied for all $\mathbf{R} \in \mathcal{R}$ that is a submatrix of $\mathbf{X}$. Now let $\mathbf{R} \in \mathcal{R}$ be a rank-1 binary matrix which covers at least one zero entry of $\mathbf{X}$. Then $\mathbf{R}$ may contain more than one element from S. However,

if it contains more than one element from S then it must also contain at least $\binom{|\operatorname{supp}(\mathbf{R})\cap S|}{2}$-many zeros as for any two distinct elements $(i_1,j_1),(i_2,j_2)$ in S we have $(i_1,j_2)\in\overline{E}$ or $(i_2,j_1)\in\overline{E}$ by the definition of isolated set. Hence, for all $\mathbf{R}\in\mathcal{R}$ such that $|\overline{E}\cap\operatorname{supp}(\mathbf{R})|>0$, constraint (22) satisfies

$$\frac{1}{k}|S\cap\operatorname{supp}(\mathbf{R})|-\frac{1}{k}|\overline{E}\cap\operatorname{supp}(\mathbf{R})|\leq\frac{1}{k}|S\cap\operatorname{supp}(\mathbf{R})|-\frac{1}{k}\binom{|S\cap\operatorname{supp}(\mathbf{R})|}{2}\leq\frac{1}{k}. \tag{26}$$

Thus we can set $\mu=\frac{1}{k}$ to get the objective value $\frac{1}{k}(i(\mathbf{X})-k)\leq\zeta_{\text{MDP}}=\zeta_{\text{MLP}}$, which provides a non-zero bound on MLP_{F} for all $k<i(\mathbf{X})$. □

The following example shows that we cannot strengthen Proposition 2 by replacing the condition $k<i(\mathbf{X})$ with the requirement that k has to be strictly smaller than the Boolean rank of $\mathbf{X}$.

EXAMPLE 3. Let $\mathbf{X}=\mathbf{J}_4-\mathbf{I}_4$, where $\mathbf{J}_4$ is the 4×4 matrix of all 1s and $\mathbf{I}_4$ is the 4×4 identity matrix. One can verify that the Boolean rank of $\mathbf{X}$ is 4 and its isolation number is 3. For $k=3$, the optimal objective value of MLP_{F} is 0 which is attained by a fractional solution in which the following 6 rank-1 binary matrices are active with weight $\frac{1}{2}$.

$$q_1=\tfrac{1}{2}\begin{bmatrix}0&0&0&0\\1&0&1&0\\0&0&0&0\\1&0&1&0\end{bmatrix}\quad q_2=\tfrac{1}{2}\begin{bmatrix}0&1&1&0\\0&0&0&0\\0&0&0&0\\0&1&1&0\end{bmatrix}\quad q_3=\tfrac{1}{2}\begin{bmatrix}0&1&0&1\\0&0&0&0\\0&1&0&1\\0&0&0&0\end{bmatrix}\quad q_4=\tfrac{1}{2}\begin{bmatrix}0&0&0&0\\1&0&0&1\\1&0&0&1\\0&0&0&0\end{bmatrix}\quad q_5=\tfrac{1}{2}\begin{bmatrix}0&0&1&1\\0&0&1&1\\0&0&0&0\\0&0&0&0\end{bmatrix}\quad q_6=\tfrac{1}{2}\begin{bmatrix}0&0&0&0\\0&0&0&0\\1&1&0&0\\1&1&0&0\end{bmatrix}$$

2.3. Exponential size formulation II. For $t\in[2^m-1]$ let $\boldsymbol{\beta}_t\in\{0,1\}^m$ be the vector denoting the binary encoding of t and note that these vectors give a complete enumeration of all non-zero binary vectors of size m. Let β_{tj} denote the j-th entry of $\boldsymbol{\beta}_t$. In [34], the authors present the following Exponential size Integer linear Program (EIP) formulation using a separate indicator variable d_t for each one of these exponentially many binary vectors $\boldsymbol{\beta}_t$,

$$\text{(EIP)}\quad \zeta_{\text{EIP}}=\min_{\alpha,z,d}\sum_{(i,j)\in E}(1-z_{ij})+\sum_{(i,j)\in\overline{E}}z_{ij} \tag{27}$$

$$\text{s.t. } z_{ij}\leq\sum_{t=1}^{2^m-1}\alpha_{it}\beta_{tj} \qquad (i,j)\in E, \tag{28}$$

$$\sum_{t=1}^{2^m-1}\alpha_{it}\beta_{tj}\leq kz_{ij} \qquad (i,j)\in\overline{E}, \tag{29}$$

$$\sum_{t=1}^{2^m-1}d_t\leq k \tag{30}$$

$$\alpha_{it}\leq d_t \qquad i\in[n],t\in[2^m-1], \tag{31}$$

$$z_{ij},d_t,\alpha_{it}\in\{0,1\} \qquad (i,j)\in E\cup\overline{E},t\in[2^m-1]. \tag{32}$$

The above formulation has an exponential number of variables and constraints but it is an integer linear program as β_{tj} are input parameters to the model. Let ELP be the LP relaxation of EIP. Observe that due to the objective function the bound constraints in ELP may be simplified to $z_{ij},\alpha_{it},d_t\geq0$ for all i,j,t and $z_{ij}\leq1$ for $(i,j)\in E$ without changing the optimum. To solve EIP or ELP explicitly, one needs to enumerate all binary vectors $\boldsymbol{\beta}_t$, which is possible only up to a very limited size. To the best of our knowledge, no method is available that avoids explicit enumeration and can guarantee the optimal solution of EIP. Previous attempts at computing a rank-k factorisation via EIP all relied on working with only a small heuristically chosen subset of vectors $\boldsymbol{\beta}_t$ [34, 35]. However, if there was an efficient method to solve ELP, the following result shows it to be as strong as the LP relaxation of MIP_{F}.

PROPOSITION 3. *The optimal objective values of* ELP *and* MLP_F *are equal.*

Proof. Note that due to constraints (17) and (18) in MLP_F and constraints (28) and (29) in ELP, it suffices to show that for any feasible solution α_{it}, d_t of ELP one can build a feasible solution q_r of MLP_F for which $\sum_{t=1}^{2^m-1} \alpha_{it}\beta_{tj} = \sum_{r\in\mathcal{R}_{(i,j)}} q_r$, and vice-versa.

First consider a feasible solution $\boldsymbol{\alpha}_t \in \mathbb{R}^n$, $d_t \in \mathbb{R}$ (for $t \in [2^m-1]$) to ELP and note that by constraint (31) we have $0 \le \alpha_{it} \le d_t$ for all $i \in [n]$ and $t \in [2^m-1]$. We can therefore express each $\boldsymbol{\alpha}_t$ as a convex combination of binary vectors in $\{0,1\}^n$ scaled by d_t,

$$\boldsymbol{\alpha}_t = d_t \sum_{s=1}^{2^n-1} \lambda_{s,t}\, \boldsymbol{a}_s \quad \boldsymbol{a}_s \in \{0,1\}^n \setminus \{\mathbf{0}\}, \quad \sum_{s=1}^{2^n-1} \lambda_{s,t} \le 1, \quad \lambda_{s,t} \ge 0, \quad s \in [2^n-1] \tag{33}$$

where $\boldsymbol{a}_s$ denotes the binary encoding of s. Note that we do not require $\lambda_{s,t}$'s to add up to 1 as we exclude the zero vector. We can therefore rewrite the solution of ELP as follows

$$\sum_{t=1}^{2^m-1} \boldsymbol{\alpha}_t\boldsymbol{\beta}_t^\top = \sum_{t=1}^{2^m-1}\sum_{s=1}^{2^n-1} d_t\,\lambda_{s,t}\,\boldsymbol{a}_s\boldsymbol{\beta}_t^\top = \sum_{t=1}^{2^m-1}\sum_{s=1}^{2^n-1} q_{s,t}\boldsymbol{a}_s\boldsymbol{\beta}_t^\top \quad \text{where } q_{s,t} := d_t\,\lambda_{s,t}. \tag{34}$$

Now it is easy to see that $\boldsymbol{a}_s\boldsymbol{\beta}_t^\top \in \mathcal{R}$ and since $\sum_{t=1}^{2^m-1} d_t \le k$ holds in any feasible solution to ELP, we get $\sum_{s=1}^{2^n-1}\sum_{t=1}^{2^m-1} q_{s,t} \le k$, which shows that $q_{s,t}$ is feasible for MLP_F.

The construction works backwards as well, as any feasible solution to MLP_F can be written as $\sum_{s=1}^{2^n-1}\sum_{t=1}^{2^m-1} q_{s,t}\boldsymbol{a}_s\boldsymbol{\beta}_t^\top$ for some rank-1 binary matrices $\boldsymbol{a}_s\boldsymbol{\beta}_t^\top \in \mathcal{R}$ and corresponding variables $q_{s,t} \ge 0$. Now let $\boldsymbol{\alpha}_t := \sum_{s=1}^{2^n-1} q_{s,t}\,\boldsymbol{a}_s$ and $d_t := \max_{i\in[n]} \alpha_{it}$ to satisfy $\alpha_{it} \le d_t$. Then since we started from a feasible solution to MLP_F, we have $\sum_{s=1}^{2^n-1}\sum_{t=1}^{2^m-1} q_{s,t} \le k$ and hence $\sum_{t=1}^{2^m-1} d_t \le k$ is satisfied too. □

3. Working under a new objective function. In the previous section, we presented formulations for k-BMF which measured the factorisation error in the squared Frobenius norm, which coincides with the entry-wise ℓ_1 norm as showed in Equation (3). In this section, we explore another objective function which introduces an asymmetry between how false negatives and false positives are treated. Whenever a 0 entry is erroneously covered in a rank-k factorisation, it may be covered by up to k rank-1 binary matrices. Our new objective function attributes a weighted error term to each 0 entry which is proportional to the number of rank-1 matrices covering that entry. As previously, by denoting $\mathbf{Z} = \mathbf{A} \circ \mathbf{B}$ a rank-k factorisation of $\mathbf{X}$, the new objective function is

$$\zeta(\rho) = \sum_{(i,j)\in E} (1 - z_{ij}) + \rho \sum_{(i,j)\in \overline{E}} \sum_{\ell=1}^{k} a_{i\ell}b_{\ell j}. \tag{35}$$

Note that the constraints $a_{i\ell}b_{\ell j} \le z_{ij} \le \sum_{\ell=1}^{k} a_{i\ell}b_{\ell j}$ encoding Boolean matrix multiplication imply that $\frac{1}{k}\sum_{\ell=1}^{k} a_{i\ell}b_{\ell j} \le z_{ij} \le \sum_{\ell=1}^{k} a_{i\ell}b_{\ell j}$. Therefore, denoting the original squared Frobenius norm objective function in Equation (3) by ζ_F, for any $\mathbf{X}$ and rank-k factorisation $\mathbf{Z}$ of $\mathbf{X}$ the following relationship holds between ζ_F and $\zeta(1)$, $\zeta(\frac{1}{k})$,

$$\zeta_F \le \zeta(1) \le \sum_{(i,j)\in E} (1 - z_{ij}) + \sum_{(i,j)\in\overline{E}} k\, z_{ij} \le k\,\zeta_F \qquad \text{and} \qquad \frac{1}{k}\zeta_F \le \zeta(\frac{1}{k}) \le \zeta_F. \tag{36}$$

We next show that this new objective function $\zeta(\rho)$ with $\rho = 1$ can overestimate the original objective ζ_F by a factor of k. But first, we need a technical result which shows that whenever the input matrix $\mathbf{X}$ contains repeated rows or columns we may assume that an optimal factorisation exists which has the same row-column repetition pattern.

LEMMA 1 **(Preprocessing)**. *Let $\boldsymbol{X}$ contain some duplicate rows and columns. Then there exists an optimal rank-k binary matrix factorisation of $\boldsymbol{X}$ under objective ζ_F (or $\zeta(\rho)$) whose rows and columns corresponding to identical copies in $\boldsymbol{X}$ are identical.*

Proof. Since the transpose of an optimal rank-k factorisation is optimal for $\mathbf{X}^\top$, it suffices to consider the rows of $\mathbf{X}$. Furthermore, it suffices to consider only one set of repeated rows of $\mathbf{X}$, so let $I \subseteq [n]$ be the index set of a set of identical rows of $\mathbf{X}$. We then need to show that there exists an optimal rank-k factorisation whose rows indexed by I are identical. Let $\mathbf{Z} = \mathbf{A} \circ \mathbf{B}$ be an optimal rank-k factorisation of $\mathbf{X}$ under objective ζ_F. For all $i_1, i_2 \in I$ we must have

$$\sum_{j:(i_1,j)\in E} (1 - z_{ij}) + \sum_{j:(i_1,j)\in \overline{E}} z_{ij} = \sum_{j:(i_2,j)\in E} (1 - z_{ij}) + \sum_{j:(i_2,j)\in \overline{E}} z_{ij} \tag{37}$$

as otherwise replacing $\mathbf{A}_{i,:}$ for each $i \in I$ with row $\mathbf{A}_{i^*,:}$ where $i^* \in I$ is a row index for which the above sum is minimised leads to a smaller error factorisation. Then since (37) holds, replacing $\mathbf{A}_{i,:}$ for each $i \in I$ with row $\mathbf{A}_{i^*,:}$ for any $i^* \in I$ leads to an optimal solution of the desired property. Similarly, if $\mathbf{Z}$ is an optimal factorisation under objective $\zeta(\rho)$, then for all $i_1, i_2 \in I$ the corresponding objective terms must equal and hence an optimal solution of the desired property exists. □

This result implies that whenever the input matrix $\mathbf{X}$ contains repeated rows or columns we may solve the following problem on a smaller matrix instead. Let $\mathbf{X}' \in \{0,1\}^{n' \times m'}$ be the binary matrix obtained from $\mathbf{X}$ by replacing each duplicate row and column by a single representative and let $\boldsymbol{r} \in \mathbb{Z}_+^{n'}$ and $\boldsymbol{c} \in \mathbb{Z}_+^{m'}$ be the counts of each unique row and column of $\mathbf{X}'$ in $\mathbf{X}$ respectively. Let E' and $\overline{E'}$ denote the non-zero and zero entry index sets of $\mathbf{X}'$ respectively. By Lemma 1 an optimal rank-k factorisation $\mathbf{Z}' = \mathbf{A}' \circ \mathbf{B}'$ of $\mathbf{X}'$ under the *updated* objective function

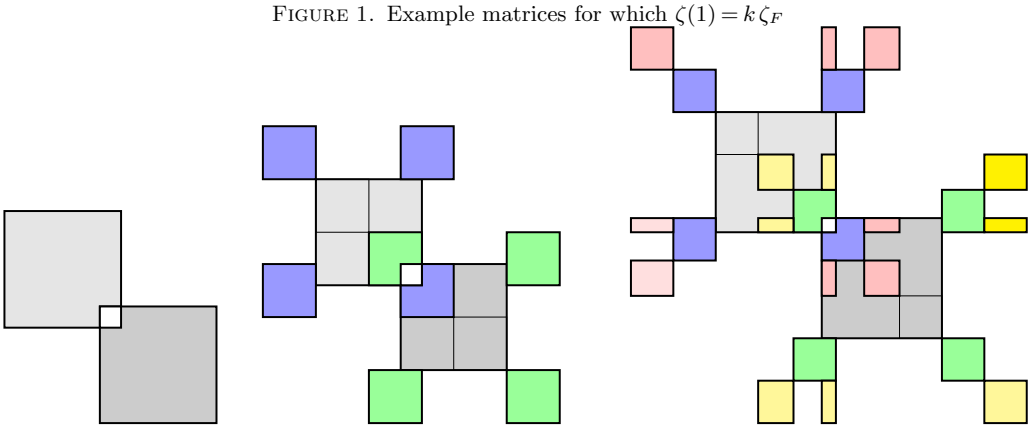
$$\zeta'_F := \sum_{(i,j)\in E'} r_i\, c_j\, (1 - z'_{ij}) + \sum_{(i,j)\in \overline{E'}} r_i\, c_j\, z'_{ij} \tag{38}$$

(or $\zeta'(\rho) := \sum_{(i,j)\in E'} r_i\, c_j\, (1 - z'_{ij}) + \rho \sum_{(i,j)\in \overline{E'}} r_i\, c_j \sum_{\ell=1}^{k} a'_{i\ell} b'_{\ell j}$) leads to an optimal rank-k factorisation of $\mathbf{X}$ under the original objective function ζ_F (or $\zeta(\rho)$).

PROPOSITION 4. *For each positive integer k there exists a matrix $\boldsymbol{X}(k)$ for which the optimal rank-k binary matrix factorisations under objectives ζ_F and $\zeta(1)$ satisfy $\zeta(1) = k\,\zeta_F$.*

Proof. The idea behind the proof is to consider a matrix $\mathbf{Z}(k)$ of exact Boolean rank-k in which all the k rank-1 components (rectangles) overlap at a unique middle entry and then replace this entry with a 0 to obtain $\mathbf{X}(k)$. Now $\mathbf{X}(k)$ and $\mathbf{Z}(k)$ are exactly at distance 1 in the squared Frobenius norm and hence $\mathbf{Z}(k)$ is a rank-k factorisation of $\mathbf{X}(k)$ with objective value $\zeta_F = 1$. On the other hand, since exactly k rectangles cover the entry at which $\mathbf{X}(k)$ and $\mathbf{Z}(k)$ differ, if $\mathbf{Z}(k)$ is taken as a rank-k factorisation of $\mathbf{X}(k)$ under objective $\zeta(1)$ it incurs an error of size k. Figure 1 shows the idea how to build such a $\mathbf{X}(k)$ for $k = 2, 4, 6$. Each colour corresponds to a rank-1 component and white areas correspond to 0s. We first consider the case when k is even. For $k = 2$ take the symmetric matrix $\mathbf{X}(2)$ as in Equation (39) which corresponds to Figure 1a. Since $\mathbf{X}(2)$ has repeated rows and columns, according to Lemma 1 we may simplify the problem by replacing $\mathbf{X}(2)$ by $\mathbf{X}'(2)$ and recording a weight vector for the rows and columns which indicate how many times each row and column is repeated. This weight vector is then used to update each entry in

FIGURE 1. Example matrices for which $\zeta(1) = k\zeta_F$



(a) $k=2$ (b) $k=4$ (c) $k=6$

the objective function with the corresponding weight. For $\mathbf{X}(2)$ the row and column weight vectors coincide as $\mathbf{X}(2)$ is symmetric and we denote it by $\boldsymbol{w}(2)$.

$$\mathbf{X}(2) = \begin{bmatrix} 1&1&0\\ 1&1&0\\ 1&1&0\\ 1&0&1\\ 0&1&1\\ 0&1&1\\ 0&1&1 \end{bmatrix} \circ \begin{bmatrix} 1&0&0\\ 1&0&0\\ 1&0&0\\ 0&1&0\\ 0&0&1\\ 0&0&1\\ 0&0&1 \end{bmatrix}^\top = \begin{bmatrix} 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&0&1&1&1\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1 \end{bmatrix} \Rightarrow \mathbf{X}'(2) = \begin{bmatrix} 1&1&0\\ 1&0&1\\ 0&1&1 \end{bmatrix} \text{ with } \boldsymbol{w}(2) = \begin{bmatrix} 3\\ 1\\ 3 \end{bmatrix} \tag{39}$$

The Boolean rank of $\mathbf{X}(2)$ is 3, which one can confirm by looking at a size 3 isolated set (shadowed entries) and an exact rank-3 factorisation shown in Equation (39). Let $\mathbf{Z}(2)$ be obtained from $\mathbf{X}(2)$ by replacing the 0 at entry $(4,4)$ by a 1. $\mathbf{Z}(2)$ clearly has Boolean rank 2, hence it is a feasible rank-2 factorisation of $\mathbf{X}(2)$. Under objective ζ_F $\mathbf{Z}(2)$ incurs an error of size 1, which is optimal as $\zeta_F \geq 1$ by $\mathbf{X}(2)$ being of Boolean rank-3. On the other hand, under objective $\zeta(1)$ $\mathbf{Z}(2)$ has objective value 2 as the middle entry is covered twice. To see that $\mathbf{Z}(2)$ is optimal under $\zeta(1)$ observe that every entry in $\mathbf{X}'(2)$ apart from the middle entry has weight strictly greater than 2. Hence not covering a 1 of $\mathbf{X}'(2)$ or covering a 0 different from the middle entry incurs an error strictly greater than 2.

For $k > 2$ even let us give a recipe to construct a symmetric matrix $\mathbf{X}'(k)$ and corresponding weight vector $\boldsymbol{w}(k)$. Let $t = \frac{k}{2} - 1$ and let the following $(4t+3) \times (4t+3)$ matrix be $\mathbf{X}'(k)$, where $\mathbf{I}_t$ is the identity matrix of size $t \times t$, $\tilde{\mathbf{I}}_t$ is the reverted identity matrix of size $t \times t$ (so $\tilde{\mathbf{I}}_2 = \left[\begin{smallmatrix} 0&1\\ 1&0 \end{smallmatrix}\right]$) and $\mathbf{J}_t$ is the all ones matrix of size $t \times t$,

$$\mathbf{X}'(k) = \begin{bmatrix} \mathbf{I}_t & & & \mathbf{1}_t & \tilde{\mathbf{I}}_t & & \\ & 1 & \mathbf{1}_t^\top & 1 & & & \\ & \mathbf{1}_t & \mathbf{J}_t & \mathbf{1}_t & & & \tilde{\mathbf{I}}_t \\ \mathbf{1}_t^\top & 1 & \mathbf{1}_t^\top & 0 & \mathbf{1}_t^\top & 1 & \mathbf{1}_t^\top \\ \tilde{\mathbf{I}}_t & & & \mathbf{1}_t & \mathbf{J}_t & \mathbf{1}_t & \\ & & & 1 & \mathbf{1}_t^\top & 1 & \\ & & \tilde{\mathbf{I}}_t & \mathbf{1}_t & & & \mathbf{I}_t \end{bmatrix}, \quad \boldsymbol{w}(k) = \begin{bmatrix} (k+1)\mathbf{1}_t \\ (k+1) \\ (k+1)\,\mathbf{1}_t \\ 1 \\ (k+1)\,\mathbf{1}_t \\ (k+1) \\ (k+1)\mathbf{1}_t \end{bmatrix}, \quad \mathbf{A}'(k) = \begin{bmatrix} \mathbf{I}_t & & \\ & 1 & \\ & \mathbf{1}_t & \tilde{\mathbf{I}}_t \\ \mathbf{1}_t^\top & 1 & 1 & \mathbf{1}_t^\top \\ \tilde{\mathbf{I}}_t & & \mathbf{1}_t & \\ & & 1 & \\ & & & \mathbf{I}_t \end{bmatrix}.$$

$\mathbf{X}'(k)$ has isolation number $i(\mathbf{X}'(k)) \geq 2t+3 = k+1$ (indicated by the shadowed entries), so no rank-k factorisation can have zero error. Let $\mathbf{Z}'(k)$ be obtained from $\mathbf{X}'(k)$ by replacing the middle

0 by a 1 and let its weight vector be the same as of $\mathbf{X}'(k)$. The Boolean rank of $\mathbf{Z}'(k)$ is then at most k as $\mathbf{Z}'(k) = \mathbf{A}'(k) \circ \mathbf{A}'^{\top}(k)$ is an exact factorisation and $\mathbf{A}'(k)$ is of dimension $(4t+3) \times k$. This factorisation is illustrated in Figure 1 for $k = 4, 6$. Therefore $\mathbf{Z}'(k)$ is a feasible rank-k factorisation of $\mathbf{X}'(k)$. Now $\mathbf{Z}'(k)$ under objective function ζ_F has error 1 and hence it is optimal. In contrast, $\mathbf{Z}'(k)$ evaluated under objective $\zeta(1)$ has error k as the middle 0 is covered k times and it has weight 1. To see that $\mathbf{Z}'(k)$ is optimal under $\zeta(1)$ as well, note that all entries of $\mathbf{X}'(k)$ apart from the middle 0 have weight strictly greater than k. Therefore, any other rank-k factorisation which does not cover a 1 or covers a 0 which is not the middle 0, incurs an error strictly greater than k, and hence $\mathbf{Z}'(k)$ is optimal under objective $\zeta(1)$ with value $k \cdot \zeta_F$.

For $k = 1$, all 1-BMFs satisfy $\zeta_F = \zeta(1)$ by definition. For $k > 1$ odd, we can obtain $\mathbf{X}'(k)$ and $\boldsymbol{w}(k)$ from $\mathbf{X}'(k+1)$ and $\boldsymbol{w}(k+1)$ by removing the first row and column of $\mathbf{X}'(k+1)$ and the corresponding first entry of $\boldsymbol{w}(k+1)$. For $\mathbf{X}'(k)$ then, the same reasoning holds as for k even. □

While Proposition 4 shows that $\zeta(1)$ can be k times larger than the Frobenius norm objective ζ_F, the matrices in the proof are quite artificial, and in practice we observe that not many zeros are covered by more than a few rank-1 matrices. In fact, our main motivation to consider this new objective function is that we observed that Exponential Formulation I. becomes computationally easier when using objective $\zeta(\rho)$ without compromising the accuracy of factorisations in practice. These numerical observations will be demonstrated in Sections 5.2.1 and 5.2.2. Therefore we consider the previously introduced formulations for k-BMF under the new objective $\zeta(\rho)$.

Let us denote a modification of formulation MIP_F with the new objective function $\zeta(\rho)$ as $\text{MIP}(\rho)$ and use the transformation $\xi_{ij} = 1 - z_{ij}$ for $(i,j) \in E$ to get

$$(\text{MIP}(\rho)) \quad \zeta_{\text{MIP}(\rho)} = \min_{\xi,q} \sum_{(i,j)\in E} \xi_{ij} + \rho \sum_{(i,j)\in \overline{E}} \sum_{r\in\mathcal{R}_{(i,j)}} q_r \tag{40}$$

$$\text{s.t.} \sum_{r\in\mathcal{R}_{(i,j)}} q_r + \xi_{ij} \geq 1 \qquad (i,j) \in E \tag{41}$$

$$\sum_{r\in\mathcal{R}} q_r \leq k \tag{42}$$

$$\xi_{ij} \geq 0,\ q_r \in \{0,1\} \qquad (i,j) \in E, r \in \mathcal{R}. \tag{43}$$

One of the imminent advantages of using objective $\zeta(\rho)$ is that we need only declare variables for entries $(i,j) \in E$ and can consequently delete the weak constraints (18) from the formulation. The LP relaxation of $\text{MIP}(\rho)$ ($\text{MLP}(\rho)$) is obtained by giving up on the integrality constraints on q_r and observing that without loss of generality we can simply write $q_r \geq 0$ for all $r \in \mathcal{R}$. We next show that the optimal solutions of the LP relaxation of MIP_F and $\text{MLP}(\rho)$ with $\rho = \frac{1}{k}$ coincide. The proof of this result is presented in the appendix of [26], but for the sake of completeness we repeat it here.

PROPOSITION 5. *The optimal solutions of the LP relaxations* MLP_F *and* $\text{MLP}(\frac{1}{k})$ *coincide.*

Proof. It suffices to observe that as MLP_F is a minimisation problem, each z_{ij} $(i,j) \in \overline{E}$ takes the value $\frac{1}{k}\sum_{r\in\mathcal{R}_{(i,j)}} q_r$ in any optimal solution to MLP_F due to constraint (18). This implies that the second terms in the objective function (16) of MIP_F and (40) of $\text{MLP}(\frac{1}{k})$ have the same value. □

Therefore one may instead solve $\text{MLP}(\frac{1}{k})$ that has fewer variables and constraints than MLP_F. In addition, for all $\rho > 0$, a corollary of Proposition 2 holds by looking at the dual of $\text{MLP}(\rho)$ ($\text{MDP}(\rho)$). Let us associate variables p_{ij} for $(i,j) \in E$ to constraints (41) and variable μ to constraint (42). Then the dual of $\text{MLP}(\rho)$ is:

$$(\text{MDP}(\rho)) \quad \zeta_{\text{MDP}(\rho)} = \max_{p,\mu} \sum_{(i,j)\in E} p_{ij} - k\,\mu \tag{44}$$

$$\text{s.t.} \sum_{(i,j)\in E\cap\text{supp}(\mathbf{R})} p_{ij} - \mu \le \rho\,|\overline{E}\cap\text{supp}(\mathbf{R})| \qquad R\in\mathcal{R} \qquad (45)$$

$$\mu \ge 0,\ p_{ij}\in[0,1] \qquad (i,j)\in E \qquad (46)$$

where $\text{supp}(R)=\{(i,j): r_{ij}=1\}$.

COROLLARY 1. *Let $\boldsymbol{X}$ have isolation number $i(\boldsymbol{X})>k$. Then for all $\rho>0$, MLP(ρ) has objective value at least $\rho\,(i(\boldsymbol{X})-k)$.*

Proof. The proof is a simple modification of Proposition 2's proof. The dual of MLP(ρ) (MDP(ρ)) differs from MDP_F by having the constant value ρ instead of dual variables s_{ij} and constraints (45) instead of (22). Therefore setting $p_{ij}=\rho$ for all $(i,j)\in S$ and 0 otherwise (where S is a maximum isolated set of $\mathbf{X}$), and $\mu=\rho$ gives the required bound of $\rho\,(i(\mathbf{X})-k)$. □

4. Computational approach. It is clearly not practical to solve the master integer program MIP(ρ) or its LP relaxation MLP(ρ) explicitly as the formulation has an exponential number of variables. *Column generation* (CG) is a well-known technique to solve large LPs iteratively by only considering the variables which have the potential to improve the objective function [2]. The column generation procedure is initialised by solving a *Restricted* Master LP (RMLP) which has a small subset of the variables of the full problem. The next step is to identify a missing variable with *negative reduced cost* to be added to RMLP. To avoid considering all missing variables explicitly, 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 RMLP is optimal for the full MLP. In this section, we detail how CG technique can be used to solve the LP relaxation of MIP(ρ) iteratively.

Each *Restricted* MLP(ρ) (RMLP(ρ)) has the same number of constraints as the full MLP(ρ) and all variables ξ_{ij} for $(i,j)\in E$ but it only has a small subset of variables q_r for $r\in\mathcal{R}'\subset\mathcal{R}$ where $|\mathcal{R}'|\ll|\mathcal{R}|$. Recall that each variable q_r corresponds to a rank-1 binary matrix $r\in\mathcal{R}$ which determines the coefficients of q_r in the constraints as well as the objective function. Hence at every iteration of the CG procedure we either need to find a rank-1 binary matrix for which the associated variable has a negative reduced cost, or, prove that no such matrix exists.

4.1. The pricing problem. At the first iteration of CG, RMLP(ρ) may be initialised with $\mathcal{R}'=\emptyset$ or can be warm started by identifying a few rank-1 matrices in $\mathcal{R}$ using a heuristic. After solving the RMLP(ρ) to optimality, one obtains an optimal dual solution $[\boldsymbol{p}^*,\mu^*]$ to the current RMLP(ρ). To identify a missing variable q_r that has negative reduced cost, we solve the following pricing problem (PP):

$$\text{(PP)}\ \ \omega(\mu^*,\boldsymbol{p}^*)=\mu^*-\max_{a,b,y}\sum_{(i,j)\in E}p^*_{ij}y_{ij}-\rho\sum_{(i,j)\in\overline{E}}y_{ij} \qquad (47)$$

$$\text{s.t. } y_{ij}=a_ib_j,\qquad a_i,b_j\in\{0,1\},\qquad i\in[n],j\in[m]. \qquad (48)$$

PP may be formulated as an integer linear program (IP_PP) by using McCormick envelopes [37] (see Section 2.1) to linearise the quadratic constrains to $y_{ij}\in MC(a_i,b_j)$. The objective of PP depends on the current dual solution $[\boldsymbol{p}^*,\mu^*]$ and its optimal solution corresponds to a rank-1 binary matrix $\boldsymbol{ab}^\top=r\in\mathcal{R}$ whose corresponding variable q_r in MLP(ρ) has the smallest reduced cost. If $\omega(\mu^*,\boldsymbol{p}^*)\ge 0$, then the current RMLP($\rho$) does not have any missing variables with negative reduced cost and consequently the current solution of RMLP(ρ) is optimal for MLP(ρ). If $\omega(\mu^*,\boldsymbol{p}^*)<0$, then the variable q_r associated with the rank-1 binary matrix $r=\boldsymbol{ab}^\top$ is added to the next RMLP(ρ) and the procedure is iterated. Moreover, any feasible solution to PP with a negative reduced cost can (also) be added to the RMLP(ρ) to continue the procedure. CG terminates with a proof of optimality if at some iteration we have $\omega(\mu^*,\boldsymbol{p}^*)\ge 0$.

4.2. Solving the master integer program. After the CG process, if the optimal solution of MLP(ρ) is integral, then it also is optimal for MIP(ρ). However, if it is fractional, then this solution only provides a lower bound on the optimal value of MIP(ρ). In this case we obtain an integer feasible solution by solving a *Restricted* MIP(ρ) (RMIP(ρ)) over the rank-1 binary matrices generated by the CG process applied to MLP(ρ). This integer feasible solution is optimal for MIP(ρ) provided that the objective value of RMIP(ρ) is equal to the ceiling of the objective value of MLP(ρ). If this is not the case, one needs to embed CG into a branch-and-bound tree [36] to solve MIP(ρ) to optimality, which is a relatively complicated process and we do not consider it in this paper.

4.3. Computing lower bounds. Note that even if the CG procedure is terminated prematurely, one can still obtain a lower bound on MLP(ρ) and therefore on MIP(ρ) by considering the dual of MLP(ρ). Let the objective value of of the current RMLP(ρ) be

$$\zeta_{\mathrm{RMLP}(\rho)} = \sum_{(i,j)\in E} \xi^*_{ij} + \rho \sum_{(i,j)\in \overline{E}} \sum_{r\in\mathcal{R}_{(i,j)}} q^*_r = \sum_{(i,j)\in E} p^*_{ij} - k\mu^* \tag{49}$$

where $[\xi^*_{ij}, q^*_r]$ is the optimal solution of RMLP(ρ) and $[\boldsymbol{p}^*, \mu^*]$ is the corresponding optimal dual solution which does not necessarily satisfy all of the constraints (45) for MDP(ρ). Now assume that we solve PP to optimality and obtain a rank-1 binary matrix with a negative reduced cost, $\omega(\mu^*, \boldsymbol{p}^*) < 0$. In this case, we can construct a feasible solution $[\boldsymbol{p}, \mu]$ to MDP(ρ) by setting $\boldsymbol{p} := \boldsymbol{p}^*$ and $\mu := \mu^* - \omega(\mu^*, \boldsymbol{p}^*)$ and obtain the following bound on the optimal value $\zeta_{\mathrm{MLP}(\rho)}$ of MLP(ρ),

$$\zeta_{\mathrm{MLP}(\rho)} \geq \sum_{(i,j)\in E} p_{ij} - k\,\mu = \sum_{(i,j)\in E} p^*_{ij} - k\,(\mu^* - \omega(\mu^*, \boldsymbol{p}^*)) = \zeta_{\mathrm{RMLP}(\rho)} + k\,\omega(\mu^*, \boldsymbol{p}^*). \tag{50}$$

If we do not have the optimal solution to PP but have a lower bound $\underline{\omega}(\mu^*, \boldsymbol{p}^*)$ on it, $\omega(\mu^*, \boldsymbol{p}^*)$ can be replaced by $\underline{\omega}(\mu^*, \boldsymbol{p}^*)$ in Equation (50) and the bound on MLP(ρ) still holds. Furthermore, this lower bound on MLP(ρ) naturally provides a valid lower bound on MIP(ρ), thus giving us a bound on the optimality gap.

4.4. Column generation for MLP$_{\mathrm{F}}$. The CG approach is described above as applied to the LP relaxation of MIP(ρ). To apply CG to MLP$_{\mathrm{F}}$ only a small modification needs to be done. The Restricted MLP$_{\mathrm{F}}$ provides dual variables for constraints (18) which are used in the objective of PP for coefficients of y_{ij} $(i,j) \in \overline{E}$.

We note that CG cannot be used to solve the LP relaxation of the *strong formulation* of MIP$_{\mathrm{F}}$ in which constraints (18) are replaced by exponentially many constraints $q_r \leq z_{ij}$ for all $r \in \mathcal{R}_{(i,j)}$ and $(i,j) \in \overline{E}$. This is due to the fact that CG could cycle and generate the same column over and over again. For example, consider applying CG to solve the strong formulation of MLP$_{\mathrm{F}}$ and start with the rank-1 binary matrix of all 1s as the first column associated with variable q_1. The objective value of the corresponding Restricted MLP$_{\mathrm{F}}$ would be $\zeta^{(1)}_{\mathrm{RMLP}} = 0 + |\overline{E}|$ for the solution vector $[\boldsymbol{\xi}^{(1)}, \boldsymbol{z}^{(1)}, \boldsymbol{q}^{(1)}] = [\mathbf{0}, \mathbf{1}, 1]$ as all entries of the input matrix are covered. Adding the same rank-1 binary matrix of all 1s in the next iteration and setting $[q_1, q_2] = [\frac{1}{2}, \frac{1}{2}]$, allows us to keep $\boldsymbol{\xi}^{(2)} = \mathbf{0}$ but reduce the value of $\boldsymbol{z}^{(2)}$ to $\frac{1}{2}\mathbf{1}$ to obtain an objective value $\zeta^{(2)}_{\mathrm{RMLP}} = 0 + \frac{1}{2}|\overline{E}|$. Therefore, repeatedly adding the same matrix of all 1s for t iterations, the objective function would become $\zeta^{(t)}_{\mathrm{RMLP}} = 0 + \frac{1}{t}|\overline{E}|$ for the solution vector $[\boldsymbol{\xi}^{(t)}, \boldsymbol{z}^{(t)}, \boldsymbol{q}^{(t)}] = [\mathbf{0}, \frac{1}{t}\mathbf{1}, \frac{1}{t}\mathbf{1}]$. Consequently, as $t \to \infty$ we would have $\zeta^{(t)}_{\mathrm{RMLP}} \to 0$ and during the column generation process we repeatedly generate the same rank-1 binary matrix.

4.5. An alternative formulation of the pricing problem. Generating rank-1 binary matrices with negative reduced cost efficiently is at the heart of the CG process. For both MLP(ρ) and MLP$_{\mathrm{F}}$, the pricing problem is a Bipartite Binary Quadratic Program (BBQP) which is NP-hard in general [19, 44]. Hence for large $\mathbf{X}$ it may take too long to solve PP to optimality via formulation IP$_{\mathrm{PP}}$ at each iteration. Introducing $\mathbf{H}$ an $n \times m$ matrix with $h_{ij} = p^*_{ij} \in [0,1]$ for $(i,j) \in E$, $h_{ij} = -\rho$ for $(i,j) \in \overline{E}$ and $h_{ij} = 0$ for $(i,j) \notin E \cup \overline{E}$, PP can be written in standard form as

$$(\mathrm{QP}_{\mathrm{PP}}) \quad \omega(\mu^*, \boldsymbol{p}^*) = \mu^* - \max_{\boldsymbol{a}\in\{0,1\}^n, \boldsymbol{b}\in\{0,1\}^m} \boldsymbol{a}^\top \mathbf{H} \boldsymbol{b}. \tag{51}$$

This explicit quadratic form QP$_{\mathrm{PP}}$ is more intuitive for thinking about heuristics than formulation IP$_{\mathrm{PP}}$. If a heuristic approach to PP returns a rank-1 binary matrix with negative reduced cost, then it is valid to add this heuristic solution as a column to the next RMLP. [19] presents several heuristics for BBQP along with a simple but powerful greedy algorithm. In Appendix A we detail this greedy algorithm and some variants of it which we use to provide a warm start to PP at every iteration of CG in Section 5.2.

5. Experiments. The integer programs and column generation approach introduced in the previous sections provide a framework for computing k-BMF with dual bounds. In this section, we present some experimental results to demonstrate the practical applicability of integer programming to obtain low-error factorisations. More specifically we detail our pricing strategies during the column generation process and present a thorough comparison of models MIP$_{\mathrm{F}}$, MIP(ρ) and CIP on synthetic and real world datasets. Our code and data can be downloaded from [24].

5.1. Data. If $\mathbf{X}$ contains rows (or columns) of all zeros, deleting these rows (or columns) leads to an equivalent problem whose solution $\mathbf{A}$ and $\mathbf{B}$ can easily be translated to a solution for the original problem by inserting a row of zeros to $\mathbf{A}$ (respectively a column of zeros to $\mathbf{B}$) in the corresponding place. In addition, if $\mathbf{X}$ contains duplicate rows or columns, by Lemma 1 there is an optimal rank-k factorisation which has the same row-column repetition pattern as $\mathbf{X}$. Hence we solve the problem on a smaller matrix $\mathbf{X}'$ which is obtained from $\mathbf{X}$ by keeping only one copy of each row and column, and use an updated objective function in which every entry is weighted proportional to the number of rows and columns it is contained in $\mathbf{X}$.

5.1.1. Synthetic data. We build our dataset of binary matrices with prescribed sparsity and Boolean rank as follows. To get a matrix $\mathbf{X} \in \{0,1\}^{n\times m}$ with Boolean rank at most κ, first we randomly generate two binary matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ of dimension $n \times \kappa$ and $\kappa \times m$, then compute their Boolean product to get $\mathbf{X}$. This ensures $\mathbf{X}$ has Boolean rank at most κ. To obtain a certain sparsity for $\mathbf{X}$, we control the probability of entries of $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ being zero. More specifically, if we generate $\tilde{a}_{i\ell}$, $\tilde{b}_{\ell j}$ to be zero with probability p, then $x_{ij} = \bigvee_{\ell=1}^{\kappa} \tilde{a}_{i\ell}\tilde{b}_{\ell j}$ is zero with probability $(1-(1-p)^2)^\kappa$. Hence, to obtain $\mathbf{X}$ with σ percent of zeros, we need to generate entries of $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ to be zero with probability $p = 1 - \sqrt{1-(\sigma/100)^{\frac{1}{\kappa}}}$.

We generate matrices as described above with $m = 20$ columns and $\kappa = 10$. The number of rows (n) is set to be $20, 35$ or 50. For each of the three dimensions $(20\times 20, 35\times 20, 50\times 20)$, we generate 10 *sparse* matrices with 75% zeroes and 10 *normal* matrices with 50% zeroes, corresponding to 10 different seed settings in the random number generation. We call this initial set of $2\cdot 3\cdot 10$ matrices the *clean* matrices. Next, we create a set of *noisy* matrices from the clean matrices by randomly flipping 5% of the entries of each matrix. The noisy matrices are not necessarily of Boolean rank at most $\kappa = 10$, but they are at most $0.05\cdot n\cdot m$ squared Frobenius distance away from a Boolean rank 10 matrix. Therefore, our test bed consists of 120 matrices corresponding to 2 noise level settings (*noisy* or *clean*), 2 sparsity levels (*sparse* or *normal*), 3 dimensions $(20\times 20, 35\times 20, 50\times 20)$ and

10 random seeds. Applying the preprocessing steps to our synthetic dataset achieves the largest dimension reduction on clean matrices, while the dimension of noisy matrices scarcely changes. A table summarising the parameters used to generate our data can be found in the Appendix B.

5.1.2. Real world data. We work with eight real world categorical datasets that were downloaded from online repositories [10, 29]. 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 option v_j for C that is $(C)_i = v_j$, then we have $(B_j)_i = 1$ and $(B_\ell)_i = 0$ for all $\ell \neq j \in [N]$. This technique of binarisation of categorical columns has been applied in [25] and [1]. If a row i has a missing value in the column of feature C, we leave the corresponding binary feature columns with missing values in row i. Table 1 shows a short summary of the resulting full-binary datasets used, in-depth details on converting categorical columns into binary, missing value treatment and feature descriptions can be found in Appendix C.

TABLE 1. Summary of binary real world datasets

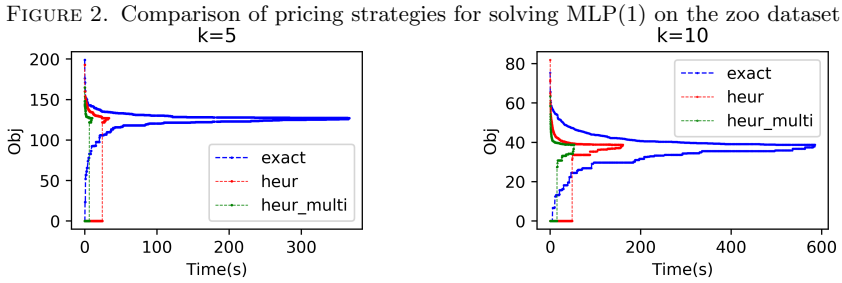
	zoo	tumor	hepatitis	heart	lymp	audio	apb	votes
$n \times m$	101 × 17	339 × 24	155× 38	242× 22	148×44	226 × 92	105× 105	435 ×16
# missing	0	670	334	0	0	899	0	392
%1s	44.3	24.3	47.2	34.4	29.0	11.3	8.0	49.2

5.2. Testing the computational approach to exponential formulation I. Since the efficiency of CG greatly depends on the speed of generating columns, let us illustrate the speed-up gained by using heuristics to solve the pricing problem. At each iteration of CG, 30 heuristic solutions are computed via the heuristics detailed in Appendix A in order to obtain initial feasible solutions to PP. Under *exact* pricing, the best heuristic solution is used as a warm start and IP_{PP} is solved to optimality at each iteration using CPLEX [8]. In simple heuristic (*heur*) pricing, if the best heuristic solution to PP has negative reduced cost then it is directly added to the next RMLP(ρ). If at some iteration, the best heuristic column does not have negative reduced cost, CPLEX is used to solve IP_{PP} to optimality for that iteration. The multiple heuristic (*heur_multi*) pricing strategy 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 RMLP(ρ).

Figure 2 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 (50) are plotted against time. Sharp increases in the dual bound for heuristic pricing strategies correspond to iterations in which CPLEX was used to solve IP_{PP}, as for the evaluation of the dual bound on MLP(1) a lower bound on $\omega(\mu^*, \boldsymbol{p}^*)$ is needed which heuristic solutions do not provide. While we observe a tailing off effect [36] on all three curves, both heuristic pricing strategies provide a significant speed-up from exact pricing, adding multiple columns at each iteration being the fastest.

In order for CG to terminate with a certificate of optimality, at least one pricing problem has to be solved to optimality. Unfortunately for larger datasets we cannot expect this to be achieved in a short amount of time. Therefore, we change the multiple heuristic pricing strategy to get a pricing strategy that we use in the rest of the experiments as follows. We impose an overall fixed time limit on the CG process and use the barrier method in CPLEX as the LP solver for RMLP at each iteration. At each iteration of CG, we add up to 2 columns with the most negative reduced cost to the next RMLP. If at an iteration, heuristics for PP do not provide a column with negative reduced cost and CPLEX has to be used to improve the heuristic solution, we do not solve IP_{PP}

FIGURE 2. Comparison of pricing strategies for solving MLP(1) on the zoo dataset



to optimality but abort CPLEX after 25 seconds 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 we wish to focus more on computing a stronger dual bound on MLP, we may continue solving IP_{PP} via CPLEX even when a heuristic negative reduced cost solution is available.

5.2.1. MLP(1) vs MLP_{F}. In this section we compare the LP relaxations of MIP(1) and MIP_{F}. According to Proposition 5 the optimal solution of MLP_{F} is equivalent to MLP($\frac{1}{k}$) and hence we solve MLP($\frac{1}{k}$) which has fewer variables and constraints than MLP_{F}. To solve MLP(1) and MLP($\frac{1}{k}$), we start off from 0 rank-1 binary matrices so $\mathcal{R}' = \emptyset$ in the first RMLP and set a total time limit of 600 seconds, so we either solve MLP to optimality under 600 seconds or run out of time and compute the gap between the last RMLP and the best dual bound MDP according to formula $100(\zeta_{\text{RMLP}} - \zeta_{\text{MDP}})/\zeta_{\text{RMLP}}$. As MLP(1) and MLP($\frac{1}{k}$) correspond to the LP relaxations of MIP(1) and MIP_{F} with integral objective coefficients, any fractional dual bound may be rounded up to give a valid bound on the master IP. Therefore, we stop CG whenever the ceiling of the dual bound reaches the objective value of RMLP.

FIGURE 3. Time taken in seconds to solve MLP(1) and MLP($\frac{1}{k}$) via CG on synthetic data

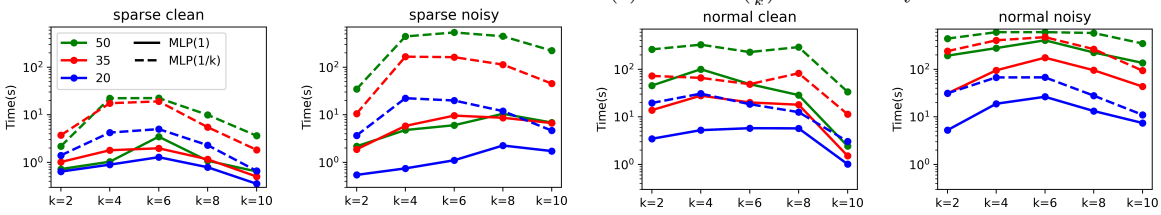


Figure 3 shows the time taken in seconds on a logarithmic scale to solve MLP(1) and MLP($\frac{1}{k}$) via CG for $k = 2, 4, \ldots, 10$ on the synthetic matrices. Each line corresponds to the average taken over 10 instances with the same dimension, sparsity and noise level. Blue lines correspond to matrices of dimension 20×20, red to 35×20 and green to 50×20. Solid lines are used for MLP(1) and dashed for MLP($\frac{1}{k}$). First, we observe that it is significantly faster to solve both MLPs on *sparse* and *clean* matrices as opposed to *normal* and *noisy* ones of the same dimension. Preprocessing is more effective in reducing the dimension for clean matrices in comparison to noisy ones (see Table 6 in Appendix B) which explains why noisy instances take longer. In addition, both MLP(1) and MLP($\frac{1}{k}$) have a number of variables and constraints directly proportional to non-zero entries of the input matrix, hence a sparse input matrix requires a smaller problem to be solved. Second, we see that $k = 10$ are solved somewhat faster. This can be explained by all matrices in our test bed being generated to have Boolean rank at most 10. For a rank-10 factorisation of *clean* matrices without noise we get 0 factorisation error under both models MIP(1) and MIP_F and hence LP relaxation objective value 0. For *noisy* matrices we observe the error to be in line with our expectation of

$0.05 \cdot n \cdot m$. We observe that in some cases it takes significantly longer to solve MLP($\frac{1}{k}$), and in all ten instances of 50×20 *normal-noisy* matrices MLP($\frac{1}{k}$) for $k = 6$ runs out of the time budget of 600 sec. In the experiments, we see the amount of time CG takes is directly proportional to the number of columns generated, MLP($\frac{1}{k}$) generating significantly more columns than MLP(1).

5.2.2. Obtaining integral solutions. Once we obtain some rank-1 binary matrices (i.e. columns) via CG applied to a master LP, we can obtain an integer feasible solution by solving either of the master IPs over the columns available. Here we explore obtaining integer feasible solutions by solving MIP(1) and MIP$_F$ over the columns generated by formulations MLP(1) and MLP($\frac{1}{k}$). We use CPLEX as our integer program solver and set a total time limit of 300 seconds.

FIGURE 4. Factorisation error in $\|\cdot\|_F^2$ of integral solutions by MIP(1) from columns by MLP(1) and MLP($\frac{1}{k}$)

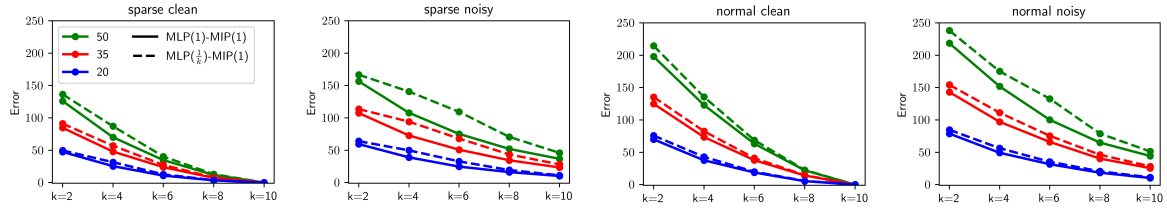


Figure 4 shows the factorisation error in $\|\cdot\|_F^2$ of integer feasible solutions obtained by solving MIP(1) over columns generated by MLP(1) and MLP($\frac{1}{k}$). As previously, each line corresponds to the average taken over 10 matrices with same dimension, sparsity and noise level. Solid lines are used to denote where the columns used were generated by MLP(1) and dashed where by MLP($\frac{1}{k}$). Comparing the error values of the dashed and solid lines we draw a crucial observation: columns generated by MLP(1) seem to be a better basis for obtaining low-error integer feasible solutions than columns by MLP($\frac{1}{k}$). We suspect this is the case as in the majority of rank-k factorisations most entries are only covered by a few rank-1 binary matrices whereas MLP($\frac{1}{k}$) favours rank-1 matrices which heavily cover 0 entries of the input matrix. This is because the coefficient in MLP($\frac{1}{k}$)'s objective function corresponding to a zero entry at position (i, j) is only $\frac{1}{k}\times$(number of rank-1 matrices covering (i, j)), hence it is cheaper for MLP($\frac{1}{k}$) to cover a 0 by a few (less than k) rank-1 matrices than to leave any 1s uncovered. We also conducted a set of experiments using formulation MIP$_F$ and we see that the factorisation error when using formulation MIP(1) to obtain the integral solutions is extremely close to that of MIP$_F$, see Appendix D Tables 7 and 8 for the precise difference in the factorisation error between the two master IPs.

FIGURE 5. Time taken in seconds to solve MIP(1) and MIP$_F$ on columns generated by MLP(1)

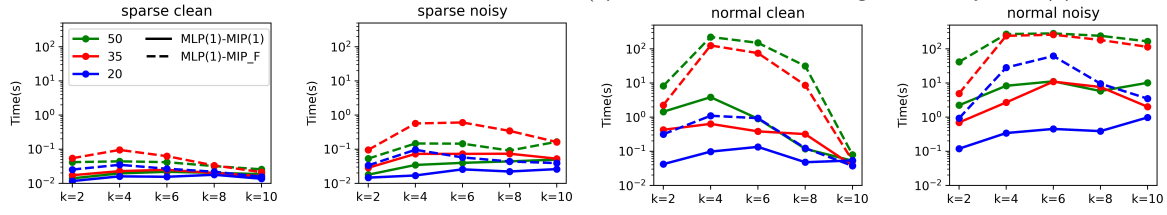


Figure 5 shows the time taken to solve the master IPs on columns generated by MLP(1). We observe that MIP(1) takes notably faster to solve than MIP$_F$ and on most normal-noisy matrices MIP$_F$ runs out of the time budget of 300 seconds. Solving both master IPs on columns by MLP($\frac{1}{k}$) also shows us that while solving MIP(1) over a larger set of columns adds only a few seconds for

most instances, MIP_F runs out of the time budget of 300 secs in about half the cases, see Appendix Table 8. These observations suggest using MIP(1) to find integer feasible solutions in the future as the solution quality is extremely close to that of MIP_F but at a fraction of computational effort.

5.3. Accuracy and speed of the IP Formulations. In this section we computationally compare the integer programs introduced in Section 2 and 3. CIP due to its polynomial size can be directly given to a general purpose IP solver like CPLEX and we set a time limit of 600 seconds on its running time. We expect solution times for CIP to grow proportional to k and density of $\mathbf{X}$ according to Proposition 1. Similarly, we may try to attack the exponential formulation EIP directly by CPLEX. Since however EIP requires the complete enumeration of 2^m binary vectors for an input matrix $\mathbf{X}$ of size $n \times m$ we can only solve its root LP under 600 seconds in a very few cases. For these few cases however, we observe the objective value of ELP to agree with $\text{MLP}(\frac{1}{k})$, which gives an experimental confirmation of Proposition 3. In the following experiments, formulation MIP_F is used on columns generated by $\text{MLP}(\frac{1}{k})$, while MIP(1) on columns by MLP(1). The final solution of MIP(1) is evaluated under the original $\|\cdot\|_F^2$ objective and that error is reported. As previously, the master LPs are solved with a time limit of 600 seconds and the master IPs with an additional time limit of 300 seconds.

Table 2 shows the factorisations error in $\|\cdot\|_F^2$ obtained by MIP_F, MIP(1) and CIP and Table 3 shows the corresponding solution times in seconds. Each row of Table 2 and 3 corresponds to the average of 10 synthetic matrices of the same size, sparsity and noise. The lowest error results are indicated in boldface. We observe that MIP(1) provides the lowest error factorisation in most cases, but CIP gives the lowest error when only looking at $k = 2$. The significantly higher error values of MIP_F are due to the lower quality columns generated by $\text{MLP}(\frac{1}{k})$ on which it is solved. We emphasise that we do not do branch-and-price when solving MIP(1) or MIP_F. Table 3 shows that MIP(1) is the fastest in all cases, while CIP runs out of its time limit on all noisy instances for $k = 5, 10$. In conclusion, CIP provides very accurate solutions for $k = 2$ but it is slower to solve than MIP(1), while for larger k's MIP(1) dominates in both accuracy and speed.

5.4. Binary matrix completion. In this section we explore how successful our approach is at recovering missing entries of incomplete binary matrices. We create an incomplete dataset of our synthetic matrices by deleting $5, 10, \ldots, 30\%$ of the entries of each matrix. This way, after computing a rank-k factorisation of the incomplete matrix, we can easily compare to the corresponding original matrix to see how many of the entries we have recovered successfully. Since our synthetic matrices are generated to be of Boolean rank at most 10, we cannot expect to recover all the entries by a rank-k completion with $k < 10$ and thus we perform the experiments with $k = 10$.

Figure 6 shows the reconstruction percentage against the percentage of missing entries when solving MIP(1) on columns generated by MLP(1) on the incomplete matrices. As previously, the three colours correspond to dimensions of the matrices: green to 50×20, red to 35×20 and blue to 20×20. We define the percentage of reconstruction as $100 * (1 - \|\mathbf{X} - \mathbf{A} \circ \mathbf{B}\|_F^2 / \|\mathbf{X}\|_F^2)$ where $\mathbf{X}$ is the original complete matrix and $\mathbf{A} \circ \mathbf{B}$ is the rank-k factorisation of the incomplete matrix. As expected the recovery percentage decreases with the percentage of missing entries and clean matrices are better recovered than noisy ones. All in all, we see a very high percentage of the entries can be recovered by MIP(1).

5.5. Comparing integer programming approaches against heuristics. In this section, we compare our integer programming approaches against the most widely used k-BMF heuristics on real-world datasets. The heuristic algorithms we evaluate include the ASSO algorithm [38, 39], the alternating iterative local search algorithm (ASSO++) of [1] which uses ASSO as a starting point, and the penalty objective formulation (pymf) of [52] via the implementation of [46]. We also compute rank-k NMF, scale rank-1 factors and then binarise them to obtain a k-BMF. The

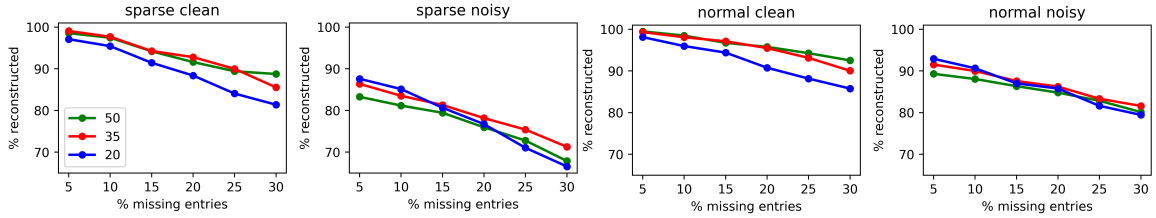
TABLE 2. Factorisation error in $\|\cdot\|_F^2$ of solutions obtained via formulations MIP_F, MIP(1) and CIP_F

data	k=2			k=5			k=10		
(n-sparsity-noise)	MIP_F	MIP(1)	CIP	MIP_F	MIP(1)	CIP	MIP_F	MIP(1)	CIP
20-sparse-clean	49.6	**47.4**	**47.4**	20.8	**16.6**	16.7	**0.0**	**0.0**	**0.0**
20-sparse-noisy	64.0	59.5	**59.3**	42.6	**30.3**	30.7	11.2	**10.2**	10.3
20-normal-clean	75.0	70.0	**68.7**	30.6	27.7	**26.5**	0.3	0.3	**0.0**
20-normal-noisy	84.6	78.9	**77.2**	47.3	40.2	**40.1**	11.2	**10.7**	11.2
35-sparse-clean	90.9	**84.7**	**84.7**	39.1	**34.5**	34.9	0.1	**0.0**	**0.0**
35-sparse-noisy	113.4	107.5	**106.9**	84.4	**60.5**	61.7	28.4	**23.3**	27.1
35-normal-clean	134.2	125.0	**121.7**	64.5	54.1	**53.4**	**0.0**	**0.0**	**0.0**
35-normal-noisy	153.6	143.1	**139.1**	101.7	**80.3**	81.7	31.1	**25.5**	31.1
50-sparse-clean	136.0	126.1	**125.6**	61.4	**50.6**	51.5	0.1	**0.0**	**0.0**
50-sparse-noisy	166.2	**156.5**	156.7	135.0	**89.8**	93.9	49.6	**36.7**	41.4
50-normal-clean	215.1	198.0	**194.3**	106.1	**91.0**	95.0	**0.0**	**0.0**	**0.0**
50-normal-noisy	237.2	218.6	**214.2**	168.6	123.9	**123.4**	62.2	**44.3**	61.3

TABLE 3. Time in seconds to obtain solutions in Table 2 via formulations MIP_F, MIP(1) and CIP_F

data	k=2			k=5			k=10		
(n-sparsity-noise)	MIP_F	MIP(1)	CIP	MIP_F	MIP(1)	CIP	MIP_F	MIP(1)	CIP
20-sparse-clean	1.1	0.4	1.6	4.6	0.4	169.7	0.7	0.4	1.9
20-sparse-noisy	2.7	0.6	21.8	233.7	0.8	601.6	10.9	1.8	602.9
20-normal-clean	15.2	3.5	56.2	303.2	5.4	600.3	3.3	1.0	15.8
20-normal-noisy	31.3	5.4	295.5	336.6	17.6	600.8	65.2	8.0	602.0
35-sparse-clean	4.0	0.8	17.3	108.4	0.9	449.8	1.9	0.5	5.3
35-sparse-noisy	12.1	1.9	147.8	514.0	6.4	602.3	275.1	6.8	605.2
35-normal-clean	76.0	14.2	188.6	378.5	21.8	600.8	23.2	1.6	80.6
35-normal-noisy	195.3	31.8	589.7	739.3	132.1	600.7	394.7	45.3	602.4
50-sparse-clean	2.6	0.6	21.9	176.3	1.1	519.9	3.8	0.7	12.9
50-sparse-noisy	28.1	2.2	285.4	827.7	6.6	602.3	523.9	6.9	605.1
50-normal-clean	362.0	46.8	509.9	692.1	153.6	602.1	187.2	2.5	139.4
50-normal-noisy	601.6	194.8	578.2	903.9	341.1	601.0	649.8	146.2	601.6

FIGURE 6. Rank-10 binary matrix completion of artificial matrices with $5-30\%$ missing entries



exact details and parameters used in the computations can be found in Appendix E. In addition, we use a new heuristic which sequentially finds k rank-1 binary matrices using any heuristic for Bipartite Binary Quadratic Programming as a subroutine. We refer to this heuristic outlined in

Algorithm 1 as *k-Greedy* as the subroutine we use to compute the rank-1 binary matrices is the greedy algorithm of [19].

Algorithm 1: Greedy algorithm for k-BMF (k-Greedy)

Input: $\mathbf{X} \in \{0,1\}^{n\times m}$, $k \in \mathbb{Z}_+$.
Set $\mathbf{H} \in \{-1,0,1\}^{n\times m}$ to $h_{ij} = 2x_{ij} - 1$ for $(i,j) \in E \cup \overline{E}$ and $h_{ij} = 0$ otherwise.
for $\ell \in [k]$ **do**
$\boldsymbol{a}, \boldsymbol{b} = \text{BBQP}(\mathbf{H})$ `// compute a rank-1 binary matrix via any algorithm for BBQP`
$\mathbf{A}_{:,\ell} = \boldsymbol{a}$
$\mathbf{B}_{\ell,:} = \boldsymbol{b}^\top$
$\mathbf{H}[\boldsymbol{a}\boldsymbol{b}^\top == 1] = 0$ `// set entries of H to zero that are covered`
end
Output: $\mathbf{A} \in \{0,1\}^{n\times k}$, $\mathbf{B} \in \{0,1\}^{k\times m}$

We solve CIP using CPLEX with a time limit of 20 mins and provide the heuristic solution of k-Greedy as a warm start to it. The column generation approach results are obtained by generating columns for 20 mins using formulation MLP(1) with a warm start of initial rank-1 binary matrices obtained from k-Greedy, then solving MIP(1) over the generated columns with a time limit of 10 mins. Table 4 shows the factorisation error in $\|\cdot\|_F^2$ after evaluating the above described methods on all real-world datasets without missing entries 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 8 out of 12 instances.

TABLE 4. Comparison of factorisation error in $\|\cdot\|_F^2$ for two IP based methods and five k-BMF heuristics

		MIP(1)	CIP	ASSO++	k-Greedy	pymf	ASSO	NMF
k=2	zoo	272	**271**	276	323	274	367	281
	heart	**1185**	1187	1187	1187	1241	1251	1267
	lymp	1192	**1184**	1202	1201	1225	1352	1272
	apb	**776**	**776**	**776**	**776**	794	778	808
k=5	zoo	**126**	129	133	218	153	354	140
	heart	**737**	738	738	738	813	887	782
	lymp	**982**	1026	1039	1053	1067	1484	1103
	apb	**684**	688	694	688	733	719	721
k=10	zoo	**39**	72	55	175	80	377	51
	heart	425	529	**419**	565	483	694	450
	lymp	**728**	829	812	859	952	1525	821
	apb	**573**	605	591	606	611	661	617

While integer programming based approaches are able to handle missing entries by simply setting the objective coefficients of the missing entries to 0, the k-BMF heuristics ASSO, ASSO++ and pymf cannot so simply be adjusted. Non-negative matrix factorisation however, has an available implementation that can handle missing entries [32, 33]. Our next experiment compares our integer programming approaches against k-Greedy and NMF on the real datasets that have missing entries. Table 5 shows the results with the lowest error results indicated in boldface. For $k = 2$, k-Greedy provides very accurate solutions which MIP(1) and CIP fail to improve on in 3 out of 4 instances.

For $k = 5, 10$ however, MIP(1) produces notably lower error factorisations than the other methods.

TABLE 5. Comparison of factorisation error in $\|\cdot\|_F^2$ for real-world data with missing entries

		MIP(1)	CIP	k-Greedy	NMF
k=2	tumor	**1352**	**1352**	**1352**	1529
	hepatitis	**1264**	1344	1416	1304
	audio	**1419**	**1419**	**1419**	1876
	votes	**1246**	**1246**	**1246**	1268
k=5	tumor	**962**	993	1004	1229
	hepatitis	**1138**	1229	1238	1172
	audio	**1064**	1078	1094	1634
	votes	**779**	853	853	900
k=10	tumor	**514**	632	646	851
	hepatitis	**907**	1048	1056	1013
	audio	**765**	881	881	1580
	votes	**240**	701	706	815

6. Conclusions and further work. In this paper we investigated the rank-k binary matrix factorisation problem from an integer programming perspective. We analysed a compact and two exponential size integer programming formulations for the problem and made a comparison on the strength of the formulations' LP-relaxations. We introduced a new objective function, which slightly differs from the traditional squared Frobenius objective in attributing a weight to zero entries of the input matrix that is proportional to the number of times the zero is erroneously covered in a rank-k factorisation. In addition, we discussed a computational approach based on column generation to solve one of the exponential size formulations and reported several computational experiments to demonstrate the applicability of our formulations on real world and artificial datasets.

Our column generation approach is rather computationally challenging and the bottleneck is to compute a tight lower bound on the pricing problem which is needed to determine the master dual bound in Equation (50). Therefore, it seems that larger datasets are currently out of reach for our methods. If however, one needs an accurate factorisation on moderate size matrices and not a tight optimality gap, our real word data experiments show that our methods provide the lowest error factorisations in most instances with the 600 seconds time limit.

To be able to obtain tighter master dual bounds, future research directions could include developing faster exact algorithms for the pricing problem. In addition, considering semidefinite programming relaxations of the pricing problem to obtain stronger lower bounds could be an interesting avenue to explore. Once computing good quality lower bounds on the pricing problem is faster, a full branch-and-price implementation would be interesting to explore. In addition, as rank-1 binary tensor factorisation has been recently explored from an integer programming perspective in [9], a natural extension would be to adapt the models presented in this paper to the rank-k binary tensor factorisation problem.

Appendix A: Heuristics for the pricing problem. The greedy algorithm of [19] to solve the Bipartite Binary Quadratic Program in Equation (51) aims to set entries of $\boldsymbol{a}$ and $\boldsymbol{b}$ to 1 which correspond to rows and columns of $\mathbf{H}$ with the largest positive weights. In the first phase of the algorithm, the row indices i of $\mathbf{H}$ 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 $\sum_{j=1}^m \max(0, \sum_{\ell=1}^{i-1} a_\ell h_{\ell j}) < \sum_{j=1}^m \max(0, \sum_{\ell=1}^{i} a_\ell h_{\ell j})$ and 0 otherwise. In the second phase, b_j is set to 1 if $(\boldsymbol{a}^\top H)_j > 0$, 0 otherwise. An efficient implementation of the greedy algorithm due to [19] is given in Algorithm 2.

Algorithm 2: Greedy Algorithm for BBQP

Input: $\mathbf{H} \in \mathbb{R}^{n \times m}$

Phase I. Order $i \in [n]$ so that $\gamma_i^+ \geq \gamma_{i+1}^+$.
Set $\boldsymbol{a} = \mathbf{0}_n$, $\boldsymbol{s} = \mathbf{0}_m^\top$.
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 $a_i = 1$, $\boldsymbol{s} = \boldsymbol{s} + \mathbf{H}_{i,:}$
end
end

Phase II.
Set $\boldsymbol{b} = \mathbf{0}_m$.
for $j \in [m]$ **do**
if $\boldsymbol{s}_j > 0$ `// s is equal to` $\boldsymbol{a}^\top$`H` **then**
Set $b_j = 1$
end
end
Output: $\boldsymbol{a} \in \{0,1\}^n, \boldsymbol{b} \in \{0,1\}^m$

There are many variants of Algorithm 2 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 $\mathbf{H}$ 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 $\mathbf{H}$ and $\mathbf{H}^\top$. In general, it is recommended to start the first phase on the smaller dimension [19]. Third, the solution from Algorithm 2 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. We summarise this alternating heuristic in Algorithm 3

In Section 5.2 we use the above described heuristics for the pricing problem in column generation. At each iteration of the column generation procedure, 30 variants of Algorithm 2 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.

Appendix B: Synthetic data. Table 6 gives a summary of the parameters used to generate our synthetic dataset. For a synthetic binary matrix $\mathbf{X}$, $n \times m$ is the dimension of $\mathbf{X}$, κ is the Boolean rank which was used to generate $\mathbf{X}$, and $n' \times m'$ is the dimension obtained after removing zero and duplicate row and columns of $\mathbf{X}$. Our synthetic data can be downloaded from [24].

Appendix C: Real world data. Our binarised real world data is available for download at [24]. The following datasets were used in the experiments:

- The Zoo dataset (*zoo*) [13] 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 matrix is 101×17.

Algorithm 3: Alternating Heuristic for BBQP

Input: $\mathbf{H} \in \mathbb{R}^{n\times m}$, $\boldsymbol{a}^{(0)} \in \{0,1\}^n, \boldsymbol{b}^{(0)} \in \{0,1\}^m$;
for $\ell = 1, 2, \ldots$ **do**
$\boldsymbol{a}^{(\ell)}[\mathbf{H}\boldsymbol{b}^{(\ell-1)} > 0] = 1$;
$\boldsymbol{a}^{(\ell)}[\mathbf{H}\boldsymbol{b}^{(\ell-1)} \leq 0] = 0$;
if $\boldsymbol{a}^{(\ell)} == \boldsymbol{a}^{(\ell-1)}$ **then**
Break
end
$\boldsymbol{b}^{(\ell)}[(\boldsymbol{a}^{(\ell)})^\top \mathbf{H} > 0] = 1$;
$\boldsymbol{b}^{(\ell)}[(\boldsymbol{a}^{(\ell)})^\top \mathbf{H} \leq 0] = 0$;
if $\boldsymbol{b}^{(\ell)} == \boldsymbol{b}^{(\ell-1)}$ **then**
Break
end
end
Output: $\boldsymbol{a}^{(\ell)} \in \{0,1\}^n, \boldsymbol{b}^{(\ell)} \in \{0,1\}^m$

TABLE 6. Parameters of the synthetic dataset

(n-sparsity-noise)	$n \times m$	κ	0s%	noise%	#instances	$n' \times m'$
20-sparse-clean	20×20	10	75	0	10	14×15
20-sparse-noisy				5		19×19
20-normal-clean			50	0		18×18
20-normal-noisy				5		19×20
35-sparse-clean	35×20	10	75	0	10	22×15
35-sparse-noisy				5		31×19
35-normal-clean			50	0		29×18
35-normal-noisy				5		34×20
50-sparse-clean	50×20	10	75	0	10	30×15
50-sparse-noisy				5		45×20
50-normal-clean			50	0		40×18
50-normal-noisy				5		48×20

- The Primary Tumor dataset (*tumor*) [22] 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 left as missing values. The final dimension of the binary matrix is 339×24 with 670 missing values.
- The Hepatitis dataset (*hepat*) [17] 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 missing values in the raw dataset are left as missing in the binary dataset as well. The final dimension of the binary matrix is 155×38 with 334 missing values.

- The SPECT Heart dataset (*heart*) [5] 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 binary matrix is 242×22.
- The Lymphography dataset (*lymp*) [23] 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 dimension of the fully binary matrix is 148×44.
- The Audiology Standardized dataset (*audio*) [45] contains 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 missing values in the raw dataset are left as missing in the binary dataset as well. The final dimension of the binary matrix is 226×92 with 899 missing values.
- The Amazon Political Books dataset (*books*) [29] 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 dimension of the binary matrix is 105×105.
- The 1984 United States Congressional Voting Records dataset (*votes*)[47] 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 16 binary features taking value 1 for "voted for", value 0 for "voted against" and a missing value indicates "did not vote". The final dimension of the binary matrix is 435×16 with 392 missing values.

Appendix D: Obtaining integer feasible solutions. In this section we give additional numerical results supporting our conclusions drawn in Section 5.2.2. Table 7 shows the factorisation error measured in $\|\cdot\|_F^2$ of integer feasible solutions obtained by solving MIP(1) and MIP$_F$ over columns generated by MLP(1). MIP(1) takes significantly faster to solve than MIP$_F$ but the absolute difference in error between solutions produced by MIP(1) and MIP$_F$ is at most 1, except for the last row in column $k = 5$ where MIP$_F$ runs out of the time budget of 300 seconds and produces higher error solutions than MIP(1).

Table 8 shows the result of an analogous experiment where the columns used are generated by MLP($\frac{1}{k}$). Since MLP($\frac{1}{k}$) is slower to solve than MLP(1), more columns are generated during CG and the master IPs have a harder task on selecting k columns from a larger set of columns in Table 8. However, while solving MIP(1) over a larger set of columns adds only a few seconds for most instances, MIP$_F$ runs out of the time budget of 300 secs in about half the cases. This is also demonstrated in the error difference, with solutions by MIP(1) having smaller error than solutions by MIP$_F$ in most cases.

Appendix E: Heuristics for k-BMF. The following methods were evaluated for the comparison in Tables 4 and 5. Our code is available at [24].

- For the alternating iterative local search algorithm of [1] (ASSO++) we obtained the code from the author's github page, see the reference. The code implements two variants of the algorithm and we report the smaller error solution from two variants of it.
- For the method of [52], we used a python implementation in the package `pymf`, see [46] and we ran it for 10000 iterations.

TABLE 7. Error in $\|\cdot\|_F^2$ (and runtime in seconds) of integer solutions by MIP(1) and MIP_F on columns by MLP(1)

data	k=2		k=5		k=10	
(n-sparsity-noise)	MIP(1)	MIP_F	MIP(1)	MIP_F	MIP(1)	MIP_F
20-sparse-clean	47 (0.0)	47 (0.0)	16 (0.0)	16 (0.0)	0 (0.0)	0 (0.0)
20-sparse-noisy	59 (0.0)	59 (0.0)	30 (0.0)	30 (0.0)	10 (0.0)	10 (0.0)
20-normal-clean	70 (0.0)	**69** (0.3)	27 (0.1)	27 (2.7)	0 (0.0)	0 (0.0)
20-normal-noisy	78 (0.1)	78 (0.9)	40 (0.5)	**39** (76.5)	10 (0.5)	10 (3.4)
35-sparse-clean	84 (0.0)	84 (0.1)	34 (0.0)	34 (0.1)	0 (0.0)	0 (0.0)
35-sparse-noisy	107 (0.0)	107 (0.1)	60 (0.0)	60 (0.6)	23 (0.1)	23 (0.2)
35-normal-clean	125 (0.4)	**124** (2.2)	54 (0.8)	**53** (154.8)	0 (0.0)	0 (0.1)
35-normal-noisy	143 (0.6)	**141** (4.9)	80 (4.1)	80 (245.4)	25 (2.0)	**24** (114.2)
50-sparse-clean	126 (0.0)	126 (0.0)	50 (0.0)	50 (0.1)	0 (0.0)	0 (0.0)
50-sparse-noisy	156 (0.0)	156 (0.1)	89 (0.0)	89 (0.2)	36 (0.0)	36 (0.2)
50-normal-clean	198 (1.4)	**197** (8.2)	91 (30.9)	91 (173.4)	0 (0.1)	0 (0.1)
50-normal-noisy	218 (2.2)	218 (41.4)	**123** (39.7)	126 (271.1)	44 (10.1)	44 (165.8)

TABLE 8. Error in $\|\cdot\|_F^2$ (and runtime in seconds) of integer solutions by MIP(1) and MIP_F on columns by MLP($\frac{1}{k}$)

data	k=2		k=5		k=10	
(n-sparsity-noise)	MIP(1)	MIP_F	MIP(1)	MIP_F	MIP(1)	MIP_F
20-sparse-clean	50 (0.0)	50 (0.2)	21 (0.0)	21 (2.6)	0 (0.0)	0 (0.0)
20-sparse-noisy	64 (0.0)	64 (0.6)	**42** (0.1)	43 (219.0)	11 (0.2)	11 (6.3)
20-normal-clean	76 (0.2)	**75** (3.9)	**30** (0.5)	31 (289.6)	0 (0.1)	0 (0.2)
20-normal-noisy	85 (0.3)	85 (6.3)	47 (1.2)	47 (300.4)	11 (0.6)	11 (54.2)
35-sparse-clean	91 (0.0)	91 (1.5)	39 (0.2)	39 (98.9)	0 (0.1)	0 (0.1)
35-sparse-noisy	114 (0.1)	113 (3.1)	**81** (0.5)	84 (300.7)	28 (0.3)	28 (229.9)
35-normal-clean	136 (1.0)	**134** (19.1)	**61** (2.0)	65 (300.8)	0 (0.8)	0 (11.9)
35-normal-noisy	154 (1.6)	154 (58.9)	**93** (6.2)	102 (301.3)	**28** (2.1)	31 (301.0)
50-sparse-clean	137 (0.0)	**136** (0.8)	61 (0.2)	61 (160.0)	0 (0.8)	0 (0.2)
50-sparse-noisy	167 (0.1)	**166** (6.5)	**128** (0.7)	135 (301.5)	**46** (0.6)	50 (301.5)
50-normal-clean	215 (2.2)	215 (131.6)	**100** (34.4)	106 (302.1)	0 (0.8)	0 (153.7)
50-normal-noisy	238 (5.7)	**237** (226.4)	**149** (95.8)	169 (302.9)	**51** (39.4)	62 (302.5)

- We evaluated the heuristic method ASSO [38] 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 form the webpage of the author: `https://people.mpi-inf.mpg.de/ pmiettin/src/DBP-progs/`. We observe that ASSO does not return monotone solutions and sometimes we get a higher error solution for a higher value of k.
- In the case of no missing entries in the binary matrix, we used the function `non_negative_factorization` from the `sklearn.decomposition` module in python for the computation of rank-k NMF. We tried all 4 possible initialisation methods: 'nndsvda', 'nndsvd', 'nndsvdar' and 'random'. For the recommendation of one of our reviewers, after obtaining the k-NMF we scale each rank-1 factor to have the same max value on the left and right hand side. Then, we binarise each rank-1 factor with a threshold of $\delta \in \{0.1, 0.2, \ldots, 0.9\}$. In Table 4 we report the best result over all these parameter settings.

  As the above python function does not allow missing entries, for incomplete binary matrices we used a Matlab implementation of NMF [33, 32]. Only random initialisation method was

available for this implementation and we used 11 different random seeds. Then we performed the same scaling and thresholding as described above and report the best result over all parameter settings in Table 5.

- The heuristic k-greedy algorithm was ran with 70 random seeds and the subroutine for BBQP used the greedy and alternating algorithms for BBQP given in Algorithms 2, 3. In addition, the k-greedy algorithm can be run on a preprocessed or original matrix and we tried both ways. For each instance the lowest error factorisation is reported.

Acknowledgments. During the completion of this work R.A.K was supported by a doctoral scholarship from The Alan Turing Institute and the Office for National Statistics.

References

[1] Barahona F, Goncalves J (2019) Local search algorithms for binary matrix factorization. URL https://github.com/IBM/binary-matrix-factorization/blob/master/code, last accessed on 2020-04-21.

[2] Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46(3):316–329, URL http://dx.doi.org/10.1287/opre.46.3.316.

[3] Beckerleg M, Thompson A (2020) A divide-and-conquer algorithm for binary matrix completion. *Linear Algebra and its Applications* 601:113–133, ISSN 0024-3795, URL http://dx.doi.org/10.1016/j.laa.2020.04.017.

[4] Chalermsook P, Heydrich S, Holm E, Karrenbauer A (2014) Nearly tight approximability results for minimum biclique cover and partition. Schulz AS, Wagner D, eds., *Algorithms - ESA 2014*, 235–246 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-662-44777-2.

[5] Cios KJ, Kurgan LA (2001) Uci machine learning repository: Spect heart data. URL https://archive.ics.uci.edu/ml/datasets/spect+heart, last accessed on 2020-06-11.

[6] Cohen JE, Rothblum UG (1993) Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and its Applications* 190:149–168, ISSN 0024-3795, URL http://dx.doi.org/https://doi.org/10.1016/0024-3795(93)90224-C.

[7] Conforti M, Cornuejols G, Zambelli G (2014) *Integer Programming* (Springer Publishing Company, Incorporated), ISBN 3319110071, 9783319110073.

[8] CPLEX Optimization (2018) *Using the CPLEX Callable Library, V.12.8*. CPLEX Optimization, Inc., Incline Village, NV.

[9] Del Pia A, Khajavirad A (2022) Rank-one boolean tensor factorization and the multilinear polytope. URL http://dx.doi.org/10.48550/ARXIV.2202.07053.

[10] Dua D, Graff C (2017) UCI machine learning repository. URL http://archive.ics.uci.edu/ml, last accessed on 2020-06-11.

[11] Fiorini S, Guo K, Macchia M, Walter M (2019) Lower bound computations for the nonnegative rank. *Proceedings of the 17th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, 41–44, https://www.ctw.ewi.utwente.nl/CTW2019ProceedingsFinal.pdf.

[12] Fiorini S, Kaibel V, Pashkovich K, Theis DO (2013) Combinatorial bounds on nonnegative rank and extended formulations. *Discrete Mathematics* 313(1):67–83, ISSN 0012-365X, URL http://dx.doi.org/https://doi.org/10.1016/j.disc.2012.09.015.

[13] 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.

[14] Garey MR, Johnson DS (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness* (New York, NY, USA: W. H. Freeman & Co.), ISBN 0716710455.

[15] Gillis N, Vavasis SA (2018) On the complexity of robust pca and l1-norm low-rank matrix approximation. *Mathematics of Operations Research* 43(4):1072–1084, URL http://dx.doi.org/10.1287/moor.2017.0895.

[16] Golub GH, Van Loan CF (1996) *Matrix Computations* (USA: Johns Hopkins University Press), 3rd edition, ISBN 0801854148.

[17] 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.

[18] Gregory DA, Pullman NJ (1983) Semiring rank: Boolean rank and nonnegative rank factorizations. *Journal of Combinatorics, Information & System Sciences* 8:223–233.

[19] Karapetyan D, Punnen AP (2013) Heuristic algorithms for the bipartite unconstrained 0-1 quadratic programming problem. arXiv 1210.3684.

[20] Kim J, Tawarmalani M, Richard JPP (2021) Convexification of permutation-invariant sets and an application to sparse principal component analysis. *Mathematics of Operations Research* .

[21] Kim K (1982) *Boolean Matrix Theory and Applications*. Monographs and textbooks in pure and applied mathematics (Dekker), ISBN 9780824717889.

[22] Kononenko I, Cestnik B (1988) Uci mach. learn. rep.: Primary tumor domain. URL https://archive.ics.uci.edu/ml/datasets/Primary+Tumor, last accessed on 2020-06-11.

[23] Kononenko I, Cestnik B (1988) Uci machine learning repository: Lymphography data set. URL https://archive.ics.uci.edu/ml/datasets/Lymphography, last accessed on 2020-06-11.

[24] Kovacs RA (2021) Code for binary matrix factorisation and completion via integer programming. URL https://github.com/kovacsrekaagnes/rank_k_Binary_Matrix_Factorisation.

[25] Kovacs RA, Gunluk O, Hauser RA (2017) Low-rank boolean matrix approximation by integer programming. *NIPS*, 1–5, Optimization for Machine Learning Workshop, https://opt-ml.org/papers/OPT2017_paper_34.pdf.

[26] Kovacs RA, Gunluk O, Hauser RA (2021) Binary matrix factorisation via column generation. *Proceedings of the AAAI Conference on Artificial Intelligence* 35(5):3823–3831, URL https://ojs.aaai.org/index.php/AAAI/article/view/16500.

[27] Koyutürk M, Grama A (2003) Proximus: A framework for analyzing very high dimensional discrete-attributed datasets. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 147–156, KDD '03 (New York, NY, USA: Association for Computing Machinery), ISBN 1581137370, URL http://dx.doi.org/10.1145/956750.956770.

[28] Koyutürk M, Grama A, Ramakrishnan N (2002) Algebraic techniques for analysis of large discrete-valued datasets. *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, 311–324, PKDD '02 (Berlin, Heidelberg: Springer-Verlag), ISBN 3540440372.

[29] Krebs V (2008) Amazon political books. URL http://moreno.ss.uci.edu/data.html#books, last accessed on 2020-06-11.

[30] Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791, URL http://dx.doi.org/10.1038/44565.

[31] Li T (2005) A general model for clustering binary data. *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 188–197, KDD '05 (New York, NY, USA: Association for Computing Machinery), ISBN 159593135X.

[32] Li Y, Ngom A (2012) The non-negative matrix factorization toolbox for biological data mining. *Source Code for Biology and Medicine* 8:10 – 10.

[33] Li Y, Ngom A (2013) The non-negative matrix factorization toolbox in matlab (the nmf matlab toolbox). URL https://sites.google.com/site/nmftool/, last accessed on 2021-07-16.

[34] Lu H, Vaidya J, Atluri V (2008) Optimal boolean matrix decomposition: Application to role engineering. *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, 297–306, ICDE '08 (Washington, DC, USA: IEEE Computer Society), URL http://dx.doi.org/10.1109/ICDE.2008.4497438.

[35] Lu H, Vaidya J, Atluri V (2014) An optimization framework for role mining. *Journal of Computer Security* 22(1):1–31, ISSN 0926-227X.

[36] Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Operations Research* 53(6):1007–1023, URL http://dx.doi.org/10.1287/opre.1050.0234.

[37] McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. *Mathematical Programming* 10(1):147–175, URL http://dx.doi.org/10.1007/BF01580665.

[38] Miettinen P, Mielikäinen T, Gionis A, Das G, Mannila H (2006) The discrete basis problem. Fürnkranz J, Scheffer T, Spiliopoulou M, eds., *Knowledge Discovery in Databases: PKDD 2006*, 335–346 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-540-46048-0.

[39] Miettinen P, Mielikäinen T, Gionis A, Das G, Mannila H (2008) The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering* 20(10):1348–1362, ISSN 1041-4347, URL http://dx.doi.org/10.1109/TKDE.2008.53.

[40] Monson SD, Pullman NJ, 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, ISSN 1183-1278.

[41] Oelze M, Vandaele A, Weltge S (2014) Computing the extension complexities of all 4-dimensional 0/1-polytopes. URL http://dx.doi.org/10.48550/ARXIV.1406.4895.

[42] Orlin J (1977) Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)* 80(5):406–424, ISSN 1385-7258, URL http://dx.doi.org/10.1016/1385-7258(77)90055-5.

[43] Padberg M (1989) The boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming* 45(1):139–172, URL http://dx.doi.org/10.1007/BF01589101.

[44] Peeters R (2003) The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics* 131(3):651–654, ISSN 0166-218X, URL http://dx.doi.org/10.1016/S0166-218X(03)00333-0.

[45] 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.

[46] Schinnerl C (2017) Pymf - python matrix factorization module. URL https://github.com/ChrisSchinnerl/pymf3, last accessed on 2021-03-11.

[47] 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.

[48] Shen BH, Ji S, Ye J (2009) Mining discrete patterns via binary matrix factorization. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 757–766, KDD '09 (New York, NY, USA: Association for Computing Machinery), ISBN 9781605584959, URL http://dx.doi.org/10.1145/1557019.1557103.

[49] Shi Z, Wang L, Shi L (2014) Approximation method to rank-one binary matrix factorization. *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, 800–805, URL http://dx.doi.org/10.1109/CoASE.2014.6899417.

[50] Simon HU (1990) On approximate solutions for combinatorial optimization problems. *SIAM Journal on Discrete Mathematics* 3(2):294–310, URL http://dx.doi.org/10.1137/0403025.

[51] Sripratak P, Punnen AP, Stephen T (2022) The bipartite boolean quadric polytope. *Discrete Optimization* 44:100657, ISSN 1572-5286, URL http://dx.doi.org/https://doi.org/10.1016/j.disopt.2021.100657.

[52] Zhang Z, Li T, Ding C, Zhang X (2007) Binary matrix factorization with applications. *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, 391–400, ICDM '07 (USA: IEEE Computer Society), URL http://dx.doi.org/10.1109/ICDM.2007.99.