Parameterized low-rank binary matrix approximation

Fedor V. Fomin, Petr A. Golovach & Fahad Panolan

Data Mining and Knowledge Discovery

ISSN 1384-5810
Volume 34
Number 2

Data Min Knowl Disc (2020) 34:478-532
DOI 10.1007/s10618-019-00669-5

Springer

Data Mining and Knowledge Discovery (2020) 34:478–532
https://doi.org/10.1007/s10618-019-00669-5

Parameterized low-rank binary matrix approximation

Fedor V. Fomin[1] · **Petr A. Golovach**[1] · **Fahad Panolan**[2]

Received: 21 March 2019 / Accepted: 9 December 2019 / Published online: 2 January 2020

Abstract
Low-rank binary matrix approximation is a generic problem where one seeks a good approximation of a binary matrix by another binary matrix with some specific properties. A good approximation means that the difference between the two matrices in some matrix norm is small. The properties of the approximation binary matrix could be: a small number of different columns, a small binary rank or a small Boolean rank. Unfortunately, most variants of these problems are NP-hard. Due to this, we initiate the systematic algorithmic study of low-rank binary matrix approximation from the perspective of parameterized complexity. We show in which cases and under what conditions the problem is fixed-parameter tractable, admits a polynomial kernel and can be solved in parameterized subexponential time.

Keywords Binary matrices · Clustering · Low-rank approximation · Fixed-parameter tractability

Responsible editor: Pauli Miettinen.

The work was done within the CEDAS center in Bergen. The preliminary version of this paper appeared as an extended abstract in the proceedings of ICALP 2018.

✉ Petr A. Golovach
Petr.Golovach@uib.no

Fedor V. Fomin
Fedor.Fomin@uib.no

Fahad Panolan
fahad@iith.ac.in

[1] Department of Informatics, University of Bergen, PB 7803, 5020 Bergen, Norway

[2] Department of Computer Science and Engineering, IIT Hyderabad, Kandi, Sangareddy, Telangana 502285, India

1 Introduction

Low-rank approximation is a generic optimization problem, in which a given data matrix has to be approximated by another matrix of low rank. It is in the heart of the basic methods in data analysis like principal component analysis (PCA) or factor analysis. It is well-known that the low-rank matrix approximation with respect to the Frobenius norm over reals can be efficiently solved with singular value decomposition (SVD). However SVD can return real-valued matrices; this makes it hard to use for interpretation of data which is originally binary or integer. For many applications in data mining and knowledge discovery it is highly desired that the low-rank matrix approximating the data matrix is also binary (Bartl et al. 2010; Miettinen et al. 2008; Miettinen and Vreeken 2011). Unfortunately, most of the interesting variants of low-rank binary matrix approximation are NP-complete.

The fact that a problem is NP-hard only means that it is "hard" in the "worst-case". It is rarely the case that the input instances we actually want to solve look like the instances on which the algorithm performs the worst. In this paper we propose the study of the computational complexity of low-rank approximation problems from the perspective of parameterized complexity. The core idea behind parameterized complexity is very general—to measure the running time in terms of both input size as well as various *parameters* that capture structural properties of the input instance. Originating in the late 80s from the foundational work of Downey and Fellows (1992), the area of parameterized algorithms and complexity has experienced tremendous growth, and is now considered one of the central subfields of theoretical computer science. The complexity of computational problems on high-dimensional data is naturally governed by various parameters and viewing high-dimensional data through parameterized complexity lens could lead to a new level of understanding of the existing methods, and yield powerful new tools for designing better heuristics, as well as provably correct and efficient algorithms.

In this paper we consider the following generic problem. Given a binary $m \times n$ matrix, that is, a matrix with entries from domain $\{0, 1\}$,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{21} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = (a_{ij}) \in \{0, 1\}^{m \times n},$$

the task is to find a "simple" binary $m \times n$ matrix $\mathbf{B}$ which approximates $\mathbf{A}$ subject to some specified constrains. One of the most widely studied error measures is the *Frobenius norm*, which for a matrix $\mathbf{A}$ is defined as

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}.$$

Here, the sums are taken over $\mathbb{R}$. Then, for a given nonnegative integer k, we want to decide whether there is a matrix $\mathbf{B}$ with certain properties such that

$$\|\mathbf{A} - \mathbf{B}\|_F^2 \leq k.$$

We consider the binary matrix approximation problems when for a given integer r, the approximation binary matrix $\mathbf{B}$

(A1) has at most r distinct columns,
(A2) is of $\mathsf{GF}(2)$-rank at most r,
(A3) is of Boolean rank at most r.

Each of these variants is very well-studied. Before defining each of the problems formally and providing an overview of the relevant results, the following observation is in order. Since we approximate a binary matrix by a binary matrix, in this case minimizing the Frobenius norm of $\mathbf{A} - \mathbf{B}$ is equivalent to minimizing the ℓ_0-norm of $\mathbf{A} - \mathbf{B}$, where the measure $\|\mathbf{X}\|_0$ is the number of nonzero entries of matrix $\mathbf{X}$. We also will be using another equivalent way of measuring the quality of approximation of a binary matrix $\mathbf{A}$ by a binary matrix $\mathbf{B}$ by taking the sum of the Hamming distances between their columns. Let us recall that the *Hamming distance* between two vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^m$, where $\mathbf{x} = (x_1, \ldots, x_m)^\mathsf{T}$ and $\mathbf{y} = (y_1, \ldots, y_m)^\mathsf{T}$, is $d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^m |x_i - y_i|$ or, in words, the number of positions $i \in \{1, \ldots, m\}$ where x_i and y_i differ. Then, for binary $m \times n$ matrix $\mathbf{A}$ with columns $\mathbf{a}^1, \ldots, \mathbf{a}^n$ and matrix $\mathbf{B}$ with columns $\mathbf{b}^1, \ldots, \mathbf{b}^n$, we define

$$d_H(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^n d_H(\mathbf{a}^i, \mathbf{b}^i).$$

In other words, $d_H(\mathbf{A}, \mathbf{B})$ is the number of positions with different entries in matrices $\mathbf{A}$ and $\mathbf{B}$. Then

$$\|\mathbf{A} - \mathbf{B}\|_F^2 = \|\mathbf{A} - \mathbf{B}\|_0 = d_H(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^n d_H(\mathbf{a}^i, \mathbf{b}^i). \tag{1}$$

Problem (A1): Binary r-Means By (1), the problem of approximating a binary $m \times n$ matrix $\mathbf{A}$ by a binary $m \times n$ matrix $\mathbf{B}$ with at most r different columns (problem (A1)) is equivalent to the following clustering problem. Given a set of n binary m-dimensional vectors $\mathbf{a}^1, \ldots, \mathbf{a}^n$ (which constitute the columns of matrix $\mathbf{A}$) and a positive integer r, BINARY r-MEANS aims to partition the vectors in at most r clusters, so as to minimize the sum of within-clusters sums of Hamming distances to their binary means. More formally,

BINARY r- MEANS

Input: An $m \times n$ matrix $\mathbf{A}$ with columns $(\mathbf{a}^1, \ldots, \mathbf{a}^n)$, a positive integer r and a nonnegative integer k.

Task: Decide whether there is a positive integer $r' \leq r$, a partition $\{I_1, \ldots, I_{r'}\}$ of $\{1, \ldots, n\}$ and vectors $\mathbf{c}^1, \ldots, \mathbf{c}^{r'} \in \{0, 1\}^m$ such that

$$\sum_{i=1}^{r'} \sum_{j \in I_i} d_H(\mathbf{c}^i, \mathbf{a}^j) \leq k.$$

To see the equivalence of BINARY r-MEANS and problem (A1), it is sufficient to observe that distinct columns of an approximate matrix $\mathbf{B}$ such that $d_H(\mathbf{A}, \mathbf{B}) \leq k$ can be used as vectors $\mathbf{c}^1, \ldots, \mathbf{c}^{r'}$, $r' \leq r$. As far as the mean vectors are selected, a partition of columns of $\mathbf{A}$ can be obtained by assigning each column-vector $\mathbf{a}^i$ to its closest mean vector $\mathbf{c}^j$ (ties breaking arbitrarily). Then, for such clustering the total sum of distances from vectors within cluster to their centers does not exceed k. Similarly, a solution to BINARY r-MEANS can be used as columns (with possible repetitions) of matrix $\mathbf{B}$ such that $d_H(\mathbf{A}, \mathbf{B}) \leq k$. For that we put $\mathbf{b}^i = \mathbf{c}^j$, where $\mathbf{c}^j$ is the closest vector to $\mathbf{a}^i$.

This problem was introduced by Kleinberg et al. (2004) as one of the examples of segmentation problems. Approximation algorithms for optimization versions of this problem were given by Alon and Sudakov (1999) and Ostrovsky and Rabani (2002), who referred to it as clustering in the Hamming cube. In bioinformatics, the case when $r = 2$ is known under the name BINARY- CONSTRUCTIVE- MEC (Minimum Error Correction) and was studied as a model for the SINGLE INDIVIDUAL HAPLOTYPING problem by Cilibrasi et al. (2007). This problem was studied by Miettinen et al. (2008) under the name DISCRETE BASIS PARTITIONING PROBLEM.

BINARY r-MEANS can be seen as a discrete variant of the well-known k-MEANS CLUSTERING. (Since in problems (A2) and (A3) we use r for the rank of the approximation matrix, we also use r in (A1) to denote the number of clusters which is commonly denoted by k in the literature on means clustering.) This problem has been studied thoroughly, particularly in the areas of computational geometry and machine learning. We refer to Agarwal et al. (2004), Badoiu et al. (2002), and Kumar et al. (2010) for further references to the works on k-MEANS CLUSTERING.

Problem (A2): Low GF(2)-Rank Approximation Let $\mathbf{A}$ be a $m \times n$ binary matrix. In this case we view the elements of $\mathbf{A}$ as elements of $\mathsf{GF}(2)$, the Galois field of two elements $\{0, 1\}$. Then the $\mathsf{GF}(2)$-rank of $\mathbf{A}$ is the minimum r such that $\mathbf{A} = \mathbf{U} \times \mathbf{V}$, where $\mathbf{U}$ and $\mathbf{V}$ are $m \times r$ and $r \times n$ binary matrices respectively, and arithmetic operations are over $\mathsf{GF}(2)$, that is, $0 + x = x + 0 = x$, $1x = x1 = x$ and $0x = x0 = 0$ for $x \in \{0, 1\}$, and $1 + 1 = 0$. Equivalently, this is the minimum number of binary vectors, such that every column (row) of $\mathbf{A}$ is a linear combination (over $\mathsf{GF}(2)$) of these vectors. Then, (A2) is the following problem.

LOW GF(2)- RANK APPROXIMATION

Input: An $m \times n$-matrix $\mathbf{A}$ over GF(2), and nonnegative integers r and k.

Task: Decide whether there is a binary $m \times n$-matrix $\mathbf{B}$ with GF(2)-rank $(\mathbf{B}) \leq r$ such that $d_H(\mathbf{A}, \mathbf{B}) \leq k$.

LOW GF(2)- RANK APPROXIMATION arises naturally in applications involving binary data sets and serves as an important tool in dimension reduction for high-dimensional data sets with binary attributes, see Dan et al. (2015), Jiang et al. (2014), Gutch et al. (2012), Koyutürk and Grama (2003), Painsky et al. (2016), Shen et al. (2009), and Yeredor (2011) for further references and numerous applications of the problem.

LOW GF(2)- RANK APPROXIMATION can be rephrased as a special variant (over GF(2)) of the problem finding the rigidity of a matrix. (For a target rank r, the *rigidity* of a matrix A over a field $\mathbb{F}$ is the minimum Hamming distance between A and a matrix of rank at most r.) Rigidity is a classical concept in Computational Complexity Theory studied due to its connections with lower bounds for arithmetic circuits (Grigoriev 1976, 1980; Valiant 1977; Razborov 1989). We refer to Lokam (2009) for an extensive survey on this topic.

LOW GF(2)- RANK APPROXIMATION is also a special case of a general class of problems approximating a matrix by a matrix with a small nonnegative rank. Already NONNEGATIVE MATRIX FACTORIZATION (NMF) is a nontrivial problem and it appears in many settings. In particular, in machine learning, approximation by a nonnegative low rank matrix has gained extreme popularity after the influential article of Lee and Seung (1999) in Nature. NMF is an ubiquitous problem and besides machine learning, it has been independently introduced and studied in combinatorial optimization (Fiorini et al. 2015; Yannakakis 1991), and communication complexity (Aho et al. 1983; Lovász and Saks 1988). An extended overview of applications of NMF in statistics, quantum mechanics, biology, economics, and chemometrics, can be found in Cohen and Rothblum (1993) and recent books of Cichocki et al. (2009), Naik (2016), and Fu (2014).

Problem (A3): Low Boolean-Rank Approximation Let $\mathbf{A}$ be a binary $m \times n$ matrix. This time we view the elements of $\mathbf{A}$ as *Boolean* variables. The *Boolean rank* of $\mathbf{A}$ is the minimum r such that $\mathbf{A} = \mathbf{U} \wedge \mathbf{V}$ for a Boolean $m \times r$ matrix $\mathbf{U}$ and a Boolean $r \times n$ matrix $\mathbf{V}$, where the product is Boolean, that is, the logical $\wedge$ plays the role of multiplication and $\vee$ the role of sum. Here, $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 1 = 1$, $0 \vee 0 = 0$, $0 \vee 1 = 1$, and $1 \vee 1 = 1$. Thus the matrix product is over the Boolean semi-ring $(0, 1, \wedge, \vee)$. This can be equivalently expressed as the normal matrix product with addition defined as $1 + 1 = 1$. Binary matrices equipped with such algebra are called *Boolean matrices*. Equivalently, $\mathbf{A} = (a_{ij}) \in \{0, 1\}^{m \times n}$ has the Boolean rank 1 if $\mathbf{A} = \mathbf{x}^\intercal \wedge \mathbf{y}$, where $\mathbf{x} = (x_1, x_2, \ldots, x_m) \in \{0, 1\}^m$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^n$ are nonzero vectors and the product is Boolean, that is, $a_{ij} = x_i \wedge y_j$. Then the Boolean rank of $\mathbf{A}$ is the minimum integer r such that $\mathbf{A} = \mathbf{A}^{(1)} \vee \cdots \vee \mathbf{A}^{(r)}$, where $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(r)}$ are matrices of Boolean rank 1; zero

matrix is the unique matrix with the Boolean rank 0. Then, LOW BOOLEAN- RANK APPROXIMATION is defined as follows.

LOW BOOLEAN- RANK APPROXIMATION

Input: A Boolean $m \times n$ matrix $\mathbf{A}$, and nonnegative integers r and k.

Task: Decide whether there is a Boolean $m \times n$ matrix $\mathbf{B}$ of Boolean rank at most r such that $d_H(\mathbf{A}, \mathbf{B}) \leq k$.

For $r = 1$ LOW BOOLEAN- RANK APPROXIMATION coincides with LOW GF(2)- RANK APPROXIMATION, but for $r > 1$, these are different problems. Boolean low-rank approximation has attracted much attention, especially in the data mining and knowledge discovery communities. In data mining, matrix decompositions are often used to produce concise representations of data. Since much of the real data is binary or even Boolean in nature, Boolean low-rank approximation could provide a deeper insight into the semantics associated with the original matrix. There is a big body of work done on LOW BOOLEAN- RANK APPROXIMATION, see e.g. Bartl et al. (2010), Belohlávek and Vychodil (2010), Dan et al. (2015), Lu et al. (2012), Miettinen et al. (2008), Miettinen and Vreeken (2011), and Vaidya (2012). In the literature the problem appears under different names like DISCRETE BASIS PROBLEM (Miettinen et al. 2008) or MINIMAL NOISE ROLE MINING PROBLEM (Vaidya et al. 2007; Lu et al. 2012; Mitra et al. 2016).

P-Matrix Approximation While at first glance LOW GF(2)- RANK APPROXIMATION and LOW BOOLEAN- RANK APPROXIMATION look very similar, algorithmically the latter problem is more challenging. The fact that GF(2) is a field allows to play with different equivalent definitions of rank like row rank and column ranks. We exploit this strongly in our algorithm for LOW GF(2)- RANK APPROXIMATION. For LOW BOOLEAN- RANK APPROXIMATION the matrix product is over the Boolean semi-ring and nice properties of the GF(2)-rank cannot be used here (see e.g. Guterman 2008). Our algorithm for LOW BOOLEAN- RANK APPROXIMATION is based on solving an auxiliary **P**-MATRIX APPROXIMATION problem, where the task is to approximate a matrix $\mathbf{A}$ by a matrix $\mathbf{B}$ whose block structure is defined by a given pattern matrix $\mathbf{P}$. It appears, that **P**-MATRIX APPROXIMATION is also an interesting problem on its own.

More formally, let $\mathbf{P} = (p_{ij}) \in \{0, 1\}^{p \times q}$ be a binary $p \times q$ matrix. We say that a binary $m \times n$ matrix $\mathbf{B} = (b_{ij}) \in \{0, 1\}^{m \times n}$ is a **P**-*matrix* if there is a partition $\{I_1, \ldots, I_p\}$ of $\{1, \ldots, m\}$ and a partition $\{J_1, \ldots, J_q\}$ of $\{1, \ldots, n\}$ such that for every $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, q\}$, $s \in I_i$ and $t \in J_j$, $b_{st} = p_{ij}$. In words, the columns and rows of $\mathbf{B}$ can be permuted such that the block structure of the resulting matrix is defined by $\mathbf{P}$. Note that by this definition, each block of $\mathbf{B}$ should be nonempty.

P- MATRIX APPROXIMATION

Input: An $m \times n$ binary matrix $\mathbf{A}$, a pattern binary matrix $\mathbf{P}$ and a nonnegative integer k.

Task: Decide whether there is an $m \times n$ **P**-matrix $\mathbf{B}$ such that $d_H(\mathbf{A}, \mathbf{B}) \leq k$.

The notion of **P**-matrix was implicitly defined by Wulff et al. (2013) as an auxiliary tool for their approximation algorithm for the related monochromatic biclustering problem. **P**-MATRIX APPROXIMATION is also closely related to the problems arising in tiling transaction databases (i.e., binary matrices), where the task is to find a tiling covers of a given binary matrix with a small number of submatrices full of 1s, see Geerts et al. (2004).

Since LOW GF(2)- RANK APPROXIMATION remains NP-complete for $r = 1$ (Gillis and Vavasis 2015), we have that **P**-MATRIX APPROXIMATION is NP-complete already for the very simple pattern matrix $\mathbf{P} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$.

1.1 Related work

In this subsection we give an overview of previous related algorithmic and complexity results for problems (A1)–(A3), as well as related problems. Since each of the problems has many practical applications, there is a tremendous amount of literature on heuristics and implementations. In this overview we concentrate on known results about algorithms with proven guarantee, with emphasis on parameterized complexity.

Problem (A1): Binary r-Means BINARY r-MEANS is trivially solvable in polynomial time for $r = 1$, and it is NP-complete for every $r \geq 2$ (Feige 2014). PTAS (polynomial time approximation scheme) for optimization variants of BINARY r-MEANS were developed by Alon and Sudakov (1999) and Ostrovsky and Rabani (2002). Approximation algorithms for more general k-MEANS CLUSTERING is a thoroughly studied topic (Agarwal et al. 2004; Badoiu et al. 2002; Kumar et al. 2010). It have been shown by Inaba et al. (1994) that the general k-MEANS CLUSTERING is solvable in n^{mr+1} time (here n is the number of vectors, m is the dimension and r the number of required clusters). We are not aware of any, except the trivial brute-force, exact algorithm for BINARY r-MEANS prior to our work.

Problem (A2): Low GF(2)-Rank Approximation When the low-rank approximation matrix **B** is not required to be binary, then the optimal Frobenius norm rank-r approximation of (not necessarily binary) matrix **A** can be efficiently found via the singular value decomposition (SVD). This is an extremely well-studied problem and we refer to surveys for an overview of algorithms for low rank approximation (Kannan and Vempala 2009; Mahoney 2011; Woodruff 2014). However, SVD does not guarantee to find an optimal solution in the case when additional structural constrains on the low-rank approximation matrix **B** (like being nonnegative or binary) are imposed.

In fact, most of these constrained variants of low-rank approximation are NP-hard. In particular, it have been shown by Gillis and Vavasis (2015) and Dan et al. (2015) that LOW GF(2)- RANK APPROXIMATION is NP-complete for every $r \geq 1$. Approximation algorithms for the optimization version of LOW BOOLEAN- RANK APPROXIMATION were considered by Jiang and Heath (2013), Jiang et al. (2014), Dan et al. (2015), Koyutürk and Grama (2003), Shen et al. (2009), and Bringmann et al. (2017) among others. Recently, polynomial time approximation schemes for this problem were obtained by Ban et al. (2019) and Fomin et al. (2018a).

Most of the known results about the parameterized complexity of the problem follows from the results for MATRIX RIGIDITY. Fomin et al. (2018a) proved that for every finite field, and in particular GF(2), MATRIX RIGIDITY is W[1]-hard being parameterized by k. This implies that LOW GF(2)- RANK APPROXIMATION is W[1]-hard when parameterized by k. However, when parameterized by k and r, the problem becomes fixed-parameter tractable. Fomin et al. (2018b) also proved that MATRIX RIGIDITY for a finite field admits a polynomial kernel. By these results, LOW GF(2)-RANK APPROXIMATION has a kernel of size $\mathcal{O}(r^2k^2)$.

For LOW GF(2)- RANK APPROXIMATION, the algorithm of Fomin et al. (2018b) runs in $2^{\mathcal{O}(f(r)\sqrt{k}\log k)}(nm)^{\mathcal{O}(1)}$ time, where f is a certain function of r. While the function $f(r)$ is not specified by Fomin et al. (2018b), the algorithm invokes enumeration of all $2^r \times 2^r$ binary matrices of rank r, and thus the running time is at least double-exponential in r.

Meesum et al. (2016) and Meesum and Saurabh (2016) considered parameterized algorithms for related problems about editing of the adjacencies of a graph (or directed graph) targeting a graph with adjacency matrix of small rank.

Problem (A3): Low Boolean-Rank Approximation It follows from the rank definitions that a matrix is of Boolean rank $r = 1$ if and only if its GF(2)-rank is 1. Thus, by the results of Gillis and Vavasis (2015) and Dan et al. (2015) LOW BOOLEAN- RANK APPROXIMATION is NP-complete already for $r = 1$. A formulation of LOW BOOLEAN-RANK APPROXIMATION as an integer programming problem with exponential number of variables and constraints was given by Lu et al. (2008).

While computing GF(2)-rank (or rank over any other field) of a matrix can be performed in polynomial time, deciding whether the Boolean rank of a given matrix is at most r is already an NP-complete problem. Thus, LOW BOOLEAN- RANK APPROXIMATION is NP-complete already for $k = 0$. This follows from the well-known relation between the Boolean rank and covering edges of a bipartite graph by bicliques (Gregory et al. 1991). Let us briefly describe this equivalence. For Boolean matrix $\mathbf{A}$, let G_A be the corresponding bipartite graph, i.e. the bipartite graph whose biadjacency matrix is $\mathbf{A}$. By the equivalent definition of the Boolean rank, $\mathbf{A}$ has Boolean rank r if and only if it is the logical disjunction of r Boolean matrices of rank 1. But, for every bipartite graph whose biadjacency matrix is a Boolean matrix of rank at most 1, its edges can be covered by at most one biclique (complete bipartite graph). Thus, deciding whether a matrix is of Boolean rank r is exactly the same as deciding whether edges of a bipartite graph can be covered by at most r bicliques. The latter BICLIQUE COVER problem is known to be NP-complete (Orlin 1977). BICLIQUE COVER is solvable in $2^{2^{\mathcal{O}(r)}} \cdot (nm)^{\mathcal{O}(1)}$ time (Gramm et al. 2008) and unless Exponential Time Hypothesis (ETH) fails, it cannot be solved in $2^{2^{o(r)}} \cdot (nm)^{\mathcal{O}(1)}$ time (Chandran et al. 2016).

For the special case $r = 1$ and $k \leq \|A\|_0/240$, an exact algorithm of running time $2^{k/\sqrt{\|A\|_0}} \cdot (nm)^{\mathcal{O}(1)}$ for LOW BOOLEAN- RANK APPROXIMATION was given in Bringmann et al. (2017).

More generally, exact algorithms for NMF were studied by Cohen and Rothblum (1993). In was shown by Arora et al. (2012) and Moitra (2016) that for a fixed value of r, NMF is solvable in polynomial time. There are also related works of Razenshteyn

et al. (2016) on weighted low-rank approximation, Clarkson and Woodruff (2015) on robust subspace approximation, and Basu et al. (2016) on PSD factorization.

Observe that all the problems studied in this paper could be seen as matrix editing problems. For BINARY r-MEANS, we can assume that $r \leq n$ as otherwise we have a trivial yes-instance. Then, the problem asks whether it is possible to edit at most k entries of the input matrix, that is, replace some 0s by 1s and some 1s by 0s, in such a way that the obtained matrix has at most r distinct columns. Respectively, LOW GF(2)- RANK APPROXIMATION asks whether it is possible to edit at most k entries of the input matrix to obtain a matrix of rank at most r. In **P**-MATRIX APPROXIMATION, we ask whether we can edit at most k elements to obtain a **P**-matrix. A lot of work in graph algorithms has been done on graph editing problems, in particular parameterized subexponential time algorithms were developed for a number of problems, including various cluster editing problems (Drange et al. 2015; Fomin et al. 2014).

1.2 Our results and methods

The main conceptual contribution of this paper is two-fold. On one hand, it demonstrates that the field of Data Mining, which remains almost unexplored from the perspective of Parameterized Complexity, is full of interesting challenges that could and should be explored. On the other hand, some of the algorithmic approaches developed in this paper, like kernelization (or preprocessing) techniques are not of theoretical interest only and potentially can bring to new practical algorithms. Below we elaborated on these statements.

The core idea behind Parameterized Complexity is very general—to measure the running time in terms of both input size as well as various *parameters* that capture structural properties of the input instance. Originating in the late 80s from the foundational work of Downey and Fellows (1992), the area of parameterized algorithms and complexity has experienced tremendous growth, and is now considered one of the central subfields of theoretical computer science. So far the scope of Parameterized Complexity has been mostly limited to problems on graphs, networks, strings, hypergraphs, and sets, with exceptions few and far between. However, *there is no inherent reason why the parameterized algorithmic approach cannot be successful in other domains*. The ideas of Parameterized Complexity hold the potential to address the need for a framework for refined algorithm analysis for different kinds of problems arising in the data analysis. The only reason why the ideas and concepts of Parameterized Complexity are not met frequently in Data Mining, where the core objects are matrices and vector spaces, is the lack of algorithmic and complexity tools to deal with such objects. Our work can be seen as one of the first steps in this direction.

While the results of the paper are mainly of a theoretical nature, we believe that at least some of our algorithms have a strong potential from practical perspective too. For example, the preprocessing procedure (Algorithm 1) for BINARY r-MEANS can be used as a subroutine in any heuristic algorithm for the problem. Moreover, while for large rank values r and k the running time of our algorithms is not practical, which is not surprising since we deal with NP-hard problems, for small values of the parameters (especially r), our algorithms can solve the problem. For example, a popular approach

Table 1 Parameterized complexity of low-rank approximation

	k	r	$k+r$
BINARY r-MEANS	$2^{\mathcal{O}(k \log k)}(nm)^{\mathcal{O}(1)}$ Theorem 1, No poly-kernel Theorem 4	NP-c for $r \geq 2$ (Feige 2014)	$2^{\mathcal{O}(\sqrt{rk \log (k+r) \log r})}$ Theorem 5, Poly-kernel Theorem 2
GF(2) APPR	W[1]-hard (Fomin et al. 2018b)	NP-c for $r \geq 1$ (Gillis and Vavasis 2015) (Dan et al. 2015)	$2^{\mathcal{O}(r\sqrt{k \log(rk)})}$ Theorem 6, Poly-kernel (Fomin et al. 2018b)
BOOL APPR	NP-c for $k = 0$ (Orlin 1977)	NP-c for $r \geq 1$ (Gillis and Vavasis 2015) (Dan et al. 2015)	$2^{\mathcal{O}(r2^r\sqrt{k \log k})}$ Theorem 8

GF(2) APPR stands for LOW GF(2)- RANK APPROXIMATION and BOOL APPR for LOW BOOLEAN- RANK APPROXIMATION. We omit the factor nm in the running times in Theorems 5, 6 and 8

to robust Principal Component Analysis (PCA) is to seek a represenation of a data matrix M as a low-rank component L and a sparse component S. That is, $M = L + S$, see e.g. Candès et al. (2011), Wright et al. (2009), and Chandrasekaran et al. (2011). By our theorems, the binary variant of robust PCA is solvable in polynomial time when the rank r is constant and $\|S\|_0$, the number of non-zero entries of the sparse matrix S, is in $\mathcal{O}(\log^2 n / \log \log n)$. Also it is solvable in polynomial time when $r \in \mathcal{O}(\sqrt{\log n})$ and $\|S\|_0 \in \mathcal{O}(\log n / \log \log n)$. This marks interesting islands of tractability for seemingly very difficult data mining problems (see Corollary 1). Observe also that it is know that LOW GF(2)- RANK APPROXIMATION is interesting even for the case $r = 1$, that is, when the data are represented as the product of two vectors that are usually called *presence* vector and *pattern* vector respectively (see e.g. Lu et al. 2011; Shi et al. 2014 and the references therein). In this case our subexponential in k Algorithm 4 may be practical. In fact, for $r = 1$, we can apply the simpler Algorithm 2 for BINARY 2-MEANS as LOW GF(2)- RANK APPROXIMATION for $r = 1$ is equivalent to BINARY 2-MEANS with the additional requirement that one of the means is the zero vector.

Description of the results We study the parameterized complexity of BINARY r-MEANS, LOW GF(2)- RANK APPROXIMATION and LOW BOOLEAN- RANK APPROXIMATION. We refer to the recent books of Cygan et al. (2015) and Downey and Fellows (2013) for an introduction to Parameterized Algorithms and Complexity. Our results are summarized in Table 1.

Our first main result concerns BINARY r-MEANS. We show (Theorem 1) that the problem is solvable in $2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$ time. Therefore, BINARY r-MEANS is FPT parameterized by k. Since LOW GF(2)- RANK APPROXIMATION parameterized by k is W[1]-hard and LOW BOOLEAN- RANK APPROXIMATION is NP-complete for any fixed $k \geq 0$, we find Theorem 1 quite surprising. The proof of Theorem 1 is based on a fundamental result of Marx (2008) about the complexity of a problem on strings, namely CONSENSUS PATTERNS. We solve BINARY r-MEANS by constructing a two-stage FPT Turing reduction to CONSENSUS PATTERNS. First, we use the color coding technique of Alon et al. (1995) to reduce BINARY r-MEANS to some special auxiliary

problem and then show that this problem can be reduced to CONSENSUS PATTERNS, and this allows us to apply the algorithm of Marx (2008). We also prove (Theorem 2) that BINARY r-MEANS admits a polynomial kernel when parameterized by r and k. That is, we give a polynomial time preprocessing algorithm that for a given instance of BINARY r-MEANS outputs an equivalent instance with $\mathcal{O}(k(k+r)$ columns and rows. Since the the kernelization algorithm outputs an equivalent instance, it can be safely pipelined with any other exact or heuristic algorithm. For parameterization by k only, we show in Theorem 4 that BINARY r-MEANS has no polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP}/\mathsf{poly}$, a standard complexity assumption.

Our second main result concerns LOW BOOLEAN- RANK APPROXIMATION. As we mentioned above, the problem is NP-complete for $k = 0$, as well as for $r = 1$, and hence is intractable being parameterized by k or by r only. On the other hand, a simpler LOW GF(2)- RANK APPROXIMATION is not only FPT parameterized by $k + r$, by Fomin et al. (2018b) it is solvable in $2^{\mathcal{O}(f(r)\sqrt{k}\log k)}(nm)^{\mathcal{O}(1)}$ time, where f is some function of r, and thus is *subexponential* in k. It is natural to ask whether a similar complexity behavior could be expected for LOW BOOLEAN- RANK APPROXIMATION. Our result, Theorem 8, shows that this is indeed the case: LOW BOOLEAN- RANK APPROXIMATION is solvable in $2^{\mathcal{O}(r2^r\cdot\sqrt{k\log k})}\cdot nm$ time. Note that the running time of our algorithm is linear in the size nm of the input matrix. The proof of this theorem is technical and consists of several steps. We first develop a subexponential algorithm for solving auxiliary **P**-MATRIX APPROXIMATION, and then construct an FPT Turing reduction from LOW BOOLEAN- RANK APPROXIMATION to **P**-MATRIX APPROXIMATION.

Let us note that due to the relation of Boolean rank computation to BICLIQUE COVER, the result of Chandran et al. (2016) implies that unless Exponential Time Hypothesis (ETH) fails, LOW BOOLEAN- RANK APPROXIMATION cannot be solved in $2^{2^{o(r)}}\cdot f(k)\cdot(nm)^{\mathcal{O}(1)}$ time for any function f. Thus the dependence in r in our algorithm cannot be improved significantly unless ETH fails.

Interestingly, the technique developed for solving **P**-MATRIX APPROXIMATION can be used to obtain algorithms of running times $2^{\mathcal{O}(\sqrt{rk\log(k+r)\log r})}\cdot nm$ for BINARY r-MEANS[1] and $2^{\mathcal{O}(r\sqrt{k\log(rk)})}\cdot nm$ for LOW GF(2)- RANK APPROXIMATION (Theorems 5 and 6 respectively). For BINARY r-MEANS, Theorem 5 provides much better running time than Theorem 1 for values of $r \in o(k\log k)$. Notice also that we achieve the linear dependence of the running time on the size of the input matrix.

For LOW GF(2)- RANK APPROXIMATION, comparing Theorem 6 and $2^{\mathcal{O}(f(r)\sqrt{k}\log k)}(nm)^{\mathcal{O}(1)}$ running time obtained by Fomin et al. (2018b), let us note that Theorem 6 not only slightly improves the exponential dependence in k by the factor $\sqrt{\log k}$; it also drastically improves the exponential dependence in r, from 2^{2^r} to $2^{r\sqrt{\log r}}$.

The remaining part of the paper is organized as follows. In Sect. 2, we introduce basic notations and obtain some auxiliary results. In Sect. 3, we show that BINARY r-MEANS is FPT when parameterized by k only. In Sect. 4, we discuss kernelization

[1] We are grateful to the anonymous reviewer who pointed to us that the running time of our algorithm can be improved from the original $2^{\mathcal{O}(r\sqrt{k\log(k+r)})}\cdot nm$ to $2^{\mathcal{O}(\sqrt{rk\log(k+r)\log r})}\cdot nm$.

for BINARY r-MEANS. In Sects. 5 and 6, we construct FPT algorithms for BINARY r-MEANS and LOW GF(2)- RANK APPROXIMATION, respectively, parameterized by k and r, that are subexponential in k. In Sect. 7, we give a subexponential algorithm for LOW BOOLEAN- RANK APPROXIMATION. We conclude our paper in Sect. 8 by stating some open problems.

2 Preliminaries

In this section we introduce the terminology used throughout the paper and obtain some properties of the solutions to our problems.

Matrices and strings All matrices and vectors considered in this paper are assumed to be (0, 1)-matrices and vectors, respectively, unless explicitly specified otherwise. Let $\mathbf{A} = (a_{ij}) \in \{0,1\}^{m\times n}$ be an $m \times n$-matrix. Thus, a_{ij}, $i \in \{1,\ldots,m\}$ and $j \in \{1,\ldots,n\}$, are the elements of $\mathbf{A}$. For $I \subseteq \{1,\ldots,m\}$ and $J \subseteq \{1,\ldots,n\}$, we denote by $\mathbf{A}[I,J]$ the $|I| \times |J|$-submatrix of $\mathbf{A}$ with the elements a_{ij} where $i \in I$ and $j \in J$. We say that two matrices $\mathbf{A}$ and $\mathbf{B}$ are *isomorphic* if $\mathbf{B}$ can be obtained from $\mathbf{A}$ by permutations of rows and columns. We use "+" and "$\sum$" to denote sums and summations over $\mathbb{R}$, and we use "$\oplus$" and "$\bigoplus$" for sums and summations over GF(2); it is assumed that the summation $\bigoplus$ over the empty set of indices is the zero-vector.

We also consider string of symbols. For two strings a and b, we denote by ab their *concatenation*. For a positive integer k, a^k denotes the concatenation of k copies of a; a^0 is assumed to be the empty string. Let $a = a_1 \cdots a_\ell$ be a string over an alphabet Σ. Recall that a string b is said to be a *substring* of a if $b = a_h a_{h+1} \cdots a_t$ for some $1 \le h \le t \le \ell$; we write that $b = a[h..t]$ in this case. Let $a = a_1 \cdots a_\ell$ and $b = b_1 \cdots b_\ell$ be strings of the same length ℓ over Σ. Similar to the the definition of Hamming distance between two (0, 1)-vectors, the *Hamming distance* $d_H(a,b)$ between two strings is defined as the number of positions $i \in \{1,\ldots,\ell\}$ where the strings differ. The Hamming distance satisfies the *triangle inequality*: for any three strings a, b, c of length n each, $d_H(a,c) \le d_H(a,b) + d_H(b,c)$.

Parameterized complexity We refer to the book of Cygan et al. (2015) for the detailed introduction to the field, see also the recent book on kernelization of Fomin et al. (2019). Here, we only briefly introduce basic notions.

A *parameterized problem* is a language $Q \subseteq \Sigma^* \times \mathbb{N}$ where Σ^* is the set of strings over a finite alphabet Σ. Respectively, an input of Q is a pair (I,k) where $I \subseteq \Sigma^*$ and $k \in \mathbb{N}$; k is the *parameter* of the problem.

A parameterized problem Q is *fixed-parameter tractable* (FPT) if it can be decided whether $(I,k) \in Q$ in $f(k) \cdot |I|^{\mathcal{O}(1)}$ time for some function f that depends on the parameter k only. Respectively, the parameterized complexity class FPT is composed by fixed-parameter tractable problems.

Parameterized complexity theory also provides tools to rule-out the existence of FPT algorithms under plausible complexity-theoretic assumptions. For this, a hierarchy of parameterized complexity classes

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}$$

was introduced by Downey and Fellows (1992), and it was conjectured that the inclusions are proper. The basic way to show that it is unlikely that a parameterized problem admit an FPT algorithm is to show that it is W[1] or W[2]-hard.

A *data reduction rule*, or simply, reduction rule, for a parameterized problem Q is a function $\phi\colon \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ that maps an instance (I,k) of Q to an equivalent instance (I',k') of Q such that ϕ is computable in time polynomial in $|I|$ and k. We say that two instances of Q are *equivalent* if the following holds: $(I,k) \in Q$ if and only if $(I',k') \in Q$. We refer to this property of the reduction rule ϕ, that it translates an instance to an equivalent one, as to the *safeness* of the reduction rule.

Informally, *kernelization* is a preprocessing algorithm that consecutively applies various data reduction rules in order to shrink the instance size as much as possible. A preprocessing algorithm takes as input an instance $(I,k) \in \Sigma^* \times \mathbb{N}$ of Q, works in polynomial in $|I|$ and k time, and returns an equivalent instance (I',k') of Q. The quality of a preprocessing algorithm $\mathcal{A}$ is measured by the size of the output. More precisely, the *output size* of a preprocessing algorithm $\mathcal{A}$ is a function $\mathrm{size}_{\mathcal{A}}\colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ defined as follows:

$$\mathrm{size}_{\mathcal{A}}(k) = \sup\{|I'| + k' : (I',k') = \mathcal{A}(I,k),\ I \in \Sigma^*\}.$$

A *kernelization algorithm*, or simply a *kernel*, for a parameterized problem Q is a preprocessing algorithm $\mathcal{A}$ that, given an instance (I,k) of Q, works in polynomial in $|I|$ and k time and returns an equivalent instance (I',k') of Q such that $\mathrm{size}_{\mathcal{A}}(k) \leq g(k)$ for some computable function $g\colon \mathbb{N} \to \mathbb{N}$. It is said that $g(\cdot)$ is the *size* of a kernel. If $g(\cdot)$ is a polynomial function, then we say that Q admits a *polynomial kernel*.

It is well-known that every FPT problem admits a kernel but, up to some reasonable complexity assumptions, there are FPT problems that have no polynomial kernels. In particular, we are using the composition technique introduced by Bodlaender et al. (2009) to show that a parameterized problem does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.

Properties of Binary r-Means We conclude this section by giving some properties of Binary r-Means that are used in Sects. 3–5.

Let $(\mathbf{A},r,k)$ be an instance of Binary r-Means where $\mathbf{A}$ is a matrix with columns $(\mathbf{a}^1,\ldots,\mathbf{a}^n)$. We say that a partition $\{I_1,\ldots,I_{r'}\}$ of $\{1,\ldots,n\}$ for $r' \leq r$ is a *solution* for $(\mathbf{A},r,k)$ if there are vectors $\mathbf{c}^1,\ldots,\mathbf{c}^{r'} \in \{0,1\}^m$ such that $\sum_{i=1}^{r'}\sum_{j\in I_i} d_H(\mathbf{c}^i,\mathbf{a}^j) \leq k$. We say that each I_i or, equivalently, the multiset of columns $\{\mathbf{a}^j \mid j \in I_i\}$ (some columns could be the same) is a *cluster* and call $\mathbf{c}^i$ the *mean* of the cluster. Observe that given a cluster $I \subseteq \{1,\ldots,n\}$, one can easily compute an optimal mean $\mathbf{c} = (c_1,\ldots,c_m)^\intercal$ that minimizes $\sum_{i\in I} d_H(c,\mathbf{a}^i)$ as follows. Let $\mathbf{a}^j = (a_{1j},\ldots,a_{mj})^\intercal$ for $j \in \{1,\ldots,n\}$. For each $i \in \{1,\ldots,m\}$, consider the multiset $S_i = \{a_{ij} \mid j \in I\}$ and put $c_i = 0$ or $c_i = 1$ according to the *majority* of elements in S_i, that is, $c_i = 0$ if at least half of the elements in S_i are 0s and $c_i = 1$ otherwise. We refer to this construction of $\mathbf{c}$ as the *majority rule*.

In the opposite direction, given a set of means $\mathbf{c}^1,\ldots,\mathbf{c}^{r'}$, we can construct clusters $\{I_1,\ldots,I_{r'}\}$ as follows: for each column $\mathbf{a}^j$, find the closest $\mathbf{c}^i$, $i \in \{1,\ldots,r'\}$, such that $d_H(\mathbf{c}^i,\mathbf{a}^j)$ is minimum and assign j to I_i. Note that this procedure does not

guarantee that all clusters are nonempty but we can simply delete empty clusters. Hence, we can define a solution as a set of means $C = \{\mathbf{c}^1, \ldots, \mathbf{c}^{r'}\}$. These arguments also imply the following observation.

Observation 1 *The task of* Binary r-Means *can equivalently be stated as follows: decide whether there exist a positive integer $r' \leq r$ and vectors $\boldsymbol{c}^1, \ldots, \boldsymbol{c}^{r'} \in \{0,1\}^m$ such that $\sum_{i=1}^{n} \min\{d_H(\boldsymbol{c}^j, \boldsymbol{a}^i) \mid 1 \leq j \leq r'\} \leq k$.*

We observe that the same columns of $\mathbf{A}$ can always be put in the same cluster. To state this property formally, we give the following definition.

Definition 1 (*Initial cluster and regular partition*) Let $\mathbf{A}$ be an $m \times n$-matrix with columns $\mathbf{a}^1, \ldots, \mathbf{a}^n$. An *initial cluster* is an inclusion maximal set $I \subseteq \{1, \ldots, n\}$ such that all the columns in the multiset $\{\mathbf{a}^j \mid j \in I\}$ are equal.

We say that a partition $\{I_1, \ldots, I_{r'}\}$ of the columns of matrix $\mathbf{A}$ is *regular* if for every initial cluster I, there is $i \in \{1, \ldots, r'\}$ such that $I \subseteq I_i$.

By the definition of the regular partition, every initial cluster of $\mathbf{A}$ is in some set I_i but the set I_i may contain many initial clusters.

Lemma 1 *Let $(\boldsymbol{A}, r, k)$ be a yes-instance of* Binary r-Means. *Then, there is a solution $\{I_1, \ldots, I_{r'}\}$, $r' \leq r$ which is regular (i.e, for any initial cluster I of $\boldsymbol{A}$, there is $i \in \{1, \ldots, r'\}$ such that $I \subseteq I_i$).*

Proof Let $\mathbf{a}^1, \ldots, \mathbf{a}^n$ be the columns of $\mathbf{A}$. By Observation 1, there are vectors $\mathbf{c}^1, \ldots, \mathbf{c}^{r'}$ for some $r' \leq r$ such that $\sum_{i=1}^{n} \min\{d_H(\mathbf{c}^j, \mathbf{a}^i) \mid 1 \leq j \leq r'\} \leq k$. Once we have the vectors $\mathbf{c}^1, \ldots, \mathbf{c}^{r'}$, a solution can be obtained by assigning each vector $\mathbf{a}^i$ to a closest vector in $\{\mathbf{c}^1, \ldots, \mathbf{c}^{r'}\}$. This implies the conclusion of the lemma. □

3 Binary r-Means parameterized by k

In this section we prove that Binary r-Means is FPT when parameterized by k. That is we prove the following theorem.

Theorem 1 Binary r-Means *is solvable in $2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$ time.*

The proof of Theorem 1 consists of two FPT Turing reductions. First we define a new auxiliary problem Cluster Selection and show how to reduce this problem to the Consensus Patterns problem. Then we can use as a black box the algorithm of Marx (2008) for this problem. The second reduction is from Binary r-Means to Cluster Selection and is based on the color coding technique of Alon et al. (1995). Note that our reductions are not classical polynomial but FPT *Turing reductions*, that is, the algorithms constructing the instances of Consensus Patterns and Cluster Selection respectively run in FPT time and, in particular, the numbers of constructed instances are bounded by functions that depend exponentially on the parameters but polynomially on the input sizes.

From Cluster Selection to Consensus Patterns In the Cluster Selection problem we are given a regular partition $\{I_1, \ldots, I_p\}$ of columns of matrix $\mathbf{A}$. Our

task is to select from each set I_i exactly one initial cluster such that the total deviation of all the vectors in these clusters from their mean is at most d. More formally,

CLUSTER SELECTION

Input: An $m \times n$-matrix $\mathbf{A}$ with columns $\mathbf{a}^1, \ldots, \mathbf{a}^n$, a regular partition $\{I_1, \ldots, I_p\}$ of $\{1, \ldots, n\}$, and a nonnegative integer d.

Task: Decide whether there is a set of initial clusters $J_1, \ldots, J_p$ and a vector $\mathbf{c} \in \{0, 1\}^m$ such that $J_i \subseteq I_i$ for $i \in \{1, \ldots, p\}$ and

$$\sum_{i=1}^{p} \sum_{j \in J_i} d_H(\mathbf{c}, \mathbf{a}^j) \leq d.$$

If $(\mathbf{A}, \{I_1, \ldots, I_p\}, d)$ is a yes-instance of CLUSTER SELECTION, then we say that the corresponding sets of initial clusters $\{J_1, \ldots, J_p\}$ and the vector $\mathbf{c}$ (or just $\{J_1, \ldots, J_p\}$ as $\mathbf{c}$ can be computed by the majority rule from the set of cluster) is a *solution* for the instance. We show that CLUSTER SELECTION is FPT when parameterized by d. Towards that, we use the results of Marx (2008) about the CONSENSUS PATTERNS problem.

CONSENSUS PATTERNS

Input: A (multi) set of p strings $\{s_1, \ldots, s_p\}$ over an alphabet Σ, a positive integer t and a nonnegative integer d.

Task: Decide whether there is a string s of length t over Σ, and a length-t substring s'_i of s_i for every $i \in \{1, \ldots, p\}$ such that $\sum_{i=1}^{p} d_H(s, s'_i) \leq d$.

Marx (2008) proved that CONSENSUS PATTERNS can be solved in $\delta^{\mathcal{O}(\delta)} \cdot |\Sigma|^{\delta} \cdot L^9$ time, where $\delta = d/p$ and L is the total length of all the strings in the input. This gives us the following lemma.

Lemma 2 (Marx 2008) CONSENSUS PATTERNS *can be solved in* $2^{\mathcal{O}(d \log d)} \cdot L^9$ *time, where* L *is the total length of all the strings in the input if the size of* Σ *is bounded by a constant.*

Now we are ready to show the following result for CLUSTER SELECTION.

Lemma 3 CLUSTER SELECTION *can be solved in* $2^{\mathcal{O}(d \log d)} \cdot (nm)^{\mathcal{O}(1)}$ *time.*

Proof Let $(\mathbf{A}, \{I_1, \ldots, I_p\}, d)$ be an instance of CLUSTER SELECTION. Let $\mathbf{a}^1, \ldots, \mathbf{a}^n$ be the columns of $\mathbf{A}$. First, we check whether there are initial clusters $J_1, \ldots, J_p$ and a vector $\mathbf{c} = \mathbf{a}^i$ for some $i \in \{1, \ldots, n\}$ such that $J_j \subseteq I_j$ for $j \in \{1, \ldots, p\}$ and $\sum_{j=1}^{p} \sum_{h \in J_j} d_H(\mathbf{c}, \mathbf{a}^h) \leq d$. Towards that we consider all possible choices of $\mathbf{c} = \mathbf{a}^i$ for $i \in \{1, \ldots, n\}$. Suppose that $\mathbf{c}$ is given. For every $j \in \{1, \ldots, p\}$, we find an initial cluster $J_j \subseteq I_j$ such that $\sum_{h \in J_j} d_H(\mathbf{c}, \mathbf{a}^h)$ is minimum. If $\sum_{j=1}^{p} \sum_{h \in J_j} d_H(\mathbf{c}, \mathbf{a}^h) \leq d$, then we return the corresponding solution, i.e., the set of initial clusters $\{J_1, \ldots, J_p\}$

and $\mathbf{c}$. Otherwise, we discard the choice of $\mathbf{c}$. It is straightforward to see that this procedure is correct and can be performed in polynomial time, as we have at most n choices of $\mathbf{c}$ and each sum $\sum_{h\in J_j} d_H(\mathbf{c}, \mathbf{a}^h)$ can be computed in polynomial time. From now on we assume that this is not the case. That is, if $(\mathbf{A}, \{I_1, \ldots, I_p\}, d)$ is a yes-instance, then $\mathbf{c} \neq \mathbf{a}^i$ for any solution. In particular, it means that for every solution $(\{J_1, \ldots, J_p\}, \mathbf{c})$, $d_H(\mathbf{c}, \mathbf{a}^j) \geq 1$ for $j \in J_1 \cup \ldots \cup J_p$. If $p > d$, we obtain that $(\mathbf{A}, \{I_1, \ldots, I_p\}, d)$ is a no-instance. In this case we return the answer and stop. Hence, from now we assume that $p \leq d$. Moreover, observe that $|J_1| + \cdots + |J_p| \leq d$ for any solution $(\{J_1, \ldots, J_p\}, \mathbf{c})$.

We consider all p-tuples of positive integers $(\ell_1, \ldots, \ell_p)$ such that $\ell_1 + \cdots + \ell_p \leq d$ and for each p-tuple check whether there is a solution $(\{J_1, \ldots, J_p\}, \mathbf{c})$ with $|J_i| = \ell_i$ for $i \in \{1, \ldots, p\}$. Note that there are at most $\binom{d}{p} \leq 2^d$ such p-tuples. If we find a solution for one of the p-tuples, we return it and stop. If we have no solution for any p-tuple, we conclude that we have a no-instance of the problem.

Assume that we are given a p-tuple $(\ell_1, \ldots, \ell_p)$. If there is $i \in \{1, \ldots, p\}$ such that there is no initial cluster $J_i \subseteq I_i$ with $|J_i| = \ell_i$, then we discard the current choice of the p-tuple. Otherwise, we reduce the instance of the problem using the following rule: if there is $i \in \{1, \ldots, p\}$ and an initial cluster $J \subseteq I_i$ such that $|J| \neq \ell_i$, then delete columns $\mathbf{a}^h$ for $h \in J$ from the matrix and set $I_i = I_i \setminus J$. By this rule, we can assume that each I_i contains only initial clusters of size ℓ_i. Let $I_i = \{J_1^i, \ldots, J_{q_i}^i\}$ where $J_1^i, \ldots, J_{q_i}^i$ are initial clusters for $i \in \{1, \ldots, p\}$.

We reduce the problem of checking the existence of a solution $(\{J_1, \ldots, J_p\}, \mathbf{c})$ with $|J_i| = \ell_i$ for $i \in \{1, \ldots, p\}$ to the Consensus Patterns problem. Towards that, we first define the alphabet $\Sigma = \{0, 1, a, b\}$ and strings

$$\overline{a} = \underbrace{a \ldots a}_{m+d}, \ \overline{b} = \underbrace{b \ldots b}_{m+d}, \quad \text{and} \quad \overline{0} = \underbrace{0 \ldots 0}_{d}.$$

Then $x = \overline{a}\overline{b} \cdots \overline{a}\overline{b}$ is defined to be the string obtained by the alternating concatenation of $d+1$ copies of $\overline{a}$ and $d+1$ copies of $\overline{b}$. Now we construct $\ell = \ell_1 + \cdots + \ell_p$ strings s_i^j for $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, \ell_i\}$. For each $i \in \{1, \ldots, p\}$, we do the following.

- For every $q \in \{1, \ldots, q_i\}$, select a column $\mathbf{a}^{f_{i,q}}$ for $f_{i,q} \in J_q^i$ and, by slightly abusing the notation, consider it to be a (0, 1)-string.
- Then for every $j \in \{1, \ldots, \ell_i\}$, set $s_i^j = x\mathbf{a}^{f_{i,1}}\overline{0}x \ldots x\mathbf{a}^{f_{i,q_i}}\overline{0}x$.

Observe that the strings s_i^j for $j \in \{1, \ldots, \ell_i\}$ are the same. We denote by $S = \{s_i^j \mid 1 \leq i \leq p,\ 1 \leq j \leq \ell_i\}$ the collection (multiset) of all constructed strings. Finally, we define $t = (m+d)(4d+5)$ and output (S, Σ, t, d) as an instance of Consensus Patterns. Now we prove the correctness of the reduction. □

Claim 1 *The instance* $(A, \{I_1, \ldots, I_p\}, d)$ *of* Cluster Selection *has a solution* $(\{J_1, \ldots, J_p\}, \boldsymbol{c})$ *with* $|J_i| = \ell_i$ *for* $i \in \{1, \ldots, p\}$ *if and only if* (S, Σ, t, d) *is a yes-instance of* Consensus Patterns.

Proof (of Claim 1) Suppose that the instance $(A, \{I_1, \ldots, I_p\}, d)$ of CLUSTER SELECTION has a solution $(\{J_1, \ldots, J_p\}, \mathbf{c})$ with $|J_i| = \ell_i$ for $i \in \{1, \ldots, p\}$. For every $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, \ell_i\}$, we select the substring $\hat{s}_i^j = x\mathbf{a}^{f_{i,j}}\overline{0}x$ where $f_{i,j} \in J_i$. By the definition, $|\hat{s}_i^j| = 2|x| + m + d = t$. We set $s = x\mathbf{c}\overline{0}x$ considering the vector $\mathbf{c}$ being a $(0, 1)$-string. Clearly, $|s| = t$. We have that

$$\sum_{i=1}^{p}\sum_{j=1}^{\ell_i} d_H(s, \hat{s}_i^j) = \sum_{i=1}^{p} \ell_i d_H(\mathbf{c}, \mathbf{a}^{f_{i,j}}) = \sum_{i=1}^{p}\sum_{f\in J_i} d_H(\mathbf{c}, \mathbf{a}^f) \leq d.$$

Therefore, (S, Σ, t, d) is a yes-instance of CONSENSUS PATTERNS.

Now we prove the reverse direction. Assume that (S, Σ, t, d) is a yes-instance of CONSENSUS PATTERNS. Let $\hat{s}_i^j$ be a substring of s_i^j of length t for $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, \ell_i\}$, and let s be a string of length t over Σ such that

$$\sum_{i=1}^{p}\sum_{j=1}^{\ell_i} d_H(s, \hat{s}_i^j) \leq d.$$

The crucial claim is that there is a positive integer $\alpha \leq t - m + 1$ such that for every $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, \ell_i\}$, $\hat{s}_i^j[\alpha..\alpha + m - 1] = \mathbf{a}^{h_{i,j}}$ for some $h_{i,j} \in I_i$. The intuition behind the claim is that the choice of every $\hat{s}_i^j$ should be "synchronized" with the choice of $\hat{s}_1^1$ due to the padding strings x. Otherwise, $\hat{s}_1^1$ and $\hat{s}_i^j$ would disagree in at least $d + 1$ symbols and we would obtain that $d_H(\hat{s}_1^1, \hat{s}_i^j) > d$. Then $d_H(\hat{s}_1^1, s) + d_H(s, \hat{s}_i^j) > d$ for any s by the triangle inequality. The two possible types of disagreements are shown in Fig. 1.

Consider the substring $\hat{s}_1^1$. Since $|\hat{s}_1^1| = t$, by the definition of the string s_1^1, we have that there is a positive integer $\beta \leq t - |x| + 1 = t - 2(m + d)(d + 1) + 1$ such that $\hat{s}_1^1[\beta..\beta + |x| - 1] = \hat{s}_1^1[\beta..\beta + 2(m + d)(d + 1) - 1] = x$. Let $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, \ell_i\}$. Suppose that $\hat{s}_i^j[\beta..\beta + 2(m + d)(d + 1) - 1] \neq x$. Recall that

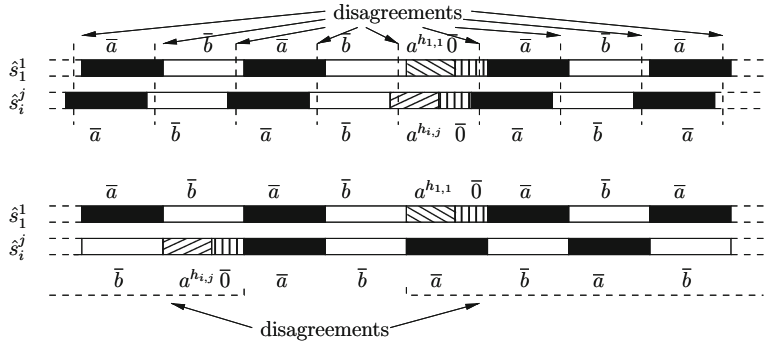

Fig. 1 Disagreements of $\hat{s}_1^1$ and $\hat{s}_i^j$

x contains $2(d+1)$ alternating copies of $\overline{a}$ and $\overline{b}$ and $|\overline{a}| = |\overline{b}| = m + d$. Because $d_H(\hat{s}_1^1, \hat{s}_i^j) \leq d_H(\hat{s}_1^1, s) + d_H(s, \hat{s}_i^j) \leq d$ (by the triangle inequality) and by the construction of the strings of S, we have that that either

(i) there is $\gamma = \beta + 2(m+d)h$ for some nonnegative integer $h \leq d$ such that $\hat{s}_i^j[\beta..\gamma-1] = x[\beta..\gamma-1]$ and $\hat{s}_i^j[\gamma..\gamma+m-1] = \mathbf{a}^{h_{i,j}}$ for some $h_{i,j} \in I_i$, or

(ii) there is $\gamma = \beta+2(m+d)h$ for some integer $1 \leq h \leq d$ such that $\hat{s}_i^j[\gamma..\beta+2(m+d)(d+1)-1] = x[\gamma..\beta+2(m+d)(d+1)-1]$ and $\hat{s}_i^j[\gamma-(m+d)..\gamma-d+1] = \mathbf{a}^{h_{i,j}}$ for some $h_{i,j} \in I_i$.

We would like to mention that in the above two cases, in one of them $x[\beta..\gamma-1]$ or $x[\gamma..\beta+2(m+d)(d+1)-1]$ may not be well defined. But at least in one of them it will be well defined. These cases are symmetric and without loss of generality we can consider only the case (i). We have that $\hat{s}_i^j[\gamma..\gamma+(m+d)-1]$ and $\hat{s}_1^1[\gamma..\gamma+(m+d)-1]$ differ in all the symbols, because all the symbols of $\hat{s}_i^j[\gamma..\gamma+(m+d)-1]$ are 0 or 1 and $\hat{s}_1^1[\gamma..\gamma+(m+d)-1] = \overline{a}$. Since $m+d > d$, it contradicts the property that $d_H(\hat{s}_1^1, \hat{s}_i^j) \leq d$. So we have that $\hat{s}_i^j[\beta..\beta+2(m+d)(d+1)-1] = x$ for every $i \in \{1,\ldots,p\}$ and $j \in \{1,\ldots,\ell_i\}$.

If $\beta > m+d$, then we set $\alpha = \beta-(m+d)$. Notice that that for every $i \in \{1,\ldots,p\}$ and $j \in \{1,\ldots,\ell_i\}$, $\hat{s}_i^j[\alpha..\alpha+m-1] = \mathbf{a}^{h_{i,j}}$ for some $h_{i,j} \in I_i$. Suppose $\beta \leq m+d$. Then we set $\alpha = \beta + 2(m+d)(d+1)$. Because $t = (m+d)(4d+5)$, it holds that $\alpha \leq t-m+1$ and for every $i \in \{1,\ldots,p\}$ and $j \in \{1,\ldots,\ell_i\}$, $\hat{s}_i^j[\alpha..\alpha+m-1] = \mathbf{a}^{h_{i,j}}$ for some $h_{i,j} \in I_i$. This concludes the proof.

Now consider $c = s[\alpha..\alpha+m-1]$. Because for every $i \in \{1,\ldots,p\}$ and $j \in \{1,\ldots,\ell_i\}$, $\hat{s}_i^j[\alpha..\alpha+m-1] = \mathbf{a}^{h_{i,j}}$ for some $h_{i,j} \in I_i$, we can assume that $\mathbf{c}$ is a $(0,1)$-string. We consider it as a vector of $\{0,1\}^m$.

Let $i \in \{1,\ldots,p\}$. We consider the columns $\mathbf{a}^{h_{i,j}}$ for $j \in \{1,\ldots,\ell_i\}$ and find among them the column $\mathbf{a}^{h_i}$ such that $d_H(\mathbf{c}, \mathbf{a}^{h_i})$ is minimum. Let $J_{r_i}^i \subseteq I_i$ for $r_i \in \{1,\ldots,q_i\}$ be an initial cluster that contains h_i. We show that $(\{J_{r_1}^1,\ldots,J_{r_p}^p\}, \mathbf{c})$ is a solution for the instance $(\mathbf{A}, \{I_1,\ldots,I_p\}, d)$ of Cluster Selection. To see it, it is sufficient to observe that the following inequality holds:

$$\sum_{i=1}^{p}\sum_{h\in J_{r_i}^i} d_H(\mathbf{c},\mathbf{a}^h) = \sum_{i=1}^{p}\ell_i d_H(\mathbf{c},\mathbf{a}^{h_i}) \leq \sum_{i=1}^{p}\sum_{j=1}^{\ell_i} d_H(\mathbf{c},\mathbf{a}^{h_{i,j}})$$

$$\leq \sum_{i=1}^{p}\sum_{j=1}^{\ell_i} d_H(s,\hat{s}_i^j) \leq d.$$

This concludes the proof of the claim. □

Using Claim 1 and Lemma 2, we solve Consensus Patterns for (S, Σ, t, d). This completes the description of our algorithm and its correctness proof. To evaluate the running time, recall first that we check in polynomial time whether we have a solution with $\mathbf{c}$ coinciding with a column of $\mathbf{A}$. If we fail to find such a solution,

then we consider at most 2^d p-tuples $(\ell_1, \ldots, \ell_p)$. Then for each p-tuple, we either discard it immediately or construct in polynomial time the corresponding instance of CONSENSUS PATTERNS, which is solved in $2^{\mathcal{O}(d \log d)} \cdot (nm)^{\mathcal{O}(1)}$ time by Lemma 2. Hence, the total running time is $2^{\mathcal{O}(d \log d)} \cdot (nm)^{\mathcal{O}(1)}$. □

Let us note that we are using Lemma 2 as a black box in our algorithm for CLUSTER SELECTION. By adapting the algorithm of Marx (2008) for CONSENSUS PATTERNS to solve CLUSTER SELECTION it is possible to improve the polynomial factor in the running time but this would demand repeating and rewriting various parts of Marx (2008).

From Binary r-Means to Cluster Selection Now we prove the main result of the section.

Proof (of Theorem 1) Our algorithm for BINARY r-MEANS uses the *color coding* technique introduced by Alon et al. (1995) (see also Cygan et al. 2015) for the introduction to this technique). In the end we obtain a deterministic algorithm but it is more convenient for us to describe a randomized Monte-Carlo algorithm and then explain how it could be derandomized.

Let $(\mathbf{A}, r, k)$ be a yes-instance of BINARY r-MEANS where $\mathbf{A} = (\mathbf{a}^1, \ldots, \mathbf{a}^n)$. Then by Lemma 1, there is a regular solution $\{I_1, \ldots, I_{r'}\}$ for this instance. Let $\mathbf{c}^1, \ldots, \mathbf{c}^{r'}$ be the corresponding means of the clusters. Recall that regularity means that for any initial cluster I, there is a cluster in the solution that contains it. We say that a cluster I_i of the solution is *simple* if it contain exactly one initial cluster and I_i is *composite* otherwise. Let I_i be a composite cluster of $\{I_1, \ldots, I_{r'}\}$ that contains $h \geq 2$ initial clusters. Then $\sum_{j \in I_i} d_H(\mathbf{c}^i, \mathbf{a}^j) \geq h - 1$. This observation immediately implies that a regular solution contains at most k composite clusters and the remaining clusters are simple. Moreover, the total number of initial clusters in the composite clusters is at most $2k$. Note also that if I_i is a simple cluster then $\mathbf{c}^i = \mathbf{a}^h$ for arbitrary $h \in I_i$, because $\sum_{j \in I_i} d_H(\mathbf{c}^i, \mathbf{a}^j) = 0$, That is, simple clusters do not contribute to the total cost of the solution.

Let $(\mathbf{A}, r, k)$ be an instance of BINARY r-MEANS where $\mathbf{A} = (\mathbf{a}^1, \ldots, \mathbf{a}^n)$. We construct the set $\mathcal{I}$ of initial clusters for the matrix $\mathbf{A}$. Let $s = |\mathcal{I}|$. The above observations imply that finding a solution for BINARY r-MEANS is equivalent to finding a set $\mathcal{I}' \subseteq \mathcal{I}$ of size at most $2k$ such that $\mathcal{I}'$ can be partitioned into at most $r - s + |\mathcal{I}'|$ composite clusters. More precisely, we are looking for $\mathcal{I}' \subseteq \mathcal{I}$ of size at most $2k$ such that there is a partition $\{P_1, \ldots, P_t\}$ of $\mathcal{I}'$ with $t \leq r - s + |\mathcal{I}'|$ and vectors $\mathbf{s}^1, \ldots, \mathbf{s}^t \in \{0, 1\}^m$ with the property that

$$\sum_{i=1}^{t} \sum_{I \in P_i} \sum_{j \in I} d_H(\mathbf{s}^i, \mathbf{a}^j) \leq k.$$

If $s \leq r$, then $(\mathbf{A}, r, k)$ is a trivial yes-instance of the problem with $\mathcal{I}$ being a solution. If $r + k < s$, then $(\mathbf{A}, r, k)$ is a trivial no-instance. Hence, we assume from now that $r < s \leq r + k$. Note that at most $2k$ initial cluster are in composite clusters of any solution. We color the elements of $\mathcal{I}$ independently and uniformly at random by $2k$ colors $1, \ldots, 2k$. Observe that if $(\mathbf{A}, r, k)$ is a yes-instance, then the initial clusters in

composite clusters of any fixed solution are colored by distinct colors with probability at least $\frac{(2k)!}{(2k)^{2k}} \geq e^{-2k}$ (see e.g. Cygan et al. 2015). We say that a solution $\{I_1, \ldots, I_{r'}\}$ for $(\mathbf{A}, r, k)$ is a *colorful* solution if all initial clusters that are included in composite clusters of $\{I_1, \ldots, I_{r'}\}$ are colored by distinct colors. We construct an algorithm for finding a colorful solution (if it exists).

Denote by $\mathcal{I}_1, \ldots, \mathcal{I}_{2k}$ the sets of color classes of initial clusters, i.e., the sets of initial clusters that are colored by $1, \ldots, 2k$, respectively. Note that some sets could be empty. We consider all possible partitions $\mathcal{P} = \{P_1, \ldots, P_t\}$ of nonempty subsets of $\{1, \ldots, 2k\}$ such that each set of $\mathcal{P}$ contains at least two elements. Notice that if $(\mathbf{A}, r, k)$ has a colorful solution $\{I_1, \ldots, I_{r'}\}$, then there is $\mathcal{P} = \{P_1, \ldots, P_t\}$ such that the following holds: a cluster I_i of the solution is a composite cluster containing initial clusters colored by a set of colors X_i if and only if there is $X_i \in \mathcal{P}$. Since we consider all possible $\mathcal{P}$, if $(\mathbf{A}, r, k)$ has a colorful solution, we will find $\mathcal{P}$ satisfying this condition. Assume that $\mathcal{P} = \{P_1, \ldots, P_t\}$ is given. If $s - |P_1| - \cdots - |P_t| + t > r$, we discard the current choice of $\mathcal{P}$. Assume from now that this is not the case.

For each $i \in \{1, \ldots, t\}$, we do the following. Let $P_i = \{i_1, \ldots, i_p\} \subseteq \{1, \ldots, 2k\}$. Let $J^i_j = \bigcup_{I \in \mathcal{I}_{i_j}} I$ and $J^i = J^i_1 \cup \cdots \cup J^i_p$. Denote by $\mathbf{A}_i$ the submatrix of $\mathbf{A}$ containing the columns $\mathbf{a}^h$ with $h \in J^i$. We use Lemma 3 to find the minimum nonnegative integer $d_i \leq k$ such that $(\mathbf{A}_i, \{J^i_1, \ldots, J^i_p\}, d_i)$ is a yes-instance of Cluster Selection. If such a value of d_i does not exist, we discard the current choice of $\mathcal{P}$. Otherwise, we find the corresponding solution $(\{L^i_1, \ldots, L^i_p\}, \mathbf{s}^i)$ of Cluster Selection. Let $L^i = L^i_1 \cup \ldots \cup L^i_p$.

If we computed d_i and constructed L^i for all $i \in \{1, \ldots, t\}$, we check whether $d_1 + \cdots + d_t \leq k$. If it holds, we return the colorful solution with the composite clusters $L^1, \ldots, L^t$ whose means are $\mathbf{s}^1, \ldots, \mathbf{s}^t$ respectively and the remaining clusters are simple. Otherwise, we discard the choice of $\mathcal{P}$. If for one of the choices of $\mathcal{P}$ we find a colorful solution, we return it and stop. If we fail to find a solution for all possible choices of $\mathcal{P}$, we return the answer NO and stop.

If the described algorithm produces a solution, then it is straightforward to verify that this is a colorful solution to $(\mathbf{A}, r, k)$ recalling that simple clusters do not contribute to the total cost of the solution. In the other direction, if $(\mathbf{A}, r, k)$ has a colorful solution $\{I_1, \ldots, I_{r'}\}$, then there is $\mathcal{P} = \{P_1, \ldots, P_t\}$ such that cluster I_i of the solution is a composite cluster containing initial clusters colored by a set of colors X_i if and only if there is $X_i \in \mathcal{P}$. Let $L_1, \ldots, L_t$ be the composite clusters of the solution that correspond to $P_1, \ldots, P_t$, respectively and denote by $\mathbf{s}^1, \ldots, \mathbf{s}^t$ their means. Let $d_i = \sum_{h \in L_i} d_H(\mathbf{s}^i, \mathbf{a}^h)$ for $i \in \{1, \ldots, t\}$. It immediately follows that for each $i \in \{1, \ldots, t\}$, it holds that if $P_i = \{i_1, \ldots, i_p\}$, then the constructed instance $(\mathbf{A}_i, \{J^i_1, \ldots, J^i_p\}, d_i)$ of Cluster Selection is a yes-instance. Hence, the algorithm returns a colorful solution.

To evaluate the running time, recall that we consider $2^{\mathcal{O}(k \log k)}$ partitions $\mathcal{P} = \{P_1, \ldots, P_t\}$ of nonempty subsets of $\{1, \ldots, 2k\}$. Then for each $\mathcal{P}$, we construct in polynomial time at most $k|\mathcal{P}|$ instances of Cluster Selection. These instances are solved in $2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$ time by Lemma 3. We conclude that the total running time of the algorithm that checks the existence of a colorful solution is $2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$.

Clearly, if for a random coloring of $\mathcal{I}$, there is a colorful solution to $(\mathbf{A}, r, k)$, then $(\mathbf{A}, r, k)$ is a yes-instance. We consider $N = \lceil e^{2k} \rceil$ random colorings of $\mathcal{I}$ and for each coloring, we check the existence of a colorful solution. If we find such a solution, we return it and stop. Otherwise, if we failed to find a solution for all colorings, we return the answer NO. Recall that if $(\mathbf{A}, r, k)$ is a yes-instance with a solution $\{I_1, \ldots, I_{r'}\}$, then the initial clusters that are included in the composite clusters of the solution are colored by distinct colors with the probability at least $\frac{(2k)!}{(2k)^{2k}} \geq e^{-2k}$. Hence, the probability that a yes-instance has no colorful solution is at most $(1 - e^{-2k})$ and, therefore, the probability that a yes-instance has no colorful solution for $N \geq e^{2k}$ random colorings is at most $(1 - e^{-2k})^{e^{2k}} \leq e^{-1}$. We conclude that our randomized algorithm returns a false negative answer with probability at most $e^{-1} < 1$. The total running time of the algorithm is $N \cdot 2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$, that is, $2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$.

By the standard derandomization technique using perfect hash families, see Alon et al. (1995) and Naor et al. (1995), our algorithm can be derandomized. Thus, we conclude that BINARY r-MEANS is solvable in $2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$ deterministic time. □

4 Kernelization for BINARY r-MEANS

In this section we show that BINARY r-MEANS admits a polynomial kernel when parameterized by r and k. Then we complement this result and Theorem 1 by proving that it is unlikely that the problem has a polynomial kernel when parameterized by k only.

4.1 Polynomial kernel with parameter $k + r$

We need the following simple lemma.

Lemma 4 *Let $(\boldsymbol{A}, r, k)$ be a yes-instance of* BINARY r-MEANS *with $\boldsymbol{A} = (\boldsymbol{a}^1, \ldots, \boldsymbol{a}^n)$. For $i, j \in \{1, \ldots, n\}$, if $d_H(\boldsymbol{a}^i, \boldsymbol{a}^j) > k$, then i and j are in distinct clusters of any solution.*

Proof Let $(I_1, \ldots, I_r)$ be a solution for $(\mathbf{A}, r, k)$. Let also $\mathbf{c}^1, \ldots, \mathbf{c}^{r'}$ be the means of $I_1, \ldots, I_{r'}$. Supposethat $d_H(\mathbf{a}^i, \mathbf{a}^j) > k$ for some $i, j \in \{1, \ldots, n\}$. For the sake of contradiction, assume that there is a cluster I_p such that $i, j \in I_p$. Then

$$\sum_{i=1}^{r'} \sum_{h \in I_i} d_H(\mathbf{c}^h, \mathbf{a}^j) \geq \sum_{h \in I_p} d_H(\mathbf{c}^p, \mathbf{a}^h) \geq d_H(\mathbf{c}^p, \mathbf{a}^i) + d_H(\mathbf{c}^p, \mathbf{a}^j)$$
$$\geq d_H(\mathbf{a}^i, \mathbf{a}^j) > k$$

by the triangle inequality, contradicting that $\{I_1, \ldots, I_{r'}\}$ is a solution. □

To construct a polynomial kernel for BINARY r-MEANS, we first describe Algorithm 1. This algorithm takes as the input an instance $(\mathbf{A}, r, k)$ of BINARY r-MEANS

Algorithm 1: Preprocessing for BINARY r-MEANS

input : An $n \times m$ matrix $\mathbf{A} = (\mathbf{a}^1, \ldots, \mathbf{a}^n)$, positive integer r and nonnegative integer k.
output: Matrices $\mathbf{A}_1, \ldots, \mathbf{A}_s$ or NO.

```
1  begin
2      Set I := {1, ..., n};
3      Construct the partition {I_1, ..., I_t} of I into initial clusters;
4      if t > k + r then
5          Return NO
6      else
7          for i := 1 to t do
8              while |I_i| > k + 1 do
9                  Select arbitrary j ∈ I_i;
10                 Set I_i := I_i\{j} and I := I\{j}
11             end
12         end
13         Set s := 0;
14         while I ≠ ∅ do
15             Set s := s + 1;
16             Select arbitrary j ∈ I;
17             Set S := {j} and I := I\{j};
18             while there are p ∈ S and q ∈ I s.t. d_H(a^p, a^q) ≤ k do
19                 Set S := S ∪ {q} and I := I\{q}
20             end
21             Set J := {1, ... m};
22             while |J| ≥ 2 and A[J, S] has a uniform row with an index h ∈ J do
23                 Set J := J\{h}
24             end
25             Set A_s := A[S, J];
26         end
27         if s > r then Return NO else Return A_1, ..., A_s
28     end
29 end
```

and ether concludes that $(\mathbf{A}, r, k)$ is a no-instance or returns $s \leq r$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_s$ of bounded in k and r size such that $(\mathbf{A}, r, k)$ is a yes-instance of BINARY r-MEANS if and only if there are positive $r_1, \ldots, r_s$ and nonnegative $k_1, \ldots k_s$ such that (i) $r_1 + \cdots + r_s \leq r$, (ii) $k_1 + \cdots + k_s \leq k$, and (iii) $(\mathbf{A}_i, r_i, k_i)$ is a yes-instance of BINARY r-MEANS for every $i \in \{1, \ldots, s\}$.

We say that a row of a binary matrix is *uniform* if all its elements are equal. Thus, a uniform row consists entirely from 0s or from 1s. Otherwise, a row is *nonuniform*.

The properties of Algorithm 1 are summarized in the following lemma.

Lemma 5 *Given an instance $(\boldsymbol{A}, r, k)$ of* BINARY r-MEANS*, Algorithm 1 runs in $\mathcal{O}(n^2 m)$ time and either correctly concludes that $(\boldsymbol{A}, r, k)$ is a no-instance of* BINARY r-MEANS *and returns* NO *or returns $s \leq r$ matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_s$ such that*

– *$(\boldsymbol{A}, r, k)$ is a yes-instance of* BINARY r-MEANS *if and only if there are positive $r_1, \ldots, r_s$ and nonnegative $k_1, \ldots k_s$ such that (i) $r_1 + \cdots + r_s \leq r$, (ii) $k_1 + \cdots + k_s \leq k$, and (iii) $(\boldsymbol{A}_i, r_i, k_i)$ is a yes-instance of* BINARY r-MEANS *for every $i \in \{1, \ldots, s\}$,*

– *for every* $i \in \{1, \ldots, s\}$, $\boldsymbol{A}_i$ *is* $m_i \times n_i$ *matrix with* $m_i \leq \max\{(\ell_i - 1)k, 1\}$, *where* ℓ_i *is the number of distinct columns of* $\boldsymbol{A}_i$, *and* $n_1 + \cdots + n_s \leq (k+1)(k+r)$,
– *the total number of distinct columns in* $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_s$ *is at most* $k + r$.

Proof Let $(\mathbf{A}, r, k)$ be an instance of BINARY r-MEANS. □

In Line 3, Algorithm 1 construct the partition $\{I_1, \ldots, I_t\}$ of $I = \{1, \ldots, n\}$ into initial clusters. Let us remind that an initial cluster is an inclusion maximal set of equal columns of the input matrix. Suppose that $(\mathbf{A}, r, k)$ is a yes-instance of BINARY r-MEANS. By Observation 1, there is a set of means $\{\mathbf{c}^1, \ldots, \mathbf{c}^{r'}\}$ for some $r' \leq r$ such that $\sum_{i=1}^{n} \min\{d_H(\mathbf{c}^j, \mathbf{a}^i) \mid 1 \leq j \leq r'\} \leq k$. It immediately implies that $\mathbf{A}$ has at most k columns that are distinct from $\mathbf{c}^1, \ldots, \mathbf{c}^{r'}$. Therefore, $\mathbf{A}$ has at most $k + r$ distinct columns. Therefore, if $t > k + r$, then $(\mathbf{A}, r, k)$ is a no-instance and the algorithm correctly returns NO in Line 5. From now on, assume that $t \leq k + r$.

In Lines 7–12, Algorithm 1 performs the following for every $i \in \{1, \ldots, t\}$: if the initial cluster has size at least $k + 2$, then we delete an arbitrary $j \in I_i$ from I_i and I. Let $I' = I \setminus \{j\}$. We show the following claim.

Claim 2 $(\boldsymbol{A}[\{1, \ldots, m\}, I], r, k)$ *is a yes-instance of* BINARY r-MEANS *if and only if* $(\boldsymbol{A}[\{1, \ldots, m\}, I'], r, k)$ *is a yes-instance.*

Proof (of Claim 2) Since $\mathbf{A}[\{1, \ldots, m\}, I']$ is a sub-matrix of $\mathbf{A}[\{1, \ldots, m\}, I]$ we have that if $(\mathbf{A}[\{1, \ldots, m\}, I], r, k)$ is a yes-instance of BINARY r-MEANS, then $(\mathbf{A}[\{1, \ldots, m\}, I'], r, k)$ is a yes-instance as well.

For the reverse direction, suppose that $(\mathbf{A}[\{1, \ldots, m\}, I'], r, k)$ is a yes-instance. Then, by Observation 1, there exist $r' \leq r$ and r' vectors $\mathbf{c}^1, \ldots, \mathbf{c}^{r'} \in \{0, 1\}^m$ such that

$$\sum_{h \in I \setminus \{j\}} \min\{d_H(\mathbf{c}^{h'}, \mathbf{a}^h) \mid 1 \leq h' \leq r'\} \leq k.$$

This implies that

$$\sum_{h \in I_i \setminus \{j\}} \min\{d_H(\mathbf{c}^{h'}, \mathbf{a}^h) \mid 1 \leq h' \leq r'\} \leq k. \tag{2}$$

Since $|I_i| > k + 1$ and all the vectors $\{\mathbf{a}^h : h \in I_i\}$ are identical, by (2), we have that there is a vector $\mathbf{c}^{h'} \in \{\mathbf{c}^1, \ldots, \mathbf{c}^{r'}\}$ such that $\mathbf{c}^{h'} = \mathbf{a}^j$. This implies that

$$\sum_{h \in I} \min\{d_H(\mathbf{c}^{h'}, \mathbf{a}^h) \mid 1 \leq h' \leq r'\} \leq k.$$

Therefore, $(\mathbf{A}[\{1, \ldots, m\}, I], r, k)$ is a yes-instance of BINARY r-MEANS. □

Claim 2 implies that for the set I obtained after the execution of Lines 7–12, $(\mathbf{A}[\{1, \ldots, m\}, I], r, k)$ is a yes-instance of BINARY r-MEANS if and only if the original instance $(\mathbf{A}, r, k)$ is a yes-instance. Notice also that $|I_i| \leq k + 1$ and $t \leq k + r$. Therefore, $|I| \leq (k + 1)(k + r)$.

In Lines 13–26, Algorithm 1 greedily constructs sets $S \subseteq I$ for $s \leq r$. Denote by $S_1, \ldots, S_s$ these sets. By the construction, these sets form a partition of I. Notice that the Hamming distance between any two columns of $\mathbf{A}$ with indices in different sets S_i and S_j is more than k. The crucial property of these sets is that every cluster of a solution is entirely in one of the sets by Lemma 4. This way $S_1, \ldots, S_s$ separate the clustering problem into subproblems. More precisely,

Claim 3 *Let $\{R_1, \ldots, R_{r'}\}$ be a solution for $(\mathbf{A}[\{1, \ldots, m\}, I], r, k)$. Then for every $i \in \{1, \ldots, r'\}$ there is $j \in \{1, \ldots, s\}$ such that $R_i \subseteq S_j$.*

By Claim 3, if $s > r$, then $(\mathbf{A}[\{1, \ldots, m\}, I], r, k)$ is a no-instance of Binary r-Means. Hence, if in Line 27, Algorithm 1 returns NO, then $(\mathbf{A}, r, k)$ is a no-instance. Assume that this is not the case.

By Claim 3, we conclude that $(\mathbf{A}[\{1, \ldots, m\}, I], r, k)$ is a yes-instance of Binary r-Means if and only if there are positive $r_1, \ldots, r_s$ and nonnegative $k_1, \ldots k_s$ such that (i) $r_1 + \cdots + r_s \leq r$, (ii) $k_1 + \cdots + k_s \leq k$ and (iii) $(\mathbf{A}[\{1, \ldots, m\}, S_i], r_i, k_i)$ is a yes-instance of Binary r-Means for $i \in \{1, \ldots, s\}$.

In Lines 13–26, besides constructing $S_1, \ldots, S_s$, Algorithm 1 deletes indices of uniform rows in each $\mathbf{A}[\{1, \ldots, m\}, S_i]$ to construct $\mathbf{A}_i$ (except one index if all the rows are uniform). It is straightforward to see that uniform rows are irrelevant for Binary r-Means and $(\mathbf{A}[\{1, \ldots, m\}, S_i], r_i, k_i)$ is a yes-instance if and only if $(\mathbf{A}_i, r_i, k_i)$ is a yes-instance.

We conclude that Algorithm 1 either correctly returns NO in Lines 5 or 27, or returns $s \leq r$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_s$ such that $(\mathbf{A}, r, k)$ is a yes-instance of Binary r-Means if and only if there are positive $r_1, \ldots, r_s$ and nonnegative $k_1, \ldots k_s$ such that (i) $r_1 + \cdots + r_s \leq r$, (ii) $k_1 + \cdots + k_s \leq k$ and (iii) $(\mathbf{A}_i, r_i, k_i)$ is a yes-instance of Binary r-Means for $i \in \{1, \ldots, s\}$.

Assume that Algorithm 1 returned $\mathbf{A}_1, \ldots, \mathbf{A}_s$. Let $\mathbf{A}_i$ be $m_i \times n_i$ matrix for $i \in \{1, \ldots, s\}$. Since $|I| \leq (k+1)(k+r)$ after the execution of Lines 7–12 and $n_1 + \cdots + n_s = |I|$, we have that $n_1 + \cdots + n_s \leq (k+1)(k+r)$.

To bound m_i from above, we show the following claim.

Claim 4 *For every $i \in \{1, \ldots, s\}$, the matrix $\mathbf{A}[\{1, \ldots, m\}, S_i]$ has at most $(\ell_i - 1)k$ nonuniform rows, where ℓ_i is the number of initial clusters in S_i.*

Proof (of Claim 4) Fix an index $i \in \{1, \ldots, s\}$ and let $\ell = \ell_i$. Recall that S_i is constructed greedily by adding the index of a column that is at distance at most k from some columns whose index is already included in S_i. Suppose that S contains initial clusters $I_{i_1}, \ldots, I_{i_\ell}$. We can assume that S_i constructed by consecutive adding $I_{i_1}, \ldots, I_{i_\ell}$. Denote by $Q_j = I_{i_1} \cup \ldots I_{i_j}$ for $j \in \{1, \ldots, \ell\}$.

For every $j \in \{1, \ldots, \ell\}$, we show inductively that $\mathbf{A}[\{1, \ldots, m\}, Q_j]$ has at most $(j-1)k$ nonuniform rows. The claim is trivial for $j = 1$. Let $p \in I_{i_1}$ and $q \in I_{i_2}$. Since $d_H(\mathbf{a}^p, \mathbf{a}^q) \leq k$, we have that $\mathbf{a}^p$ and $\mathbf{a}^q$ differ in at most k positions. Therefore, $\mathbf{A}[\{1, \ldots, m\}, Q_2]$ has at most k nonuniform rows.

Now consider the case $j \geq 3$. Let $p \in I_{i_j}$. By the induction hypothesis, $\mathbf{A}[\{1, \ldots, m\}, Q_{j-1}]$ has at least $m - (j-2)k$ uniform rows. Denote by $J \subseteq \{1, \ldots, m\}$, the set of indices of uniform rows of $\mathbf{A}[\{1, \ldots, m\}, Q_{j-1}]$. Since

$d_H(\mathbf{a}^p, \mathbf{a}^q) \leq k$ for some $q \in Q_{i-1}$, there are at most k positions where $\mathbf{a}^p$ and $\mathbf{a}^q$ are distinct. In particular, this implies that there are at most k indices of J for which the corresponding elements of $\mathbf{a}^p$ and $\mathbf{a}^q$ are distinct. This immediately implies that $\mathbf{A}[\{1,\ldots,m\}, Q_j]$ has at least $|J| - k \geq m - (j-2)k - k = m - (j-1)k$ uniform rows. Hence, $\mathbf{A}[\{1,\ldots,m\}, Q_j]$ has at most $(j-1)k$ nonuniform rows. □

By Claim 4, we obtain that $m_i \leq \max\{(\ell_i - 1)k, 1\}$ for $i \in \{1,\ldots,s\}$.

Notice that each initial cluster is included in some S_i for $\in \{1,\ldots,s\}$. Since $t \leq k + r$, this implies that the total number of distinct columns in $\mathbf{A}_1,\ldots,\mathbf{A}_s$ is at most $k + r$. This concludes the correctness proof.

To evaluate the running time, observe that Lines 2 and 3 can be done in $\mathcal{O}(n^2m)$ time. Lines 7–12 can be done in $\mathcal{O}(n)$ time. Then Lines 13–26 can be done in $\mathcal{O}(n^2m)$. This implies that the total running time is $\mathcal{O}(n^2m)$. □

We believe that the preprocessing Algorithm 1 is interesting by itself as it reduces an instance of our clustering problem to a collection of instances with matrices of bounded size that are independent of each other and may be solved separately. In other words, the algorithm gives a *Turing kernel* for LOW GF(2)- RANK APPROXIMATION [we refer to Fomin et al. (2019) for the definition of the notion]. It is also useful to make the following observation.

Observation 2 *Algorithm 1 can be implemented to run in $\mathcal{O}(k(k+r)mn)$ time.*

Proof Observe that in Lines 3, we construct the partition $\{I_1,\ldots,I_t\}$ of $\{1,\ldots,n\}$ into initial clusters, and then, in Lines 4–5, we stop and return NO if $t > k + r$. Instead, we can stop immediately if constructing the partition into initial clusters we discover $k + r + 1$ distinct columns of $\mathbf{A}$. This allows to reduce the running time to $\mathcal{O}((k+r)nm)$. Then after executing Lines 7–12, we have that $|I| \leq (k+1)(k+r)$ and the remaining steps can be done in $\mathcal{O}(k(k+r)mn)$ time. □

Now we show that BINARY r-MEANS admits a polynomial kernel when parameterized by r and k.

Theorem 2 BINARY r-MEANS *admits a kernel of size $\mathcal{O}(k^2(k+r)^2)$. Moreover, the kernelization algorithm in $(k+r)^{\mathcal{O}(1)} \cdot nm$ time either solves the problem or outputs an equivalent instance of* BINARY r-MEANS *such that the matrix in this instance has at most $k + r$ pairwise distinct columns and $\mathcal{O}(k(k+r))$ pairwise distinct rows and the values of r and k are the same as in the input instance.*

Proof Let $(\mathbf{A}, r, k)$ be an instance of BINARY r-MEANS. Let $\mathbf{a}^1,\ldots,\mathbf{a}^n$ be the columns of $\mathbf{A}$. □

We apply Algorithm 1. If the algorithm returns NO, we output a trivial no-instance, say, the instance with $\mathbf{A} = (0, 1)$, $r = 1$ and $k = 0$. Assume that this is not the case. Then, by Lemma 5, the algorithm returns $s \leq r$ matrices $\mathbf{A}_1,\ldots,\mathbf{A}_s$ such that

– $(\mathbf{A}, r, k)$ is a yes-instance of BINARY r-MEANS if and only if there are positive $r_1,\ldots,r_s$ and nonnegative $k_1,\ldots k_s$ such that (i) $r_1 + \cdots + r_s \leq r$, (ii) $k_1 + \cdots + k_s \leq k$, and (iii) $(\mathbf{A}_i, r_i, k_i)$ is a yes-instance of BINARY r-MEANS for every $i \in \{1,\ldots,s\}$,

– for every $i \in \{1, \ldots, s\}$, $\mathbf{A}_i$ is $m_i \times n_i$ matrix with $m_i \leq \max\{(\ell_i - 1)k, 1\}$, where ℓ_i is the number of distinct columns of $\mathbf{A}_i$, and $n_1 + \cdots + n_s \leq (k+1)(k+r)$,
– the total number of distinct columns in $\mathbf{A}_1, \ldots, \mathbf{A}_s$ is at most $k + r$.

To construct a kernel, we "glue" the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_s$ into a single matrix.

Let $\ell = \max_{1 \leq i \leq s} m_i$. For $i \in \{1, \ldots, s\}$, we construct $\ell \times n_i$ matrix $\mathbf{A}'_i$ from $\mathbf{A}_i$ by adding $\ell - m_i$ zero rows. For $i \in \{1, \ldots, s\}$, $\mathbb{0}_i$ and $\mathbb{1}_s$ denote $\lceil (k+1)/2 \rceil \times n_i$-matrices with all the elements 0 and 1, respectively. We use the matrices $\mathbf{A}'_i$, $\mathbb{0}_i$ and $\mathbb{1}_i$ as blocks to define

$$\mathbf{B} = \left(\begin{array}{c|c|c|c} \mathbf{A}'_1 & \mathbf{A}'_2 & \cdots & \mathbf{A}'_s \\ \hline \mathbb{1}_1 & \mathbb{0}_2 & \cdots & \mathbb{0}_s \\ \hline \mathbb{0}_1 & \mathbb{1}_2 & \cdots & \mathbb{0}_s \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \mathbb{0}_1 & \mathbb{0}_2 & \cdots & \mathbb{1}_s \end{array}\right).$$

We assume that the columns of $\mathbf{A}_i$ are indexed by the set of indexes S_i for $i \in \{1, \ldots, s\}$ and the columns of $\mathbf{B}$ are denoted by $\mathbf{b}^i$ with $i \in S_1 \cup \ldots \cup S_s$ following the convention that the columns of $\left(\begin{array}{c} \mathbf{A}'_i \\ \hline \vdots \end{array}\right)$ are indexed by $j \in S_i$ according to the indexing of the corresponding columns of $\mathbf{A}_i$. We also assume that the rows or $\mathbf{B}$ are indexed by $j \in J$.

Then, our kernelization algorithm returns the instance $(\mathbf{B}, r, k)$ of Binary r-Means. The correctness of the algorithm is based on the following claim.

Claim 5 *$(\boldsymbol{A}, r, k)$ is a yes-instance of* Binary r-Means *if and only if $(\boldsymbol{B}, r, k)$ is a yes-instance.*

Proof (of Claim 5) Suppose that $(\mathbf{A}, r, k)$ is a yes-instance. By Lemma 5, there are positive $r_1, \ldots, r_s$ and nonnegative $k_1, \ldots k_s$ such that (i) $r_1 + \cdots + r_s \leq r$, (ii) $k_1 + \cdots + k_s \leq k$ and (iii) $(\mathbf{A}_i, r_i, k_i)$ is a yes-instance of Binary r-Means for $i \in \{1, \ldots, s\}$. Then the union of solutions for $(\mathbf{A}_i, r_i, k_i)$ for $i \in \{1, \ldots, s\}$ gives a solution for $(\mathbf{B}, r, k)$. Therefore, $(\mathbf{B}, r, k)$ is a yes-instance.

Assume that $(\mathbf{B}, r, k)$ is a yes-instance. Notice that if $p \in S_i$ and $q \in S_j$ for distinct $i, j \in \{1, \ldots, s\}$, then $d_H(\mathbf{b}^p, \mathbf{b}^q) > k$. Then by Lemma 4, p and q are in distinct clusters of every solution. This implies that there are positive $r_1, \ldots, r_s$ and nonnegative $k_1, \ldots k_s$ such that (i) $r_1 + \cdots + r_s \leq r$, (ii) $k_1 + \cdots + k_s \leq k$, and (iii) $(\mathbf{B}[J, S_i], r_i, k_i)$ is a yes-instance of Binary r-Means for every $i \in \{1, \ldots, s\}$. By the construction of $\mathbf{B}$, $(\mathbf{A}_i, r_i, k_i)$ is a yes-instance of Binary r-Means for $i \in \{1, \ldots, s\}$. Then by Lemma 5, $(\mathbf{A}, r, k)$ is a yes-instance. □

To bound the size of $\mathbf{B}$, note that $\mathbf{B}$ has $n_1 + \cdots + n_s \leq (k+1)(k+r)$ columns and at most $k + r$ of them are pairwise distinct. The matrices $\mathbf{A}'_1, \ldots, \mathbf{A}'_s$ have ℓ rows. Because $s \leq r$, we obtain that $\mathbf{B}$ has at most $\ell + \lceil (k+1)/2 \rceil r$ rows and at most $\ell + r$ distinct rows. Since $\ell = \max_{1 \leq i \leq s} m_i$ and $m_i \leq \max\{(\ell_i - 1)k, 1\}$ for $i \in \{1, \ldots, s\}$, $\ell \leq k(k+r)$. Therefore, $\mathbf{B}$ has $\mathcal{O}(k(k+r))$ (pairwise distinct) rows. Respectively, the number of elements of $\mathbf{B}$ is $\mathcal{O}(k^2(k+r)^2)$.

Finally, to evaluate the running time, note that Algorithm 1 can be implemented to run in $\mathcal{O}(k(k+r)mn)$ time by Observation 2. Then, the construction of $\mathbf{B}$ can be done in $(k+r)^{\mathcal{O}(1)}$ time. Thus, the total running time is $(k+r)^{\mathcal{O}(1)} \cdot nm$. □

4.2 Ruling out polynomial kernel with parameter *k*

Our next aim is to show that Binary r-Means parameterized by k does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. We do this in two steps. First, we use the composition technique introduced by Bodlaender et al. (2009) (see also Cygan et al. (2015) for the introduction to this technique) to show that it is unlikely that the Consensus String with Outliers problem introduced by Boucher et al. (2011) has a polynomial kernel. Then, we use this result to prove the claim for Binary r-Means.

Consensus String with Outliers

Input: A (multi) set of n strings $S = \{s_1, \ldots, s_n\}$ of the same length ℓ over an alphabet Σ, a positive integer r, and a nonengative integer d.

Task: Decide whether there is a string s of length ℓ over Σ and $I \subseteq \{1, \ldots, n\}$ with $|I| = r$ such that $\sum_{i \in I} d_H(s, s_i) \leq d$.

The parameterized complexity of Consensus String with Outliers was investigated by Boucher et al. (2011) and the authors obtained a number of approximation and inapproximability results. In particular, they proved that the problem is FPT when parameterized by d. We show that it is unlikely that this problem has a polynomial kernel for this parameterization.

Theorem 3 Consensus String with Outliers *has no polynomial kernel when parameterized by d unless NP $\subseteq$ coNP/poly even for strings over the binary alphabet. Moreover, the result holds for the instances with $r \leq d$.*

Proof We use the fact that Consensus String with Outliers is NP-complete for strings over the binary alphabet $\Sigma = \{0, 1\}$ (Boucher et al. 2011) and construct a composition algorithm for the problem parameterized by d. □

Let $(S_1, r, d), \ldots, (S_t, r, d)$ be instances of Consensus String with Outliers where $S_1, \ldots, S_t$ are (multi) sets of binary strings of the same length ℓ. Denote by $\bar{0}$ and $\bar{1}$ the strings of length $d+1$ composed by 0s and 1s, respectively. That is,

$$\bar{0} = \underbrace{0 \ldots 0}_{d+1} \text{ and } \bar{1} = \underbrace{1 \ldots 1}_{d+1}.$$

For each $i \in \{1, \ldots, t\}$, we define the set of strings

$$S'_i = \{s\bar{0}^{i-1}\bar{1}\bar{0}^{t-i} \mid s \in S_i\}.$$

Then, we put $S^* = \bigcup_{i=1}^{t} S'_i$. Our composition algorithm outputs the instance (S^*, r, d) of Consensus String with Outliers.

We show that (S^*, r, d) is a yes-instance of Consensus String with Outliers if and only if there exists $i \in \{1, \ldots, t\}$ such that (S_i, r, d) is a yes-instance. Before proving the correctness let us define some notations. Assume that for $i \in \{1, \ldots, t\}$, the strings of S_i and S'_i are indexed by indices from a set I_i, where $I_1, \ldots, I_t$ are disjoint, and denote the strings of $S = \bigcup_{i=1}^{t} S_i$ and S^* by s_j and s'_j, respectively, for $j \in \bigcup_{i=1}^{t} I_i$.

Claim 6 *(S^*, r, d) is a yes-instance of* Consensus String with Outliers *if and only if there exists $i \in \{1, \ldots, t\}$ such that (S_i, r, d) is a yes-instance.*

Proof (of Claim 6) Suppose there exists $i \in \{1, \ldots, t\}$ such that (S_i, r, d) is a yes-instance of Consensus String with Outliers. Then, there exist $I \subseteq I_i$ and a binary string s of length ℓ such that $|I| = r$ and $\sum_{i \in I} d_H(s, s_i) \leq d$. Let $s' = s\bar{0}^{i-1}\bar{1}\bar{0}^{t-i}$. Since $I \subseteq I_i$, we have that

$$\sum_{i \in I} d_H(s', s'_i) = \sum_{i \in I} d_H(s, s_i) \leq d.$$

That is, (S^*, r, d) is a yes-instance.

For the other direction, assume that (S^*, r, d) is a yes-instance of Consensus String with Outliers. Then, there exit $I \subseteq \bigcup_{i=}^{t} I_i$ and a binary string s' of length $\ell + (d+1)t$ such that $|I| = r$ and $\sum_{i \in I} d_H(s', s'_i) \leq d$. We show that there is $i \in \{1, \ldots, t\}$ such that $I \subseteq I_i$. To obtain a contradiction, assume that there are distinct $i, j \in \{1, \ldots, t\}$ such that $I \cap I_i \neq \emptyset$ and $I \cap I_j \neq \emptyset$. Let $p \in I \cap I_i$ and $q \in I \cap I_j$. We conclude that

$$\begin{aligned}\sum_{h \in I} d_H(s', s'_h) &\geq d_H(s', s'_p) + d_H(s', s'_q) \geq d_H(s'_p, s'_q) \\ &\geq d_H(\bar{0}^{i-1}\bar{1}\bar{0}^{t-i}, \bar{0}^{j-1}\bar{1}\bar{0}^{t-j}) \geq 2(d+1) > d.\end{aligned}$$

This is a contradiction. Hence, there is $i \in \{1, \ldots, t\}$ such that $I \subseteq I_i$. Let s be a substring of s' containing the first ℓ symbols. Then, we have that

$$\sum_{h \in I} d_H(s, s_h) \leq \sum_{h \in I} d_H(s', s'_h) \leq d.$$

That is, (S_i, r, d) is a yes-instance of Consensus String with Outliers. □

Observe that every string of S^* is of length $\ell + (d+1)t$ and that $|S^*| = \sum_{i=1}^{t} |S_i|$. This means that the size of (S^*, r, d) is polynomial in the sum of the sizes of (S_i, r, d) and t. Note also that the parameter d remains the same. By the results of Bodlaender et al. (2009), we conclude that Consensus String with Outliers has no polynomial kernel when parameterized by d unless NP $\subseteq$ coNP /poly.

To see that the result holds even if $r \leq d$, we observe that for $r \geq d+1$, CONSENSUS STRING WITH OUTLIERS is solvable in polynomial time. Let (S, r, d) be a yes-instance of CONSENSUS STRING WITH OUTLIERS where $S = \{s_1, \ldots, s_n\}$ and $r \geq d+1$. Then, there is a string s and $I \subseteq \{1, \ldots, n\}$ with $|I| = r$ such that $\sum_{i \in I} d_H(s, s_i) \leq d$. Since $|I| = r \geq d+1$, there is $i \in I$ such that $s = s_i$. That is, the mean string s is one of the input strings. This brings us to the following simple algorithm. Let (S, r, d) be an instance of CONSENSUS STRING WITH OUTLIERS with $S = \{s_1, \ldots, s_n\}$ and $r \geq d+1$. For each $i \in \{1, \ldots, n\}$, we check whether the instance has a solution with $s = s_i$. We do it by the greedy selection of r strings closest to s. It is straightforward to see that this is a polynomial time algorithm solving the problem. □

We use Theorem 3 to obtain our kernelization lower bound for BINARY r-MEANS.

Theorem 4 BINARY r-MEANS *has no polynomial kernel when parameterized by* k *unless* NP $\subseteq$ coNP/poly.

Proof We reduce CONSENSUS STRING WITH OUTLIERS for strings over the binary alphabet to BINARY r-MEANS.

Let $(S, \widehat{r}, d)$ be an instance of CONSENSUS STRING WITH OUTLIERS, where $S = \{s_1, \ldots, s_n\}$ is a (multi) set of binary strings of length ℓ and $\widehat{r} \leq d$. Denote by $\bar{0}$ and $\bar{1}$ the strings of length $d+1$ composed by 0s and 1s, respectively. For $i \in \{1, \ldots, n\}$, we set $s'_i = s_i \bar{0}^{i-1} \bar{1} \bar{0}^{n-i}$. We construct the matrix $\mathbf{A}$ considering $s'_1, \ldots, s'_n$ to be vectors of $\{0,1\}^{\ell+(d+1)n}$ composing the columns of $\mathbf{A}$. Slightly abusing notation, we use $s'_1, \ldots, s'_n$ to denote the columns of $\mathbf{A}$. We set $k = (d+1)\widehat{r} + d$ and set $r' = n - \widehat{r} + 1$.

We claim that $(S, \widehat{r}, d)$ is a yes-instance of CONSENSUS STRING WITH OUTLIERS if and only if $(\mathbf{A}, r', k)$ is a yes-instance of BINARY r-MEANS.

Suppose that $(S, \widehat{r}, d)$ is a yes-instance of CONSENSUS STRING WITH OUTLIERS. Let $I \subseteq \{1, \ldots, n\}$ and a string s of length ℓ be a solution, that is, $|I| = \widehat{r}$ and $\sum_{i \in I} d_H(s, s_i) \leq d$. Denote $s' = s\bar{0}^n$. By the definition of $s'_1, \ldots, s'_n$, we have that

$$\sum_{i \in I} d_H(s', s'_i) \leq d + (d+1)\widehat{r}.$$

We construct clusters $I_1, \ldots, I_{r'}$ for $(\mathbf{A}, r', k)$ as follows. We put $I_1 = I$ and let $I_2, \ldots, I_{r'}$ be one-element disjoint subsets of $\{1, \ldots, n\} \setminus I$. Then, we define the means $\mathbf{c}^1, \ldots, \mathbf{c}^n$ as follows. We set $\mathbf{c}^1 = s'$ considering s' to be a binary vector. For every single-element cluster $I_i = \{j\}$ for $i \in \{2, \ldots, r'\}$, we set $\mathbf{c}^i = s'_j$. Note that $d_H(\mathbf{c}^i, s_j) = 0$ for $i \in \{2, \ldots, r'\}$ and $j \in I_i$. Then, we have

$$\sum_{i=1}^{r'} \sum_{j \in I_i} d_H(\mathbf{c}^i, s'_j) = \sum_{j \in I} d_H(\mathbf{c}^1, s'_j) = \sum_{j \in I} d_H(s', s'_j) \leq d + (d+1)\widehat{r} \leq k.$$

That is, $\{I_1, \ldots, I_{r'}\}$ is a solution for $(\mathbf{A}, r', k)$. This means that $(\mathbf{A}, r', k)$ is a yes-instance of BINARY r-MEANS.

Next, assume that $(\mathbf{A}, r', k)$ is a yes-instance of BINARY r-MEANS. Let $\{I_1, \ldots, I_p\}$, $p \leq r'$, be a solution. Denote by $\mathbf{c}^1, \ldots, \mathbf{c}^p$ the corresponding means obtained by the

majority rule. If $\widehat{r} = 1$, then $(S, \widehat{r}, d)$ is a trivial yes-instance of CONSENSUS STRING WITH OUTLIERS. Let $\widehat{r} \geq 2$. Then, because $r' = n - \widehat{r} + 1$, we have that there are clusters I_j with at least two elements. We assume that $|I_j| \geq 2$ for $j \in \{1, \ldots, q\}$ for some $q \leq p$ and $|I_j| = 1$ for $j \in \{q+1, \ldots, p\}$.

We show that $q = 1$. Targeting towards a contradiction, let us assume that $q \geq 2$. We have that $\sum_{i=1}^{q} |I_i| = n - (p - q)$, that is, $n - (p - q)$ columns of $\mathbf{A}$ are in clusters with at least two elements. By the construction of $\mathbf{A}$, we have that the last $n(d+1)$ elements of each mean $\mathbf{c}^i$ are 0s for $i \in \{1, \ldots, q\}$, because the means were constructed by the majority rule. This implies that $d_H(\mathbf{c}^i, s'_j) \geq d + 1$ for $j \in I_i$ and $i \in \{1, \ldots, q\}$. Clearly, $d_H(\mathbf{c}^i, s'_j) = 0$ for $j \in I_i$ and $i \in \{q+1, \ldots, p\}$ as these clusters I_i contain one element each. Then,

$$\sum_{i=1}^{p} \sum_{j \in I_i} d_H(\mathbf{c}^i, s'_j) = \sum_{i=1}^{q} \sum_{j \in I_i} d_H(\mathbf{c}^i, s'_j) \geq \sum_{i=1}^{q} |I_i|(d+1) = (n - (p-q))(d+1)$$
$$\geq (n - r' + 2)(d+1) = (\widehat{r}+1)(d+1) > k.$$

This is a contradiction to the assumption that $\{I_1, \ldots, I_p\}$ is a solution. Therefore, $q = 1$. Thus, we have that $|I_1| = n - p + 1 \geq n - r' + 1 = \widehat{r}$. Let $I \subseteq I_1$ with $|I| = \widehat{r}$. Recall that we defined the string $s'_j = s_j \bar{0}^{j-1} \bar{1} \bar{0}^{n-j}$ for $j \in \{1, \ldots, n\}$ and we consider these strings as the columns of $\mathbf{A}$. In particular, the first ℓ elements correspond to s_j and the last $n(d+1)$ elements correspond to the string $\bar{0}^{j-1} \bar{1} \bar{0}^{n-j}$. As above, we have that the last $n(d+1)$ elements of $\mathbf{c}^1$ are 0s. Denote by s the vector composed by the first ℓ elements of $\mathbf{c}^1$. We have that

$$\sum_{j \in I} d_H(\mathbf{c}^1, s'_j) \leq \sum_{j \in I_1} d_H(\mathbf{c}^1, s'_j) \leq \sum_{i=1}^{p} \sum_{j \in I_i} d_H(\mathbf{c}^i, s'_j) \leq k = (d+1)\widehat{r} + d.$$

Since

$$\sum_{j \in I} d_H(\mathbf{c}^1, s'_j) = \sum_{j \in I} (d_H(s, s_j) + (d+1)) = \sum_{j \in I} d_H(s, s_j) + (d+1)\widehat{r},$$

we conclude that $\sum_{j \in I} (d_H(s, s_j) \leq d$. This implies that $(S, \widehat{r}, d)$ is a yes-instance of CONSENSUS STRING WITH OUTLIERS.

Summarizing, we have constructed a parameterized reduction from CONSENSUS STRING WITH OUTLIERS to BINARY r-MEANS. Notice that for the new parameter k of BINARY r-MEANS, we have that $k = (d+1)\widehat{r} + d = \mathcal{O}(d^2)$ since $\widehat{r} \leq d$. This observation together with Theorem 3 implies that BINARY r-MEANS has no polynomial kernel when parameterized by k unless $\mathsf{NP} \subseteq \mathsf{coNP}/\mathrm{poly}$. □

5 Subexponential algorithms for BINARY r-MEANS

We have already seen that BINARY r-MEANS is solvable in $2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$ time (Theorem 1) In this section, we show that with respect to the combined parameterization by k and r, there is an algorithm for BINARY r-MEANS which runs in subexponential in k time. For constant rank r, the running time of this algorithm is $2^{\mathcal{O}(\sqrt{k \log k})} \cdot nm$, which outperforms the algorithms from Theorem 1. On the other hand, we are paying for the improvements on the dependency on k by making the dependency on r worse. Formally, the main result of this section is stated as follows.

Theorem 5 BINARY r-MEANS *is solvable in* $2^{\mathcal{O}(\sqrt{rk \log(k+r) \log r})} \cdot nm$ *time.*

Towards the proof of Theorem 5, we prove some auxiliary lemmas. We will be seeking for solutions of a special type.

Definition 2 Let $\mathbf{A}$ be an $m \times n$-matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_m$. We say that a vector $\mathbf{c} = (c_1, \ldots, c_m)^\intercal \in \{0,1\}^m$ *agrees* with $\mathbf{A}$ if $c_i = c_j$ whenever $\mathbf{a}_i = \mathbf{a}_j$ for $i, j \in \{1, \ldots, m\}$.

We will be using the following properties of vectors that agree with matrix $\mathbf{A}$.

Lemma 6 *Let* $(\boldsymbol{A}, r, k)$ *be a yes-instance of* BINARY r-MEANS. *Then* $(\boldsymbol{A}, r, k)$ *has a solution such that for each cluster of the solution its mean agrees with* $\boldsymbol{A}$.

Proof Let $\{I_1, \ldots, I_{r'}\}$ be a solution to $(\mathbf{A}, r, k)$. For $i \in \{1, \ldots, r'\}$, let $\mathbf{c}^i = (c^i_1, \ldots, c^i_m)^\intercal \in \{0,1\}^m$ be the mean of the cluster I_i computed by the majority rule. Then, if $\mathbf{a}_j$ and $\mathbf{a}_h$ are rows of $\mathbf{A}$ and $\mathbf{a}_j = \mathbf{a}_h$, then $c^i_j = c^i_h$ because the majority rule computes the same value. Hence, we have that $\mathbf{c}^i$ agrees with $\mathbf{A}$ for each $i \in \{1, \ldots, r'\}$. □

Lemma 7 *Let* $\boldsymbol{A}$ *be a binary* $m \times n$ *matrix with at most* t *different rows,* $\boldsymbol{a}$ *be a column of* $\boldsymbol{A}$ *and* h *be a positive integer. Then, there are at most* $\sum_{i=1}^{h} \binom{t}{i}$ *binary vectors* $\boldsymbol{b} \in \{0,1\}^m$ *agreeing with* $\boldsymbol{A}$ *that are within the Hamming distance at most* h *from* $\boldsymbol{a}$.

Proof Let $I_1, \ldots, I_t$ be the partition of rows of $\mathbf{A}$ into inclusion-maximal sets of equal rows. Clearly, vector $\mathbf{a}$ agrees with $\mathbf{A}$. Also, for every vector $\mathbf{b}$ that agrees with $\mathbf{A}$ there is $J \subseteq \{1, \ldots, t\}$, such that $\mathbf{b}$ is obtained from $\mathbf{a}$ by changing for every $i \in J$ the coordinates corresponding to all rows from I_i. But since the distance from $\mathbf{a}$ and $\mathbf{b}$ is at most h, the size of J is at most h. Hence, the number of such vectors is at most $\sum_{i=1}^{h} \binom{t}{i}$. □

Now we are ready to describe the algorithm.

Let $(\mathbf{A}, r, k)$ be an instance of BINARY r-MEANS with $A = (\mathbf{a}^1, \ldots, \mathbf{a}^n)$. First, we preprocess the instance using the kernelization algorithm from Theorem 2. If the algorithm solves the problem, we return the answer and stop. Assume that this is not the case. Then, the algorithm returns an instance of BINARY r-MEANS where the matrix has at most $k + r$ pairwise distinct columns and $\mathcal{O}(k(k + r))$ pairwise distinct rows. To simplify notations, we use the same notation $(\mathbf{A}, r, k)$ for the obtained instance and

we use m and n to denote the number of rows and columns of $\mathbf{A}$. Observe that now $n, m = (k+r)^{\mathcal{O}(1)}$. Denote by w the number of pairwise distinct rows of $\mathbf{A}$.

Informally, our algorithm does the following. For a given partial clustering of some columns of $\mathbf{A}$, budget d, and a new subset I of columns of $\mathbf{A}$ which have to be clustered, it tries to extend the partial solution by not exceeding the budget d. Some of the columns from I can go to the existing cluster and some can form new clusters. Suppose that we know the minimum distance h from vectors in new clusters to their means. Then, all vectors which are within the distance less than h to the already existing means, can be assigned to the existing clusters. Eventually, we will be left with two options. Either the number of columns to be assigned to new clusters does not exceed $\sqrt{dr \log w / \log r}$; in this case we brute-force in all possible partitions of I. Or we can bound $h \leq \sqrt{k \log r/(r \log w)}$ and invoke recursive arguments branching on the choices of a new mean that is at distance h from the unclustered columns.

Algorithm 2: EXTEND- MEANS

input : A set $I \subseteq \{1, \ldots, n\}$, a set of vectors $S \subseteq \{0,1\}^m$ of size at most r that agree with $\mathbf{A}$, and a nonnegative integer d.

output: A set of vectors $C \subseteq \{0,1\}^m$ of size at most r such that each of the vectors agrees with $\mathbf{A}$, $S \subseteq C$ and $\sum_{i \in I} \min\{d_H(\mathbf{c}, \mathbf{a}^i) \mid \mathbf{c} \in C\} \leq d$ or NO.

begin
  if $\sum_{i \in I} \min\{d_H(\boldsymbol{s}, \boldsymbol{a}^i) \mid \boldsymbol{s} \in S\} \leq d$ **then** Return $C = S$ and **quit** **if** $|S| = r$ **then** Return NO and **quit** Set $h := 0$;
  while $h \leq d$ **do**
    foreach $i \in I$ **do**
      if $\ell = \min\{d_H(\boldsymbol{s}, \boldsymbol{a}^i) \mid \boldsymbol{s} \in S\} \leq h - 1$ **then**
        Set $I := I \backslash \{i\}$ and $d := d - \ell$
      end
    end
    if $d \geq 0$ *and* $|I| \leq \sqrt{dr \log w / \log r}$ **then**
      for $p := 1$ **to** $\min\{|I|, r - |S|\}$ **do**
        foreach *partition* $\{J_0, \ldots, J_p\}$ *of* I, *where* J_0 *may be empty* **do**
          for $j := 1$ **to** p **do**
            find the optimal mean $\mathbf{s}^j$ for the cluster J_j using the majority rule
          end
        end
        Set $S := S \cup \{\mathbf{s}^1, \ldots, \mathbf{s}^p\}$;
        if $\sum_{i \in I} \min\{d_H(\boldsymbol{s}, \boldsymbol{a}^i) \mid \boldsymbol{s} \in S\} \leq d$ **then** Return $C = S$ and **quit**
      end
    end
    if $h \leq d/|I|$ **then**
      foreach *vector* $\boldsymbol{s} \in \{0,1\}^m$ *s.t.* $\boldsymbol{s}$ *agrees with* $\boldsymbol{A}$ *and* $d_H(\boldsymbol{s}, \boldsymbol{a}^i) = h$ *for some* $i \in I$ **do**
        Call EXTEND- MEANS$\{I, S \cup \{\mathbf{s}\}, d\}$;
        if *the algorithm returns a solution* C **then** return C and **quit**
      end
    end
    Set $h := h + 1$
  end
  Return NO
end

Now we give a formal description of the algorithm. Towards that we design the recursive algorithm EXTEND- MEANS (Algorithm 2). The input of EXTEND- MEANS is a set $I \subseteq \{1, \ldots, n\}$, a set of vectors $S \subseteq \{0,1\}^m$ of size at most r that agree with $\mathbf{A}$, and a nonnegative integer d. The ouput of EXTEND- MEANS is either a set of vectors $C \subseteq \{0,1\}^m$ of size at most r such that each of the vectors agrees with $\mathbf{A}$, $S \subseteq C$ and $\sum_{i \in I} \min\{d_H(\mathbf{c}, \mathbf{a}^i) \mid \mathbf{c} \in C\} \leq d$ or NO if such a set does not exist. We say that such a set C is a *solution*. Thus, we are looking for a solution extending the partial solution S for the set of column vectors indexed by I.

To solve BINARY r-MEANS, we call EXTEND- MEANS$(I = \{1, \ldots, n\}, S = \emptyset, d = k)$. Recall that w is the number of pairwise distinct rows of $\mathbf{A}$.

Now we are ready to prove Theorem 5.

Proof (of Theorem 5) We first show the correctness of Algorithm 2 and then evaluate the running time.

Correctness First of all, by its construction, if Algorithm 2 returns a set of vectors C, then each of the vectors from C agrees with $\mathbf{A}$, $|C| \leq r$, $S \subseteq C$, and $\sum_{i \in I} \min\{d_H(\mathbf{c}, \mathbf{a}^i) \mid \mathbf{c} \in C\} \leq d$, because the algorithm verifies this condition in Lines 2 and 19 before returning C.

Now we show that if there is a solution to (I, S, d) then the algorithm returns a solution. The proof is by induction on $r - |S|$. We assume that there is a solution to (I, S, d). The base case is when $|S| = r$. Then, $C = S$ is a solution and the algorithm returns C in Step 1.

Now we consider the induction step. That is $|S| < r$. By induction hypothesis, we have that for any $S' \supset S$ of size at most r, $I' \subseteq \{1, \ldots, n\}$ and a nonnegative integer d' such that each vector from S' agrees with $\mathbf{A}$, the algorithm returns a solution to the input (S', I', d') if such a solution exists.

If S is a solution, then the algorithm outputs it in Line 2 and we are done. Now we assume that S is not a solution. By assumption there is a solution to (I, S, d). We choose a solution C^* for (I, S, d) such that for each $\mathbf{c} \in C^* \setminus S$, there exists $i \in I$, for all $\mathbf{s} \in S$, $d_H(\mathbf{c}, \mathbf{a}^i) \leq d_H(\mathbf{s}, \mathbf{a}^i)$ (i.e., each vector in $C^* \setminus S$ is a mean of a non-empty cluster). Let h be the smallest integer such that there exists $i \in I$ and $\mathbf{c} \in C^* \setminus S$, for all $s \in S$, $d_H(\mathbf{s}, \mathbf{a}^i) \geq d_H(\mathbf{c}, \mathbf{a}^i) = h$. Clearly, $h \leq d$. We claim that the algorithm outputs a solution in the while loop in Lines 5–29 for this value of h unless it already produced a solution for some lesser value of h. In the later case we are done. So now we have that the loop is executed for this value of h. Let the set $J \subseteq I$ and an integer d' be constructed as follows:

- set $J = I$ and $d' = d$,
- for every $i \in J$, if $\ell = \min\{d_H(\mathbf{s}, \mathbf{a}^i) \mid \mathbf{s} \in S\} \leq h - 1$, then set $J = J \setminus \{i\}$ and $d' = d' - \ell$.

Notice that for every $j \in I \setminus J$, $\min\{d_H(\mathbf{s}, \mathbf{a}^j) \mid \mathbf{s} \in S\} \leq \min\{d_H(\mathbf{c}, \mathbf{a}^j) \mid \mathbf{c} \in C^* \setminus S\}$. Therefore, C^* is a solution for (J, S, d'). Observe that by the choice of h, and by the construction of J, we have that for every $\mathbf{c} \in C^*$ and $j \in J$, $d_H(\mathbf{c}, \mathbf{a}^j) \geq h$. Let $\mathbf{c}^* \in C^* \setminus S$ and $i^* \in J$ be such that $d_H(\mathbf{c}^*, \mathbf{a}^{i^*}) = h$. Observe that $\mathbf{c}^*$ and $i^\star$ exist by the choice of h and C^*. Note that for the set I and the integer d constructed in

Line 6–9, we have that $I = J$ and $d = d'$. Now, we have two cases according to the cardinality of J.

Case 1 $|J| \leq \sqrt{d'r \log w / \log r}$. This case is considered in Lines 11–21. Let $C^* \setminus S = \{\mathbf{c}^1, \ldots, \mathbf{c}^p\}$. We construct partition $\{J_0, \ldots, J_p\}$ of J as follows. For each $i \in J$, find $t = \min\{d_H(\mathbf{c}, \mathbf{a}^i) \mid \mathbf{c} \in C^*\}$. If $t = d_H(\mathbf{c}, \mathbf{a}^i)$ for $\mathbf{c} \in S$, then include $i \in J_0$; note that J_0 may be empty. Otherwise, find minimum $i \in \{1, \ldots, p\}$ such that $t = d_H(\mathbf{c}^i, \mathbf{a}^i)$ and include i in J_i. Assume without loss of generality that $J_1, \ldots, J_p$ are nonempty (otherwise, we can just ignore empty sets and consider a solution with less means). For each $i \in \{1, \ldots, p\}$, let $\mathbf{s}^i$ be the optimum mean for the cluster J_i constructed by the majority rule. Clearly, $\sum_{j \in J_i} d_H(\mathbf{s}^i, \mathbf{a}^j) \leq \sum_{j \in J_i} d_H(\mathbf{c}^i, \mathbf{a}^j)$. Recall also that the majority rule constructs vectors that agree with $\mathbf{A}$. This implies that $C' = S \cup \{\mathbf{s}^1, \ldots, \mathbf{s}^p\}$ is a solution. Since, we consider all $p \leq \min\{|I|, r - |S|\}$, and all partitions of I into $p + 1$ subsets in Lines 12–20, the algorithm outputs a solution C' in Line 19.

Case 2 $|J| > \sqrt{d'r \log w / \log r}$. This case is analyzed in Lines 22–27. Note that for every $\mathbf{c} \in C^*$ and $j \in J$, $d_H(\mathbf{c}, \mathbf{a}^j) \geq h$. Hence, $h \leq d'/|J|$. In Lines 23–26, for each vector $\mathbf{s} \in \{0, 1\}^m$ which agrees with $\mathbf{A}$ and which is at the Hamming distance h from some $\mathbf{a}^j$, $j \in J$, we call Extend- Means$(J, S \cup \{\mathbf{s}\}, d')$. We have that in some branch of the algorithm, we call Extend- Means$(J, S \cup \{\mathbf{c}^*\}, d')$, because $d_H(\mathbf{c}^*, \mathbf{a}^{i^*}) = h$ and $i^* \in J$. By the induction hypothesis, the algorithm outputs a solution C' for the input $(J, S \cup \{\mathbf{c}^*\}, d')$. Then, C' is a solution to (I, S, d) and Algorithm 2 returns it in Line 25.

To complete the correctness proof, note that the depth of the recursion is bounded from above by r. It follows that the algorithm perform finite number of steps and returns either a solution or the answer NO.

Running time To evaluate the running time, recall that the kernelization algorithm from Theorem 2 preforms the preprocessing in $(k + r)^{\mathcal{O}(1)} \cdot nm$ time and afterwards we have that $n, m = (k + r)^{\mathcal{O}(1)}$. Note that Lines 2–3 can be done in polynomial time. In the while loop in Lines 5–29, we consider $d + 1 \leq k + 1$ values of h and for each h, we perform Lines 6–28. The step in Lines 6–11 is done in polynomial time. Since $|I| \leq \sqrt{dr \log w / \log r}$ and $p \leq r$ in Lines 12–20, we consider $2^{\mathcal{O}(\log r \sqrt{rk \log w / \log r})} = 2^{\sqrt{rk \log w \log r}}$ partitions of I. Because $w = \mathcal{O}(k(k + r))$, these lines can be done in $2^{\mathcal{O}(\sqrt{rk \log(k(k+r)) \log r})} \cdot (nm)^{\mathcal{O}(1)}$ time. In Lines 22–27, we have that $h \leq d/|I| \leq \sqrt{d \log r / (r \log w)} \leq \sqrt{k \log r / (r \log w)}$. Recall that $\mathbf{A}$ has at most $r + k$ pairwise-distinct columns. Hence, to construct $\mathbf{s}$ in Line 23, we consider at most $r + k$ columns $\mathbf{a}^i$ for $i \in I$. Recall also that $\mathbf{A}$ has w distinct rows. Since $\mathbf{s}$ agrees with $\mathbf{A}$, by Lemma 7, we have that there are at most $2^{\mathcal{O}(\log w \sqrt{k \log r / (r \log w)})}$ vectors $\mathbf{s}$ at the Hamming distance at most h from $\mathbf{a}^i$. It follows that the steps in Lines 23–26 without recursive calls of Extend- Means can be performed in $2^{\mathcal{O}(\log w \sqrt{k \log r / (r \log w)})} \cdot (nm)^{\mathcal{O}(1)}$ time and we have $2^{\mathcal{O}(\sqrt{k \log r \log w / r})}$ recursive calls of the algorithm. The depth of the recursion is bounded from above by r. Hence, the number of leaves of the search tree is $2^{\mathcal{O}(\sqrt{rk \log r \log w})}$. Notice that the steps Lines 11–21 are performed only for the leaves of the search tree. Using the property that $w = \mathcal{O}(k(k+r))$, we have that the

running time of our recursive branching algorithm is $2^{\mathcal{O}(\sqrt{rk\log k(k+r)\log r})} \cdot (nm)^{\mathcal{O}(1)}$ or $2^{\mathcal{O}(\sqrt{rk\log(k+r)\log r})} \cdot (nm)^{\mathcal{O}(1)}$. Using the fact that $n, m = (k+r)^{\mathcal{O}(1)}$, we have that it runs in $2^{\mathcal{O}(\sqrt{rk\log(k+r)\log r})}$ time. Taking into account the preprocessing, the total running time is $2^{\mathcal{O}(\sqrt{rk\log(k+r)\log r})} + (k+r)^{\mathcal{O}(1)}nm$ and can be bounded from above by $2^{\mathcal{O}(\sqrt{rk\log(k+r)\log r})} \cdot nm$. □

6 Subexponential algorithm for Low GF(2)-Rank Approximation

Recall that it was proved by Fomin et al. (2018b) that Low GF(2)- Rank Approximation is FPT when parameterized by k and r. Throughout the section, by rank, we mean the GF(2)-rank of the concerned matrix. To demonstrate that the total dependency on $k+r$ could be relatively small, we describe the simple recursive algorithm RankAppox (Algorithm 3) that, given an $m \times n$ matrix $\mathbf{A}$, nonnegative integers r and k, either returns an $m \times n$ matrix $\mathbf{B}$ of GF(2)-rank set at most r such that $d_H(\mathbf{A}, \mathbf{B}) \leq k$ or NO if such a matrix does not exist. For $m \times n$ matrix $\mathbf{A} = (a_{ij})$ over GF(2) and a pair of integers $i \in \{1, \ldots m\}$ and $j \in \{1, \ldots, m\}$, $\mathsf{flip}_{ij}(\mathbf{A})$ denotes the matrix obtained from $\mathbf{A}$ by the replacement of a_{ij} by $1 \oplus a_{ij}$, that is, by replacing the value of a_{ij} by the opposite.

Algorithm 3: RankApprox

input : An $m \times n$ matrix $\mathbf{A}$, nonnegative integers r and k.
output: An $m \times n$ matrix $\mathbf{B}$ of GF(2)-rank set at most r such that $d_H(\mathbf{A}, \mathbf{B}) \leq k$ or NO.

```
1  begin
2    if rank(A) ≤ r then return B = A and quit if k = 0 then return NO and quit else Find
     I ⊆ {1, ..., m} and J ⊆ {1, ..., n} s.t. |I| = |J| = r + 1 and rank(A(I, J)) = r + 1;
3    foreach i ∈ I and j ∈ J do
4      Call RankApprox(flip_ij(A), r, k − 1);
5      if the algorithm outputs a matrix B then
6        return B and quit
7      end
8    end
9    Return NO
10 end
```

The algorithm gives us the following proposition.

Proposition 1 Low GF(2)- Rank Approximation *is solvable in* $2^{\mathcal{O}(k\log r)} \cdot (nm)^{\mathcal{O}(1)}$ *time.*

Proof To see the correctness of Algorithm 3, It is convenient to interpret Low GF(2)-Rank Approximation as a matrix editing problem: given a matrix $\mathbf{A}$ over GF(2), a positive integer r and a nonnegative integer k, decide whether it is possible to obtain a matrix $\mathbf{B}$ from $\mathbf{A}$ with $\mathrm{rank}(\mathbf{B}) \leq r$, by editing at most k elements, i.e., by replacing 0s by 1s and 1s by 0s. Algorithm 3 uses the following simple observation. Let $\mathbf{B}$ be an

$m \times n$-matrix of rank at most r. If $\text{rank}(\mathbf{A}[I, J]) > r + 1$ for some $I \subseteq \{1, \ldots, m\}$ and $J \subseteq \{1, \ldots, n\}$, then $d_H(\mathbf{A}[I, J], \mathbf{B}[I, J]) \geq 1$, i.e, $\mathbf{A}[I, J]$ and $\mathbf{B}[I, J]$ differ in at least one element. Respectively, if the rank of the input matrix is at least $r + 1$ and $k > 0$, then the algorithm finds an $(r + 1) \times (r + 1)$ submatrix of rank $r + 1$ and branches on its elements reducing k.

To evaluate the running time, notice that we can compute $\text{rank}(\mathbf{A})$ in polynomial time, and if $\text{rank}(\mathbf{A}) > r$, then we can find in polynomial time an $(r + 1) \times (r + 1)$-submatrix of $\mathbf{A}$ of rank $r + 1$. Then, we have $(r + 1)^2$ branches in our algorithm. Since we decrease the parameter k in every recursive call, the depth of the recurrence tree is at most k. This implies that the algorithm runs in $(r + 1)^{2k} \cdot (nm)^{\mathcal{O}(1)}$ time. □

Similarly to Theorem 5, we show that we can improve the dependency on k and obtains an algorithm for Low GF(2)- Rank Approximation that runs in subexponential in k time. Again, we are paying by making the dependency on r worse. The main result of this section is the following theorem.

Theorem 6 Low GF(2)- Rank Approximation *is solvable in* $2^{\mathcal{O}(r\sqrt{k \log(rk)})} \cdot nm$ *time.*

In particular, this gives the following tractability island for Low GF(2)- Rank Approximation.

Corollary 1 *If either r is a fixed constant and $k \in \mathcal{O}(\log^2 n / \log\log n)$ or if $r \in \mathcal{O}(\sqrt{\log n})$ and $k \in \mathcal{O}(\log n / \log\log n)$, then* Low GF(2)- Rank Approximation *can be solved in polynomial time.*

Proof If r is a constant, then the running time from Theorem 1 may be written as $2^{\mathcal{O}(\sqrt{k \log k})} \cdot nm$. Because $\log^2 n / \log\log n \leq \log^2 n$, we have that for $k \in \mathcal{O}(\log^2 n / \log\log n)$, the running time is $2^{\mathcal{O}(\log n)} \cdot n^{\mathcal{O}(1)}$, that is, it is polynomial in n. Similarly, since $\log n / \log\log n \leq \log n$, it holds that the running time is $2^{\mathcal{O}(\log n)} \cdot n^{\mathcal{O}(1)}$ whenever $r \in \mathcal{O}(\sqrt{\log n})$ and $k \in \mathcal{O}(\log n / \log\log n)$. □

The general idea of parameterized subexponential time algorithm for Low GF(2)- Rank Approximation (Theorem 6) is similar to the parameterized subexponential time algorithm for Binary r-Means. However, because we cannot use the majority rule like in the case of Binary r-Means, we have to overcome some technical difficulties. We start with some auxiliary claims and then prove Theorem 6.

In our algorithm, it is more convenient to use the alternative formulation of Low GF(2)- Rank Approximation.

Observation 3 *The task of* Low GF(2)- Rank Approximation *of binary matrix $\boldsymbol{A}$ with columns $\boldsymbol{a}^1, \ldots, \boldsymbol{a}^n$ can equivalently be stated as follows: decide whether there is a nonnegative integer $r' \leq r$ and linearly independent vectors $\boldsymbol{c}^1, \ldots, \boldsymbol{c}^{r'} \in \{0, 1\}^m$ over GF(2) such that*

$$\sum_{i=1}^{n} \min\{d_H(\boldsymbol{s}, \boldsymbol{a}^i) \mid \boldsymbol{s} = \bigoplus_{j \in I} \boldsymbol{c}^j,\ I \subseteq \{1, \ldots, r'\}\} \leq k;$$

if $r' = 0$, then the set of vectors is empty.

The vectors $\mathbf{c}^1, \ldots, \mathbf{c}^{r'}$ form a basis of the column space of the matrix of rank r' approximating $\mathbf{A}$. Throughout this section we say that a set of linearly independent vectors $C = \{\mathbf{c}^1, \ldots, \mathbf{c}^{r'}\}$ satisfying the condition

$$\sum_{i=1}^{n} \min\{d_H(\mathbf{s}, \mathbf{a}^i) \mid \mathbf{s} = \bigoplus_{j \in I} \mathbf{c}^j,\ I \subseteq \{1, \ldots, r'\}\} \leq k$$

is a *solution* for $(\mathbf{A}, r, k)$. Note that C could be empty; in this case $\mathbf{A}$ is approximated by the zero matrix.

To construct our algorithm for Binary r-Means, we first preprocessed the input instance. We follow the same strategy and use the kernelization algorithm for Low GF(2)- Rank Approximation given by Fomin et al. (2018b).

Lemma 8 (Fomin et al. 2018b) Low GF(2)- Rank Approximation *admits a kernel such that the output matrix has at most* $(r + 1)(k + 1)$ *rows and columns and the values of* r *and* k *are the same as in the input instance. Moreover, the kernelization algorithm runs in* $(rk)^{\mathcal{O}(1)} \cdot nm$ *time.*

Note that Fomin et al. (2018a) do not claim that the running time of their algorithm is $(rk)^{\mathcal{O}(1)} \cdot nm$ but this is implicit in the paper.

Let $(\mathbf{A}, r, k)$ be an instance of Low GF(2)- Rank Approximation with $\mathbf{A} = (\mathbf{a}^1, \ldots, \mathbf{a}^n)$. First, we preprocess the instance using the kernelization algorithm from Lemma 8 that returns an instance with at most $(r + 1)(k + 1)$ columns and rows. For convenience, we use the same notation for the obtained instance and the columns of the matrix as before. Note that $m, n \leq (r + 1)(k + 1)$.

As in Algorithm 2, we construct an algorithm extending a partial solution. For the simplicity of explanation, we solve the decision version of the problem. Our algorithm could be easily modified to produce a solution if it exists. Towards that we design a recursive algorithm Extend- Basis (Algorithm 4). This algorithm extends a column basis of a matrix approximating $\mathbf{A}$. The input of Extend- Basis is a set $I \subseteq \{1, \ldots, n\}$, a p-sized set of linearly independent vectors $S = \{\mathbf{s}^1, \ldots, \mathbf{s}^p\} \subseteq \{0, 1\}^m$ over GF(2), and a nonnegative integer d. The output of Extend- Basis is either YES if there is a set of linearly independent vectors $C = \{\mathbf{c}^1, \ldots, \mathbf{c}^{r'}\} \subseteq \{0, 1\}^m$ over GF(2), such that $|C| \leq r$, $S \subseteq C$ and $\sum_{i \in I} \min\{d_H(\mathbf{s}, \mathbf{a}^i) \mid \mathbf{s} = \bigoplus_{j \in J} \mathbf{c}^j,\ J \subseteq \{1, \ldots, r'\}\} \leq d$ or NO if there is no such a set. We say that such a set C is a *solution*, and call (I, S, d) an *instance* of Extend- Basis. Then, to solve Low GF(2)- Rank Approximation, we call Extend- Basis$(\{1, \ldots, n\}, \emptyset, k)$.

We denote by $\mathbf{S} = (\mathbf{s}^1, \ldots, \mathbf{s}^p)$ the $m \times p$-matrix whose columns are the vectors of S and denote by $\mathbf{s}_1, \ldots, \mathbf{s}_m$ its rows. For $p \leq r' \leq r$, we say that an r'-tuple of $(\mathbf{x}^1, \ldots, \mathbf{x}^{r'})$ with $\mathbf{x}^i \in \{0, 1\}^p$ for $i \in \{1, \ldots, r'\}$ is a zero-extended row basis of $\mathbf{S}$ if $\mathbf{x}^1, \ldots, \mathbf{x}^p$ is a row basis of $\mathbf{S}$ and $\mathbf{x}^{p+1}, \ldots, \mathbf{x}^{r'}$ are zero-vectors.

For $I \subseteq \{1, \ldots, n\}$, we define $\mathbf{A}^I = \mathbf{A}[\{1, \ldots, m\}, I]$. We denote by $\mathbf{a}^I_1, \ldots, \mathbf{a}^I_m$ the rows of $\mathbf{A}^I$. Our algorithm uses the following properties of $\mathbf{S}$ and $\mathbf{A}^I$.

Lemma 9 *Let* $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{r'})$ *be a zero-extended row basis of* $\boldsymbol{S}$*. A solution* C *for an instance* (I, S, d) *exists if and only if there is* $r' \leq r$ *and an* r'*-tuples of vectors*

$(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^{r'})$ *with* $\boldsymbol{y}^i \in \{0,1\}^{|I|}$ *for* $i \in \{1, \ldots, r'\}$ *such that*

$$\sum_{i=1}^{m} \min\{d_H(\boldsymbol{y}, (\boldsymbol{a}_i^I)^\mathsf{T}) \mid J \subseteq \{1, \ldots, r'\}, \boldsymbol{y} = \bigoplus_{j \in J} \boldsymbol{y}^j, \text{ and } \boldsymbol{s}_i^\mathsf{T} = \bigoplus_{j \in J} \boldsymbol{x}^j\} \leq d.$$

Proof For an $m \times p$-matrix $\mathbf{X}$ and an $m \times q$-matrix $\mathbf{Y}$, we denote by $(\mathbf{X}|\mathbf{Y})$ the *augmentation* of $\mathbf{X}$ by $\mathbf{Y}$, that is, the $m \times (p+q)$-matrix whose first p columns are the columns of $\mathbf{X}$ and the last q columns are the columns of $\mathbf{Y}$.

Note that we have a solution C for an instance (I, S, d) if and only if there is an $m \times |I|$-matrix $\hat{\mathbf{A}}^I$ such that $\operatorname{rank}(\mathbf{S}|\hat{\mathbf{A}}^I) = r' \leq r$ and $d_H(\mathbf{A}^I, \hat{\mathbf{A}}^I) \leq d$. This observation immediately implies the lemma. □

Now we are ready to describe EXTEND- BASIS formally (see Algorithm 4) for LOW GF(2)- RANK APPROXIMATION.

Proof (of Theorem 6) First, we prove the correctness of Algorithm 4. Then we evaluate the running time.

Correctness The construction of Algorithm 4 imply that if the algorithm returns YES, then there is a solution C for the instance (I, S, d). To see this, it is sufficient to observe that the algorithm verifies whether there is a solution in Line 2 and it verifies the conditions of Lemma 9 in Line 15 before returning YES.

Now we show the reverse direction. That is, we show that if there is a solution C for (I, S, d), then the algorithm returns YES. We assume that there is a solution C for (I, S, d). The proof is by induction on $r - |S|$. The base case is when $|S| = r$. In this case $C = S$, and algorithm returns YES in Line 2.

Now consider the induction step. That is, $|S| < r$. By the induction hypothesis we have that for every $I' \subseteq \{1, \ldots, n\}$, a set of size at most r linearly independent vectors $S' \supset S$ and a nonnegative integer d', the algorithm returns YES if a solution for the input (S', I', d') exists. If the algorithm output YES in Line 2, then we are done. Otherwise, we have that S is not a solution. Since $|S| < r$, the algorithm does not return NO in Line 3. Since S is not a solution, we have that every solution $C = \{\mathbf{c}^1, \ldots, \mathbf{c}^{r'}\} \supset S$ minimizing the sum $\sum_{i \in I} \min\{d_H(\mathbf{c}, \mathbf{a}^i) \mid \mathbf{c} = \bigoplus_{j \in J} \mathbf{c}^j,\ J \subseteq \{1, \ldots, r'\}\}$ satisfies the following property: there exists $i \in I$ such that for every linear combination $\mathbf{s} \in \{0,1\}^m$ of vectors from S we have $d_H(\mathbf{s}, \mathbf{a}^i) > h$, where $h = \min\{d_H(\mathbf{c}, \mathbf{a}^i) \mid \mathbf{c} = \bigoplus_{j \in J} \mathbf{c}^j,\ J \subseteq \{1, \ldots, r'\}\}$. We choose a solution $C = C^*$ and $i = i^* \in I$ in such a way that the value of h is minimum. Notice that there is a subset $J^* \subseteq \{1, \ldots, r'\}$ such that $\mathbf{c}^* = \bigoplus_{j \in J^*} \mathbf{c}^j$, $h = d_H(\mathbf{c}^*, \mathbf{a}^{i^*})$ and $\{\mathbf{c}^j : j \in J^*\} \backslash S \neq \emptyset$. We assume that $\mathbf{c}^* \in C^*$. Otherwise, we just add $\mathbf{c}^*$ to the solution and exclude arbitrary $\mathbf{c}^j \notin S$ with $j \in J^*$ from C^* (because $\mathbf{c}^* = \bigoplus_{j \in J^*} \mathbf{c}^j$). Let $C^* = \{\mathbf{c}^1, \ldots, \mathbf{c}^{r'}\}$. Clearly, $h \leq d$. We claim that the algorithm outputs YES in the while loop in Lines 5–24 for this value of h unless it already produced the same answer for some smaller value of h. Let the set $I' \subseteq I$ and the integer d' be constructed as follows:

- set $I' = I$ and $d' = d$,
- for every $i \in I'$, if $\ell = \min\{d_H(\mathbf{s}, \mathbf{a}^i) \mid \mathbf{s} = \bigoplus_{j \in J} \mathbf{s}^j,\ J \subseteq \{1, \ldots, p\}\} \leq h - 1$, then set $I' = I' \backslash \{i\}$ and $d' = d' - \ell$.

Algorithm 4: EXTEND- BASIS

input : A set $I \subseteq \{1, \ldots, n\}$, a p-sized set of linearly independent vectors $S = \{\mathbf{s}^1, \ldots, \mathbf{s}^p\} \subseteq \{0,1\}^m$ over GF(2), and a nonnegative integer d.
output: YES or NO.

1 **begin**
2 **if** $\sum_{i \in I} \min\{d_H(\boldsymbol{s}, \boldsymbol{a}^i) \mid \boldsymbol{s} = \bigoplus_{j \in J} \boldsymbol{s}^j,\ J \subseteq \{1, \ldots, p\}\} \leq d$ **then** Return YES and **quit if** $p = r$ **then** Return NO and **quit** Set $h := 0$;
3 **while** $h \leq d$ **do**
4 **foreach** $i \in I$ **do**
5 **if** $\ell = \min\{d_H(\boldsymbol{s}, \boldsymbol{a}^i) \mid \boldsymbol{s} = \bigoplus_{j \in J} \boldsymbol{s}^j,\ J \subseteq \{1, \ldots, p\}\} \leq h - 1$ **then**
6 Set $I := I \backslash \{i\}$ and $d := d - \ell$
7 **end**
8 **end**
9 **if** $d \geq 0$ *and* $|I| \leq \sqrt{d \log(rk)}$ **then**
10 **for** $r' := p$ **to** r **do**
11 construct a zero-extended row basis $(\mathbf{x}^1, \ldots, \mathbf{x}^{r'})$ of $\mathbf{S}$;
12 **foreach** *r'-tuple of vectors $(\mathbf{y}^1, \ldots, \mathbf{y}^{r'})$ with $\mathbf{y}^i \in \{0,1\}^{|I|}$ for $i \in \{1, \ldots, r'\}$* **do**
13 **if** $\sum_{i=1}^{m} \min\{d_H(\boldsymbol{y}, (\boldsymbol{a}_i^I)^\mathsf{T}) \mid \boldsymbol{y} = \bigoplus_{j \in J} \boldsymbol{y}^j,\ J \subseteq \{1, \ldots, r'\}$ *and* $\boldsymbol{s}_i^\mathsf{T} = \bigoplus_{j \in J} \boldsymbol{x}^j\} \leq d$ **then** Return YES and **quit**
14 **end**
15 **end**
16 **end**
17 **if** $h \leq d/|I|$ **then**
18 **foreach** *vector $\boldsymbol{s} \in \{0,1\}^m$ such that $\boldsymbol{s}$ is linearly independent with the vectors of S and $d_H(\boldsymbol{s}, \boldsymbol{a}^i) = h$ for some $i \in I$* **do**
19 Call EXTEND- BASIS$\{I, S \cup \{\mathbf{s}\}, d\}$;
20 **if** *the algorithm returns YES* **then** return YES and **quit**
21 **end**
22 **end**
23 Set $h := h + 1$
24 **end**
25 Return NO
26 **end**

Notice that for every $j \in I \backslash I'$,

$$\min\{d_H(\mathbf{s}, \mathbf{a}^j) \mid \mathbf{s} = \bigoplus_{j \in J} \mathbf{s}^j,\ J \subseteq \{1, \ldots, p\}\} \leq \min\{d_H(\mathbf{c}, \mathbf{a}^j) \mid \mathbf{c} = \bigoplus_{j \in J} \mathbf{c}^j,\ J \subseteq \{1, \ldots, r'\}\} \tag{3}$$

Because of the choice of h, we have a solution for (I', S, d') if and only if there is a solution for the original input (I, S, d). Moreover, C^* is a solution for (I', S, d'). Observe also that by the choice of h we have that $i^* \in I'$. Note that for the set I and an integer d constructed in Lines 6–10, we have that $I = I'$ and $d = d'$.

Case 1 $|I'| \leq \sqrt{d' \log(rk)}$. This case is considered in Lines 11–17. By Lemma 9, there is $p \leq r'' \leq r$ and an r''-tuple of vectors $(\mathbf{y}^1, \ldots, \mathbf{y}^{r''})$ with $\mathbf{y}^i \in \{0,1\}^{|I'|}$ for

$i \in \{1, \ldots, r''\}$ such that for a zero-extended row basis $(\mathbf{x}^1, \ldots, \mathbf{x}^{r'})$ of S it holds that

$$\sum_{i=1}^{m} \min\{d_H(\mathbf{y}, \mathbf{a}_i^\mathsf{T}) \mid \mathbf{y} = \bigoplus_{j \in J} \mathbf{y}^j,\ J \subseteq \{1, \ldots, r''\} \text{ and } \mathbf{s}_i^\mathsf{T} = \bigoplus_{j \in J} \mathbf{x}^j\} \leq d'. \quad (4)$$

Since our algorithm considers all $r'' \leq r$ in the for loop in Lines 12–16 and all possible r''-tuples of vectors $(\mathbf{y}^1, \ldots, \mathbf{y}^{r''})$ with $\mathbf{y}^i \in \{0, 1\}^{|I'|}$ for $i \in \{1, \ldots, r''\}$ in Lines 14–16 and verifies (4), we obtain that the algorithm outputs YES in Line 15.

Case 2 $|I'| > \sqrt{d' \log(rk)}$. This case is analyzed in Lines 19–23. Note that for every $j \in I'$ and $J \subseteq \{1, \ldots, r'\}$, $d_H(\mathbf{c}, \mathbf{a}^j) \geq h$ for $\mathbf{c} = \bigoplus_{i \in J} \mathbf{c}^i$. Hence, $h \leq d'/|I'|$. In the loop in Lines 20–23, we consider every vector $\mathbf{s} \in \{0, 1\}^m$ such that $\mathbf{s}$ is linearly independent with the vectors of S and $d_H(\mathbf{s}, \mathbf{a}^j) = h$ for some $j \in I'$, and call EXTEND- BASIS$(I', S \cup \{s\}, d')$. We have that in some recursive call of the algorithm, we call EXTEND- BASIS$(J, S \cup \{\mathbf{c}^*\}, d')$, because $d_H(\mathbf{c}^*, \mathbf{a}^{i^*}) = h$ and $i^* \in I'$. By the induction hypothesis, the algorithm outputs YES for the input $(I', S \cup \{\mathbf{c}^*\}, d')$. Then we output YES for (I, S, d) in Line 22.

To complete the correctness proof, note that the depth of the recursion is at most r. It follows that the algorithm performs a finite number of steps and correctly reports that (I, S, d) is a yes-instance or a no-instance.

Running time To evaluate the running time, recall that the algorithm from Lemma 8 runs in $(rk)^{\mathcal{O}(1)} \cdot nm$ time and afterwards we have $n, m \leq (r+1)(k+1)$. Now we evaluate the running time of Algorithm 4.

Observe that the steps in Lines 2 and 3 can be done in polynomial time.

In the while loop in Lines 5–26, we consider $d + 1 \leq k + 1$ values of h and for each h, perform steps in Lines 6–24. Clearly, the steps in Lines 6–10 can be done in polynomial time.

Consider Lines 11–18. Clearly, a zero-extended row basis $(\mathbf{x}^1, \ldots, \mathbf{x}^{r'})$ of S can be found in polynomial time. Since $r' \leq r$, $|I| \leq \sqrt{d \log(rk)} \leq \sqrt{k \log(rk)}$ and $p \leq r$, we consider $2^{\mathcal{O}(r\sqrt{k \log(rk)})}$ r'-tuples of vectors $(\mathbf{y}^1, \ldots, \mathbf{y}^{r'})$. It implies that the steps in Lines 11–18 take $2^{\mathcal{O}(r\sqrt{k \log(rk)})} \cdot (nm)^{\mathcal{O}(1)}$ time.

In Lines 19–24, we have that $h \leq d/|I| \leq d/\sqrt{d \log(rk)} \leq \sqrt{k/\log(rk)}$. Recall that $\mathbf{A}$ has at most $(r+1)(k+1)$ columns. Hence, to construct $\mathbf{s}$ is Line 20, we consider at most $(r+1)(k+1)$ columns $\mathbf{a}^i$ for $i \in I$. Recall also that $\mathbf{A}$ has at most $(r+1)(k+1)$ rows. Then there are at most $(r+1)(k+1) \cdot ((r+1)(k+1))^{\sqrt{k/\log(rk)}}$ vectors $\mathbf{s}$ at distance h from $\mathbf{a}^i$. It follows that the steps in Lines 19-24 without recursive calls of EXTEND- BASIS can be done in $2^{\mathcal{O}(r\sqrt{k \log(rk)})} \cdot (nm)^{\mathcal{O}(1)}$ time and we have $2^{\mathcal{O}(r\sqrt{k \log(rk)})}$ recursive calls of the algorithm.

The depth of the recursion is at most r. Note that Lines 12–18 are executed only for the leaves of the search tree. Therefore, the total running time of the branching algorithm is $2^{\mathcal{O}(r\sqrt{k \log(rk)})} \cdot (nm)^{\mathcal{O}(1)}$. Since $n, m \leq (r+1)(k+1)$, we have that its running time can be bounded from above by $2^{\mathcal{O}(r\sqrt{k \log(rk)})}$. Taking into account the preprocessing, the total running time is $2^{\mathcal{O}(r\sqrt{k \log(rk)})} \cdot nm$. □

7 Subexponential algorithms for P-MATRIX APPROXIMATION and LOW BOOLEAN-RANK APPROXIMATION

In this section we give a parameterized subexponential algorithm for LOW BOOLEAN-RANK APPROXIMATION. This algorithm is more complicated than the one for LOW GF(2)- RANK APPROXIMATION. This is because $(\{0,1\}, \vee, \wedge)$ do not form a field, and thus many nice properties of matrix-rank cannot be used here. The way we handle this issue is to solve the **P**-MATRIX APPROXIMATION problem. As soon as we obtain a subexponential algorithm for **P**-MATRIX APPROXIMATION, a simple reduction will provide an algorithm for LOW BOOLEAN- RANK APPROXIMATION.

We will be using the following observation which follows directly from the definition of a **P**-matrix.

Observation 4 *Let $\boldsymbol{P}$ be a binary $p \times q$ matrix. Then every $\boldsymbol{P}$-matrix $\boldsymbol{B}$ has at most p pairwise distinct rows and at most q pairwise distinct columns.*

In our algorithm for **P**-MATRIX APPROXIMATION, we need a subroutine for checking whether a matrix $\mathbf{A}$ is a **P**-matrix. For that we employ the following brute-force algorithm. Let $\mathbf{A}$ be an $m \times n$-matrix. Let $\mathbf{a}_1, \ldots, \mathbf{a}_m$ be the rows of $\mathbf{A}$, and let $\mathbf{a}^1, \ldots, \mathbf{a}^n$ be the columns of $\mathbf{A}$. Let $\mathcal{I} = \{I_1, \ldots, I_s\}$ be the partition of $\{1, \ldots, m\}$ into inclusion-maximal sets of indices such that for every $i \in \{1, \ldots, s\}$ the rows $\mathbf{a}_j$ for $j \in I_i$ are equal. Similarly, let $\mathcal{J} = \{J_1, \ldots, J_t\}$ be the partition of $\{1, \ldots, n\}$ into inclusion-maximal sets such that for every $i \in \{1, \ldots, t\}$, the columns $\mathbf{a}^j$ for $j \in I_i$ are equal. We say that $(\mathcal{I}, \mathcal{J})$ is the *block partition* of $\mathbf{A}$.

Observation 5 *There is an algorithm which given an $m \times n$-matrix $\boldsymbol{A} = (a_{ij}) \in \{0,1\}^{m \times n}$ and a $p \times q$-matrix $\boldsymbol{P} = (p_{ij}) \in \{0,1\}^{p \times q}$, runs in $2^{\mathcal{O}(p \log p + q \log q)} \cdot (nm)^{\mathcal{O}(1)}$ time, and decides whether $\boldsymbol{A}$ is a $\boldsymbol{P}$-matrix or not.*

Proof Let $(\mathcal{I} = \{I_1, \ldots, I_s\}, \mathcal{J} = \{J_1, \ldots, J_t\})$ be the block partition of $\mathbf{A}$ and let $(\mathcal{X} = \{X_1, \ldots, X_{p'}\}, \mathcal{Y} = \{Y_1, \ldots, Y_{q'}\})$ be the block partition of $\mathbf{P}$. Observe that $\mathbf{A}$ is a **P**-matrix if and only if $s = p'$, $t = q'$, and there are permutations α and β of $\{1, \ldots, p'\}$ and $\{1, \ldots, q'\}$, respectively, such that the following holds for every $i \in \{1, \ldots, p'\}$ and $j \in \{1, \ldots, q'\}$:

(i) $|I_i| \geq |X_{\alpha(i)}|$ and $|J_j| \geq |Y_{\beta(j)}|$,
(ii) $a_{i'j'} = p_{i''j''}$ for $i' \in I_i$, $j \in J_j$, $i'' \in X_{\alpha(i)}$ and $j'' \in Y_{\beta(i)}$.

Thus, in order to check whether $\mathbf{A}$ is a **P**-matrix, we check whether $s = p'$ and $t = q'$, and if it holds, we consider all possible permutations α and β, and verify (i) and (ii). Note that the block partitions of $\mathbf{A}$ and $\mathbf{P}$ can be constructed in polynomial time. Since there are $p'! \in 2^{\mathcal{O}(p \log p)}$ and $q'! \in 2^{\mathcal{O}(q \log q)}$ permutations of $\{1, \ldots, p'\}$ and $\{1, \ldots, q'\}$, respectively, and (i)–(ii) can be verified in polynomial time, we obtain that the algorithm runs in $2^{\mathcal{O}(p \log p + q \log q)} \cdot (nm)^{\mathcal{O}(1)}$ time. □

Let $(\mathbf{A}, \mathbf{P}, k)$ be an instance of **P**-MATRIX APPROXIMATION. We say that a matrix $\mathbf{B}$ is a *solution* for $(\mathbf{A}, \mathbf{P}, k)$ if $d_H(\mathbf{A}, \mathbf{B}) \leq k$, and $\mathbf{B}$ is a **P**-matrix. Next, we observe that **P**-MATRIX APPROXIMATION admits a polynomial kernel when parameterized by the size of the pattern matrix and k.

Lemma 10 *Let $\boldsymbol{P}$ be a $p \times q$ binary matrix. Then $\boldsymbol{P}$-*MATRIX APPROXIMATION *admits a kernel such that the output matrix has at most $(\max\{k, p\} + 1)(p + k)$ rows, and at most $(\max\{k, q\} + 1)(q + k)$ columns. Moreover, the kernelization algorithm runs in $\mathcal{O}((p + q + k)nm)$ time, and either solves the problem or outputs an equivalent instance of $\boldsymbol{P}$-*MATRIX APPROXIMATION *such that the matrix in this instance has at least p rows, at least q columns, at most $p + k$ pairwise distinct rows and at most $q + k$ distinct columns.*

Proof Let $(\mathbf{A}, \mathbf{P}, k)$ be an instance of $\mathbf{P}$-MATRIX APPROXIMATION. We apply the following reduction rules. □

Reduction Rule 1 *If $\boldsymbol{A}$ has at most $p - 1$ rows or at most $q - 1$ columns or at least $p + k + 1$ pairwise distinct rows or at least $q + k + 1$ pairwise distinct columns, then return a trivial no-instance (say, $((1), (0), 0)$) and stop.*

Clearly, if $\mathbf{A}$ has at most $p-1$ rows or at most $q-1$ columns, then $(\mathbf{A}, \mathbf{P}, k)$ is a no-instance. Assume that this is not the case, and suppose that $(\mathbf{A}, \mathbf{P}, k)$ is a yes-instance. Then, there is a solution $\mathbf{B}$, that is, an $m \times n$ binary $\mathbf{P}$-matrix with $d_H(\mathbf{A}, \mathbf{B}) \leq k$. By Observation 4, $\mathbf{B}$ has at most p pairwise distinct rows and at most q pairwise distinct columns. Since $d_H(\mathbf{A}, \mathbf{B}) \leq k$, we conclude that $\mathbf{A}$ has at most $p + k$ pairwise distinct rows and at most $q + k$ pairwise distinct columns. Hence, if $\mathbf{A}$ has at least $p + k + 1$ pairwise distinct rows or at least $q + k + 1$ pairwise distinct columns, then $(\mathbf{A}, \mathbf{P}, k)$ is a no-instance.

Assume that we did not stop by Reduction Rule 1. Then, $\mathbf{A}$ has at most $p + k$ pairwise distinct rows and at most $q + k$ pairwise distinct columns.

Reduction Rule 2 *If $\boldsymbol{A}$ has at least $\max\{p, k\} + 2$ identical rows, then delete one of these rows that is chosen arbitrarily.*

Claim 7 *Reduction Rule 2 is safe.*

Proof (of Claim 7) Let $\mathbf{a}_1, \ldots, \mathbf{a}_m$ be the rows of $\mathbf{A}$ and assume that the rows $\mathbf{a}_i$ for $i \in I \subseteq \{1, \ldots, m\}$ are the same and it holds that $|I| \geq \max\{p, k\} + 2$. Let $\mathbf{A}'$ be the matrix obtained from $\mathbf{A}$ by the deletion of $\mathbf{a}_h$ for some $h \in I$.

Suppose that $(\mathbf{A}, \mathbf{P}, k)$ is a yes-instance of $\mathbf{P}$-MATRIX APPROXIMATION. Then, there is a $\mathbf{P}$-matrix $\mathbf{B} = (b_{st})$ with $d_H(\mathbf{A}, \mathbf{B}) \leq k$. Since $\mathbf{B}$ is a $\mathbf{P}$-matrix, there a partition $\{I_1, \ldots, I_p\}$ of $\{1, \ldots, m\}$, and a partition $\{J_1, \ldots, J_q\}$ of $\{1, \ldots, n\}$ such that for every $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, q\}$, $s \in I_i$, and $t \in J_j$, $b_{st} = p_{ij}$. Since $|I| \geq p + 2$, there is $i \in \{1, \ldots, p\}$ such that $|I_i| \geq 2$ and $I_i \cap I \neq \emptyset$. Let $h' \in I_i \cap I$, and denote by $\mathbf{A}''$ and $\mathbf{B}''$ the matrices obtained from $\mathbf{A}$ and $\mathbf{B}$, respectively, by the deletion of h'-th row. Since $|I_i| \geq 2$, we have that $\mathbf{B}''$ is a $\mathbf{P}$-matrix. Clearly, $d_H(\mathbf{A}'', \mathbf{B}'') \leq d_H(\mathbf{A}, \mathbf{B}) \leq k$. Because $\mathbf{A}'$ and $\mathbf{A}''$ are the same up to a permutation of rows, we have that $(\mathbf{A}', \mathbf{P}, k)$ is a yes-instance.

Assume that $(\mathbf{A}', \mathbf{P}, k)$ is a yes-instance of $\mathbf{P}$-MATRIX APPROXIMATION. We have that there is a $\mathbf{P}$-matrix $B' = (b_{st})$ with $d_H(\mathbf{A}', \mathbf{B}') \leq k$. We assume that the rows of $\mathbf{A}'$ and $\mathbf{B}'$ are indexed by the elements of $\{1, \ldots, m\} \setminus \{h\}$. Since $\mathbf{B}'$ is a $\mathbf{P}$-matrix, there a partition $\{I_1, \ldots, I_p\}$ of $\{1, \ldots, m\} \setminus \{h\}$, and a partition $\{J_1, \ldots, J_q\}$ of $\{1, \ldots, n\}$ such that for every $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, q\}$, $s \in I_i$ and $t \in J_j$, $b_{st} = p_{ij}$.

Because $|I\backslash\{h\}| \geq k+1$ and $d_H(\mathbf{A}', \mathbf{B}') \leq k$, there is $i \in \{1, \ldots, p\}$ such that $I \cap I_i \neq \emptyset$, and there is $h' \in I \cap I_i$ with h'-th row of $\mathbf{B}'$ is the same as $\mathbf{a}_{h'}$. Let $\mathbf{B}$ be the matrix obtained from $\mathbf{B}'$ by inserting $\mathbf{a}_h$ as the h-th row. Observe that $d_H(\mathbf{A}, \mathbf{B}) = d_H(\mathbf{A}', \mathbf{B}') \leq k$. We also have that $\mathbf{B}$ is a $\mathbf{P}$-matrix. To see this, it is sufficient to modify the partition $\{I_1, \ldots, I_p\}$ by replacing I_i by $I_i \cup \{h\}$. We conclude that $(\mathbf{A}, \mathbf{P}, k)$ is a yes-instance of $\mathbf{P}$-Matrix Approximation. This completes the safeness proof.

□

The safeness of the next rule is proved by the same arguments using the symmetry between rows and columns.

Reduction Rule 3 *If $\mathbf{A}$ has at least $\max\{q, k\}+2$ identical columns, then delete one of these columns that is chosen arbitrarily.*

We apply Reduction Rules 2 and 3 exhaustively. Denote by $(\mathbf{A}', \mathbf{P}, k)$ the obtained instance of $\mathbf{P}$-Matrix Approximation. Because $\mathbf{A}'$ has at most $p+k$ pairwise distinct rows and at most $q+k$ pairwise distinct columns, and Reduction Rules 2 and 3 cannot be applied, we have that $\mathbf{A}'$ has at most $(\max\{k, p\}+1)(p+k)$ rows and at most at most $(\max\{k, q\}+1)(q+k)$ columns. Notice that the values of the elements of $\mathbf{P}$ were never used in the reduction rules.

The running time in the same way as in the proofs of Theorem 2 and Lemma 8: we list pairwise distinct columns and rows in $\mathcal{O}((p+q+k)nm)$ time and immediately stop if we obtained at least $p+k+1$ distinct rows or at least $q+k+1$ distinct row. In parallel, we count the number of columns and rows that are same. □

Observe that Lemma 10 immediately implies the following straightforward algorithm for $\mathbf{P}$-Matrix Approximation. First, we apply the kernelization algorithm from Lemma 10. We obtain an instance of $\mathbf{P}$-Matrix Approximation such that the input matrix $\mathbf{A}$ has at most $(\max\{k, p\}+1)(\max\{k, q\}+1)(p+k)(q+k)$ elements. We verify whether $\mathbf{A}$ is a $\mathbf{P}$-matrix using Observation 5. If it holds, then we return YES as an answer. Otherwise, if $k \geq 1$, we branch on all possible edits of one element of $\mathbf{A}$ and reduce the parameter. This leads to a trivial branching algorithm, and gives the following proposition.

Proposition 2 *For a $(p \times q)$ binary matrix $\mathbf{P}$, $\mathbf{P}$-Matrix Approximation can be solved in $2^{\mathcal{O}(k \log[(k+p)(k+q)]+p \log p+q \log q)} \cdot nm$ time.*

Similarly to the algorithms for Binary r-Means and Low GF(2)- Rank Approximation in Theorems 5 and 6, respectively, we construct a recursive branching algorithm for $\mathbf{P}$-Matrix Approximation that is subexponential in k. Recall that in the algorithms for Binary r-Means and Low GF(2)- Rank Approximation, we solved auxiliary problems. In these auxiliary problems one has to extend a partial solution while reducing the set of "undecided" columns. In particular, if the set of these undecided columns is sufficiently small, we use brute force algorithms to solve the problems. Here, we use a similar strategy, but the auxiliary extension problem is slightly more complicated. Following is the auxiliary extension problem.

EXTENDABLE **P**- MATRIX APPROXIMATION

Input: An $m \times n$ binary matrix $\mathbf{A}$, a pattern $p \times q$-matrix $\mathbf{P}$, a partition $\{X, Y, Z\}$ of $\{1, \ldots, n\}$, where some sets could be empty, such that $|X| + |Y| = q$, and a nonnegative integer k.

Task: Decide whether there is an $m \times n$-matrix $\mathbf{B}$ such that (i) $\mathbf{B}[\{1, \ldots, m\}, X] = \mathbf{A}[\{1, \ldots, m\}, X]$, (ii) $\mathbf{B}[\{1, \ldots, m\}, X \cup Y]$ is a $\mathbf{P}$-matrix and each column of $\mathbf{B}$ coincides with a column of $\mathbf{B}[\{1, \ldots, m\}, X \cup Y]$, and (iii) $d_H(\mathbf{A}, \mathbf{B}) \leq k$.

We call a matrix $\mathbf{B}$ satisfying (i)–(iii) a *solution* for EXTENDABLE **P**-MATRIX APPROXIMATION. Next, we prove that **P**-MATRIX APPROXIMATION is a special case of EXTENDABLE **P**-MATRIX APPROXIMATION.

Lemma 11 *There is an algorithm that, given an instance $(\boldsymbol{A}, \boldsymbol{P}, k)$ of* **P**-MATRIX APPROXIMATION *such that input matrix $\boldsymbol{A}$ has at most ℓ pairwise distinct columns, in $\ell^q(nm)^{\mathcal{O}(1)}$ time outputs a collection $\mathcal{I} = \{(\boldsymbol{A}, \boldsymbol{P}, \{X_i = \emptyset, Y_i, Z_i\}, k) \colon 1 \leq i \leq t\}$ of $t \leq \ell^q$ instances of* EXTENDABLE **P**-MATRIX APPROXIMATION *such that $\boldsymbol{B}$ is a solution of $(\boldsymbol{A}, \boldsymbol{P}, k)$ if and only if $\boldsymbol{B}$ is a solution of at least one instance in $\mathcal{I}$.*

Proof Let $\mathbf{B} = (\mathbf{b}^1, \ldots, \mathbf{b}^n)$ be a solution to the instance $(\mathbf{A}, \mathbf{P}, k)$ of **P**-MATRIX APPROXIMATION. We say that a set $J \subseteq \{1, \ldots, n\}$ *represents* $\mathbf{P}$ with respect to $\mathbf{B}$ if (a) $|J| = q$, (b) $\mathbf{B}[\{1, \ldots, m\}, J]$ is a $\mathbf{P}$-matrix, (c) for each $j \in \{1, \ldots, n\} \setminus J$, $\mathbf{b}^j = \mathbf{b}^i$ for some $i \in J$. Observe that for every solution $\mathbf{B}$, there is $J \subseteq \{1, \ldots, n\}$ that represents $\mathbf{P}$ with respect to $\mathbf{B}$.

We say that two sets of indices $J, J' \subseteq \{1, \ldots, n\}$ are *equivalent* with respect to $\mathbf{A}$ if the matrices $\mathbf{A}[\{1, \ldots, m\}, J]$ and $\mathbf{A}[\{1, \ldots, m\}, J']$ are the same up to a column permutation. Observe that if J, J' are equivalent with respect to $\mathbf{A}$, then $(\mathbf{A}, \mathbf{P}, k)$ has a solution $\mathbf{B}$ such that J represents $\mathbf{P}$ with respect to $\mathbf{B}$ if and only if the same instance has a solution $\mathbf{B}'$ such that J' represents $\mathbf{P}$ with respect to $\mathbf{B}'$.

Because $\mathbf{A}$ has at most ℓ pairwise distinct columns, there are at most ℓ^q sets of indices $J \in \{1, \ldots, n\}$ of size q that are pairwise nonequivalent with respect to $\mathbf{A}$. We consider such sets, and for each such set J_i, we construct the instance $(\mathbf{A}, \mathbf{P}, \{X_i, Y_i, Z_i\}, k)$ of EXTENDABLE **P**-MATRIX APPROXIMATION with $X_i = \emptyset$, $Y_i = J$, and $Z_i = \{1, \ldots, n\} \setminus J_i$. Let $\mathcal{I}$ be the set of constructed instances. Clearly, $\mathcal{I}$ can be constructed in $\ell^q(nm)^{\mathcal{O}(1)}$ time.

If $\mathbf{B}$ is a solution for the instance $(\mathbf{A}, \mathbf{P}, k)$ of **P**-MATRIX APPROXIMATION with J_i representing $\mathbf{P}$ with respect to $\mathbf{B}$, then $\mathbf{B}$ is a solution to $(\mathbf{A}, \mathbf{P}, \{X_i, Y_i, Z_i\}, k)$ of EXTENDABLE **P**-MATRIX APPROXIMATION, that is, $\mathcal{I}$ contains a yes-instance of EXTENDABLE **P**-MATRIX APPROXIMATION. For the opposite direction, observe that if $\mathbf{B}$ is a solution of some $(\mathbf{A}, \mathbf{P}, \{X_i, Y_i, Z_i\}, k) \in \mathcal{I}$, then $\mathbf{B}$ is a solution for the instance $(\mathbf{A}, \mathbf{P}, k)$ of **P**-MATRIX APPROXIMATION. This completes the proof of the lemma. □

The following lemma will be used in the algorithm for **P**-MATRIX APPROXIMATION when the sum $|Y| + |Z|$ is sufficiently small.

Lemma 12 EXTENDABLE ***P***-MATRIX APPROXIMATION *can be solved in* $2^{\mathcal{O}(p(r+\log k)+p\log p+q\log q)}\cdot(nm)^{\mathcal{O}(1)}$ *time, where* $r=|Y|+|Z|$, *if the input matrix* $\mathbf{A}$ *has at least* p *rows, at least* q *columns, at most* $p+k$ *pairwise-distinct rows and at most* $q+k$ *pairwise-distinct columns.*

Proof Let $(\mathbf{A},\mathbf{P},\{X,Y,Z\},k)$ be an instance of EXTENDABLE $\mathbf{P}$-MATRIX APPROXIMATION. Let $\mathbf{a}^1,\ldots,\mathbf{a}^n$ be the columns and $\mathbf{a}_1,\ldots,\mathbf{a}_m$ be the rows of matrix $\mathbf{A}=(a_{ij})\in\{0,1\}^{m\times n}$.

Suppose that matrix $\mathbf{B}$ with the rows $\mathbf{b}_1,\ldots,\mathbf{b}_m$ is a solution to the instance $(\mathbf{A},\mathbf{P},\{X,Y,Z\},k)$. We say that a set $I\subseteq\{1,\ldots,m\}$ *represents* $\mathbf{P}$ with respect to $\mathbf{B}$ if (a) $|I|=p$, (b) $\mathbf{B}[I,X\cup Y]$ is a $\mathbf{P}$-matrix and each column of $\mathbf{B}[I,\{1,\ldots,n\}]$ coincides with a column of $\mathbf{B}[I,X\cup Y]$, and (c) for every $i\in\{1,\ldots,m\}\setminus I$, there is $j\in I$ such that $\mathbf{b}_i=\mathbf{b}_j$. Clearly, for every solution $\mathbf{B}$, there is $I\subseteq\{1,\ldots,m\}$ that represents $\mathbf{P}$ with respect to $\mathbf{B}$.

We say that two sets of indices $I,I'\subseteq\{1,\ldots,m\}$ are *equivalent* with respect to $\mathbf{A}$ if the matrices $\mathbf{A}[I,\{1,\ldots,n\}]$ and $\mathbf{A}[I',\{1,\ldots,n\}]$ are the same up to a permutation of rows. Observe that if I,I' are equivalent with respect to $\mathbf{A}$, then $(\mathbf{A},\mathbf{P},\{X,Y,Z\},k)$ has a solution $\mathbf{B}$ such that I represents $\mathbf{P}$ with respect to $\mathbf{B}$ if and only if the same instance has a solution $\mathbf{B}'$ such that I' represents $\mathbf{P}$ with respect to $\mathbf{B}'$.

Since $\mathbf{A}$ has at most $p+k$ pairwise-distinct rows, there are at most $(p+k)^p$ pairwise nonequivalent with respect to $\mathbf{A}$ sets of indices $I\subseteq\{1,\ldots,m\}$ of size p. We consider such sets, and for each I, we check whether there is a solution $\mathbf{B}$ for $(\mathbf{A},\mathbf{P},\{X,Y,Z\},k)$ with I representing $\mathbf{P}$ with respect to $\mathbf{B}$. If we find that there is a solution, we return YES, and we return NO if there is no solution for every choice of I. Now on, we assume that I is given.

Our aim now is to check the existence of a solution $\mathbf{B}=(b_{ij})\in\{0,1\}^{m\times n}$ with I representing $\mathbf{P}$ with respect to $\mathbf{B}$. We denote by $\mathbf{b}_1,\ldots,\mathbf{b}_m$ the rows of $\mathbf{B}$. We consider all possible matrices $\mathbf{B}[I,\{1,\ldots,n\}]$. Recall that for each $i\in\{1,\ldots,m\}$, we should have that $b_{ij}=a_{ij}$ for $j\in X$. It implies that there are at most $2^{|Y|+|Z|}$ possibilities to construct $\mathbf{b}_i$ for $i\in I$, and there are at most $2^{(|Y|+|Z|)p}$ possible matrices $\mathbf{B}[I,\{1,\ldots,n\}]$. Since the matrix should satisfy the condition (b), we use Observation 5 to check this condition in $2^{\mathcal{O}(p\log p+q\log q)}\cdot(nm)^{\mathcal{O}(1)}$ time. If it is violated, we discard the current choice of $\mathbf{B}[I,\{1,\ldots,n\}]$. Otherwise, we check whether $\mathbf{B}[I,\{1,\ldots,n\}]$ can be extended to a solution $\mathbf{B}$ satisfying the condition (c). For $i,j\in\{1,\ldots,m\}$, we define

$$d_H^*(\mathbf{b}_i,\mathbf{a}_j)=\begin{cases}d_H(\mathbf{b}_i,\mathbf{a}_j) & \text{if } b_{ih}=a_{jh} \text{ for } h\in X,\\ +\infty & \text{otherwise.}\end{cases}$$

We observe that $\mathbf{B}[I,\{1,\ldots,n\}]$ can be extended to a solution $\mathbf{B}$ satisfying the condition (c) if and only if

$$\sum_{i\in I}d_H(\mathbf{b}_i,\mathbf{a}_i)+\sum_{j\in\{1,\ldots,m\}\setminus I}\min\{d_H^*(\mathbf{b}_i,\mathbf{a}_j)\mid i\in I\}\leq k.$$

We verify the condition, and return YES. Otherwise we discard the current choice of $\mathbf{B}[I, \{1, \ldots, n\}]$. If we fail for all choices of $\mathbf{B}[I, \{1, \ldots, n\}]$, we return NO and stop.

It remains to evaluate the running time. We can check in polynomial time whether $\mathbf{A}$ has at most $p+k$ rows and at most $q+k$ columns. Then, we construct at most $(p+k)^p$ pairwise nonequivalent (with respect to $\mathbf{A}$) sets of indices $I \in \{1, \ldots, m\}$ of size p, and for each I, we consider at most $2^{(|Y|+|Z|)p}$ possible matrices $\mathbf{B}[I, \{1, \ldots, n\}]$. Then, for each choice of matrix $\mathbf{B}[I, \{1, \ldots, n\}]$, we first check in $2^{\mathcal{O}(p \log p + q \log q)} \cdot (nm)^{\mathcal{O}(1)}$ time whether this matrix satisfies the condition (b), and then check in polynomial time whether $\mathbf{B}[I, \{1, \ldots, n\}]$ can be extended to a solution. We obtain that the total running time is $2^{\mathcal{O}(p(\log k + |Y| + |Z|) + p \log p + q \log q)} \cdot (nm)^{\mathcal{O}(1)}$. □

Now we are ready to prove the main technical result of this section.

Theorem 7 ***P***-MATRIX APPROXIMATION *is solvable in time*

$$2^{\mathcal{O}((p+q)\sqrt{k \log(p+k)} + p \log p + q \log q + q \log(q+k))} \cdot nm.$$

Proof Let $(\mathbf{A}, \mathbf{P}, k)$ be an instance of **P**-MATRIX APPROXIMATION, where $\mathbf{P}$ is a $p \times q$-matrix. Let $\mathbf{a}_1, \ldots, \mathbf{a}_m$ and $\mathbf{a}^1, \ldots, \mathbf{a}^n$ denote the rows and columns of $\mathbf{A}$, respectively.

First, we preprocess the input instance using the kernelization algorithm from Lemma 10. If the algorithm solves the problem, we return the answer and stop. Assume that this is not the case and the algorithm returns an equivalent instance. For simplicity, we use the same notation for it. Recall that $\mathbf{A}$ has at most $p+k$ pairwise-distinct rows, at most $q+k$ pairwise-distinct columns and $p \leq m \leq (\max\{k, p\} + 1)(p+k)$ and $q \leq n \leq (\max\{k, q\} + 1)(q+k)$.

Next, we apply Lemma 11 and we get a collection

$$\mathcal{I} = \{(\mathbf{A}, \mathbf{P}, \{X_i = \emptyset, Y_i, Z_i\}, k) \colon 1 \leq i \leq t\}$$

of $t \leq (q+k)^q$ instances of EXTENDABLE **P**-MATRIX APPROXIMATION such that $\mathbf{B}$ is a solution of $(\mathbf{A}, \mathbf{P}, k)$ if and only if $\mathbf{B}$ is a solution of at least one instance in $\mathcal{I}$. Therefore, it is enough to solve EXTENDABLE **P**-MATRIX APPROXIMATION for each instance $\mathcal{I}$.

In what follows, we design a recursive branching algorithm for EXTENDABLE **P**-MATRIX APPROXIMATION called EXTEND-**P**-SOLUTION (Algorithm 5). The algorithm takes as an input a matrix $\mathbf{A}$, disjoint sets of indices $X, Y, Z \subseteq \{1, \ldots, n\}$ such that $|X| + |Y| = q$, and a nonnegative integer k and outputs either YES if $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$ is a yes-instance of EXTENDABLE **P**-MATRIX APPROXIMATION or NO otherwise.

Correctness To prove correctness, first we show that if Algorithm 5 returns YES, then $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$ is a yes-instance of EXTENDABLE **P**-MATRIX APPROXIMATION.

Suppose that $Y = \emptyset$. This case is considered in Lines 2–8. If the algorithms returns YES, then it does not stop in Lines 3–5. Hence, $\mathbf{A}[\{1, \ldots, m\}, X]$ is a **P**-matrix. If $Z = \emptyset$, we have that $\mathbf{B} = \mathbf{A}[\{1, \ldots, m\}, X] = \mathbf{A}$ is a solution for the instance $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$ of EXTENDABLE **P**-MATRIX APPROXIMATION, and the algorithm correctly returns YES in Line 6, since the condition in this line trivially holds. Let

Algorithm 5: EXTEND- P- SOLUTION

input : A matrix $\mathbf{A}$, disjoint sets $X, Y, Z \subseteq \{1, \ldots, n\}$ s.t. $|X| + |Y| = q$, and $k \geq 0$.
output: YES or NO.

1 **begin**
2 **if** $Y = \emptyset$ **then**
3 **if** $\boldsymbol{A}[\{1, \ldots, m\}, X]$ *is not a* $\boldsymbol{P}$*-matrix* **then**
4 Return NO and **quit**
5 **end**
6 **if** $\sum_{i \in Z} \min\{d_H(\boldsymbol{a}^i, \boldsymbol{a}^j) \mid j \in X\} \leq k$ **then** Return YES and **quit else** Return NO and **quit**
7 **end**
8 Set $h := 0$;
9 **while** $h \leq k$ **do**
10 **foreach** $i \in Z$ **do**
11 **if** $\ell = \min\{d_H(\boldsymbol{a}^i, \boldsymbol{a}^j) \mid j \in X\} \leq h$ **then**
12 Set $Z := Z \backslash \{i\}$ and $k := k - \ell$
13 **end**
14 **end**
15 **if** $k \geq 0$ *and* $|Y| + |Z| \leq \sqrt{k \log(p + k)}$ **then**
16 solve the problem for the instance $(\mathbf{A}[\{1, \ldots, m\}, X \cup Y \cup Z], \mathbf{P}, \{X, Y, Z\}, k)$ using the algorithm from Lemma 12, return the answer and **quit**
17 **end**
18 **if** $h \leq k/(|Y| + |Z|)$ **then**
19 **foreach** $i \in Y \cup Z$ *and a vector* $\hat{\boldsymbol{a}}^i \in \{0, 1\}^m$ *s.t.* $d_H(\boldsymbol{a}^i, \hat{\boldsymbol{a}}^i) = h$ **do**
20 **if** $i \in Y$ **then**
21 call EXTEND- **P**- SOLUTION$(\hat{\mathbf{A}}, X \cup \{i\}, Y \backslash \{i\}, Z, k - h)$, where $\hat{\mathbf{A}}$ is obtained from $\mathbf{A}$ by replacing the column $\mathbf{a}^i$ by $\hat{\mathbf{a}}^i$;
22 **if** *the algorithm returns YES* **then**
23 Return YES and **quit**
24 **end**
25 **end**
26 **if** $i \in Z$ **then**
27 **foreach** $j \in Y$ **do**
28 Call EXTEND- **P**- SOLUTION$(\hat{\mathbf{A}}, X \cup \{j\}, Y \backslash \{j\}, Z, k - d_H(\mathbf{a}^j, \hat{\mathbf{a}}^i))$, where $\hat{\mathbf{A}}$ is obtained from $\mathbf{A}$ by replacing the column $\mathbf{a}^j$ by $\hat{\mathbf{a}}^i$;
29 **if** *the algorithm returns YES* **then**
30 Return YES and **quit**
31 **end**
32 **end**
33 **end**
34 **end**
35 **end**
36 Set $h := h + 1$
37 **end**
38 Return NO
39 **end**

$Z \neq \emptyset$. For every $i \in X$, we set $\mathbf{b}^i = \mathbf{a}^i$, and for each $i \in Z$, we define the vector $\mathbf{b}^i = \mathbf{a}^h$ for $h \in X$, where $d_H(\mathbf{a}^i, \mathbf{a}^h) = \min\{d_H(\mathbf{a}^i, \mathbf{a}^j) \mid j \in X\}$. Consider the matrix $\mathbf{B}$ composed of the columns $\mathbf{b}^i$ for $i \in X \cup Y \cup Z$. It is easy to verify that $\mathbf{B}$ is a solution to $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$. Hence, if the algorithm returns YES in Line 6, then $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$ is a yes-instance.

Now consider the case $Y \neq \emptyset$. This case is considered in Lines 9–38 and if the algorithm returns YES, then it is done in the while loop in these lines. Let $h \in \{0, \ldots, k\}$ be such that algorithm returns YES for this value of h.

Denote by Z' the set obtained from Z in the for loop in Lines 11–15, and let k' be the value of k obtained after the execution of these steps. Observe that (a) if $(\mathbf{A}[\{1, \ldots, m\}, X \cup Y \cup Z'], \mathbf{P}, \{X, Y, Z'\}, k')$ is a yes-instance, then $(\mathbf{A} = \mathbf{A}[\{1, \ldots, m\}, X \cup Y \cup Z], \mathbf{P}, \{X, Y, Z\}, k)$ is a yes-instance as well. Indeed, let $\hat{\mathbf{B}}$ be a solution to $(\mathbf{A}[\{1, \ldots, m\}, X \cup Y \cup Z'], \mathbf{P}, \{X, Y, Z'\}, k')$ with the columns $\hat{\mathbf{b}}^i$ for $i \in X \cup Y \cup Z'$. We define $\mathbf{B}$ with the columns $\mathbf{b}^i$ for $i \in X \cup Y \cup Z$ as follows: for $i \in X \cup Y \cup Z'$, $\mathbf{b}^i = \hat{\mathbf{b}}^i$, and for $i \in Z \backslash Z'$, $\mathbf{b}^i = \mathbf{a}^s = \hat{\mathbf{b}}^s$ for $s \in X$ such that $d_H(\mathbf{a}^i, \mathbf{a}^s) = \min\{d_H(\mathbf{a}^i, \mathbf{a}^j) \mid j \in X\}$. It is easy to see that $\mathbf{B}$ is a solution for $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$.

Therefore, if the algorithm returns YES in Line 17, then, by the statement (a), $(\mathbf{A}[\{1, \ldots, m\}, X \cup Y \cup Z'], \mathbf{P}, \{X, Y, Z'\}, k')$ and $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$ are yes-instances.

Suppose that the algorithm returns YES in Lines 19–36 for $i \in Y \cup Z'$. Here, we have two cases. The first case is when Algorithm 5 returns YES in Line 24. Then $i \in Y$ and $(\hat{\mathbf{A}}[\{1, \ldots, m\}, X \cup Y \cup Z'], \mathbf{P}, \{X \cup \{i\}, Y \backslash \{i\}, Z'\}, k' - h)$ is a yes-instance. Since $d_H(\mathbf{a}^i, \hat{\mathbf{a}}^i) = h$, we have that every solution to $(\hat{\mathbf{A}}[\{1, \ldots, m\}, X \cup Y \cup Z'], \mathbf{P}, \{X \cup \{i\}, Y \backslash \{i\}, Z'\}, k' - h)$ is also a solution to $(\mathbf{A}[\{1, \ldots, m\}, X \cup Y \cup Z'], \mathbf{P}, \{X, Y, Z'\}, k')$. This implies that $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$ is a yes-instance (by the statement (a)).

The second case is when the algorithm returns YES in Line 31. Then $i \in Z'$. Since the algorithm returns YES in Lines 27–34 for the value i, we have that there exists $j \in Y$ such that EXTEND-$\mathbf{P}$-SOLUTION$(\hat{\mathbf{A}}, X \cup \{j\}, Y \backslash \{j\}, Z', k' - d_H(\mathbf{a}^j, \hat{\mathbf{a}}^i))$ returns YES, where $\hat{\mathbf{A}}$ is the matrix obtained from $\mathbf{A}$ by replacing column $\mathbf{a}^j$ by $\hat{\mathbf{a}}^i$. This implies that $(\mathbf{A}[\{1, \ldots, m\}, X \cup Y \cup Z'], \mathbf{P}, \{X, Y, Z'\}, k')$. Hence, by by the statement (a), $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$ is a yes-instance.

Next we prove that (b) if $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$ is a yes-instance, then the algorithm will output YES. We show this using induction on $|Y|$.

The base case is when $|Y| = 0$, i.e., $Y = \emptyset$. This case is considered in Lines 2–8 of Algorithm 5. Let $\mathbf{B}$ be a solution for $(\mathbf{A}, \mathbf{P}, \{X, Y, Z\}, k)$. Since $\mathbf{A}[\{1, \ldots, m\}, X] = \mathbf{B}[\{1, \ldots, m\}, X]$, we have that $\mathbf{A}[\{1, \ldots, m\}, X]$ is a $\mathbf{P}$-matrix. In particular, the algorithm does not stop in Lines 3–4. If $Z = \emptyset$, then the algorithm returns YES in Line 6. Notice that, when $Z = \emptyset$, $\mathbf{A} = \mathbf{A}[\{1, \ldots, m\}, X]$ is a $\mathbf{P}$-matrix, and hence $\mathbf{A}$ is a solution. Now, consider the case $Z \neq \emptyset$. Notice that because $\mathbf{A}[\{1, \ldots, m\}, X] = \mathbf{B}[\{1, \ldots, m\}, X]$ and $\mathbf{B}$ is a solution we have that

$$\sum_{i \in Z} \min\{d_H(\mathbf{a}^i, \mathbf{a}^j) \mid j \in X\} \leq \sum_{i \in Z} d_H(\mathbf{a}^i, \mathbf{b}^i) \leq k.$$

Hence, the algorithm returns YES in Line 6.

Now we consider the induction step. That is, we assume that $|Y| > 0$ and we the statement (b) holds for sets Y' whose cardinality is less than that of Y. Let $h^* = \min\{d_H(\mathbf{a}^i, \mathbf{b}^i) \mid i \in Y \cup Z\}$. Let also $i^* \in Y \cup Z$ be such that $h^* = d_H(\mathbf{a}^{i^*}, \mathbf{b}^{i^*})$.

We claim that EXTEND-**P**-SOLUTION returns YES in the while loop in Lines 10–38 for $h = h^*$ unless it does not return YES before.

Denote by Z^* the set obtained from Z in Lines 11–15 and let k^* the value of k obtained in Lines 11–15 for $h = h^*$. By the choice of h^*, it follows that $(\mathbf{A}[\{1,\ldots,m\}, X \cup Y \cup Z^*], \mathbf{P}, \{X, Y, Z\}, k^*)$ is a yes-instance of EXTENDABLE **P**-MATRIX APPROXIMATION. If $|Y| + |Z^*| \leq \sqrt{k^* \log(p+k^*)}$, we solve EXTENDABLE **P**-MATRIX APPROXIMATION for $(\mathbf{A}[\{1,\ldots,m\}, X \cup Y \cup Z^*], \mathbf{P}, \{X, Y, Z^*\}, k^*)$ directly using Lemma 12. Hence, the algorithm returns YES in Line 17.

So, now on, we assume that $|Y| + |Z^*| > \sqrt{k^* \log(p+k^*)}$. Notice that for each $i \in Z^*$, we have that $d_H(\mathbf{a}^i, \mathbf{b}^i) > h^*$. Hence,

$$k^* \geq \sum_{i \in Y \cup Z^*} d_H(\mathbf{a}^i, \mathbf{b}^i) > h^* \cdot (|Y| + |Z^*|)$$

and $h^* < k^*/(|Y| + |Z^*|) \leq \sqrt{k^*/\log(p+k^*)}$. Hence Lines 20–35 will be executed. In these lines, we consider every $i \in Y \cup Z^*$ and each vector $\hat{\mathbf{a}}^i \in \{0,1\}^m$ such that $d_H(\mathbf{a}^i, \hat{\mathbf{a}}^i) = h^*$. In particular, we consider the execution of Lines 21–34, when $i = i^*$ and $\hat{\mathbf{a}}^{i^*} = \mathbf{b}^{i^*}$. Here, we have two cases. In the first case $i^* \in Y$. Then, $(\hat{\mathbf{A}}[\{1,\ldots,m\}, X \cup Y \cup Z^*], \mathbf{P}, \{X \cup \{i^*\}, Y \setminus \{i^*\}, Z^*\}, k^* - h^*)$ is a yes-instance, where $\hat{\mathbf{A}}$ is obtained from $\mathbf{A}$ by the replacing column a^{i^*} by $\hat{\mathbf{a}}^{i^*} = \mathbf{b}^{i^*}$. By the induction hypothesis, we have that the algorithm returns YES for this branch in Lines 23-24.

In the second case $i^* \in Z^*$. We know that $\hat{\mathbf{a}}^{i^*} = \mathbf{b}^{i^*}$ is a column in $\mathbf{B}[\{1,\ldots,m\}, X \cup Y]$. We claim that $\hat{\mathbf{a}}^{i^*} = \mathbf{b}^j$ for some $j \in Y$. For the sake of contradiction assume that $j \in X$. Then,

$$\begin{aligned}
d_H(\mathbf{a}^{i^*}, \mathbf{a}^j) &\leq d_H(\mathbf{a}^{i^*}, \hat{\mathbf{a}}^{i^*}) + d_H(\hat{\mathbf{a}}^{i^*}, \mathbf{a}^j) && \text{(by the triangle inequality)} \\
&= h^* + d_H(\hat{\mathbf{a}}^{i^*}, \mathbf{a}^j) \\
&= h^* + d_H(\mathbf{b}^j, \mathbf{a}^j) && \text{(Because } \hat{\mathbf{a}}^{i^*} = \mathbf{b}^{i^*}\text{)} \\
&= h^* + 0 && \text{(Because } j \in X\text{)}
\end{aligned}$$

Since $d_H(\mathbf{a}^{i^*}, \mathbf{a}^j) \leq h^*$, in the for loop in Lines 11–15, i^* will be deleted from Z and hence $i^* \notin Z^*$. This is a contradiction to the assumption that $i^* \in Z^*$. So, now we have that $j \in Y$. Then, $(\hat{\mathbf{A}}, X \cup \{j\}, Y \setminus \{j\}, Z^*, k - d_H(\mathbf{a}^j, \hat{\mathbf{a}}^{i^*}))$ is a yes-instance, where $\hat{\mathbf{A}}$ is the matrix obtained from $\mathbf{A}$ by replacing column $\mathbf{a}^j$ by $\hat{\mathbf{a}}^{i^*}$. By the induction hypothesis, we have that the algorithm returns YES in Lines 27-34 for this value of j in the for loop in these lines. This completes the correctness proof of the algorithm EXTEND-**P**-SOLUTION.

Running time Recall that the kernelization algorithm from Lemma 10 runs in $\mathcal{O}((p+q+k)nm)$ time and it holds that $m \leq (\max\{k,p\}+1)(p+k)$ and $n \leq (\max\{k,q\}+1)(q+k)$ after this phase. Lemma 11 runs in $(q+k)^{\mathcal{O}(q)}(p+q+k)^{\mathcal{O}(1)}$ time, because $m \leq (\max\{k,p\}+1)(p+k)$ and $n \leq (\max\{k,q\}+1)(q+k)$ and it outputs a collection of at most $(q+k)^q$ instance of **P**-MATRIX APPROXIMATION. Now we evaluate the running time of Algorithm 5.

By Observation 5, Lines 3–4 can be done in $2^{\mathcal{O}(p \log p + q \log q)} \cdot (nm)^{\mathcal{O}(1)}$ time. Clearly, Lines 6-7 are executed in polynomial time. In the while loop in Lines 10–38, we consider $k+1$ values of h and for each h perform Lines 11–37. Lines 11–15 take polynomial time. The steps in Lines 16–18 can be done in $2^{\mathcal{O}(p(\sqrt{k \log(p+k)} + \log k) + p \log p + q \log q)} \cdot (nm)^{\mathcal{O}(1)}$ time by Lemma 12. In Lines 19–36, we consider at most $|Y| + |Z| \leq q + k$ values of i in the for loop in Lines 20–35, and for each i, consider all vectors $\hat{\mathbf{a}}^i \in \{0,1\}^m$ such that $d_H(\mathbf{a}^i, \hat{\mathbf{a}}^i) = h \leq \sqrt{k/\log(p+k)}$. Recall that after the preprocessing, we have that $m \leq (p+k)^2$. Hence, we have at most $(p+k)^{2\sqrt{k/\log(p+k)}}$ vectors $\hat{\mathbf{a}}^i$. The number of choices for j in Lines 28–33 is at most q. This implies that in Lines 20–35 we have $2^{\mathcal{O}(\sqrt{k \log(p+k)})}$ recursive calls. On each recursive call, we reduce the size of Y. It means that the depth of the recursion is at most q. Then the total running time is

$$2^{\mathcal{O}(p\sqrt{k \log(p+k)} + q \log \sqrt{k \log(p+k)} + p \log p + q \log q + q \log(q+k))} \cdot (nm)^{\mathcal{O}(1)}.$$

Since $m \leq (\max\{k, p\} + 1)(p+k)$ and $n \leq (\max\{k, q\} + 1)(q+k)$, the branching algorithm runs in

$$2^{\mathcal{O}(p\sqrt{k \log(p+k)} + q \log \sqrt{k \log(p+k)} + p \log p + q \log q + q \log(q+k))}$$

time and after taking into account the kernelization algorithm and Lemma 11, the total running time can be bounded from above by

$$2^{\mathcal{O}(p\sqrt{k \log(p+k)} + q \log \sqrt{k \log(p+k)} + p \log p + q \log q + q \log(q+k))} \cdot nm.$$

This competes the proof of the theorem. □

Note that the running time in Theorem 7 is asymmetric in p and q due to the fact that we treat rows and columns in different way but, trivially, the instances $(\mathbf{A}, \mathbf{P}, k)$ and $(\mathbf{A}^\mathsf{T}, \mathbf{P}^\mathsf{T}, k)$ of **P**-MATRIX APPROXIMATION are equivalent. If p and q are assumed to be constants, then **P**-MATRIX APPROXIMATION is solvable in $2^{\mathcal{O}(\sqrt{k \log k})} \cdot nm$ time.

Notice that we can invoke Theorem 7 to solve LOW GF(2)- RANK APPROXIMATION as follows. We observe that $(\mathbf{A}, r, k)$ is a yes-instance of LOW GF(2)- RANK APPROXIMATION if and only if there is a $p \times q$-matrix $\mathbf{P}$ of GF(2)-rank at most r for $p = \min\{m, 2^r\}$ and $q = \min\{n, 2^r\}$ such that $(\mathbf{A}, \mathbf{P}, k)$ is a yes-instance of **P**-MATRIX APPROXIMATION. A $p \times q$-matrix $\mathbf{P}$ is of GF(2)-rank r if and only if it can be represented as a product $\mathbf{P} = \mathbf{U} \cdot \mathbf{V}$, where $\mathbf{U}$ is $p \times r$ and $\mathbf{V}$ is $r \times q$ binary matrix and arithmetic operations are over GF(2). There are at most 2^{r2^r} different binary $p \times r$-matrices $\mathbf{U}$, and at most 2^{r2^r} different binary $r \times q$ -matrices $\mathbf{V}$. Thus there are at most 2^{2r2^r} candidate matrices $\mathbf{P}$. For each such matrix $\mathbf{P}$, we check whether $(\mathbf{A}, \mathbf{P}, k)$ is a yes-instance of **P**-MATRIX APPROXIMATION by invoking Theorem 7. However this approach gives a double exponential dependence in r, which is much worse the bound provided by Theorem 6. Still, this approach is useful if we consider the variant of LOW GF(2)- RANK APPROXIMATION for Boolean matrices.

Let us remind that binary matrix $\mathbf{A}$ has the Boolean rank 1 if $A = \mathbf{x} \wedge \mathbf{y}^\intercal$ where $\mathbf{x} \in \{0,1\}^m$ and $\mathbf{y} \in \{0,1\}^n$ are nonzero vectors. The Boolean rank of $\mathbf{A}$ is the minimum integer r such that $\mathbf{A} = \mathbf{A}^{(1)} \vee \cdots \vee \mathbf{A}^{(r)}$ where $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(r)}$ are matrices of Boolean rank 1.

Theorem 8 LOW BOOLEAN- RANK APPROXIMATION *admits an algorithm running in time* $2^{\mathcal{O}(r2^r\sqrt{k \log k})} \cdot nm$.

Proof Let $\mathbf{A}$ be a Boolean $m \times n$-matrix with the Boolean rank $r \geq 1$. Then $\mathbf{A} = \mathbf{A}^{(1)} \vee \ldots \vee \mathbf{A}^{(r)}$ where $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(r)}$ are matrices of Boolean rank 1. It implies that $\mathbf{A}$ has at most 2^r pairwise-distinct rows and at most 2^r pairwise-distinct columns. Hence, the Boolean rank of $\mathbf{A}$ is at most r if and only if there is a $p \times q$-matrix $\mathbf{P}$ of Boolean rank at most r for $p = \min\{m, 2^r\}$ and $q = \min\{n, 2^r\}$ such that $\mathbf{A}$ is a $\mathbf{P}$-matrix. Respectively, the LOW BOOLEAN- RANK APPROXIMATION problem can be reformulated as follows: Decide whether there is a $p \times q$-pattern matrix $\mathbf{P}$ with the Boolean rank at most r and an $m \times n$ $\mathbf{P}$-matrix $\mathbf{B}$ such that $d_H(\mathbf{A}, \mathbf{B}) \leq k$.

We generate all $p \times q$-matrices $\mathbf{P}$ of Boolean rank at most r, and then for each matrix $\mathbf{P}$, we solve $\mathbf{P}$-MATRIX APPROXIMATION for the instance $(\mathbf{A}, \mathbf{P}, k)$. We return YES if we obtain at least one yes-instance of $\mathbf{P}$-MATRIX APPROXIMATION, and we return NO otherwise.

By definition, the Boolean rank of $\mathbf{P}$ is r if and only if $\mathbf{P} = \mathbf{U} \wedge \mathbf{V}$ for a Boolean $p \times r$ matrix $\mathbf{U}$ and a Boolean $r \times q$ matrix $\mathbf{V}$. Since there are at most 2^{r2^r} $p \times r$-matrices ($r \times q$-matrices respectively), we construct all the $p \times q$-matrices $\mathbf{P}$ with the Boolean rank at most r in $2^{\mathcal{O}(r2^r)}$ time. By Theorem 7, the considered instances of $\mathbf{P}$-MATRIX APPROXIMATION is solvable in $2^{\mathcal{O}(r2^r\sqrt{k \log k})} \cdot nm$ time. This implies that the total running time is $2^{\mathcal{O}(r2^r\sqrt{k \log k})} \cdot nm$. □

To conclude the section, we observe that we hardly can avoid the double exponential dependence on r for LOW BOOLEAN- RANK APPROXIMATION. It was proved by Chandran et al. (2016) that the BICLIQUE COVER problem that asks, given a bipartite graph G, whether the set of edges of G could be covered by at most r bicliques (that is, complete bipartite graphs) cannot be solved in $2^{2^{o(r)}} \cdot |V(G)|^{\mathcal{O}(1)}$ time, unless the *Exponential Time Hypothesis* (ETH) is false (we refer to Cygan et al. (2015) for the introduction to the algorithmic lower bounds based on ETH). Since BICLIQUE COVER is equivalent to deciding whether the bipartite adjacency matrix of G has the Boolean rank at most r, LOW BOOLEAN- RANK APPROXIMATION cannot be solved in time $2^{2^{o(r)}} \cdot (nm)^{\mathcal{O}(1)}$ for $k = 0$ unless ETH fails. This implies the following proposition.

Proposition 3 *For any computable function* f, LOW BOOLEAN- RANK APPROXIMATION *cannot be solved in* $2^{2^{o(r)}} f(k) \cdot (nm)^{\mathcal{O}(1)}$ *time unless ETH fails.*

Proof Let $(\mathbf{A}, r, k)$ be an instance of LOW BOOLEAN- RANK APPROXIMATION. Consider the $m \times (k+1)n$-matrix $\mathbf{A}^*$ obtained from $\mathbf{A}$ by taking $k+1$ copies of each column of $\mathbf{A}$. Observe that $(\mathbf{A}, r, k)$ is a yes-instance of LOW BOOLEAN- RANK APPROXIMATION if and only if $(\mathbf{A}^*, r, 0)$ is a yes-instance. Then any algorithm solving LOW BOOLEAN- RANK APPROXIMATION in $2^{2^{o(r)}} \cdot f(k) \cdot (nm)^{\mathcal{O}(1)}$ time would imply that BICLIQUE COVER can be solved in $2^{2^{o(r)}} \cdot |V(G)|^{\mathcal{O}(1)}$ time contradicting ETH by Chandran et al. (2016). □

8 Conclusion and open problems

In this paper we provide a number of parameterized algorithms for a number of binary matrix-approximation problems. Our results uncover some parts of the complexity landscape of these fascinating problems. We hope that our work will facilitate further investigation of this important and exciting area. We conclude with the following concrete open problems about bivariate complexity of Binary r-Means, Low GF(2)- Rank Approximation, and Low Boolean- Rank Approximation.

We have shown that Binary r-Means is solvable in $2^{\mathcal{O}(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$ time. A natural question is whether this running time is optimal. While the lower bound of the kind $2^{o(k)} \cdot (nm)^{\mathcal{O}(1)}$ or $2^{o(k \log k)} \cdot (nm)^{\mathcal{O}(1)}$ seems to be most plausible here, we do not know any strong argument against, say a $2^{o(k)} \cdot (nm)^{\mathcal{O}(1)}$ time algorithm. At least for the number of distinct columns $r \in \mathcal{O}(k^{1-\varepsilon})$ with $\varepsilon > 0$, we have a subexponential in k algorithm, so maybe we can solve the problem in subexponential in k time for any value of r?

For Low GF(2)- Rank Approximation, we have an algorithm solving the problem in $2^{\mathcal{O}(r \cdot \sqrt{k \log rk})}(nm)^{\mathcal{O}(1)}$ time. Here, shaving off the $\sqrt{\log k}$ factor in the exponent seems to be a reasonable thing. However, we do not know how to do it even at the cost of a worse dependence in r. In other words, could the problem be solvable in $2^{\mathcal{O}(f(r) \cdot \sqrt{k})}(nm)^{\mathcal{O}(1)}$ time for some function f? On the other hand, we also do not know how to rule out algorithms running in $2^{o(r) \cdot o(k)}(nm)^{\mathcal{O}(1)}$ time.

For Low Boolean- Rank Approximation, how far is our upper bound $2^{\mathcal{O}(r2^r \cdot \sqrt{k \log k})}(nm)^{\mathcal{O}(1)}$ from the optimal? For example, we know that for any function f, the solvability of the problem in $2^{2^{o(r)}} f(k)(nm)^{\mathcal{O}(1)}$ time refutes ETH. Could we rule out any $2^{o(\sqrt{k})} f(r)(nm)^{\mathcal{O}(1)}$ algorithm?

Acknowledgements We thank Daniel Lokshtanov, Syed Mohammad Meesum and Saket Saurabh for helpful discussions on the topic of the paper. We also are very grateful to the anonymous reviewers whose suggestions helped us to improve our results.

Funding The research leading to these results have been supported by the Research Council of Norway via the projects "CLASSIS" (grant 249994) and "MULTIVAL" (grant 263317).

References

Agarwal PK, Har-Peled S, Varadarajan KR (2004) Approximating extent measures of points. J ACM 51(4):606–635

Aho AV, Ullman JD, Yannakakis M (1983) On notions of information transfer in VLSI circuits. In: Proceedings of the 15th annual ACM symposium on theory of computing (STOC), ACM, pp 133–139

Alon N, Sudakov B (1999) On two segmentation problems. J Algorithms 33(1):173–184

Alon N, Yuster R, Zwick U (1995) Color-coding. J ACM 42(4):844–856

Arora S, Ge R, Kannan R, Moitra A (2012) Computing a nonnegative matrix factorization—provably. In: Proceedings of the 44th annual ACM symposium on theory of computing (STOC), ACM, pp 145–162

Badoiu M, Har-Peled S, Indyk P (2002) Approximate clustering via core-sets. In: Proceedings of the 34th annual ACM symposium on theory of computing (STOC). ACM, pp 250–257

Ban F, Bhattiprolu V, Bringmann K, Kolev P, Lee E, Woodruff DP (2019) A PTAS for ℓ_p-low rank approximation. In: Proceedings of the thirtieth annual ACM-SIAM symposium on discrete algorithms, SODA 2019, San Diego, California, USA, 6–9 Jan 2019. SIAM, pp 747–766

Bartl E, Belohlávek R, Konecny J (2010) Optimal decompositions of matrices with grades into binary and graded matrices. Ann Math Artif Intell 59(2):151–167
Basu A, Dinitz M, Li X (2016) Computing approximate PSD factorizations. CoRR arXiv:1602.07351
Belohlávek R, Vychodil V (2010) Discovery of optimal factors in binary data via a novel method of matrix decomposition. J Comput Syst Sci 76(1):3–20
Bodlaender HL, Downey RG, Fellows MR, Hermelin D (2009) On problems without polynomial kernels. J Comput Syst Sci 75(8):423–434
Boucher C, Lo C, Lokshtanov D (2011) Outlier detection for DNA fragment assembly. CoRR arXiv:1111.0376
Bringmann K, Kolev P, Woodruff DP (2017) Approximation algorithms for ℓ_0-low rank approximation. In: Advances in neural information processing systems 30 (NIPS), pp 6651–6662
Candès EJ, Li X, Ma Y, Wright J (2011) Robust principal component analysis? J ACM 58(3):11:1–11:37
Chandran LS, Issac D, Karrenbauer A (2016) On the parameterized complexity of biclique cover and partition. In: Proceedings of the 11th international symposium on parameterized and exact computation (IPEC), Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, LIPIcs, vol 63, pp 11:1–11:13
Chandrasekaran V, Sanghavi S, Parrilo PA, Willsky AS (2011) Rank-sparsity incoherence for matrix decomposition. SIAM J Optim 21(2):572–596
Cichocki A, Zdunek R, Phan AH, Si Amari (2009) Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. Wiley, Hoboken
Cilibrasi R, van Iersel L, Kelk S, Tromp J (2007) The complexity of the single individual SNP haplotyping problem. Algorithmica 49(1):13–36
Clarkson KL, Woodruff DP (2015) Input sparsity and hardness for robust subspace approximation. In: Proceedings of the 56th annual symposium on Foundations of Computer Science (FOCS). IEEE Computer Society, pp 310–329
Cohen JE, Rothblum UG (1993) Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. Linear Algebra Appl 190:149–168
Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) Parameterized algorithms. Springer, Berlin
Dan C, Hansen KA, Jiang H, Wang L, Zhou Y (2015) On low rank approximation of binary matrices. CoRR arXiv:1511.01699
Downey RG, Fellows MR (1992) Fixed-parameter tractability and completeness. In: Proceedings of the 21st Manitoba conference on numerical mathematics and computing Congressus Numerantium, vol 87, pp 161–178
Downey RG, Fellows MR (2013) Fundamentals of parameterized complexity. Texts in computer science. Springer, Berlin
Drange PG, Reidl F, Villaamil FS, Sikdar S (2015) Fast biclustering by dual parameterization. CoRR arXiv:1507.08158
Feige U (2014) NP-hardness of hypercube 2-segmentation. CoRR arXiv:1411.0821
Fiorini S, Massar S, Pokutta S, Tiwary HR, de Wolf R (2015) Exponential lower bounds for polytopes in combinatorial optimization. J ACM 62(2):17
Fomin FV, Kratsch S, Pilipczuk M, Pilipczuk M, Villanger Y (2014) Tight bounds for parameterized complexity of cluster editing with a small number of clusters. J Comput Syst Sci 80(7):1430–1447
Fomin FV, Golovach PA, Lokshtanov D, Panolan F, Saurabh S (2018a) Approximation schemes for low-rank binary matrix approximation problems. CoRR arXiv:1807.07156
Fomin FV, Lokshtanov D, Meesum SM, Saurabh S, Zehavi M (2018b) Matrix rigidity from the viewpoint of parameterized complexity. SIAM J Discrete Math 32(2):966–985
Fomin FV, Lokshtanov D, Saurabh S, Zehavi M (2019) Kernelization. Theory of parameterized preprocessing. Cambridge University Press, Cambridge
Fu Y (2014) Low-rank and sparse modeling for visual analysis, 1st edn. Springer, Berlin
Geerts F, Goethals B, Mielikäinen T (2004) Tiling databases. In: Proceedings of the 7th international conference on discovery science, (DS), pp 278–289
Gillis N, Vavasis SA (2015) On the complexity of robust PCA and ℓ_1-norm low-rank matrix approximation. CoRR arXiv:1509.09236
Gramm J, Guo J, Hüffner F, Niedermeier R (2008) Data reduction and exact algorithms for clique cover. ACM J Exp Algorithmics. https://doi.org/10.1145/1412228.1412236
Gregory DA, Pullman NJ, Jones KF, Lundgren JR (1991) Biclique coverings of regular bigraphs and minimum semiring ranks of regular matrices. J Comb Theory Ser B 51(1):73–89

Grigoriev D (1976) Using the notions of separability and independence for proving the lower bounds on the circuit complexity (in Russian). Notes of the Leningrad branch of the Steklov Mathematical Institute, Nauka

Grigoriev D (1980) Using the notions of separability and independence for proving the lower bounds on the circuit complexity. J Sov Math 14(5):1450–1456

Gutch HW, Gruber P, Yeredor A, Theis FJ (2012) ICA over finite fields—separability and algorithms. Sig Process 92(8):1796–1808

Guterman AE (2008) Rank and determinant functions for matrices over semirings. In: Surveys in contemporary mathematics, London Mathematical Society lecture note series, vol 347. Cambridge University Press, Cambridge, pp 1–33

Inaba M, Katoh N, Imai H (1994) Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. In: Proceedings of the 10th annual symposium on computational geometry. ACM, pp 332–339

Jiang P, Heath MT (2013) Mining discrete patterns via binary matrix factorization. In: ICDM workshops. IEEE Computer Society, pp 1129–1136

Jiang P, Peng J, Heath M, Yang R (2014) A clustering approach to constrained binary matrix factorization. Springer, Berlin, pp 281–303

Kannan R, Vempala S (2009) Spectral algorithms. Found Trends Theor Comput Sci 4(3–4):157–288

Kleinberg J, Papadimitriou C, Raghavan P (2004) Segmentation problems. J ACM 51(2):263–280

Koyutürk M, Grama A (2003) Proximus: a framework for analyzing very high dimensional discrete-attributed datasets. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining (KDD). ACM, New York, pp 147–156

Kumar A, Sabharwal Y, Sen S (2010) Linear-time approximation schemes for clustering problems in any dimensions. J ACM 57(2):5:1–5:32

Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791

Lokam SV (2009) Complexity lower bounds using linear algebra. Found Trends Theor Comput Sci 4:1–155

Lovász L, Saks ME (1988) Lattices, möbius functions and communication complexity. In: Proceedings of the 29th annual symposium on Foundations of Computer Science (FOCS). IEEE, pp 81–90

Lu H, Vaidya J, Atluri V (2008) Optimal boolean matrix decomposition: application to role engineering. In: Proceedings of the 24th international conference on data engineering, (ICDE), pp 297–306

Lu H, Vaidya J, Atluri V, Shin H, Jiang L (2011) Weighted rank-one binary matrix factorization. In: Proceedings of the eleventh SIAM international conference on data mining, SDM 2011, 28–30 Apr 2011, Mesa, Arizona, USA. SIAM/Omnipress, pp 283–294

Lu H, Vaidya J, Atluri V, Hong Y (2012) Constraint-aware role mining via extended boolean matrix decomposition. IEEE Trans Dependable Secur Comput 9(5):655–669

Mahoney MW (2011) Randomized algorithms for matrices and data. Found Trends Mach Learn 3(2):123–224

Marx D (2008) Closest substring problems with small distances. SIAM J Comput 38(4):1382–1410

Meesum SM, Saurabh S (2016) Rank reduction of directed graphs by vertex and edge deletions. In: Proceedings of the 12th Latin American symposium on (LATIN), lecture notes in computer science, vol 9644. Springer, pp 619–633

Meesum SM, Misra P, Saurabh S (2016) Reducing rank of the adjacency matrix by graph modification. Theoret Comput Sci 654:70–79

Miettinen P, Vreeken J (2011) Model order selection for boolean matrix factorization. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining (KDD). ACM, pp 51–59

Miettinen P, Mielikäinen T, Gionis A, Das G, Mannila H (2008) The discrete basis problem. IEEE Trans Knowl Data Eng 20(10):1348–1362

Mitra B, Sural S, Vaidya J, Atluri V (2016) A survey of role mining. ACM Comput Surv 48(4):50:1–50:37

Moitra A (2016) An almost optimal algorithm for computing nonnegative rank. SIAM J Comput 45(1):156–173

Naik GR (2016) Non-negative matrix factorization techniques. Springer, Berlin

Naor M, Schulman LJ, Srinivasan A (1995) Splitters and near-optimal derandomization. In: Proceedings of the 36th annual symposium on Foundations of Computer Science (FOCS). IEEE, pp 182–191

Orlin J (1977) Contentment in graph theory: covering graphs with cliques. Nederl Akad Wetensch Proc Ser A 80=Indag Math 39(5):406–424

Ostrovsky R, Rabani Y (2002) Polynomial-time approximation schemes for geometric min-sum median clustering. J ACM 49(2):139–156

Painsky A, Rosset S, Feder M (2016) Generalized independent component analysis over finite alphabets. IEEE Trans Inf Theory 62(2):1038–1053

Razborov AA (1989) On rigid matrices. Manuscript in Russian

Razenshteyn IP, Song Z, Woodruff DP (2016) Weighted low rank approximations with provable guarantees. In: Proceedings of the 48th annual ACM symposium on theory of computing (STOC). ACM, pp 250–263

Shen BH, Ji S, Ye J (2009) Mining discrete patterns via binary matrix factorization. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining (KDD). ACM, New York, pp 757–766

Shi Z, Wang L, Shi L (2014) Approximation method to rank-one binary matrix factorization. In: 2014 IEEE international conference on automation science and engineering, CASE 2014, New Taipei, Taiwan, 18–22 Aug 2014. IEEE, pp 800–805

Vaidya J (2012) Boolean matrix decomposition problem: theory, variations and applications to data engineering. In: Proceedings of the 28th IEEE international conference on data engineering (ICDE). IEEE Computer Society, pp 1222–1224

Vaidya J, Atluri V, Guo Q (2007) The role mining problem: finding a minimal descriptive set of roles. In: Proceedings of the 12th ACM symposium on access control models and (SACMAT), pp 175–184

Valiant LG (1977) Graph-theoretic arguments in low-level complexity. In: Mathematical foundations of computer science (MFCS), Lecture Notes in Computer Science, vol 53. Springer, pp 162–176

Woodruff DP (2014) Sketching as a tool for numerical linear algebra. Found Trends Theor Comput Sci 10(1–2):1–157

Wright J, Ganesh A, Rao SR, Peng Y, Ma Y (2009) Robust principal component analysis: exact recovery of corrupted low-rank matrices via convex optimization. In: Proceedings of 23rd annual conference on neural information processing systems (NIPS). Curran Associates, Inc., pp 2080–2088

Wulff S, Urner R, Ben-David S (2013) Monochromatic bi-clustering. In: Proceedings of the 30th international conference on machine learning, (ICML), JMLR.org, JMLR workshop and conference proceedings, vol 28, pp 145–153

Yannakakis M (1991) Expressing combinatorial optimization problems by linear programs. J Comput Syst Sci 43(3):441–466

Yeredor A (2011) Independent component analysis over Galois fields of prime order. IEEE Trans Inf Theory 57(8):5342–5359