Approximation Schemes for Low-rank Binary Matrix Approximation Problems

FEDOR V. FOMIN and PETR A. GOLOVACH, Department of Informatics, University of Bergen, Norway
DANIEL LOKSHTANOV, University of California Santa Barbara, USA
FAHAD PANOLAN, Department of Computer Science and Engineering, IIT Hyderabad, India
SAKET SAURABH, The Institute of Mathematical Sciences, HBNI, India

We provide a randomized linear time approximation scheme for a generic problem about clustering of binary vectors subject to additional constraints. The new constrained clustering problem generalizes a number of problems and by solving it, we obtain the first linear time-approximation schemes for a number of well-studied fundamental problems concerning clustering of binary vectors and low-rank approximation of binary matrices. Among the problems solvable by our approach are Low GF(2)-Rank Approximation, Low Boolean-Rank Approximation, and various versions of Binary Clustering. For example, for Low GF(2)-Rank Approximation problem, where for an $m \times n$ binary matrix A and integer $r > 0$, we seek for a binary matrix B of GF(2) rank at most r such that the ℓ_0-norm of matrix $\mathbf{A} - \mathbf{B}$ is minimum, our algorithm, for any $\epsilon > 0$ in time $f(r, \epsilon) \cdot n \cdot m$, where f is some computable function, outputs a $(1 + \epsilon)$-approximate solution with probability at least $(1 - \frac{1}{e})$. This is the first linear time approximation scheme for these problems. We also give (deterministic) PTASes for these problems running in time $n^{f(r)\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}}$, where f is some function depending on the problem. Our algorithm for the constrained clustering problem is based on a novel sampling lemma, which is interesting on its own.

CCS Concepts: • **Theory of computation → Design and analysis of algorithms**; • **Mathematics of computing → Probability and statistics**;

Additional Key Words and Phrases: Binary matrix factorization, clustering, approximation scheme, random sampling

ACM Reference format:
Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2019. Approximation Schemes for Low-rank Binary Matrix Approximation Problems. *ACM Trans. Algorithms* 16, 1, Article 12 (November 2019), 39 pages.
https://doi.org/10.1145/3365653

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416) and from the Norwegian Research Council via grants MULTIVAL and CLASSIS.

Authors' addresses: F. V. Fomin and P. A. Golovach, Department of Informatics, University of Bergen, PB 7803, Bergen, 5020, Norway; emails: {fedor.fomin, petr.golovach}@uib.no; D. Lokshtanov, Department of Computer Science, University of California Santa Barbara, 2104, USA; email: daniello@ucsb.edu; F. Panolan, Department of Computer Science and Engineering, IIT Hyderabad, Kandi, Sangareddy, 502285, India; email: fahad@iith.ac.in; S. Saurabh, The Institute of Mathematical Sciences, HBNI, 4th CrossStreet, CIT Campus, Tharamani, Chennai, Tamil Nadu, 600113 and Department of Informatics, University of Bergen, PB 7803, Bergen, 5020; email: saket@imsc.res.in.

1549-6325/2019/11-ART12 $15.00
https://doi.org/10.1145/3365653

1 INTRODUCTION

Low-rank matrix approximation is a generic optimization problem, in which a given data matrix has to be approximated by another matrix of low rank. It is at the heart of the methods used in Machine Learning and Data Analysis like Principal Component Analysis (PCA) or Factor Analysis. A recent trend in many applications from data mining and knowledge discovery is the study of low-rank approximation of binary matrices. This is due to the fact that in various settings, like in latent semantic indexing, approximating a binary matrix by a low rank binary matrix is an easy way to interpret data succinctly [7, 28, 29]. There are well-known and efficient techniques like Singular Value Decomposition (SVD) for computing optimal low-rank approximation with respect to the Frobenius norm for matrices over *reals*. Unfortunately, these methods are often inapplicable for handling binary data. Moreover, it appears that most of the interesting variants of low-rank binary matrix approximation are NP-complete. This is the reason why the majority of the approaches used in practice rely on heuristics with no provable guarantees. In this article, we show that despite of their worst-case intractability, many problems around low-rank binary matrix approximation admit efficient approximation algorithms, and their behavior can be analyzed rigorously.

To obtain approximation algorithms for low-rank approximation problems, we design approximation algorithms for a "constrained" version of binary clustering.

A k-ary relation R is a set of binary k-tuples with elements from $\{0,1\}$. A k-tuple $t = (t_1, \ldots, t_k)$ *satisfies* R, we write $t \in R$, if t is equal to one of the k-tuples from R.

Definition 1 (Vectors Satisfying $\mathcal{R}$). Let $\mathcal{R} = \{R_1, \ldots, R_m\}$ be a set of k-ary relations. We say that a set $C = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$ of binary m-dimensional vectors *satisfies* $\mathcal{R}$ and write $< C, \mathcal{R} >$, if $(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i]) \in R_i$ for all $i \in \{1, \ldots, m\}$.

For example, for $m = 2$, $k = 3$, $R_1 = \{(0,0,1), (1,0,0)\}$, and $R_2 = \{(1,1,1), (1,0,1), (0,0,1)\}$, the set of vectors

$$\mathbf{c}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \ \mathbf{c}_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ \mathbf{c}_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

satisfies $\mathcal{R} = \{R_1, R_2\}$, because $(\mathbf{c}_1[1], \mathbf{c}_2[1], \mathbf{c}_3[1]) = (0,0,1) \in R_1$ and $(\mathbf{c}_1[2], \mathbf{c}_2[2], \mathbf{c}_3[2]) = (1,0,1) \in R_2$.

Let us recall that the *Hamming distance* between two vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^m$, where $\mathbf{x} = (x_1, \ldots, x_m)^\intercal$ and $\mathbf{y} = (y_1, \ldots, y_m)^\intercal$, 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. For a set of vectors C and a vector $\mathbf{x}$, we define $d_H(\mathbf{x}, C)$, the Hamming distance between $\mathbf{x}$ and C, as the minimum Hamming distance between $\mathbf{x}$ and a vector from C. Thus, $d_H(\mathbf{x}, C) = \min_{\mathbf{c} \in C} d_H(\mathbf{x}, \mathbf{c})$.

Then, we define the following problem:

Binary Constrained Clustering
Input: A set $X \subseteq \{0,1\}^m$ of n vectors, a positive integer k, and a set of k-ary relations $\mathcal{R} = \{R_1, \ldots, R_m\}$.
Task: Among all vector sets $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\} \subseteq \{0,1\}^m$ satisfying $\mathcal{R}$, find a set C minimizing the sum $\sum_{\mathbf{x} \in X} d_H(\mathbf{x}, C)$.

First, we prove the following theorem:

Theorem 1. *There is a deterministic algorithm that, given an instance of* Binary Constrained Clustering *and $\varepsilon > 0$, runs in time $m \cdot n^{O(\frac{k^2}{\varepsilon^2} \log \frac{1}{\varepsilon})}$, and outputs a $(1+\varepsilon)$-approximate solution.*

Theorem 1 is a warm-up for our main theorem, which is stated as follows:

Theorem 2. *There is an algorithm that for a given instance of* Binary Constrained Clustering *and* $\varepsilon > 0$ *in time* $2^{O(\frac{k^4}{\varepsilon^2}\log\frac{1}{\varepsilon})} \cdot (\frac{1}{\varepsilon})^{O(\frac{k}{\varepsilon^2}\log\frac{1}{\varepsilon})} n \cdot m$ *outputs a* $(1+\varepsilon)$*-approximate solution with probability at least* $(1-\frac{1}{e})$.

In other words, for any constant k and ε, the algorithm in linear time outputs a set of vectors $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\} \subseteq \{0,1\}^m$ satisfying $\mathcal{R}$ such that $\sum_{\mathbf{x}\in X} d_H(\mathbf{x}, C) \leq (1+\varepsilon)\cdot OPT$, where OPT is the value of the optimal solution.

Theorems 1 and 2 have a number of interesting applications.

1.1 Applications of Theorems 1 and 2

Binary matrix factorization is the following problem: Given a binary $m \times n$ matrix; that is a matrix with entries from the domain $\{0,1\}$,

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

the task is to find a "simple" binary $m \times n$ matrix $\mathbf{B}$ that approximates $\mathbf{A}$ subject to some specified constraints. One of the most widely studied error measures is the *Frobenius norm*, which for the 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, we want to find a matrix $\mathbf{B}$ with certain properties such that

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

is minimum.

In particular, two variants of the problem were studied in the literature. In the first variant, one seeks a matrix $\mathbf{B}$ of GF(2)-rank at most r. In the second variant, matrix $\mathbf{B}$ should be of Boolean rank at most r. Depending on the selection of the rank, we obtain two different optimization problems.

Low GF(2)-Rank Approximation. Here the task is to approximate a given binary matrix $\mathbf{A}$ by a matrix $\mathbf{B}$ that has GF(2)-rank at most r.

Low GF(2)-Rank Approximation
Input: An $m \times n$-matrix $\mathbf{A}$ over GF(2) and a positive integer r.
Task: Find a binary $m \times n$-matrix $\mathbf{B}$ with GF(2)-rank$(\mathbf{B}) \leq r$ such that $\|\mathbf{A}-\mathbf{B}\|_F^2$ is minimum.

Low Boolean-rank Approximation. Let $\mathbf{A}$ be a binary $m \times n$ matrix. Now, we consider the elements of $\mathbf{A}$ to be *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 semiring $(0, 1, \wedge, \vee)$. 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*.

Low Boolean-Rank Approximation
Input: A Boolean $m \times n$ matrix $\mathbf{A}$ and a positive integer r.
Task: Find a Boolean $m \times n$ matrix $\mathbf{B}$ of Boolean rank at most r such that $\|\mathbf{A} - \mathbf{B}\|_F^2$ is minimum.

Low-rank matrix approximation problems can be also treated as special cases of Binary Constrained Clustering. To keep the flow of the article, we postpone the proof of the following lemmata until Section 9.

Lemma 1. *For any instance* $(\mathbf{A}, r)$ *of* Low GF(2)-Rank Approximation, *one can construct in time* $O(m + n + 2^{2r})$ *an instance* $(X, k = 2^r, \mathcal{R})$ *of* Binary Constrained Clustering *with the following properties:*

- *for any* α*-approximate solution* C *of* $(X, k, \mathcal{R})$, *there is an algorithm that in time* $O(rmn)$ *returns an* α*-approximate solution* $\mathbf{B}$ *of* $(\mathbf{A}, r)$, *and*
- *for any* α*-approximate solution* $\mathbf{B}$ *of* $(\mathbf{A}, r)$, *there is an algorithm that in time* $O(rmn)$ *returns an* α*-approximate solution* C *of* $(X, k, \mathcal{R})$.

Similarly for Low Boolean-Rank Approximation, we have the following lemma:

Lemma 2. *For any instance* $(\mathbf{A}, r)$ *of* Low Boolean-Rank Approximation, *one can construct in time* $O(m + n + 2^{2r})$ *an instance* $(X, k = 2^r, \mathcal{R})$ *of* Binary Constrained Clustering *with the following properties:*

- *for any* α*-approximate solution* C *of* $(X, k, \mathcal{R})$, *there is an algorithm that in time* $O(rmn)$ *returns an* α*-approximate solution* $\mathbf{B}$ *of* $(\mathbf{A}, r)$, *and*
- *for any* α*-approximate solution* $\mathbf{B}$ *of* $(\mathbf{A}, r)$, *there is an algorithm that in time* $O(rmn)$ *returns an* α*-approximate solution* C *of* $(X, k, \mathcal{R})$.

Hence, to design approximation schemes for Low Boolean-Rank Approximation and Low GF(2)-Rank Approximation, it suffices to give an approximation scheme for Binary Constrained Clustering.

For $\alpha > 1$, we say that an algorithm is an α-approximation algorithm for the low-rank approximation problem if for a matrix $\mathbf{A}$ and an integer r it outputs a matrix $\mathbf{B}$ satisfying the required constraints such that $\|\mathbf{A} - \mathbf{B}\|_F^2 \leq \alpha \cdot \|\mathbf{A} - \mathbf{B}_r\|_F^2$, where $\mathbf{B}_r = \operatorname{argmin}_{\operatorname{rank}(\mathbf{B}_r)=r} \|\mathbf{A} - \mathbf{B}_r\|_F^2$. By Theorems 1 and 2 and Lemmata 1 and 2, we obtain the following:

Corollary 1. *There is an algorithm that for a given instance of* Low Boolean-Rank Approximation (Low GF(2)-Rank Approximation) *and* $\varepsilon > 0$ *in time*

$$\left(\frac{1}{\varepsilon}\right)^{(\frac{2^{O(r)}}{\varepsilon^2} \log \frac{1}{\varepsilon})} \cdot n \cdot m$$

outputs a $(1 + \varepsilon)$*-approximate solution with probability at least* $(1 - \frac{1}{e})$.

Corollary 2. *There is a deterministic algorithm that for a given instance of* Low Boolean-Rank Approximation (Low GF(2)-Rank Approximation) *and* $\varepsilon > 0$ *in time* $m \cdot n^{O(\frac{2^{2r}}{\varepsilon^2} \log \frac{1}{\varepsilon})}$ *outputs a* $(1 + \varepsilon)$*-approximate solution.*

Let us observe that our results also yield randomized approximation scheme for the "dual" maximization versions of the low-rank matrix approximation problems. In these problems one wants to maximize the number of elements that are the same in $\mathbf{A}$ and $\mathbf{B}$ or, in other words,

to maximize the value of $nm - \|\mathbf{A}-\mathbf{B}\|_F^2$. It is easy to see that for every binary $m \times n$ matrix $\mathbf{A}$ there is a binary matrix $\mathbf{B}$ with GF(2)-rank$(\mathbf{B}) \leq 1$ such that $\|\mathbf{A}-\mathbf{B}\|_F^2 \leq mn/2$. This implies that $\|\mathbf{A}-\mathbf{B}_r\|_F^2 \leq (mn - \|\mathbf{A}-\mathbf{B}_r\|_F^2)$, where $\mathbf{B}_r = \operatorname{argmin}_{\text{rank}(\mathbf{B}_r)=r}\|\mathbf{A}-\mathbf{B}_r\|_F^2$. This observation implies that for any matrix $\mathbf{B}$ satisfying $\|\mathbf{A}-\mathbf{B}\|_F^2 \leq (1+\varepsilon)\cdot\|\mathbf{A}-\mathbf{B}_r\|_F^2$,

$$\begin{aligned}(mn - \|\mathbf{A}-\mathbf{B}_r\|_F^2) - (mn - \|\mathbf{A}-\mathbf{B}\|_F^2) &= \|\mathbf{A}-\mathbf{B}\|_F^2 - \|\mathbf{A}-\mathbf{B}_r\|_F^2\\ &\leq \varepsilon\|\mathbf{A}-\mathbf{B}_r\|_F^2 \leq \varepsilon(mn - \|\mathbf{A}-\mathbf{B}_r\|_F^2).\end{aligned}$$

1.2 Binary k-means

The special case of Binary Constrained Clustering where no constraints are imposed on the centers of the clusters is Binary k-Means.

Binary k-Means
Input: A set $X \subseteq \{0,1\}^m$ of n vectors and a positive integer k.
Task: Find a set $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\} \subseteq \{0,1\}^m$ minimizing the sum $\sum_{\mathbf{x}\in X} d_H(\mathbf{x}, C)$.

Equivalently, in Binary k-Means, we seek to partition a set of binary vectors X into k clusters $\{X_1, \ldots, X_k\}$ such that after we assign to each cluster its mean, which is a binary vector $\mathbf{c}_i$ (not necessarily from X) closest to X_i, then the sum $\sum_{i=1}^k \sum_{\mathbf{x}\in X_i} d_H(\mathbf{c}_i, \mathbf{x})$ is minimum.

Of course, Binary Constrained Clustering generalizes Binary k-Means: For given instance (X,k) of Binary k-Means by taking sets R_i, $1 \leq i \leq m$, consisting of all possible k-tuples $\{0,1\}^k$, we construct in time $O(n+m+2^k)$ an instance $(X,k,\mathcal{R})$ of Binary Constrained Clustering equivalent to (X,k). Note that, since all the sets R_i are the same, it is sufficient to keep just one copy of the set for the instance $(X,k,\mathcal{R})$. That is, any $(1+\varepsilon)$-approximation to one instance is also a $(1+\varepsilon)$-approximation to another. Theorems 1 and 2 imply the following corollaries:

Corollary 3. *There is an algorithm that for a given instance of* Binary k-Means *and $\varepsilon > 0$ in time $(\frac{1}{\varepsilon})^{O(\frac{k^4}{\varepsilon^2}\log\frac{1}{\varepsilon})}\cdot n\cdot m$ outputs a $(1+\varepsilon)$-approximate solution with probability at least $(1-\frac{1}{e})$.*

Corollary 4. *There is a deterministic algorithm that for a given instance of* Binary k-Means *and $\varepsilon > 0$ in time $m \cdot n^{O(\frac{k^2}{\varepsilon^2}\log\frac{1}{\varepsilon})}$ outputs a $(1+\varepsilon)$-approximate solution.*

1.3 Variants of Binary Clustering

Theorems 1 and 2 can be used for many other variants of binary clustering. Let us briefly mention some other clustering problems that fit in our framework.

For example, the following generalization of binary clustering can be formulated as Binary Constrained Clustering. Here the centers of clusters are linear subspaces of bounded dimension r. (For $r=1$ this is Binary k-Means and for $k=1$ this is Low GF(2)-Rank Approximation.) More precisely, in Binary Projective Clustering, we are given a set $X \subseteq \{0,1\}^m$ of n vectors and positive integers k and r. The task is to find a family of r-dimensional linear subspaces $C = \{C_1, \ldots, C_k\}$ over GF(2) minimizing the sum

$$\sum_{\mathbf{x}\in X} d_H\left(\mathbf{x}, \cup_{i=1}^k C_i\right).$$

To see that Binary Projective Clustering is the special case of Binary Constrained Clustering, we observe that the condition that C_i is an r-dimensional subspace over GF(2) can be encoded (as in Lemma 1) by 2^r constraints. For completeness, we state the following lemma and a proof sketch of it is given in Section 9:

Lemma 3. *For any instance* (X, k, r) *of* Binary Projective Clustering, *one can construct in time* $O(m + n + 2^{k \cdot r})$ *an instance* $(X, k' = 2^{k \cdot r}, \mathcal{R})$ *of* Binary Constrained Clustering *such that any* α*-approximate solution* C *of* $(X, k', \mathcal{R})$ *is also an* α*-approximate solution of* (X, k, r)*, and vice versa.*

Similar arguments hold also for the variant of Binary Projective Clustering when instead of r-dimensional subspaces, we use r-flats (r-dimensional affine subspaces).

In Correlative k-Bicluster Editing, we are given a bipartite graph, and the task is to change the minimum number of adjacencies such that the resulting graph is a disjoint union of at most k complete bipartite graphs [3]. This is the special case of Binary Constrained Clustering where X is the set of column vectors of the bipartite adjacency matrix and each constrain R_i consists of k-tuples and each of the k-tuples contains exactly one element 1 and all the other elements are 0. Another problem that can be reduced to Binary Constrained Clustering is the following variant of the Biclustering problem [40]. Here, for matrix $\mathbf{A}$ and positive integers k, r, we want to find a binary $m \times n$-matrix $\mathbf{B}$ with at most r pairwise-distinct rows and k pairwise-distinct columns such that $\|\mathbf{A} - \mathbf{B}\|_F^2$ is minimum.

1.4 Previous Work

Low-rank binary matrix approximation. Low-rank matrix approximation is a fundamental and extremely well-studied problem. When the measure of the similarity between $\mathbf{A}$ and $\mathbf{B}$ is the Frobenius norm of the matrix $\mathbf{A} - \mathbf{B}$, the rank-r approximation (for any r) of matrix $\mathbf{A}$ can be efficiently found via the singular value decomposition (SVD). This is an extremely well-studied problem and we refer to surveys and books [22, 27, 39] for an overview of this topic. However, SVD is not guaranteed to find an optimal solution in the case when additional structural constraints on the low-rank approximation matrix $\mathbf{B}$ (like being non-negative or binary) are imposed. In fact, most of the variants of low-rank approximation with additional constraints are NP-hard.

For a long time the predominant approaches for solving such low-rank approximation problems with NP-hard constraints were either heuristic methods based on convex relaxations or optimization methods. Recently, there has been considerable interest in the rigorous analysis of such problems [4, 11, 32, 35].

Low GF(2)-Rank Approximation arises naturally in applications involving binary data sets and serve as important tools in dimension reduction for high-dimensional data sets with binary attributes (see References [12, 18, 21, 24, 34, 36, 41] for further information). Due to the numerous applications of low-rank binary matrix approximation problems, various heuristic algorithms for these problems could be found in the literature [14, 20, 21, 24, 36].

When it concerns a rigorous analysis of algorithms for Low GF(2)-Rank Approximation, the previous results include the following: Gillis and Vavasis [15] and Dan et al. [12] have shown that Low GF(2)-Rank Approximation is NP-complete for every $r \geq 1$. A subset of the authors studied parameterized algorithms for Low GF(2)-Rank Approximation in Reference [13].

The first approximation algorithm for Low GF(2)-Rank Approximation is due to Shen et al. [36] who gave a 2-approximation algorithm for the special case of $r = 1$. Shen et al. [36] formulated the rank-one problem as Integer Linear Programming and proved that its relaxation gives a 2-approximation. They also observed that the efficiency of their algorithm can be improved by reducing the linear program to the Max-Flow problem. Jiang et al. [21] found a much simpler algorithm by observing that for the rank-one case, simply selecting the best column of the input matrix yields a 2-approximation. Bringmann et al. [9] developed a 2-approximation algorithm for $r = 1$ that runs in sublinear time. Thus, even for the special case $r = 1$, no polynomial time approximation scheme was known prior to our work.

For rank $r > 1$, Dan et al. [12] have shown that a $(r/2 + 1 + \frac{r}{2(2^r-1)})$-approximate solution can be formed from r columns of the input matrix $\mathbf{A}$. Hence, by trying all possible r columns of $\mathbf{A}$, we can obtain $r/2 + 1 + \frac{r}{2(2^r-1)}$-approximation in time $n^{O(r)}$. Even the existence of a linear time algorithm with a constant-factor approximation for $r > 1$ was open.

Low Boolean-Rank Approximation in the case of $r = 1$ coincides with Low GF(2)-Rank Approximation. Thus, by the results of Gillis and Vavasis [15] and Dan et al. [12] Low Boolean-Rank Approximation is NP-complete already for $r = 1$. 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. This follows from the well-known relation between the Boolean rank and covering edges of a bipartite graph by bicliques [17]. Thus, for fixed r, the problem is solvable in time $2^{2^{O(r)}}(nm)^{O(1)}$ [13, 16] and unless Exponential Time Hypothesis (ETH) fails, it cannot be solved in time $2^{2^{o(r)}}(nm)^{O(1)}$ [10].

There is a large body of work on Low Boolean-Rank Approximation, 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 such as word-document data is binary or even Boolean in nature, the 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., References [7, 8, 12, 26, 28, 29, 37]. In the literature the problem appears under different names such as Discrete Basis Problem [28] or Minimal Noise Role Mining Problem [26, 30, 38].

Since for $r = 1$ Low Boolean-Rank Approximation is equivalent to Low GF(2)-Rank Approximation, the 2-approximation algorithm for Low GF(2)-Rank Approximation in the case of $r = 1$ is also a 2-approximation algorithm for Low Boolean-Rank Approximation. For rank $r > 1$, Dan et al. [12] described a procedure that produces a $2^{r-1} + 1$-approximate solution to the problem.

Let us note that independently Ban et al. [6] announced a very similar algorithmic result. To compare with our result, their algorithm is for matrices over any finite field but takes a slightly worse running time $(\frac{1}{\varepsilon})^{2^{O(r)}/\varepsilon^2} n^{1+o(1)} \cdot m$, where $o(1) = (\log\log n)^{1.1}/\log n$.

Binary k-Means. This problem was introduced by Kleinberg, Papadimitriou, and Raghavan [23] as one of the examples of segmentation problems. Ostrovsky and Rabani [33] gave a randomized PTAS for Binary k-Means. In other words they show that for any $\gamma > 0$ and $0 < \varepsilon < 1/8$ there is an algorithm finding a $(1 + 8\varepsilon)^2$-approximate solution with probability at least $1 - n^{-\gamma}$. The running time of the algorithm of Ostrovsky and Rabani is $n^{f(\varepsilon,k)}$ for some function f.

For the dual maximization problem, where one wants to maximize $nm - \sum_{i=1}^{k} \sum_{\mathbf{x}\in X_i} d_H(\mathbf{c}_i, \mathbf{x})$, a significantly faster approximation is known. Alon and Sudakov [2] gave a randomized EPTAS. For a fixed k and $\varepsilon > 0$ the running time of the $(1 - \varepsilon)$-approximation algorithm of Alon and Sudakov is linear in the input length.

Binary k-Means can be seen as a discrete variant of the well-known k-Means Clustering. This problem has been studied thoroughly, particularly in the areas of computational geometry and machine learning. We refer to References [1, 5, 25] for further references to the works on k-Means Clustering. In particular, the ideas from the algorithm for k-Means Clustering of Kumar et al. [25] form the starting point of our algorithm for Binary Constrained Clustering.

The comparison of our results with the previous work is summarized in Table 1.

1.5 Our Approach

Sampling lemma and deterministic algorithm. Our algorithms are based on Sampling Lemma (Lemma 4). Suppose we have a relation $R \subseteq \{0,1\}^k$ and a weight tuple $\mathbf{w} = (w_1, \ldots, w_k)$, where $w_i \geq 0$ for all $i \in \{1, \ldots, k\}$. Then Sampling Lemma says that for any $\varepsilon > 0$, there is a

Table 1. Comparison of Our Results with Previously Known Results

Problem	Our results		Previous results	
	Approx	Runtime	Approx	Runtime
Low GF(2)-Rank Approximation	$(1+\varepsilon)$	$f(r,\varepsilon)\cdot n\cdot m$	2 (for $r=1$)	$n^{O(1)}$ [9, 21, 36]
			$(\frac{r}{2}+1+\frac{r}{2(2^r-1)})$	$n^{O(r)}$[12]
Low Boolean Rank Approximation	$(1+\varepsilon)$	$f(r,\varepsilon)\cdot n\cdot m$	2 (for $r=1$)	$n^{O(1)}$ [9, 21, 36]
			$2^{r-1}+1$	$n^{O(2^r)}m$ [12]
Binary k-Means	$(1+\varepsilon)$	$f(k,\varepsilon)\cdot n\cdot m$	$(1+\varepsilon)$	$n^{f(\varepsilon,k)}$ [33]
Binary Projective Clustering	$(1+\varepsilon)$	$f(k,r,\varepsilon)\cdot n\cdot m$	–	–

Recently and independently Ban et al. [6] obtained the same approximation ration for Low Boolean Rank, Low GF(2)-Rank, and Binary k-Means. To compare with our result, their algorithm is for matrices over any finite field but has a worse running time $f(r,\varepsilon)\cdot n\cdot m\cdot \log^{2^r} n$. For Binary Projective Clustering, we are not aware of any known approximation algorithms.

constant $r=\Theta(\frac{k}{\varepsilon^2}\log\frac{1}{\varepsilon})$ such that for any tuple $\mathbf{p}=(p_1,\ldots,p_k)$, $0\le p_i\le 1$, r random samples from Bernoulli distribution $B(p_i)$ for each $i\in\{1,\ldots,k\}$ give a *good estimate* of the minimum weighted by $\mathbf{w}$ distance of $\mathbf{p}$ from the tuples in R. For more details, we refer to Lemma 4. We believe that Sampling Lemma, which proves that random samples of constant size provide a good estimate to minimum weighted distance of probability distributions, will be of independent interest.

With a small additional work our sampling lemma implies a deterministic PTAS for Binary Constrained Clustering. Let $J=(X,k,\mathcal{R}=\{R_1,\ldots,R_m\})$ be an instance of Binary Constrained Clustering and $\varepsilon>0$. Let $C=\{\mathbf{c}_1,\ldots,\mathbf{c}_k\}$ be an optimum solution to J. Let $X_1\uplus X_2\ldots\uplus X_k$ be a partition of X into cluster corresponding to C, that is such that $\sum_{\mathbf{x}\in X} d_H(\mathbf{x},C)=\sum_{i=1}^{k}\sum_{\mathbf{x}\in X_i} d_H(\mathbf{x},\mathbf{c}_i)$. By Sampling Lemma, for constant $r=\Theta(\frac{k}{\varepsilon^2}\log\frac{1}{\varepsilon})$ and weight $w_i=|X_i|$, sampling r vectors uniformly at random chosen with repetition from X_i for each $i\in\{1,\ldots,k\}$, produces a small instance of the problem whose solution is a $(1+\varepsilon)$-approximate solution to J (see Lemma 5). Thus, to find an approximate solution, we brute-force over all choices of $w_1,\ldots,w_k\in[n]$ and k-sized families of r-sized (multi)sets. For each choice, we find a solution and then return the best one. This almost immediately brings us to the deterministic PTAS claimed in Theorem 1. However, to obtain EPTAS with linear running time, we need several additional ideas.

Linear time algorithm (Theorem 2). The general idea of our linear time algorithm for Binary Constrained Clustering is inspired by the algorithm of Kumar et al. [25]. Very informally, the algorithm of Kumar et al. [25] for k-Means Clustering is based on repeated sampling and does the following: For any (optimum) solution, there is a cluster of size at least $\frac{1}{k}$-th of the number of input vectors. Then when we sample a constant number of vectors from the input uniformly at random, with a good probability, the sampled vectors will be from the largest cluster. Moreover, if we sample sufficiently many (but still constant) number of vectors, they not only will belong to the largest cluster with a good probability, but taking the mean of the sample as the center of the whole cluster in the solution, we obtain a vector "close to optimum." This procedure succeeds if the size of the largest cluster is a large fraction of the number of vectors we used to sample from. Then the main idea behind the algorithm of Kumar et al. is to assign vectors at a small distance from the guessed center vectors to their clusters. Moreover, once some vectors are assigned to clusters, the next largest cluster will be a constant (depending on k and ε) fraction the size of the yet-unassigned vectors. With the right choice of parameters, it is possible to show that with a good probability this procedure will be a good approximation to the optimum solution.

On a very superficial level, we want to implement a similar procedure: iteratively sample, identify centers from samples, assign some of the unassigned vectors to the centers, then sample again, identify centers, and so on. Unfortunately, it is not that simple. The main difficulty is that in Binary Constrained Clustering, even though we could guess vectors from the largest cluster, we cannot select a center vector for this cluster, because the centers of "future" clusters should satisfy constraints from $\mathcal{R}$—selection of one center could influence the "future" in a very unpredicted way. Since we cannot select a good center, we cannot assign vectors to the cluster, and thus we cannot guarantee that sampling will find the next cluster. The whole procedure just falls apart!

Surprisingly, the sampling idea still works for Binary Constrained Clustering, but we have to be more careful. The main idea behind our approach is that if we sample all "big" clusters simultaneously and assign centers to these clusters such that the assigned centers "partially" satisfy $\mathcal{R}$, then with a good probability this choice does not mess up the solution much. After sampling vectors from all big clusters, we

(i) find centers for the clusters sampled simultaneously, and
(ii) make these centers to be a subset of our final solution.

The proof that (i) works correctly is based on Sampling Lemma (Lemma 6). To show that we can perform (ii), we prove that even after finding "approximately close centers" for the big clusters, there exist centers for small clusters that together with the already found centers form a *good* approximate solution (i.e, they obey the relation $\mathcal{R}$ and the cost of the corresponding solution is small). As far as we succeed in finding with a good probability a subset of "good" center vectors, we assign some of the remaining input vectors to the clusters around the centers. For the remaining vectors, we proceed iteratively.

To implement this approach to work in linear time, we do the following: Let us remind that in each iteration of the algorithm, after identifying some centroid vectors, we assign the remaining vectors that are close to the identified centroids to the clusters around them. In fact, we show that if the number of such vectors (vectors that are close to already found centers) is at most one half of the remaining input vectors, then there exists at least one cluster (whose center is yet to be computed) that contains a constant fraction of the remaining vectors. In the other case, we know that the number of vectors assigned to already identified clusters is at least one half. This leads us to the following recurrence relation:

$$T(n,k) = T\left(\frac{n}{2},k\right) + cT(n,k-k') + c'n \cdot m,$$

for $n,k \geq 1$, and $T(n,0) = T(0,k) = 1$, where c and c' are constants depending on k and ε, and $k' \geq 1$. The above recurrence solves to $f(k,\varepsilon) \cdot n \cdot m$ for some function f. However, there is one more complication. To apply Sampling Lemma, we need to know the weights w_i or the sizes of the optimum clusters in X. Thus, to compute optimum cluster centers from the samples, we have to guess the values of w_i, but this is too costly if we target the linear running time. Instead, we know that if the sizes of large clusters are *comparable* and if know them *approximately*, then we could compute approximate cluster centers in linear time (see Lemma 6).

Organization of the article. The remaining part of the article is organized as follows: In Section 2, we give notations, definitions, and some known results that we use throughout the article. In Section 3, we give notations related to Binary Constrained Clustering. In Section 4, we prove the Sampling Lemma, which we use to design both the PTAS and the linear time randomized approximation scheme for Binary Constrained Clustering. Then, in Section 5, we use Sampling Lemma to design a deterministic PTAS for the problem. The subsequent sections are building towards obtaining the linear time approximation scheme for the problem. In Section 9, we provide

the proofs that EPTASs for Low Boolean-Rank Approximation and Low GF(2)-Rank Approximation follows from Theorem 2.

2 PRELIMINARIES

We use $\mathbb{N}$ to denote the set $\{1, 2, \ldots\}$. For an integer $n \in \mathbb{N}$, we use $[n]$ as a shorthand for $\{1, \ldots, n\}$. For a set U and a non-negative integer i, 2^U and $\binom{U}{i}$ denote the set of subsets of U and set of i sized subsets of U, respectively. For a tuple $b = (b_1, \ldots, b_k) \in \{0,1\}^k$ and an index $i \in \{1, \ldots, k\}$, $b[i]$ denotes the ith entry of t, i.e, $b[i] = b_i$. We use log to denote the logarithm with base 2.

In the course of our algorithm, we will be constructing a solution iteratively. When we find a set of vectors $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_r\}$, $r < k$, which will be part of a solution, these vectors should satisfy relations in $\mathcal{R}$. Thus, we have to guarantee that for some index set $I \subset \{1, \ldots, k\}$ of size r, the set of vectors C satisfies the part of $\mathcal{R}$ "projected" on I. More precisely,

Definition 2 (Projection of $\mathcal{R}$ on I, $\mathrm{proj}_I(\mathcal{R})$). Let $R \subseteq \{0,1\}^k$ be a relation and $I = \{i_1, \ldots, i_r\} \subseteq \{1, \ldots, k\}$ be a subset of indices, where $i_1 < i_2 < \cdots < i_r$. We say that a relation $R' \subseteq \{0,1\}^r$ is a *projection of R on I*, denoted by $\mathrm{proj}_I(R)$, if R' is a set of r-tuples from $\{0,1\}^r$ such that $u = (u_1, \ldots, u_r) \in \mathrm{proj}_I(R)$ if and only if there exists $t \in R$ such that $t[i_j] = u_j$ for all $j \in \{1, \ldots, r\}$. In other words, the tuples of $\mathrm{proj}_I(R)$ are obtained from tuples of R by leaving only the entries with coordinates from I. For a family $\mathcal{R} = \{R_1, \ldots, R_m\}$ of relations, where $R_i \subseteq \{0,1\}^k$, we use $\mathrm{proj}_I(\mathcal{R})$ to denote the family $\{\mathrm{proj}_I(R_1), \ldots, \mathrm{proj}_I(R_m)\}$.

Thus, a set of vectors $\mathbf{c}_1, \ldots, \mathbf{c}_r \in \{0,1\}^m$ satisfies $\mathrm{proj}_I(\mathcal{R})$ if and only if for every $\ell \in \{1, \ldots, m\}$ there exists $\mathbf{t} \in R_\ell$ such that for every $j \in \{1, \ldots, r\}$, $\mathbf{c}_\ell[j] = \mathbf{t}[i_j]$, where $I = \{i_1, \ldots, i_r\}$ and $i_1 < i_2 < \cdots < i_r$. As far as we fix a part of the solution $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_r\}$ and index set I of size r, such that C satisfies $\mathrm{proj}_I(\mathcal{R})$, we can reduce the family of relations $\mathcal{R}$ by deleting from each relation $R_i \in \mathcal{R}$ all k-tuples not compatible with C and I. More precisely, for every $1 \leq i \leq m$, we can leave only k-tuples whose projections on I are equal to $(\mathbf{c}_1[i], \ldots, \mathbf{c}_r[i])$. Let the reduced family of relations be $\mathcal{R}|_{(I,C)}$. Then in every solution S extending C, the set of vectors $S \setminus C$ should satisfy the projection of $\mathcal{R}|_{(I,C)}$ on $\bar{I} = \{1, \ldots, k\} \setminus I$. This brings us to the following definitions:

Definition 3 (Reducing Relations $\mathcal{R}$ to $\mathcal{R}|_{(I,C)}$). Let $R \subseteq \{0,1\}^k$ be a relation and $I = \{i_1, \ldots, i_r\} \subseteq \{1, \ldots, k\}$ be a subset of indices, where $i_1 < i_2 < \cdots < i_r$, and let $u = (u_1, \ldots, u_r) \in \mathrm{proj}_I(R)$ be an r-tuple. We say that relation $R' \subseteq R$ is *obtained from R subject to I and u* and write $R' = R|_{(I,u)}$, if

$$R' = \{t \in R \mid t[i_j] = u_j \text{ for all } j \in \{1, \ldots, r\}\}.$$

For a set of vectors $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_r\}$, set $I \subseteq \{1, \ldots, k\}$ of size r, and a family of relations $\mathcal{R} = \{R_1, \ldots, R_m\}$, we denote by $\mathcal{R}|_{(I,C)}$ the family of relations $\{R'_1, \ldots, R'_m\}$, where $R'_i = R_i|_{(I,(\mathbf{c}_1[i], \ldots, \mathbf{c}_r[i]))}$ for all $i \in [m]$.

Definition 4 ($\mathcal{R}(I,C)$: Projection of $\mathcal{R}|_{(I,C)}$ on $\bar{I}$). For a relation $R \subseteq \{0,1\}^k$, a subset of indices $I \subseteq \{1, \ldots, k\}$ of size r, and an r-tuple $u = (u_1, \ldots, u_r) \in \mathrm{proj}_I(R)$, we use $R(I,u)$ to denote the projection of $R|_{(I,u)}$ on $\bar{I} = \{1, \ldots, k\} \setminus I$.

For a family $\mathcal{R} = \{R_1, \ldots, R_m\}$ of relations, a set of $r \leq k$ vectors $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_r\}$ from $\{0,1\}^m$, and an r-sized set of indices $I \subseteq \{1, \ldots, k\}$, we use $\mathcal{R}(I,C)$ to denote the family $\{R'_1, \ldots, R'_m\}$, where $R'_i = R_i(I, t_i) = \mathrm{proj}_{\bar{I}}(R_i|_{(I,t_i)})$ and $t_i = (\mathbf{c}_1[i], \ldots, \mathbf{c}_r[i])$ for all $i \in [m]$.

In other words, $R(I,u)$ consists of all $(k-r)$-tuples v, such that "merging" of u and v results in a k-tuple from R.

We also use $\mathbf{0}$ and $\mathbf{1}$ to denote the vectors with all entries equal to 0 and 1, respectively, where the dimension of the vectors will be clear from the context. For a vector $\mathbf{x} \in \{0,1\}^m$ and a set

$X \subset \{0,1\}^m$, we use $d_H(\mathbf{x}, C)$ to denote the minimum Hamming distance (the number of different coordinates) between $\mathbf{x}$ and vectors in C. For two sets $X, Y \subset \{0,1\}^m$, we define

$$\text{cost}(X, Y) = \sum_{\mathbf{x} \in X} d_H(\mathbf{x}, Y).$$

For a vector $\mathbf{x} \in \{0,1\}^m$ and an integer $\ell > 0$, we use $\mathcal{B}(\mathbf{x}, \ell)$ to denote the open ball of radius ℓ centered at $\mathbf{x}$; that is, the set of vectors in $\{0,1\}^m$ at a Hamming distance less than ℓ from $\mathbf{x}$.

Probability. In the analysis of our algorithm, we will be using well-known tail inequalities like the Markov's and Hoeffding's inequalities [19, 31].

Proposition 2.1 (Markov's Inequality). *Let X be a non-negative random variable and $a > 0$. Then*

$$\Pr(X \geq a \cdot \mathbf{E}[X]) \leq \frac{1}{a}.$$

Proposition 2.2 (Hoeffding's Inequality). *Let $X_1, \ldots, X_n$ be independent random variables such that each X_i is strictly bounded by the intervals $[a_i, b_i]$. Let $X = \sum_{i=1}^{n} X_i$ and $t > 0$. Then*

$$\Pr(X - \mathbf{E}[X] \geq t) \leq e^{\left(-\frac{2t^2}{\sum_{i \in [n]} (b_i - a_i)^2}\right)}.$$

3 NOTATIONS RELATED TO BINARY CONSTRAINED CLUSTERING

Let $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ be an instance of Binary Constrained Clustering and $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ be a solution to J; that is, a set of vectors satisfying $\mathcal{R}$. Then the cost of C is $\text{cost}(X, C)$. Given a set C, there is a natural way we can partition the set of vectors in X into k sets $X_1 \uplus \cdots \uplus X_k$ such that

$$\text{cost}(X, C) = \sum_{i=1}^{k} \text{cost}(X_i, \{\mathbf{c}_i\}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i).$$

Thus, for each vector $\mathbf{x}$ in X_i, the closest to $\mathbf{x}$ vector from C is $\mathbf{c}_i$. We call such partition *clustering of X induced by C* and refer to sets $X_1, \ldots, X_k$ as to *clusters corresponding to C*.

We use $\text{OPT}(J)$ to denote the optimal solution to J. That is,

$$\text{OPT}(J) = \min\{\text{cost}(X, C) \mid \langle C, \mathcal{R} \rangle\}.$$

Note that in the definition of a vector set C satisfiying relations $\mathcal{R}$, we require that the size of C is k. We also need a relaxed notion for vector sets of size smaller than k to satisfy a part of $\mathcal{R}$.

Definition 5 (Vectors Respecting $\mathcal{R}$). Let $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_i\} \subseteq \{0,1\}^m$ be a set of binary vectors, where $i \leq k$, we say that C respects $\mathcal{R}$ if there is an index set $I \subseteq \{1, \ldots, k\}$ such that $< C, \text{proj}_I(\mathcal{R}) >$; that is, C satisfies $\text{proj}_I(\mathcal{R})$. In other words, C is a solution to $(X, i, \text{proj}_I(\mathcal{R}))$.

Notice that given a set C' of $i \leq k$ vectors that respects $\mathcal{R}$, one can extend it to a set C in time linear in the size of J such that C satisfies $\mathcal{R}$. Thus, C is a (maybe non-optimal) solution to J such that $\text{cost}(X, C) \leq \text{cost}(X, C')$. We will use this observation in several places and thus state it as a proposition.

Proposition 3.1. *Let $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ be an instance of* Binary Constrained Clustering *and $C' = \{\mathbf{c}_1, \ldots, \mathbf{c}_i\} \subseteq \{0,1\}^m$ for $i \leq k$ be a set of vectors respecting $\mathcal{R}$. Then there is linear time algorithm that finds a solution C to J such that* $\text{cost}(X, C) \leq \text{cost}(X, C')$.

Definition 6. Let $J = (X, k, \mathcal{R})$ be an instance of Binary Constrained Clustering. For $i \in \{1, \ldots, k\}$, we define

$$\text{OPT}_i(J) = \min\{\text{cost}(X, C\} : |C| = i \text{ and } C \text{ respects } \mathcal{R}\}.$$

An equivalent way of defining $\mathsf{OPT}_i(J)$ is

$$\mathsf{OPT}_i(J) = \min\{\mathsf{OPT}(X, i, \mathrm{proj}_I(\mathcal{R})) : I \subseteq \{1, \ldots, k\} \text{ and } |I| = i\}.$$

Notice that

$$\mathsf{OPT}_1(J) \geq \mathsf{OPT}_2(J) \geq \cdots \geq \mathsf{OPT}_k(J) = \mathsf{OPT}(J).$$

4 SAMPLING PROBABILITY DISTRIBUTIONS

One of the main ingredients of our algorithms is the lemma about sampling of specific probability distributions.

Informally, Sampling Lemma proves the following: Suppose we have k bins numbered by $\{1, \ldots, k\}$. Each bin contains a certain amount of balls, each ball is labelled either by 1 or by 0. Each bin is assigned a non-negative weight w_i. For example, it can be the number of balls in the bin, but generally the lemma works for any weight. Let p_i be the ratio of balls labelled by 1 in the bin i to the total number of balls in this bin. We want to solve the following problem: Given a relation $R \subseteq \{0,1\}^k$ (that is a set of binary k-tuples), we want to find the values $u_i \in \{0,1\}$, $1 \leq i \leq k$, minimizing the value

$$\sum_{i=1}^{k} w_i |u_i - p_i|, \tag{1}$$

and such that the k-tuple $\mathbf{u} = (u_1, \ldots, u_k)$ belongs to R; that is, $\mathbf{u}$ is equal to one of the k-tuples of R. Without constraints, the values u_i minimizing Equation (1) are selected by the majority rule: if bin i contains more 1s than 0s (that is, $p_i > 1/2$), we put $u_i = 1$, otherwise, we put $u_i = 0$. However, with the constraints the situation changes and now the weights of the bins come into play. We still can solve this problem by trying all k-tuples from R and selecting the one minimizing the sum. Since all k-tuples are binary, we make at most 2^k tries in total.

However, for the purposes of our algorithm, we need to know how to optimize Equation (1) in the case when the values p_i are not known to us. We know the weight of each bin but we cannot look inside and count the number of balls with 1s and 0s. We are allowed to observe some random balls from each bin but each sample is costly. Then Sampling Lemma asserts that for every ε and k, we can select a constant r depending on ε and k only such that sampling only r balls from each bin can be used for a good approximation of Equation (1). A bit more precisely, for each sample, we compute the value q_i, the ratio of balls labelled by 1 from this sample to r, and find a k-tuple $\mathbf{v} = (v_1, \ldots, v_k)$ belongs to R and minimizing

$$\sum_{i=1}^{k} w_i |v_i - q_i|. \tag{2}$$

Let OPT be the minimum value in Equation (1) subject to R. Then Sampling Lemma states that

$$\mathbf{E}\left[\sum_{i=1}^{k} w_i |v_i - p_i|\right] \leq (1+\varepsilon)OPT.$$

Thus, the random variable $\mathbf{v}$ brings us to a pretty good estimation in Equation (1) and to find a good approximation for Equation (1) it suffices to find the best fit to Equation (2) subject to R.

To state the lemma, we use the following notations: For a real p between 0 and 1, we will denote by $B(p)$ the Bernoulli distribution that assigns probability p to 1 and $1-p$ to 0. We will write $X \sim B(p)$ to denote that X is a random variable with distribution $B(p)$.

Definition 7 (Weighted Distance $d^{\mathbf{w}}$). For two k-tuples $\mathbf{u} = (u_1, \ldots, u_k)$ and $\mathbf{v} = (v_1, \ldots, v_k)$ over reals and k-tuple $\mathbf{w} = (w_1, \ldots, w_k)$ with $w_i \geq 0$, the *distance from* $\mathbf{u}$ *to* $\mathbf{v}$ *weighted by* $\mathbf{w}$ is defined

as

$$d^{\mathbf{w}}(\mathbf{u},\mathbf{v}) = \sum_{i=1}^{k} w_i|u_i - v_i|.$$

Lemma 4 (Sampling Lemma). *There exists $c > 0$ such that for every $\varepsilon > 0$, positive integers k and $r \geq c \cdot \frac{k}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon}$, k-tuples $\mathbf{p} = (p_1, \ldots, p_k)$ with $0 \leq p_i \leq 1$, and $\mathbf{w} = (w_1, \ldots, w_k)$ with $0 \leq w_i$, and relation $R \subseteq \{0,1\}^k$, the following is satisfied:*

For every $1 \leq i \leq k$ and $1 \leq j \leq r$, let $X_i^j \sim B(p_i)$, and let $\mathbf{Q} = (Q_1, \ldots, Q_k)$ be the k-tuple of random variables, where $Q_i = \frac{1}{r}\sum_{j=1}^{r} X_i^j$. Let d_{min} be the minimum distance weighted by $\mathbf{w}$ from $\mathbf{p}$ to a k-tuple from R. Let $\mathbf{q}$ be a k-tuple from R within the minimum weighted by $\mathbf{w}$ distance to $\mathbf{Q}$—that is, $\mathbf{q} = \operatorname{argmin}_{\mathbf{x}\in R} d^{\mathbf{w}}(\mathbf{x},\mathbf{Q})$—and let $D = d^{\mathbf{w}}(\mathbf{q},\mathbf{p})$. Then $\mathbf{E}[D] \leq (1+\varepsilon)d_{min}$.

Proof. Let $\mathbf{u} = \operatorname{argmin}_{\mathbf{x}\in R} d^{\mathbf{w}}(\mathbf{x},\mathbf{p})$. Then $d_{min} = d^{\mathbf{w}}(\mathbf{u},\mathbf{p})$. Let R_{small} be the set of all tuples $\mathbf{v} \in R$ such that $d^{\mathbf{w}}(\mathbf{v},\mathbf{p}) \leq (1+\frac{\varepsilon}{2})d_{min}$. Let $R_{big} = R \setminus R_{small}$. We will prove the following claim:

Claim 1. *For every $\mathbf{v} \in R_{big}$,*

$$\Pr(d^{\mathbf{w}}(\mathbf{v},\mathbf{Q}) \leq d^{\mathbf{w}}(\mathbf{u},\mathbf{Q})) \leq \frac{d_{min}}{d^{\mathbf{w}}(\mathbf{v},\mathbf{p})} \cdot \frac{\varepsilon}{2^{k+1}}.$$

Assuming Claim 1, we complete the proof of the lemma:

$$\begin{aligned}
\mathbf{E}[D] &= \sum_{v\in R_{small}} d^{\mathbf{w}}(\mathbf{v},\mathbf{p}) \cdot \Pr(\mathbf{q}=\mathbf{v}) + \sum_{v\in R_{big}} d^{\mathbf{w}}(\mathbf{v},\mathbf{p}) \cdot \Pr(\mathbf{q}=\mathbf{v}) \\
&\leq d_{min}\left(1+\frac{\varepsilon}{2}\right) + \sum_{v\in R_{big}} d^{\mathbf{w}}(\mathbf{v},\mathbf{p}) \cdot \Pr(d^{\mathbf{w}}(\mathbf{v},\mathbf{Q}) \leq d^{\mathbf{w}}(\mathbf{u},\mathbf{Q})) \\
&\leq d_{min}\left(1+\frac{\varepsilon}{2}\right) + \sum_{v\in R_{big}} d^{\mathbf{w}}(\mathbf{v},\mathbf{p}) \cdot \frac{d_{min}}{d^{\mathbf{w}}(\mathbf{v},\mathbf{p})} \cdot \frac{\varepsilon}{2^{k+1}} \\
&\leq d_{min}\left(1+\frac{\varepsilon}{2}\right) + d_{min} \cdot \frac{\varepsilon}{2} \\
&\leq d_{min}(1+\varepsilon).
\end{aligned}$$

Hence, all that remains to prove the lemma is to prove Claim 1.

Proof of Claim 1. We will assume without loss of generality that $\varepsilon \leq \frac{1}{10}$. By renaming 0 to 1 and vice versa at the coordinates i where $u_i = 1$, we may assume that $\mathbf{u} = \mathbf{0}$. Thus, $d_{min} = d^{\mathbf{w}}(\mathbf{0},\mathbf{p})$. We may now rewrite the statement of the claim as:

$$\Pr(d^{\mathbf{w}}(\mathbf{v},\mathbf{Q}) \leq d^{\mathbf{w}}(\mathbf{0},\mathbf{Q})) \leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{d^{\mathbf{w}}(\mathbf{v},\mathbf{p})} \cdot \frac{\varepsilon}{2^{k+1}}. \tag{3}$$

Consider now the weight k-tuple $\mathbf{w}' = (w_1', \ldots, w_k')$ where $w_i' = w_i$ if $v_i = 1$ and $w_i' = 0$ if $v_i = 0$. We have that $\Pr(d^{\mathbf{w}}(\mathbf{v},\mathbf{Q}) \leq d^{\mathbf{w}}(\mathbf{0},\mathbf{Q})) = \Pr(d^{\mathbf{w}'}(\mathbf{v},\mathbf{Q}) \leq d^{\mathbf{w}'}(\mathbf{0},\mathbf{Q}))$, and that $\frac{d^{\mathbf{w}'}(\mathbf{0},\mathbf{p})}{d^{\mathbf{w}'}(\mathbf{v},\mathbf{p})} \leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{d^{\mathbf{w}}(\mathbf{v},\mathbf{p})}$. Hence, to prove Equation (3), it is sufficient to prove

$$\Pr(d^{\mathbf{w}'}(\mathbf{v},\mathbf{Q}) \leq d^{\mathbf{w}'}(\mathbf{0},\mathbf{Q})) \leq \frac{d^{\mathbf{w}'}(\mathbf{0},\mathbf{p})}{d^{\mathbf{w}'}(\mathbf{v},\mathbf{p})} \cdot \frac{\varepsilon}{2^{k+1}}.$$

In other words, it is sufficient to prove Equation (3) under the additional assumption that $w_i = 0$ whenever $v_i = 0$. Under this assumption, we have that $d^{\mathbf{w}}(\mathbf{v},\mathbf{Q}) = d^{\mathbf{w}}(\mathbf{1},\mathbf{Q})$, and that

$d^{\mathbf{w}}(\mathbf{v},\mathbf{p}) = d^{\mathbf{w}}(\mathbf{1},\mathbf{p})$. Thus, it is suffices to prove that $d^{\mathbf{w}}(\mathbf{1},\mathbf{p}) \geq (1+\frac{\varepsilon}{2}) \cdot d^{\mathbf{w}}(\mathbf{0},\mathbf{p})$ implies that

$$\Pr(d^{\mathbf{w}}(\mathbf{1},\mathbf{Q}) \leq d^{\mathbf{w}}(\mathbf{0},\mathbf{Q})) \leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{d^{\mathbf{w}}(\mathbf{1},\mathbf{p})} \cdot \frac{\varepsilon}{2^{k+1}}. \tag{4}$$

Let $w^* = \sum_{i=1}^{k} w_i$. We have that $d^{\mathbf{w}}(\mathbf{0},\mathbf{p}) + d^{\mathbf{w}}(\mathbf{1},\mathbf{p}) = w^*$. Thus, $d^{\mathbf{w}}(\mathbf{1},\mathbf{p}) \geq (1+\frac{\varepsilon}{2}) \cdot d^{\mathbf{w}}(\mathbf{0},\mathbf{p})$ implies that $d^{\mathbf{w}}(\mathbf{0},\mathbf{p}) \leq \frac{w^*}{2+\frac{\varepsilon}{2}}$. We also have that $d^{\mathbf{w}}(\mathbf{1},\mathbf{Q}) + d^{\mathbf{w}}(\mathbf{0},\mathbf{Q}) = w^*$, and therefore $d^{\mathbf{w}}(\mathbf{1},\mathbf{Q}) \leq d^{\mathbf{w}}(\mathbf{0},\mathbf{Q})$ if and only if $d^{\mathbf{w}}(\mathbf{0},\mathbf{Q}) \geq \frac{w^*}{2}$. Furthermore, $d^{\mathbf{w}}(\mathbf{1},\mathbf{p}) \leq w^*$. Hence, to prove the claim (in particular, Equation (4)) it is sufficient to show that $d^{\mathbf{w}}(\mathbf{0},\mathbf{p}) \leq \frac{w^*}{2+\frac{\varepsilon}{2}}$ implies

$$\Pr\left(d^{\mathbf{w}}(\mathbf{0},\mathbf{Q}) \geq \frac{w^*}{2}\right) \leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{w^*} \cdot \frac{\varepsilon}{2^{k+1}}. \tag{5}$$

We now prove Equation (5) distinguishing between two cases.

Case 1, $d^{\mathbf{w}}(\mathbf{0},\mathbf{p}) > \frac{w^*}{128k}$.

We have that $d^{\mathbf{w}}(\mathbf{0},\mathbf{Q}) = \sum_{i=1}^{k}\sum_{j=1}^{r} \frac{w_i}{r} X_i^j$. Thus, $d^{\mathbf{w}}(\mathbf{0},\mathbf{Q})$ is the sum of kr independent random variables, grouped into groups of size r, where all variables in group i take value $\frac{w_i}{r}$ with probability p_i and value 0 with probability $1-p_i$. It follows that $\mathbf{E}[d^{\mathbf{w}}(\mathbf{0},\mathbf{Q})] = d^{\mathbf{w}}(\mathbf{0},\mathbf{p}) \leq \frac{w^*}{2+\frac{\varepsilon}{2}}$. Hence, $\frac{w^*}{2} - \mathbf{E}[d^{\mathbf{w}}(\mathbf{0},\mathbf{Q})] \geq \frac{w^*}{2} - \frac{w^*}{2+\frac{\varepsilon}{2}} > \frac{\varepsilon w^*}{10}$, where the last inequality follows from the assumption that $\varepsilon \leq \frac{1}{10}$. Thus, we may use Proposition 2.2 to upper bound $\Pr(d^{\mathbf{w}}(\mathbf{0},\mathbf{Q}) \geq \frac{w^*}{2})$.

$$\begin{aligned}
\Pr\left(d^{\mathbf{w}}(\mathbf{0},\mathbf{Q}) \geq \frac{w^*}{2}\right) &\leq \Pr\left(d^{\mathbf{w}}(\mathbf{0},\mathbf{Q}) - \mathbf{E}[d^{\mathbf{w}}(\mathbf{0},\mathbf{Q})] > \frac{\varepsilon w^*}{10}\right) \\
&\leq \exp\left(-\frac{2\varepsilon^2 (w^*)^2}{100\Sigma_{i=1}^{k}\Sigma_{j=1}^{r}\left(\frac{w_i}{r}\right)^2}\right) \\
&\leq \exp\left(-\frac{\varepsilon^2 r}{50}\right) \\
&\leq \frac{w^*}{128k} \cdot \frac{128k}{w^*} \cdot \frac{\varepsilon}{128k \cdot 2^{k+1}} \\
&\leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{w^*} \cdot \frac{\varepsilon}{2^{k+1}}.
\end{aligned}$$

Here the second transition is by Proposition 2.2, while the fourth is by the choice of $r = \Omega(\frac{k}{\varepsilon^2} \cdot \log\frac{1}{\varepsilon})$.

Case 2, $d^{\mathbf{w}}(\mathbf{0},\mathbf{p}) \leq \frac{w^*}{128k}$.

For every i such that $w_i \geq \frac{w^*}{4k}$, we have $p_i \leq \frac{1}{32}$. Since Q_i is binomially distributed, we have that

$$\Pr\left(Q_i \geq \frac{1}{4}\right) = \sum_{t=\lceil \frac{r}{4} \rceil}^{r} \binom{r}{t} p_i^t (1-p_i)^{r-t} \leq 2^r \cdot p_i^{\frac{r}{4}} \leq p_i \cdot 2^r \cdot \left(\frac{1}{32}\right)^{\frac{r}{4}-1} \leq p_i \cdot 2^{-\frac{r}{4}-5} \leq \frac{p_i}{4k^2} \cdot \frac{\varepsilon}{2^{k+1}}.$$

Here the last inequality follows from the fact that $r = \Omega(\frac{k}{\varepsilon^2} \cdot \log\frac{1}{\varepsilon})$.

For every i such that $w_i \geq \frac{w^*}{4k}$, we have that $p_i \leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{w^*} \cdot 4k$, since otherwise $w_i p_i > d^{\mathbf{w}}(\mathbf{0},\mathbf{p})$, a contradiction. Thus, for every such i, we have

$$\Pr\left(Q_i \geq \frac{1}{4}\right) \leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{w^* k} \cdot \frac{\varepsilon}{2^{k+1}}.$$

By the union bound, we have that

$$\Pr\left(\sum_{i\,:\,w_i \geq \frac{w^*}{4k}} w_i Q_i \geq \frac{w^*}{4}\right) \leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{w^*} \cdot \frac{\varepsilon}{2^{k+1}}.$$

Since

$$\sum_{i:w_i < \frac{w^*}{4k}} w_i Q_i < \frac{w^*}{4}$$

(with probability 1), it follows that

$$\Pr\left(d^{\mathbf{w}}(\mathbf{0},\mathbf{Q}) \geq \frac{w^*}{2}\right) \leq \frac{d^{\mathbf{w}}(\mathbf{0},\mathbf{p})}{w^*} \cdot \frac{\varepsilon}{2^{k+1}}. \qquad \square$$

This proves the claim, and completes the proof of Lemma 4. $\square$

5 WARM-UP: DETERMINISTIC PTAS

As a warm-up, let us show how Sampling Lemma can be used to obtain a deterministic PTAS for Binary Constrained Clustering. Towards that, we need the following definition:

Definition 8. Let $k, m \in \mathbb{N}$ and $\mathcal{R} = \{R_1, \ldots, R_m\}$ be a family of relations, where $R_i \subseteq \{0,1\}^k$ for each $i \in \{1, \ldots, m\}$. Let $\{S_1, \ldots, S_k\}$ be a family of multisets of vectors from $\{0,1\}^m$ and $w_1, \ldots, w_k \in \mathbb{R}_{\geq 0}$. For a vector (multi)set $B \subseteq \{0,1\}^m$, let $z^{(i)}(B)$ be the number of vectors in B with the ith entry equal to 0 and let $d^{(i)}(B) = |B| - z^{(i)}(B)$ be the number of vectors in B with the ith entry equal to 1. Then $\text{best}_{\mathcal{R}}(S_1, \ldots, S_k, w_1, \ldots, w_k)$ is a set of vectors $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ satisfying $\mathcal{R}$, which is defined as follows: For $i \in \{1, \ldots, m\}$ and a k-tuple $\mathbf{b} = (b_1, \ldots, b_k) \in R_i$, let

$$f_i(b_1, \ldots, b_k) = \sum_{j \in I_b} w_j \cdot z^{(i)}(S_j) + \sum_{j \in [k] \setminus I_b} w_j \cdot d^{(i)}(S_j),$$

where $I_b = \{j \in \{1, \ldots, k\} : b_j = 1\}$. The set $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ is such that $(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i]) \in R_i$ and $f_i(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i]) = \min_{\mathbf{b} \in R_i} f_i(\mathbf{b})$, $1 \leq i \leq m$.

Lemma 5. *Let $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ of* Binary Constrained Clustering, *$\varepsilon > 0$, and $r = \Theta(\frac{k}{\varepsilon^2} \log \frac{1}{\varepsilon})$ is the constant defined in Lemma 4. Then there exist $w_1, \ldots, w_k \in [n]$ and a family $\{S'_1, \ldots, S'_k\}$ of r sized multisets of vectors from X, such that*

$$\text{cost}(X, \text{best}_{\mathcal{R}}(S'_1, \ldots, S'_k, w_1, \ldots, w_k)) \leq (1+\varepsilon)\text{OPT}(J).$$

Proof. Let $C^* = \{\mathbf{c}^*_1, \ldots, \mathbf{c}^*_k\}$ be an optimal solution to J with corresponding clusters $P_1, \ldots, P_k$. That is, $\text{OPT}(J) = \text{cost}(X, C^*) = \sum_{i=1}^k \text{cost}(P_i, \{\mathbf{c}^*_i\})$. For each $i \in \{1, \ldots, k\}$, we set $w_i = |P_i|$. For each $i \in \{1, \ldots, k\}$, we define a multiset S_i of r vectors, where each vector in S_i is chosen uniformly at random with repetition from P_i. To prove the lemma it is enough to prove that

$$\mathbf{E}[\text{cost}(X, \text{best}_{\mathcal{R}}(S_1, \ldots, S_k, w_1, \ldots, w_k))] \leq (1+\varepsilon)\text{OPT}(J). \tag{6}$$

Recall the definition of functions f_i, $1 \leq i \leq m$ (see Definition 8). For $i \in \{1, \ldots, m\}$ and a k-tuple $\mathbf{b} = (b_1, \ldots, b_k) \in R_i$,

$$f_i(b_1, \ldots, b_k) = \sum_{j \in I_b} w_j \cdot z^{(i)}(S_j) + \sum_{j \in [k] \setminus I_b} w_j \cdot d^{(i)}(S_j), \tag{7}$$

where $I_b = \{j \in \{1, \ldots, k\} : b_j = 1\}$. The set $\text{best}_{\mathcal{R}}(S_1, \ldots, S_k, w_1, \ldots, w_k) = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ is such that $(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i]) \in R_i$ and $f_i(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i]) = \min_{\mathbf{b} \in R_i} f_i(\mathbf{b})$, $1 \leq i \leq m$.

Notice that $\text{best}_{\mathcal{R}}(S_1, \ldots, S_k, w_1, \ldots, w_k) = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ is a random variable. We define a random variable $Y = \sum_{i=1}^k \text{cost}(P_i, \{\mathbf{c}_i\})$. To prove Equation (6), it is enough to prove that

$\mathbf{E}[Y] \leq (1+\varepsilon)\mathrm{OPT}(J)$. We define functions g_i for all $i \in [m]$, which denotes the cost of each element in the relation R_i with respect to the partition $P_1 \uplus \ldots \uplus P_k$ of X. Formally, for each $\mathbf{b} = (b_1, \ldots, b_k) \in R_i$, we put

$$g_i(\mathbf{b}) = \sum_{j \in I_b} z^{(i)}(P_j) + \sum_{j \in [k] \setminus I_b} d^{(i)}(P_j), \tag{8}$$

where $I_b = \{j \in \{1, \ldots, k\} : b_j = 1\}$. Let $V_i = g_i(\mathbf{c}_1^*[i], \ldots, \mathbf{c}_k^*[i])$. Notice that $\mathrm{OPT}(J) = \sum_{i=1}^m V_i$. Let $Y_i = g_i(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i])$, $1 \leq i \leq m$. By Equation (8) and the definition of Y, we have that $Y = \sum_{i=1}^m Y_i$. By the linearity of expectation, we have that $\mathbf{E}[Y] = \sum_{i=1}^m \mathbf{E}[Y_i]$. Thus, to prove that $\mathbf{E}[Y] \leq (1+\varepsilon)\mathrm{OPT}(J)$, it is sufficient to prove that $\mathbf{E}[Y_i] \leq (1+\varepsilon)V_i$ for every $i \in [m]$.

Claim 2. *For every $i \in [m]$, $\mathbf{E}[Y_i] \leq (1+\varepsilon)V_i$.*

Proof. Fix an index $i \in [m]$. Let $z_j = z^{(i)}(P_j)$ and $d_j = d^{(i)}(P_j)$. Thus, z_j (d_j) is the number of vectors from P_j whose ith coordinate is 0 (1). Let $n_j = |P_j|$ and $p_j = \frac{d_j}{n_j}$, $j \in \{1, \ldots, k\}$. Since each vector from S_j is equally likely to be any vector in P_j, for every vector $v \in S_j$,

$$\Pr(v[i] = 1) = \frac{d_j}{n_j} = p_j \qquad \text{and} \qquad \Pr(v[i] = 0) = \frac{z_j}{n_j} = 1 - p_j. \tag{9}$$

Let $\mathbf{p} = (p_1, \ldots, p_k)$. We define k-tuple $\mathbf{w} = (w_1, \ldots, w_k)$ and claim that for every $\mathbf{b} = (b_1, \ldots, b_k) \in R_i$, $d^{\mathbf{w}}(\mathbf{b}, \mathbf{p}) = g_i(\mathbf{b})$. Indeed,

$$\begin{aligned} d^{\mathbf{w}}(\mathbf{b}, \mathbf{p}) &= \sum_{j=1}^{k} w_j |b_j - p_j| \\ &= \sum_{j \in I_b} w_j (1 - p_j) + \sum_{j \in [k] \setminus I_b} w_j \cdot p_j \\ &= \sum_{j \in I_b} z_j + \sum_{j \in [k] \setminus I_b} d_j \qquad \text{(Because } w_j = |P_j| = n_j\text{)} \\ &= g_i(\mathbf{b}). \end{aligned} \tag{10}$$

For each set $S_j = \{v_1, \ldots, v_r\}$, $1 \leq j \leq k$, and $1 \leq q \leq r$, we define random variable X_j^q, which is 1 when $v_q[i] = 1$ and 0 otherwise. By Equation (9), we have that $X_j^q \sim B(p_j)$ for all $1 \leq j \leq k$ and $1 \leq q \leq r$. Let $\mathbf{Q} = (Q_1, \ldots Q_k)$ be the k-tuple of random variables, where $Q_j = \frac{1}{r} \sum_{q=1}^r X_j^q$. From the definitions of $z^{(i)}(S_j)$, $d^{(i)}(S_j)$ and X_j^q, we have that

$$z^{(i)}(S_j) = \sum_{q=1}^{r} \left(1 - X_j^q\right) \quad \text{and} \quad d^{(i)}(S_j) = \sum_{q=1}^{r} X_j^q. \tag{11}$$

For $\mathbf{b} \in R_i$, we express $d^{\mathbf{w}}(\mathbf{b}, \mathbf{Q})$ in terms of $f_i(\mathbf{b})$.

$$\begin{aligned} d^{\mathbf{w}}(\mathbf{b}, \mathbf{Q}) &= \sum_{j=1}^{k} w_j |b_i - Q_j| \\ &= \sum_{j \in I_b} w_j (1 - Q_j) + \sum_{j \in [k] \setminus I_b} w_j \cdot Q_j \\ &= \sum_{j \in I_b} w_j \cdot \left(1 - \frac{1}{r} \sum_{q=1}^{r} X_j^q\right) + \sum_{j \in [k] \setminus I_b} w_j \cdot \left(\frac{1}{r} \sum_{q=1}^{r} X_j^q\right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{r}\left(\sum_{j\in I_b} w_j \cdot \left(\sum_{q=1}^{r}\left(1-X_j^q\right)\right) + \sum_{j\in[k]\setminus I_b} w_j \cdot \left(\sum_{q=1}^{r} X_j^q\right)\right)\\
&= \frac{1}{r}\left(\sum_{j\in I_b} w_j \cdot z^{(i)}(S_j) + \sum_{j\in[k]\setminus I_b} w_j \cdot d^{(i)}(S_j)\right) \qquad \text{(by (11))}\\
&= \frac{1}{r} f_i(\mathbf{b}). \qquad \text{(by (7))} \qquad (12)
\end{aligned}
$$

Let

$$\mathbf{q} = \operatorname*{argmin}_{\mathbf{x}\in R_i} d^{\mathbf{w}}(\mathbf{x}, \mathbf{Q}).$$

By Equation (12) and by the definition of the vector set $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, we have that $\mathbf{q} = (\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i])$.

We also define k-tuple

$$\mathbf{u} = \operatorname*{argmin}_{\mathbf{x}\in R_i} d^{\mathbf{w}}(\mathbf{x}, \mathbf{p}).$$

This implies that

$$d^{\mathbf{w}}(\mathbf{u}, \mathbf{p}) = g_i(\mathbf{c}_1^*[i], \ldots, \mathbf{c}_k^*[k]) = V_i. \qquad (13)$$

Thus, the minimum weighted by $\mathbf{w}$ distance d_{min} from $\mathbf{p}$ to a k-tuple from R_i is equal to $d^{\mathbf{w}}(\mathbf{u}, \mathbf{p})$. Let D be the random variable that is a minimum weighted by $\mathbf{w}$ distance from $\mathbf{q}$ to $\mathbf{p}$. By Lemma 4,

$$\mathbf{E}[d^{\mathbf{w}}(\mathbf{q}, \mathbf{p})] = \mathbf{E}[D] \le (1+\varepsilon) d_{min} = (1+\varepsilon) d^{\mathbf{w}}(\mathbf{u}, \mathbf{p}). \qquad (14)$$

Finally, we upper bound $\mathbf{E}[Y_i]$.

$$
\begin{aligned}
\mathbf{E}[Y_i] &= \mathbf{E}[g_i(\mathbf{q})]\\
&= \mathbf{E}[d^{\mathbf{w}}(\mathbf{q}, \mathbf{p})] \qquad \text{(by (10))}\\
&\le (1+\varepsilon) \cdot d^{\mathbf{w}}(\mathbf{u}, \mathbf{p}) \qquad \text{(by (14))}\\
&\le (1+\varepsilon) V_i. \qquad \text{(by (13)).}
\end{aligned}
$$

This completes the proof of the claim. □

By Claim 3, the fact that $\mathrm{OPT}(J) = \sum_{i=1}^{m} V_i$, and by the linearity of expectation, we have that $\mathbf{E}[Y] \le (1+\varepsilon)\mathrm{OPT}(J)$. This completes the proof of the lemma. □

Lemma 5 implies Theorem 1, because of the following: We go over all choices of $w_1, \ldots, w_k \in \{1, \ldots, n\}$ and collections of r sized (multi)sets $\{S_1', \ldots, S_k'\}$ of vectors from X, and then compute $\mathrm{cost}(X, \mathrm{best}_{\mathcal{R}}(S_1', \ldots, S_k', w_1, \ldots, w_k))$. Then, we choose the best solution accordingly. The remaining part of the article is built towards obtaining a linear time randomized approximation scheme for Binary Constrained Clustering.

6 SAMPLING INSTANCES WITH LARGE CLUSTERS

In this section, we prove the algorithmic variant of Sampling Lemma, which will be the main engine of our randomized algorithm. Informally, the lemma says that if there is an optimal solution C for the set of vectors X such that each of the clusters corresponding to C contains a large fraction of the vectors from X, then sampling a constant number of vectors from X for each cluster is a good estimate for a good approximate solution. In fact, we need a stronger property: We want to derive a good clustering of a subset of vectors $Z \subseteq X$ that is unknown to us (a hidden subset).

Lemma 6 (Algorithmic Sampling Lemma). *Let $X \subseteq \{0,1\}^m$ be a set of n binary vectors and $Z \subseteq X$ be (an unknown) set of vectors. Let $J = (Z, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ be an instance of* Binary

Constrained Clustering. *Suppose that there exists a solution* $C^* = \{\mathbf{c}_1^*, \ldots, \mathbf{c}_k^*\}$ *(not necessarily optimal) to* J *with corresponding clusters* $P_1, \ldots, P_k$ *and* $1 \geq \beta > 0$ *such that* $|P_j| \geq n\beta$ *for all* $j \in \{1, \ldots, k\}$. *We denote the cost of* C^* *by* $V = \mathrm{cost}(Z, C^*) = \sum_{i=1}^{k} \mathrm{cost}(P_i, \{\mathbf{c}_i^*\})$.

Then there exists an algorithm $\mathcal{A}$ *with the following specifications:*

- *Input of* $\mathcal{A}$ *is* X, k, $\mathcal{R} = \{R_1, \ldots, R_m\}$, $\delta, \varepsilon > 0$, $0 < \beta \leq 1$, *and values* $w_1, \ldots, w_k$ *(promised bounds on the sizes of clusters* P_i*) such that for some constant* c, *for each* $j \in \{1, \ldots, k\}$,
$$\frac{|P_j|}{c} \leq w_j \leq \frac{(1+\delta)|P_j|}{c}.$$
- *Output of* $\mathcal{A}$ *is a solution* $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ *to* J *such that* $\sum_{i=1}^{k} \mathrm{cost}(P_i, \{\mathbf{c}_i\}) \leq (1+\varepsilon)^2(1+\delta)V$ *with probability at least* $\frac{\varepsilon \cdot \beta^{r \cdot k}}{1+\varepsilon}$, *where* $r = \Theta(\frac{k}{\varepsilon^2} \log \frac{1}{\varepsilon})$.
- $\mathcal{A}$ *runs in time* $O((\frac{k}{\varepsilon})^2 \log \frac{1}{\varepsilon} \cdot \sum_{i=1}^{m} |R_i|)$.

Proof. Let r be the constant defined for k and ε in Lemma 4. That is, $r = \Theta(\frac{k}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon})$. For a vector $\mathbf{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$, we define $I_b = \{j \in \{1, \ldots, k\} : b_j = 1\}$. For a vector set $B \subseteq X$, let $z^{(i)}(B)$ be the number of vectors in B with the ith entry equal to 0. Similarly, let $d^{(i)}(B) = |B| - z^{(i)}(B)$ be the number of vectors in B with the ith entry equal to 1.

Algorithm. The algorithm $\mathcal{A}$ is very simple. We sample k times (with possible repetitions) uniformly at random r vectors from X. Thus, we obtain k sets of vectors $S_1, \ldots, S_k$, each of the sets is of size r. Based on these samples, we output a solution $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ as follows: For each index $i \in \{1, \ldots, m\}$ and a k-tuple $\mathbf{b} = (b_1, \ldots, b_k) \in R_i$, let

$$f_i(b_1, \ldots, b_k) = \sum_{j \in I_b} w_j \cdot z^{(i)}(S_j) + \sum_{j \in [k] \setminus I_b} w_j \cdot d^{(i)}(S_j). \tag{15}$$

Then C is a set of vectors minimizing functions f_i subject to constraints in $\mathcal{R}$. More precisely, we define a vector set $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ such that $(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i]) \in R_i$ and $f_i(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i]) = \min_{\mathbf{b} \in R_i} f_i(\mathbf{b})$, for all $i \in [m]$. In other words, $C = \mathrm{best}_{\mathcal{R}}(S_1, \ldots, S_k, w_1, \ldots, w_k)$. Clearly, C is a solution to J.

Running time. We assume that input vectors X are stored in an array and that we can sample a vector u.a.r from a set of n vectors stored in an array in constant time. For each $i \in [m]$ and $\mathbf{b} \in R_i$, the computation of $f_i(\mathbf{b})$ takes time $O(r \cdot k)$. Then computations of functions $f_i(\mathbf{b})$ for all $i \in [m]$ and $b \in R_i$ require $O((\sum_{i=1}^{m} |R_i|)(\frac{k}{\varepsilon})^2 \log \frac{1}{\varepsilon})$ time. For each i, we use an array of length k to store the k-tuple from R_i, which gives the minimum of f_i computed so far during the computation. Therefore, the running time of the algorithm follows.

Correctness. Let $\mathscr{E}$ be the event that for all $j \in \{1, \ldots, k\}$, $S_j \subseteq P_j$. Since $|P_j| \geq |X| \cdot \beta$ for all $j \in \{1, \ldots, k\}$, we have that

$$\Pr(\mathscr{E}) \geq \beta^{r \cdot k}. \tag{16}$$

From now on, we assume that the event $\mathscr{E}$ happened. Therefore, we can think that each vector in S_j is chosen uniformly at random from P_j (with repetitions). Notice that the output $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ is a random variable. We define a random variable $Y = \sum_{i=1}^{k} \mathrm{cost}(P_j, \{\mathbf{c}_j\})$.

Now, we prove that $\mathbf{E}[Y \mid \mathscr{E}] \leq (1+\varepsilon)(1+\delta)V$. We define functions g_i for all $i \in [m]$, which denotes the cost of each element in the relation R_i with respect to the partition $P_1 \uplus \ldots \uplus P_k$ of Z. Formally, for each $\mathbf{b} = (b_1, \ldots, b_k) \in R_i$, we put

$$g_i(\mathbf{b}) = \sum_{j \in I_b} z^{(i)}(P_j) + \sum_{j \in [k] \setminus I_b} d^{(i)}(P_j). \tag{17}$$

Let $V_i = g_i(\mathbf{c}_1^*[i], \ldots, \mathbf{c}_k^*[i])$. Notice that $V = \sum_{i=1}^m V_i$. Let $Y_i = g_i(\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i])$, $1 \le i \le m$. By Equation (17) and the definition of Y, we have that $Y = \sum_{i=1}^m Y_i$. By the linearity of conditional expectation, we have that $\mathbf{E}[Y \mid \mathscr{E}] = \sum_{i=1}^m \mathbf{E}[Y_i \mid \mathscr{E}]$. Thus, to prove that $\mathbf{E}[Y \mid \mathscr{E}] \le (1+\varepsilon)(1+\delta)V$, it is sufficient to prove that $\mathbf{E}[Y_i \mid \mathscr{E}] \le (1+\varepsilon)(1+\delta)V_i$ for every $i \in [m]$.

Claim 3. *For every $i \in [m]$, $\mathbf{E}[Y_i \mid \mathscr{E}] \le (1+\varepsilon)(1+\delta)V_i$.*

Proof. Fix an index $i \in [m]$. Let $z_j = z^{(i)}(P_j)$ and $d_j = d^{(i)}(P_j)$. Thus, z_j (d_j) is the number of vectors from P_j whose ith coordinate is 0 (1). Let $n_j = |P_j|$ and $p_j = \frac{d_j}{n_j}$, $j \in \{1, \ldots, k\}$. Since we assume that $\mathscr{E}$ happened, each vector from S_j is equally likely to be any vector in P_j. That is, for every vector $v \in S_j$, we have that

$$\Pr(v[i] = 1 \mid \mathscr{E}) = \frac{d_j}{n_j} = p_j \qquad \text{and} \qquad \Pr(v[i] = 0 \mid \mathscr{E}) = \frac{z_j}{n_j} = 1 - p_j. \tag{18}$$

Let $\mathbf{p} = (p_1, \ldots, p_k)$. We define a k-tuple $\mathbf{y} = (n_1, \ldots, n_k)$ and claim that for every $\mathbf{b} = (b_1, \ldots, b_k) \in R_i$, $d^{\mathbf{y}}(\mathbf{b}, \mathbf{p}) = g_i(\mathbf{b})$. Indeed,

$$\begin{aligned} d^{\mathbf{y}}(\mathbf{b}, \mathbf{p}) &= \sum_{j=1}^{k} n_j |b_j - p_j| \\ &= \sum_{j \in I_b} n_j (1 - p_j) + \sum_{j \in [k] \setminus I_b} n_j \cdot p_j \\ &= \sum_{j \in I_b} z_j + \sum_{j \in [k] \setminus I_b} d_j \\ &= g_i(\mathbf{b}). \end{aligned} \tag{19}$$

For each set $S_j = \{v_1, \ldots, v_r\}$, $1 \le j \le k$ and $1 \le q \le r$, we define a random variable L_j^q, which is 1 when $v_q[i] = 1$ and 0 otherwise. Let us denote by X_j^q the random variable $L_j^q \mid \mathscr{E}$. By Equation (18), we have that $X_j^q \sim B(p_j)$ for all $1 \le j \le k$ and $1 \le q \le r$. Let $\mathbf{Q} = (Q_1, \ldots Q_k)$ be the k-tuple of random variables, where $Q_j = \frac{1}{r} \sum_{q=1}^r X_j^q$. From the definitions of $z^{(i)}(S_j)$, $d^{(i)}(S_j)$ and X_j^q, we have that

$$z^{(i)}(S_j) = \sum_{q=1}^{r} \left(1 - X_j^q\right) \quad \text{and} \quad d^{(i)}(S_j) = \sum_{q=1}^{r} X_j^q. \tag{20}$$

Let $\mathbf{w} = (w_1, \ldots, w_k)$. Next, for any $\mathbf{b} \in R_i$, we express $d^{\mathbf{w}}(\mathbf{b}, \mathbf{Q})$ in terms of $f_i(\mathbf{b})$.

$$\begin{aligned} d^{\mathbf{w}}(\mathbf{b}, \mathbf{Q}) &= \sum_{j=1}^{k} w_j |b_i - Q_j| \\ &= \sum_{j \in I_b} w_j (1 - Q_j) + \sum_{j \in [k] \setminus I_b} w_j \cdot Q_j \\ &= \sum_{j \in I_b} w_j \cdot \left(1 - \frac{1}{r} \sum_{q=1}^{r} X_j^q\right) + \sum_{j \in [k] \setminus I_b} w_j \cdot \left(\frac{1}{r} \sum_{q=1}^{r} X_j^q\right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{r}\left(\sum_{j\in I_b} w_j \cdot \left(\sum_{q=1}^{r}(1-X_j^q)\right) + \sum_{j\in[k]\setminus I_b} w_j \cdot \left(\sum_{q=1}^{r} X_j^q\right)\right)\\
&= \frac{1}{r}\left(\sum_{j\in I_b} w_j \cdot z^{(i)}(S_j) + \sum_{j\in[k]\setminus I_b} w_j \cdot d^{(i)}(S_j)\right) && \text{(by (20))}\\
&= \frac{1}{r} f_i(\mathbf{b}). && \text{(by (15))} \qquad (21)
\end{aligned}
$$

Let

$$\mathbf{q} = \operatorname*{argmin}_{\mathbf{x}\in R_i} d^{\mathbf{w}}(\mathbf{x}, \mathbf{Q}).$$

By Equation (21) and by the definition of the vector set $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, we have that $\mathbf{q} = (\mathbf{c}_1[i], \ldots, \mathbf{c}_k[i])$.

We also define k-tuples

$$\mathbf{u} = \operatorname*{argmin}_{\mathbf{x}\in R_i} d^{\mathbf{w}}(\mathbf{x}, \mathbf{p}) \text{ and } \mathbf{u}^* = \operatorname*{argmin}_{\mathbf{x}\in R_i} d^{\mathbf{y}}(\mathbf{x}, \mathbf{p}).$$

Thus, the minimum weighted by $\mathbf{w}$ distance d_{min} from $\mathbf{p}$ to a k-tuple from R_i is equal to $d^{\mathbf{w}}(\mathbf{u}, \mathbf{p})$. Let D be the random variable that is the minimum weighted by $\mathbf{w}$ distance from $\mathbf{q}$ to $\mathbf{p}$. By Lemma 4,

$$\mathbf{E}[d^{\mathbf{w}}(\mathbf{q}, \mathbf{p}) \mid \mathscr{E}] = \mathbf{E}[D] \leq (1+\epsilon) d_{min} = (1+\epsilon) d^{\mathbf{w}}(\mathbf{u}, \mathbf{p}). \qquad (22)$$

Since $(\mathbf{c}_1^*[i], \ldots, \mathbf{c}_k^*[i]) \in R_i$, and because $d^{\mathbf{y}}(\mathbf{b}, \mathbf{p}) = g_i(\mathbf{b})$ for each $\mathbf{b} \in R_i$ (by Equation (19)), we have that

$$d^{\mathbf{y}}(\mathbf{u}^*, \mathbf{p}) \leq g_i(\mathbf{c}_1^*[i], \ldots, \mathbf{c}_k^*[k]) = V_i. \qquad (23)$$

Finally, we upper bound $\mathbf{E}[Y_i \mid \mathscr{E}]$.

$$
\begin{aligned}
\mathbf{E}[Y_i \mid \mathscr{E}] &= \mathbf{E}[g_i(\mathbf{q}) \mid \mathscr{E}]\\
&= \mathbf{E}[d^{\mathbf{y}}(\mathbf{q}, \mathbf{p}) \mid \mathscr{E}] && \text{(by (19))}\\
&\leq c \cdot \mathbf{E}[d^{\mathbf{w}}(\mathbf{q}, \mathbf{p}) \mid \mathscr{E}] && \text{(because } n_j \leq c w_j \text{ for all } j \in [k])\\
&\leq c \cdot (1+\varepsilon) \cdot d^{\mathbf{w}}(\mathbf{u}, \mathbf{p}) && \text{(by (22))}\\
&\leq c \cdot (1+\varepsilon) \cdot d^{\mathbf{w}}(\mathbf{u}^*, \mathbf{p}) && \text{(by the choice of } \mathbf{u})\\
&\leq (1+\varepsilon)(1+\delta) d^{\mathbf{y}}(\mathbf{u}^*, \mathbf{p}) && \text{(because } c w_j \leq (1+\delta) n_j \text{ for all } j \in [k])\\
&\leq 0(1+\varepsilon)(1+\delta) V_i. && \text{(by (23))}
\end{aligned}
$$

This completes the proof of the claim. □

By Claim 3, the fact that $V = \sum_{i=1}^{m} V_i$, and by the linearity of expectation, we have that

$$\mathbf{E}[Y \mid \mathscr{E}] \leq (1+\varepsilon)(1+\delta) V.$$

Combined with the Markov's inequality (Proposition 2.1), this implies that

$$\Pr\left(Y \geq (1+\varepsilon)^2(1+\delta) V \;\middle|\; \mathscr{E}\right) \leq \frac{1}{1+\varepsilon}.$$

Therefore,

$$\Pr\left(Y \leq (1+\varepsilon)^2(1+\delta) V \;\middle|\; \mathscr{E}\right) \geq \frac{\varepsilon}{1+\varepsilon}. \qquad (24)$$

Finally,

$$\begin{aligned}\Pr(Y \leq (1+\varepsilon)^2(1+\delta)V) &\geq \Pr\left((Y \leq (1+\varepsilon)^2(1+\delta)V) \cap \mathscr{E}\right)\\ &= \Pr\left(Y \leq (1+\varepsilon)^2(1+\delta)V \;\middle|\; \mathscr{E}\right)\cdot \Pr(\mathscr{E})\\ &\geq \frac{\varepsilon}{1+\varepsilon}\cdot \beta^{r\cdot k}. \qquad \text{(by (24) and (16))}\end{aligned}$$

This completes the proof of the lemma. □

7 NON-IRREDUCIBLE INSTANCES AND EXTENDABLE SOLUTIONS

For the algorithm of k-means clustering, Kumar et al. [25] used the notion of *irreducible* instances. We introduce a similar notion for Binary Constrained Clustering.

7.1 Non-irreducible Instances

Definition 9. Let $J = (X, k, \mathcal{R})$ be an instance of Binary Constrained Clustering, $j \in \{2, \ldots, k\}$ and $\alpha > 0$. We say that J is (j, α)-irreducible if $\mathrm{OPT}_{j-1}(J) \geq (1+\alpha)\mathrm{OPT}_j(J)$.

The following property of irreducible instances plays an important role in our algorithm:

Lemma 7. *Let $J = (X, k, \mathcal{R})$ be an instance of* Binary Constrained Clustering. *For every $0 < \epsilon \leq 4$ and $0 < \alpha \leq \frac{\epsilon}{8k}$, the following holds: Let*

$$\widehat{k} = \begin{cases} 1, & \text{if } J \text{ is not } (i,\alpha)\text{-irreducible for all } i \in \{2,\ldots,k\},\\ \max\{i : J \text{ is } (i,\alpha)\text{-irreducible}\}, & \text{otherwise.}\end{cases}$$

Then $\mathrm{OPT}_{\widehat{k}}(J) \leq (1+\frac{\epsilon}{4})\mathrm{OPT}(J)$.

Proof. By the choice of $\widehat{k}$, for every $\widehat{k} \leq i < k$, we have that $\mathrm{OPT}_i(J) \leq (1+\alpha)\mathrm{OPT}_{i+1}(J)$. Thus, $\mathrm{OPT}_{\widehat{k}}(J) \leq (1+\alpha)^k \mathrm{OPT}_k(J) = (1+\alpha)^k \mathrm{OPT}(J)$. Since

$$\begin{aligned}(1+\alpha)^k &\leq \left(1+\frac{\epsilon}{8k}\right)^k = \sum_{i=0}^{k}\binom{k}{i}\left(\frac{\epsilon}{8k}\right)^i\\ &\leq \sum_{i=0}^{k}\left(\frac{\epsilon}{8}\right)^i = \left(1+\frac{\epsilon}{8}\sum_{i=0}^{k-1}\left(\frac{\epsilon}{8}\right)^i\right)\\ &\leq \left(1+\frac{\epsilon}{8}\sum_{i=0}^{k-1}\left(\frac{1}{2}\right)^i\right) \leq \left(1+\frac{\epsilon}{4}\right), \qquad \text{(because } \epsilon \leq 4\text{)}\end{aligned}$$

we have that $\mathrm{OPT}_{\widehat{k}}(J) \leq (1+\frac{\epsilon}{4})\mathrm{OPT}(J)$. □

Due to Proposition 3.1 and Lemma 7, to obtain a $(1+\epsilon)$-approximate solution to Binary Constrained Clustering, it is sufficient to learn how to approximate irreducible instances. Indeed, let $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_k\})$ be an instance to Binary Constrained Clustering and suppose that for $\epsilon > 0$ and $\alpha \leq \frac{\epsilon}{8k}$, instance J is not (k, α)-irreducible. Then for the index $\widehat{k}$ defined in Lemma 7, we have that $\widehat{k} < k$ and $\mathrm{OPT}_{\widehat{k}}(J) \leq (1+\frac{\epsilon}{4})\mathrm{OPT}(J)$. The definition of $\mathrm{OPT}_{\widehat{k}}(J)$, implies that

$$\mathrm{OPT}_{\widehat{k}}(J) = \min\{\mathrm{OPT}(X, \widehat{k}, \mathrm{proj}_I(\mathcal{R})) \mid I \subseteq \{1,\ldots,k\} \text{ and } |I| = \widehat{k}\}.$$

We can make a guess for the value of $\widehat{k}$ and then guess an index subset $I \subset \{1, \ldots, k\}$ of size $\widehat{k}$. For each of the $(k-1)\cdot 2^{k-1}$ guesses of $\widehat{k}$ and I, we form instance $J' = (X, \widehat{k}, \mathrm{proj}_I(\mathcal{R}))$. We know

that for at least one of our guesses, we will have a $(\widehat{k}, \alpha)$-irreducible instance J' with $\mathsf{OPT}(J') = \mathsf{OPT}_{\widehat{k}}(J)$. By Lemma 7 and Proposition 3.1, any $(1+\epsilon/4)$-approximate solution to J' is extendable in linear time to a $(1+\epsilon/4)^2$-approximate solution (and hence a $(1+\epsilon)$-approximate solution) to J. As a result, if J is not (k,α)-irreducible, then a $(1+\epsilon/4)$-approximate solution to J' will bring us to a $(1+\epsilon)$-approximate solution to J. Hence, everything boils down to the approximation of (k,α)-irreducible instances.

7.2 Extendable Solutions

By Lemma 6, if there is a solution such that the sizes of all corresponding clusters are constant fractions of the number of the input vectors, then sampling produces a good approximate solution. However, there is no guarantee that such a favorable condition will occur. To overcome this, we sample vectors for large clusters and then identify some vectors in the input that we can *safely* delete and make the next largest remaining cluster a constant fraction of the rest of the vectors. Towards that, we need the following definition:

Definition 10 (δ-extension of a Solution). Let $J = (X, k, \mathcal{R})$ be an instance of Binary Constrained Clustering and $\delta \geq 0$. Let $B \subseteq X$ and $C_1 \subseteq \{0,1\}^m$ be a set of k_1 vectors for some $k_1 \leq k$. We say that the pair (C_1, B) is *δ-extendable for the instance J* if there is a vector set $C_2 \subseteq \{0,1\}^m$ of size $k - k_1$ such that $C_1 \cup C_2$ satisfies $\mathcal{R}$ and $\mathsf{cost}(B, C_1) + \mathsf{cost}(X \setminus B, C_1 \cup C_2) \leq (1+\delta)\mathsf{OPT}(J)$. We also say that C_2 is a *δ-extension of* (C_1, B).

In particular, if (C_1, B) is δ-extendable for J, then there is a set $C \supseteq C_1$ such that C is a solution to J with cost at most $(1+\delta)\mathsf{OPT}(J)$ even when B is assigned to the clusters corresponding to the center vectors in C_1. This implies that after we find such a set C_1, it is safe to delete the vectors in B from the input set of vectors. If C_2 is a δ-extension of (C_1, B), then there exists index subset $I \subseteq \{1, \ldots, k\}$ of size $|C_1|$ such that C_1 satisfies $\mathsf{proj}_I(\mathcal{R})$ and C_2 satisfies $\mathcal{R}(I, C_1)$. The proof of the next observation follows directly from the definition of δ-extension.

Observation 7.1. *Let $J = (X, k, \mathcal{R})$ be an instance of* Binary Constrained Clustering, *$\delta' \geq \delta \geq 0$, and (C_1, B) be a δ-extendable pair for J. Then*

- *(C_1, B) is also δ'-extendable for J,*
- *if C_2 is a δ-extension of (C_1, B), then C_2 is also a δ'-extension of (C_1, B), and*
- *for each $B' \subseteq B$, the pair (C_1, B') is δ-extendable for J.*

Let (C_1, B) be a pair that is δ-extendable for J and $C \supseteq C_1$ be such that $C \setminus C_1$ is a δ-extension of (C_1, B). While the set $C \setminus C_1$ is not known to us and we do not know yet how to compute it, as we will see in the following lemma, we can proceed successfully even if only the minimum Hamming distance t between vectors in C_1 and $C \setminus C_1$ is known. We show that if we know t, then we can find a set of vectors $B' \subseteq X \setminus B$ such that B' is not only safe to delete but the number of vectors in $X \setminus (B \cup B')$ that will be assigned to clusters corresponding to C_1 (in the solution C) is at most a constant fraction of the number of vectors in $X \setminus (B \cup B')$. In other words, the number of vectors that will be assigned to clusters corresponding to $C \setminus C_1$ will be at least a constant fraction of the number of vectors in $X \setminus (B \cup B')$. This information is of crucial importance; it yields that from the remaining set of vectors at least one of the clusters assigned to $C \setminus C_1$ is large, and thus could be found by sampling.

Lemma 8. *Let $J = (X, k, \mathcal{R})$ be an instance of* Binary Constrained Clustering *and $\delta \geq 0$. Let (C_1, B), where $B \subseteq X$ and $C_1 \subseteq \{0,1\}^m$, be a δ-extendable pair for J. Let C_2 be a δ-extension of (C_1, B) and $t = \min\{d_H(\mathbf{c}, \mathbf{c}') : \mathbf{c} \in C_1, \mathbf{c}' \in C_2\}$. Let (Z_1, Z_2) be a partition of $X \setminus B$ such that*

$\text{cost}(X \setminus B, C_1 \cup C_2) = \text{cost}(Z_1, C_1) + \text{cost}(Z_2, C_2)$. *Let* $B' = \bigcup_{\mathbf{c} \in C_1} \mathcal{B}(\mathbf{c}, t/2) \cap (X \setminus B)$. *Then the following conditions hold:*

(*i*) $B' \subseteq Z_1$. *Moreover,* B' *consists of the first* $|B'|$ *vectors of* $X \setminus B$ *in the ordering according to the non-decreasing distance* $d_H(\mathbf{x}, C_1)$ *(where* $\mathbf{x} \in X \setminus B$*).*

(*ii*) $\text{cost}(X \setminus (B \cup B'), C_1 \cup C_2) = \text{cost}(Z_1 \setminus B', C_1) + \text{cost}(Z_2, C_2)$. *Moreover,* $(C_1, B \cup B')$ *is* δ*-extendable for* J *and* C_2 *is a* δ*-extension of* $(C_1, B \cup B')$.

(*iii*) *If* J *is* $(k, 5\delta')$*-irreducible for some* $\delta' \geq \delta$*, then* $|Z_2| \geq (\frac{\delta'}{1+\delta'})|X \setminus (B \cup B')|$. *If in addition,* $|B'| \leq \frac{|X \setminus B|}{2}$*, then* $|Z_2| \geq (\frac{\delta'}{2(1+\delta')})|X \setminus B|$.

Proof. We start with (*i*). Since $t = \min\{d_H(\mathbf{c}, \mathbf{c}') : \mathbf{c} \in C_1, \mathbf{c}' \in C_2\}$, for any vector $\mathbf{x}$ in $\bigcup_{\mathbf{c} \in C_1} \mathcal{B}(\mathbf{c}, t/2)$, the value $d_H(\mathbf{x}, C_2)$ is strictly greater than $t/2$. Because $\text{cost}(X \setminus B, C_1 \cup C_2) = \text{cost}(Z_1, C_1) + \text{cost}(Z_2, C_2)$, we conclude that $B' \subseteq Z_1$. Now if we order the vectors of $X \setminus B$ according to their distances to C_1 (the smallest distance comes first and ties are broken arbitrarily), then the first $|B'|$ vectors in this ordering are within distance strictly less than $t/2$ from C_1, while for any other vector $\mathbf{x} \notin B'$, $d_H(\mathbf{x}, C_1) \geq t/2$. This concludes the proof of (*i*).

Before proceeding with (*ii*) and (*iii*), we observe the following: First, since C_2 is a δ-extension of (C_1, B), we have that

$$\text{cost}(X, C_1 \cup C_2) \leq \text{cost}(B, C_1) + \text{cost}(X \setminus B, C_1 \cup C_2) \leq (1 + \delta)\text{OPT}(J).$$

By the definition of the sets Z_1 and Z_2, we have that

$$\begin{aligned} \text{cost}(B, C_1) + \text{cost}(X \setminus B, C_1 \cup C_2) &= \text{cost}(B, C_1) + \text{cost}(Z_1, C_1) + \text{cost}(Z_2, C_2) \\ &= \text{cost}(B \cup Z_1, C_1) + \text{cost}(Z_2, C_2). \end{aligned}$$

Thus,

$$\text{cost}(X, C_1 \cup C_2) \leq \text{cost}(B \cup Z_1, C_1) + \text{cost}(Z_2, C_2) \leq (1 + \delta)\text{OPT}(J). \tag{25}$$

We need one more observation and its proof follows directly from the definition of the sets Z_1 and Z_2.

Observation 7.2. *For every* $\mathbf{x} \in Z_1$, $d_H(\mathbf{x}, C_1 \cup C_2) = d_H(\mathbf{x}, C_1)$ *and for every* $\mathbf{y} \in Z_2$, $d_H(\mathbf{y}, C_1 \cup C_2) = d_H(\mathbf{y}, C_2)$.

Now, we prove condition (*ii*). First, by Observation 7.2 and the fact that $B' \subseteq Z_1$, we have that

$$\text{cost}(X \setminus (B \cup B'), C_1 \cup C_2) = \text{cost}(Z_1 \setminus B', C_1) + \text{cost}(Z_2, C_2). \tag{26}$$

To prove that $(C_1, B \cup B')$ is δ-extendable for J and that C_2 is a δ-extension of $(C_1, B \cup B')$, we note that

$$\begin{aligned} \text{cost}(B \cup B', C_1) \quad &+ \quad \text{cost}(X \setminus (B \cup B'), C_1 \cup C_2) \\ &= \quad \text{cost}(B \cup B', C_1) + \text{cost}(Z_1 \setminus B', C_1) + \text{cost}(Z_2, C_2) \qquad \text{(by (26))} \\ &\leq \quad (1 + \delta)\text{OPT}(J). \qquad \text{(by (25) and because } B' \subseteq Z_1\text{)} \end{aligned}$$

Now, we prove condition (*iii*) of the lemma. Let $C_2 = \{\mathbf{c}_1, \ldots, \mathbf{c}_\ell\}$, where $\ell = k - |C_1|$. Let $Y_1 \uplus \cdots \uplus Y_\ell$ be a partition of Z_2 such that

$$\text{cost}(Z_2, C_2) = \sum_{j=1}^{\ell} \sum_{\mathbf{y} \in Y_j} d_H(\mathbf{c}_j, \mathbf{y}).$$

Let vector $\mathbf{c} \in C_1$ and index $r \leq \ell$ be such that $t = d_H(\mathbf{c}, \mathbf{c}_r)$.

Claim 4. *If* J *is* $(k, 5\delta')$*-irreducible for some* $\delta' \geq \delta$*, then* $|Z_1 \setminus B'| \leq \frac{1}{\delta'} \cdot |Y_r|$.

Proof. For the sake of contradiction, assume that $|Z_1 \setminus B'| > \frac{1}{\delta'} \cdot |Y_r|$. This implies that $\text{cost}(Z_1 \setminus B', C_1) \geq \frac{t}{2\delta'}|Y_r|$. Thus,

$$t|Y_r| \leq 2\delta' \text{cost}(Z_1 \setminus B', C_1) \leq 2\delta' \text{cost}(Z_1, C_1). \tag{27}$$

Now, we upper bound $\text{cost}(X, C_1 \cup (C_2 \setminus \{c_r\})) - \text{cost}(X, C_1 \cup C_2)$, by considering the the change in cost when we reassign the vectors in Y_r to the cluster with center $\mathbf{c}$.

$$\begin{aligned} \text{cost}(X, C_1 \cup (C_2 \setminus \{\mathbf{c}_r\})) - \text{cost}(X, C_1 \cup C_2) &\leq \sum_{\mathbf{y} \in Y_r} (d_H(\mathbf{y}, \mathbf{c}) - d_H(\mathbf{y}, \mathbf{c}_r)) \\ &\leq \sum_{\mathbf{y} \in Y_r} d_H(\mathbf{c}, \mathbf{c}_r) \quad \text{(by the triangle inequality)} \\ &= |Y_r| \cdot t \\ &\leq 2\delta' \cdot \text{cost}(Z_1, C_1). \qquad \text{(by (27))} \end{aligned} \tag{28}$$

But then

$$\begin{aligned} \text{OPT}_{k-1}(J) &\leq \text{cost}(X, C_1 \cup (C_2 \setminus \{\mathbf{c}_r\})) \\ &\leq \text{cost}(X, C_1 \cup C_2) + 2\delta' \text{cost}(Z_1, C_1) && \text{(by (28))} \\ &\leq (1+\delta) \cdot \text{OPT}(J) + 2\delta'(1+\delta) \cdot \text{OPT}(J) && \text{(by (25))} \\ &\leq (1 + \delta + 4\delta') \cdot \text{OPT}(J) && \text{(since } \delta \leq 1\text{)} \\ &\leq (1 + 5\delta') \cdot \text{OPT}(J). && \text{(since } \delta \leq \delta'\text{)} \end{aligned}$$

This contradicts the assumption that J is $(k, 5\delta')$-irreducible. □

Since $Y_r \subseteq Z_2$, by Claim 4, we have that $|Z_1 \setminus B'| \leq \frac{1}{\delta'} \cdot |Z_2|$. The sets Z_1 and Z_2 form a partition of $X \setminus B$ and, because $B' \subseteq Z_1$, we have that $X \setminus (B \cup B') = (Z_1 \setminus B') \cup Z_2$. This implies that $|X \setminus (B \cup B')| \leq |Z_2|(1 + 1/\delta')$, and hence

$$|Z_2| \geq \frac{\delta'}{1+\delta'} \cdot |X \setminus (B \cup B')|.$$

Finally, we prove the last condition in (iii). If $|B'| \leq \frac{|X \setminus B|}{2}$, then

$$\begin{aligned} |Z_2| &\geq \frac{\delta'}{1+\delta'} \cdot |X \setminus (B \cup B')| \\ &\geq \frac{\delta'}{1+\delta'} \cdot (|X \setminus B| - |B'|) \\ &\geq \frac{\delta'}{2(1+\delta')} \cdot |X \setminus B|. \end{aligned}$$

This completes the proof of the lemma. □

Due to Lemma 8, once we have a partial solution C_1, we can identify a set B' of vectors such that the number of vectors in the largest cluster corresponding to $C \setminus C_1$ (here C is a *good* solution that contains C_1) is at least a constant fraction of the remaining vectors. This allows us to further use Lemma 6 as formally explained in Lemma 10. To make the statements in Lemma 10 easier, we make use of one of the *best*δ-extension of (C_1, B) as derived in the following lemma:

Definition 11 (Good δ-extension). For an instance $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ of Binary Constrained Clustering, $\delta \geq 0$, $B \subseteq X$ and $C_1 \subseteq \{0,1\}^m$ of size k', a δ-extension C_2 of (C_1, B) is a *good δ-extension of* (C_1, B) if there is a partition $Z_1 \uplus Z_2 = X \setminus B$ and an index set $I \subseteq \{1, \ldots, k\}$ of size k' such that

- $\mathrm{cost}(X \setminus B, C_1 \cup C_2) = \mathrm{cost}(Z_1, C_1) + \mathrm{cost}(Z_2, C_2)$,
- C_1 satisfies $\mathrm{proj}_I(\mathcal{R})$, and
- C_2 is an optimal solution to $J' = (Z_2, k - k', \mathcal{R}' = \mathcal{R}(I, C_1))$.

We will refer to J' as the C_2-optimal reduced instance.

Lemma 9. *Let $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ be an instance of* Binary Constrained Clustering *and $\delta \geq 0$. Let $B \subseteq X$ and $C_1 \subseteq \{0,1\}^m$ be a set of k' vectors for some $k' \leq k$ such that (C_1, B) is δ-extendable for J. Then there is a good δ-extension C_2 of (C_1, B).*

Proof. Among all the δ-extensions of (C_1, B), let C_2 be a δ-extension of (C_1, B) such that $\mathrm{cost}(X \setminus B, C_1 \cup C_2)$ is minimized. We prove that C_2 is the required δ-extension of (C_1, B).

Let

$$\eta = \mathrm{cost}(X \setminus B, C_1 \cup C_2).$$

Since C_2 is a δ-extension of (C_1, B), there is a set $I \subseteq \{1, \ldots, k\}$ of size k' and a partition $Z_1 \uplus Z_2$ of $X \setminus B$ such that C_1 and C_2 satisfy families of relations $\mathrm{proj}_I(\mathcal{R})$ and $\mathcal{R}(I, C_1)$ correspondingly, and

$$\eta = \mathrm{cost}(Z_1, C_1) + \mathrm{cost}(Z_2, C_2) \leq (1 + \delta)\mathrm{OPT}(J).$$

By the choice of C_2, for every δ-extension C_2' of (C_1, B), we have that $\mathrm{cost}(X \setminus B, C_1 \cup C_2') \geq \eta$. For the sake of contradiction, assume that C_2 is not an optimal solution to $(Z_2, k - k', \mathcal{R}(I, C_1))$. Let C_2^* be an optimal solution to $(Z_2, k - k', \mathcal{R}(I, C_1))$. Then $\mathrm{cost}(Z_2, C_2^*) < \mathrm{cost}(Z_2, C_2)$. Moreover, $C_1 \cup C_2^*$ satisfies $\mathcal{R}$ and hence is a solution to J. Since $\mathrm{cost}(Z_2, C_2^*) < \mathrm{cost}(Z_2, C_2)$, we conclude that

$$\mathrm{cost}(X \setminus B, C_1 \cup C_2^*) \leq \mathrm{cost}(Z_1, C_1) + \mathrm{cost}(Z_2, C_2^*) < \mathrm{cost}(Z_1, C_1) + \mathrm{cost}(Z_2, C_2) = \eta.$$

This contradicts the choice of C_2, which, in turn, completes the proof of the lemma. □

To state the next lemma, we need one more definition.

Definition 12 (Set of κ-heavy Clusters). Let $\kappa > 0$, X be a vector set, and $X_1, \ldots, X_\ell$ be a clustering of X. We say that a subset of clusters $\mathcal{X} \subseteq \{X_1, \ldots, X_\ell\}$ is a set of *κ-heavy clusters* if for every $Y \in \mathcal{X}$ and $Z \notin \mathcal{X}$, $|Y| > \kappa|Z|$.

The following lemma says that if we have an irreducible instance J with δ-extendable pair (C_1, B), then it is possible to construct a larger extendable pair by adding to C_1 a set of "approximate" centers of heavy clusters from optimal clustering of the instance J' obtained from J by "subtracting" (C_1, B).

Lemma 10. *Let $\delta \geq 0$ and $0 \leq \alpha \leq 1$. Let $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ be a $(k, 5\delta')$-irreducible instance of* Binary Constrained Clustering *for some $\delta' \geq \delta$. Let the pair (C_1, B) where $B \subseteq X$ and $C_1 \subseteq \{0,1\}^m$, $|C_1| < k$, be δ-extendable for J. Let $C_2 = \{\mathbf{c}_1^*, \ldots, \mathbf{c}_\ell^*\}$ be a good δ-extension of (C_1, B) and $J' = (Z_2, \ell = k - |C_1|, \mathcal{R}' = \mathcal{R}(I, C_1))$ be the corresponding C_2-optimal reduced instance. (The existence of such C_2 is guaranteed by Lemma 9.) Let also $X_1, \ldots, X_\ell$ be the set of clusters corresponding to the solution C_2 of J' and let $I' \subseteq \{1, \ldots, \ell\}$, $|I'| = k'$, be the set of indices of the $\frac{k}{\alpha}$-heavy clusters from this set.*

Then for every solution $C_2' = \{\mathbf{c}_i : i \in I'\}$ to $J_1' = (\bigcup_{i \in I'} X_i, k', \mathrm{proj}_{I'}(\mathcal{R}'))$ satisfying condition

$$\sum_{j \in I'} \mathrm{cost}(X_j, \{\mathbf{c}_j\}) \leq (1 + \alpha) \cdot \sum_{j \in I'} \mathrm{cost}(X_j, \{\mathbf{c}_j^*\}),$$

the pair $(C_1 \cup C_2', B)$ is $(5\delta + 4\alpha)$-extendable for J.

Proof. Because C_2 is a δ-extension of (C_1, B), we have that

$$\mathrm{cost}(B, C_1) + \mathrm{cost}(X \setminus B, C_1 \cup C_2) \leq (1 + \delta)\mathrm{OPT}(J).$$

The assumption that C_2 is a good δ-extension of (C_1, B), yields that there are $Z_1 \uplus Z_2 = X \setminus B$ and $I \in \binom{[k]}{|C_1|}$ such that

- $\text{cost}(X \setminus B, C_1 \cup C_2) = \text{cost}(Z_1, C_1) + \text{cost}(Z_2, C_2)$,
- C_1 satisfies $\text{proj}_I(\mathcal{R})$, and
- C_2 is an optimal solution to $J' = (Z_2, \ell = k - |C_1|, \mathcal{R}' = \mathcal{R}(I, C_1))$.

Hence,

$$\text{cost}(X, C_1 \cup C_2) \leq \text{cost}(B \cup Z_1, C_1) + \text{cost}(Z_2, C_2) \leq (1 + \delta)\text{OPT}(J). \tag{29}$$

For the ease of presentation, we assume that $I' = \{1, \ldots, k'\}$. Let $W_1 = \bigcup_{j=I'} X_j$ and $W_2 = Z_2 \setminus W_1$. Let $C_3 = \{\mathbf{c}_{k'+1}, \ldots, \mathbf{c}_\ell\}$ be an optimal solution to $J_2' = (W_2, \ell - k', \mathcal{R}'(I', C_2'))$. From the solution C_3 of J_2', we define a solution $C_3' = \{\mathbf{c}'_{k'+1}, \ldots, \mathbf{c}'_\ell\}$ to J_2' as follows: For each $i \in [m]$, we set

$$(\mathbf{c}'_{k'+1}[i], \ldots, \mathbf{c}'_\ell[i]) = \begin{cases} (\mathbf{c}^*_{k'+1}[i], \ldots, \mathbf{c}^*_\ell[i]), & \text{if } (\mathbf{c}_1[i], \ldots, \mathbf{c}_{k'}[i]) = (\mathbf{c}^*_1[i], \ldots, \mathbf{c}^*_{k'}[i]), \\ (\mathbf{c}_{k'+1}[i], \ldots, \mathbf{c}_\ell[i]), & \text{otherwise.} \end{cases}$$

Observation 7.3. *C_3' is a solution to J_2'.*

Proof. From the definition of C_3', we have that for any $i \in [m]$, $(\mathbf{c}'_{k'+1}[i], \ldots, \mathbf{c}'_\ell[i]) \in \mathcal{R}'(I', C_2')$. Hence, C_3' is a solution to J_2. □

Recall that we define $\mathcal{R}' = \mathcal{R}(I, C_1)$. Because $C_1 \in \text{proj}_I(\mathcal{R})$, $C_2' \in \text{proj}_{I'}(\mathcal{R}')$, and $C_3' \in \mathcal{R}'(I', C_2')$, we have that $C_1 \cup C_2' \cup C_3'$ is a solution to J. We will prove that C_3' is indeed a $(5\delta + 4\alpha)$-extension of $C_1 \cup C_2'$. Towards that, we define

$$\Delta_1 = \sum_{j \in I'} \sum_{\mathbf{x} \in X_j} d_H(\mathbf{x}, \mathbf{c}^*_j) = \sum_{j \in I'} \text{cost}(X_j, \{\mathbf{c}^*_j\}), \tag{30}$$

and

$$\Delta_2 = \sum_{j \in [\ell] \setminus I'} \sum_{\mathbf{x} \in X_j} d_H(\mathbf{x}, \mathbf{c}^*_j) = \sum_{j \in [\ell] \setminus I'} \text{cost}(X_j, \{\mathbf{c}^*_j\}). \tag{31}$$

By counting the number of mismatches at each of the coordinates of the vectors in W_1 with its corresponding center in $\{\mathbf{c}^*_1, \ldots, \mathbf{c}^*_{k'}\}$, we get that

$$\sum_{r=1}^{m} \sum_{j \in I'} \sum_{\mathbf{x} \in X_j : \mathbf{x}[r] \neq \mathbf{c}^*_j[r]} 1 = \sum_{j \in I'} \text{cost}(X_j, \{\mathbf{c}^*_j\}) = \Delta_1, \tag{32}$$

where the last equality follows from Equation (30). Since C is an optimal solution to J' with corresponding clusters $X_1, \ldots, X_\ell$, we have that $\text{OPT}(J') = \Delta_1 + \Delta_2$. Combining Equation (30) with the assumption $\sum_{j \in I'} \text{cost}(X_j, \{\mathbf{c}_j\}) \leq (1 + \alpha)(\sum_{j \in I'} \text{cost}(X_j, \{\mathbf{c}^*_j\}))$, we have that

$$\text{cost}(W_1, C_2') \leq \sum_{j \in I'} \text{cost}(X_j, \{\mathbf{c}_j\}) \leq (1 + \alpha)\Delta_1. \tag{33}$$

By arguments similar to the reasoning for Equation (32) and by Equation (33), we obtain the following:

$$\sum_{r=1}^{m} \sum_{j \in I'} \sum_{\mathbf{x} \in X_j : \mathbf{x}[r] \neq \mathbf{c}_j[r]} 1 = \sum_{j \in I'} \text{cost}(X_j, \{\mathbf{c}_j\}) = (1 + \alpha)\Delta_1. \tag{34}$$

We claim that

Claim 5. $\text{cost}(W_2, C_3') \leq \Delta_2 + (3\alpha \cdot \Delta_1)$

Before proving the claim, let us show how it concludes the proof of the lemma. By Equation (33) and Claim 5, we have that

$$\begin{aligned} \mathrm{cost}(W_1, C_2') + \mathrm{cost}(W_2, C_3') &\leq (1+\alpha)\Delta_1 + \Delta_2 + (3\alpha \cdot \Delta_1) \\ &\leq (1+4\alpha)(\Delta_1 + \Delta_2) \\ &\leq (1+4\alpha)\mathrm{cost}(Z_2, C_2), \end{aligned} \tag{35}$$

where the last inequality follows from the fact that $C_2 = \{\mathbf{c}_1^*, \ldots, \mathbf{c}_\ell^*\}$ is a solution to J' with the corresponding clusters $X_1, \ldots, X_\ell$ and Equations (30) and (31).

Now, we prove that C_3' is a $(5\delta + 4\alpha)$-extension of $(C_1 \cup C_2', B)$ for J. Towards that, we upper bound $\eta = \mathrm{cost}(B, C_1 \cup C_2') + \mathrm{cost}(X \setminus B, C_1 \cup C_2' \cup C_3')$.

$$\begin{aligned} \eta &\leq \mathrm{cost}(B, C_1) + \mathrm{cost}(Z_1, C_1) + \mathrm{cost}(W_1, C_2') + \mathrm{cost}(W_2, C_3') \\ &\leq \mathrm{cost}(B, C_1) + \mathrm{cost}(Z_1, C_1) + (1+4\alpha)\mathrm{cost}(Z_2, C_2) && \text{(by (35))} \\ &\leq (1+\delta)\mathrm{OPT}(J) + (4\alpha)\mathrm{cost}(Z_2, C_2) && \text{(by (29))} \\ &\leq (1+\delta)\mathrm{OPT}(J) + (4\alpha)(1+\delta)\mathrm{OPT}(J) && \text{(by (29))} \\ &\leq (1+5\delta+4\alpha)\mathrm{OPT}(J). && \text{(since } \alpha \leq 1\text{)} \end{aligned}$$

This, subject to the proof of Claim 5, completes the proof of the lemma. We need one more technical claim to prove Claim 5.

Claim 6. *For any $j \in [\ell] \setminus I'$,*

$$\sum_{r \in [m]: \mathbf{c}_j'[r] \neq \mathbf{c}_j^*[r]} |X_j| \leq \frac{3\alpha}{k} \cdot \Delta_1.$$

Proof. Fix $j \in [\ell] \setminus I'$. For any $r \in [m]$ such that $\mathbf{c}_j'[r] \neq \mathbf{c}_j^*[r]$, by the definition of $(\mathbf{c}_{k'+1}'[r], \ldots, \mathbf{c}_k'[r])$, we have that $(\mathbf{c}_1[r], \ldots, \mathbf{c}_{k'}[r]) \neq (\mathbf{c}_1^*[r], \ldots, \mathbf{c}_{k'}^*[r])$. That is, for any $r \in [m]$ such that $\mathbf{c}_j'[r] \neq \mathbf{c}_j^*[r]$, there is $j_r \in I'$ such that $\mathbf{c}_{j_r}[r] \neq \mathbf{c}_{j_r}^*[r]$. Define a map f from $I_j = \{r \in [m] : \mathbf{c}_j'[r] \neq \mathbf{c}_j^*[r]\}$ to I' as follows: $f(r) = j_r$, where j_r is an arbitrary index in I' such that $\mathbf{c}_{j_r}[r] \neq \mathbf{c}_{j_r}^*[r]$. We define sets

$$A[r] = \{\mathbf{x} \in X_{f(r)} \mid \mathbf{x}[r] \neq \mathbf{c}_{f(r)}[r]\},$$

and

$$B[r] = \{\mathbf{x} \in X_{f(r)} \mid \mathbf{x}[r] \neq \mathbf{c}_{f(r)}^*[r]\}.$$

Since $|X_j| < \frac{\alpha}{k}|X_{f(r)}|$, for every $r \in I_j$ (by the assumption that the clusters with the indices from I' are $\frac{k}{\alpha}$-heavy), we have that

$$|X_j| \leq \frac{\alpha}{k}|X_{f(r)}| = \frac{\alpha}{k}\left(\sum_{\mathbf{x} \in A[r]} 1 + \sum_{\mathbf{x} \in B[r]} 1\right).$$

Therefore,

$$\begin{aligned}
\sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]} |X_j| &\leq \sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]} \frac{\alpha}{k}\left(\sum_{\mathbf{x}\in A[r]} 1 + \sum_{\mathbf{x}\in B[r]} 1\right)\\
&\leq \frac{\alpha}{k}\left(\left(\sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]}\ \sum_{\mathbf{x}\in A[r]} 1\right) + \left(\sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]}\ \sum_{\mathbf{x}\in B[r]} 1\right)\right)\\
&\leq \frac{\alpha}{k}\left(\left(\sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]}\ \sum_{\mathbf{x}\in A[r]} 1\right) + \Delta_1\right) \qquad \text{(by (32))}\\
&\leq \frac{\alpha}{k}\left((1+\alpha)\Delta_1 + \Delta_1\right) \qquad \text{(by (34))}\\
&\leq \frac{3\alpha}{k}\Delta_1. \qquad \text{(since } \alpha \leq 1\text{)}
\end{aligned}$$

□

Now, we are ready to proceed with the proof of Claim 5.

Proof of Claim 5. We bound

$$\begin{aligned}
\mathrm{cost}(W_2, C'_3) &\leq \sum_{j\in[\ell]\setminus I'}\ \sum_{\mathbf{x}\in X_j} d_H(\mathbf{x}, \mathbf{c}'_j)\\
&= \sum_{j\in[\ell]\setminus I'}\ \sum_{\mathbf{x}\in X_j}\ \sum_{r=1}^{m} |\mathbf{x}[r] - \mathbf{c}'_j[r]|\\
&= \sum_{j\in[\ell]\setminus I'}\ \sum_{\mathbf{x}\in X_j}\left(\sum_{r\in[m]:\mathbf{c}'_j[r]=\mathbf{c}^*_j[r]} |\mathbf{x}[r] - \mathbf{c}'_j[r]| + \sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]} |\mathbf{x}[r] - \mathbf{c}'_j[r]|\right)\\
&\leq \sum_{j\in[\ell]\setminus I'}\ \sum_{\mathbf{x}\in X_j}\left(d_H(\mathbf{x}, \mathbf{c}^*_j) + \sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]} |\mathbf{x}[r] - \mathbf{c}'_j[r]|\right)\\
&= \Delta_2 + \sum_{j\in[\ell]\setminus I'}\ \sum_{\mathbf{x}\in X_j}\ \sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]} |\mathbf{x}[r] - \mathbf{c}'_j[r]| \qquad \text{(by (31))}\\
&= \Delta_2 + \sum_{j\in[\ell]\setminus I'}\ \sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]}\ \sum_{\mathbf{x}\in X_j} |\mathbf{x}[r] - \mathbf{c}'_j[r]|\\
&\leq \Delta_2 + \sum_{j\in[\ell]\setminus I'}\ \sum_{r\in[m]:\mathbf{c}'_j[r]\neq\mathbf{c}^*_j[r]} |X_j|\\
&\leq \Delta_2 + \sum_{j\in[\ell]\setminus I'} \frac{3\alpha}{k}\cdot\Delta_1 \qquad \text{(by Claim 6)}\\
&\leq \Delta_2 + (3\alpha\cdot\Delta_1).
\end{aligned}$$

□

The last inequality completes the proof of Claim 5, which, in turn, completes the proof of the lemma. □

8 PUTTING ALL TOGETHER: PROOF OF THEOREM 2

As we already mentioned in Section 7, the most essential part of the proof of Theorem 2 is to approximate irreducible instances. We start from the approximation algorithm for irreducible instances and then use this algorithm to prove Theorem 2.

8.1 Approximating Irreducible Instances

Now, we have sufficient ingredients to design an algorithm for outputting a $(1+\varepsilon)$-approximate solution under the assumption that the given instance is $(k, \frac{\varepsilon}{8k^*})$-irreducible for some $k^* \geq k$. More precisely, in this section, we prove the following theorem:

Theorem 3. *There is an algorithm with the following specifications:*

- *The input of the algorithm is an instance $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ of* Binary Constrained Clustering, *$\varepsilon > 0$, and $k^* \geq k$ such that J is $(k, \frac{\varepsilon}{8k^*})$-irreducible.*
- *The output of the algorithm is a solution C to J such that $\mathrm{cost}(X, C) \leq (1 + \frac{\varepsilon}{40k^*})\mathrm{OPT}(J)$ with probability at least*

$$p(k^*, k, \varepsilon) = \left(\frac{\varepsilon}{1+\varepsilon}\right)^k \cdot \left(\frac{\varepsilon^k}{\left(40k \cdot k^* \cdot (5^k - 1)\right)^{k-1} \cdot 2k \cdot (40k^* + \varepsilon)}\right)^{\frac{c' \cdot k^2}{\varepsilon^2} \log \frac{1}{\varepsilon}}.$$

- *The running time of the algorithm is $2^{O(k^2 + k\log(k^*))} \cdot n \cdot m \cdot (\frac{1}{\varepsilon})^{O(k)}$.*

First, we provide an overview of the algorithm informally without mentioning actual values of the parameters. Then, we reason about how to set different parameters that will lead to the required algorithm. And then, we give a formal proof of Theorem 3.

On a high level, our algorithm works as follows: Let $J = (X, k, \mathcal{R})$ be the input instance. We consider a tuple (C', S, δ) to be a partial solution where $S \subseteq X$, $\delta \geq 0$, and $C' \subseteq \{0,1\}^m$, $|C'| \leq k$. We say that a partial solution (C', S, δ) is a *good* partial solution, if $(C', X \setminus S)$ is δ-extendable for J; that is, C' is a set of cluster centers. Moreover C' can be extended to a $(1+\delta)$-approximate solution with $X \setminus S$ being assigned to the clusters corresponding to C'. Initially, we set our set of partial solution to be $\mathcal{P} = \{(\emptyset, X, 0)\}$. Clearly $(\emptyset, X, 0)$ is a good partial solution. At each iteration, we maintain an invariant that $\mathcal{P}$ contains a good partial solution with high probability. At any step if there is a partial solution $(C_1, S, \delta) \in \mathcal{P}$ such that $|C_1| < k$ and $S \neq \emptyset$, we do the following: First, we delete (C_1, S, δ) from $\mathcal{P}$ and then we add the following partial solutions to $\mathcal{P}$: Let $\mathbf{v}_1, \ldots, \mathbf{v}_{|S|}$ be the vectors in S ordered according to the non-decreasing order of $d_H(\mathbf{v}_i, C_1)$.

(i) We add (C_1, S', δ), where S' is the last $\lceil \frac{|S|}{2} \rceil$ vectors in the order $\mathbf{v}_1, \ldots, \mathbf{v}_{|S|}$.
(ii) For any choice of $k' \in \{1, \ldots, k - |C_1|\}$, we extend the set C_1 assuming $(C_1, X \setminus S)$ is δ-extendable (see the below paragraph for a brief explanation).

Assume that C_2 is a good δ-extension of $(C_1, X \setminus S)$ (see Definition 11). Since C_2 is a good δ-extension of $(C_1, X \setminus S)$, there exist $I \in \binom{[k]}{|C_1|}$ and a partition $Z_1 \uplus Z_2$ of S such that C_2 is an optimum solution to $J' = (Z_2, \ell = k - |C_1|, \mathcal{R}(I, C_1))$ and $\mathrm{cost}(S, C_1 \cup C_2) = \mathrm{cost}(Z_1, C_1) + \mathrm{cost}(Z_2, C_2)$ (see Lemma 9). Let B' be the set defined in Lemma 8. If $|B'| \geq \lfloor \frac{|S|}{2} \rfloor$, then by Lemma 8($ii$) and Observation 7.1, (C_1, S', δ) is δ-extendable for J (this is covered in item (i) above). Suppose the size of B' is at most $\lfloor \frac{|S|}{2} \rfloor$. Then, by Lemma 8($iii$), we know that $|Z_2|$ is a constant fraction of $|S|$. Therefore, the size of the largest cluster among the clusters corresponding to C_2 in the instance J' is also a constant fraction of $|S|$. Let $X_1, \ldots, X_\ell$ be the clusters corresponding to the solution C_2, where $\ell = |C_2|$. For the ease of presentation, assume that $|X_1| \geq |X_2| \ldots \geq |X_\ell|$. Let k' be the smallest

integer such that $|X_{k'}|$ is "much larger than" $|X_{k'+1}|$. In other words, for any $i \in \{1, \ldots, k'-1\}$, the size of X_i is at most a constant times $|X_{i+1}|$. This implies that the size of X_j is a constant fraction of $|S|$ for any $j \in \{1, \ldots, k'\}$. Therefore, we use Lemma 6 to compute an approximate set of centers C' for the clusters $X_1, \ldots, X_{k'}$. Then, we add $(C_1 \cup C', S, 5\delta + 4\alpha)$ to $\mathcal{P}$ (this case is covered in item (ii) above). The explanation for setting the parameter α and how to estimate weights w_i used in Lemma 6 is given in the next paragraph. At the end, for any tuple $(C, S, \delta) \in \mathcal{P}$, either $|C| = k$ or $S = \emptyset$. Then, we output the best solution among all the tuples in $\mathcal{P}$. The correctness of the algorithm will follow from the invariant that $\mathcal{P}$ contains a good partial solution with high probability at each iteration.

Now, we explain how to set different parameters for the algorithm, which are used in Lemmas 6, 8, and 10. We have an assumption that J is $(k, \frac{\varepsilon}{8k^*})$-irreducible. In Lemmas 8 and 10, we want J to be $(k, 5\delta')$-irreducible. So, we set $\delta' = \frac{\varepsilon}{40k^*}$. Initially, we set $\mathcal{P} = \{(\emptyset, X, 0)\}$. At each iteration, we extend a partial solution (C_1, S, δ_1) either by deleting half of vectors from S or by computing a set of center vectors using Lemma 6 and the correctness of the step (assuming $(C_1, X \setminus S)$ is δ_1-extendable) follows from Lemma 10. For the initial application of Lemma 10 (i.e., where $(C_1, S, \delta_1) = (\emptyset, X, 0)$), we have that $\delta_1 = 0$. Then after each application of Lemma 10, we get a partial solution that we expect to be $(5\delta_1 + 4\alpha)$-extendable (we fix α later). The number of times Lemma 10 is applied to get a particular solution in $\mathcal{P}$ (at the end of the algorithm) is at most k. This implies that at the end some solution in $\mathcal{P}$ is $\gamma(k)$-extendable (by Observation 7.1), where $\gamma(k)$ can be obtained from the following recurrence relation:

$$\gamma(0) = 0, \text{ and}$$
$$\gamma(k') = 5\gamma(k'-1) + 4\alpha, \text{ for } k' \in \{1, 2, \ldots, k\}.$$

The above recurrence relation solves to $\gamma(k') = (5^{k'} - 1)\alpha$ for any $k' \in \{0, 1, \ldots, k\}$. Moreover, by Lemma 10, we need $\gamma(k')$ to be at most δ' for all $k' \in \{0, 1, \ldots, k\}$. Thus, we set $\alpha = \frac{\delta'}{(5^k - 1)}$. From Lemma 6, we expect a $(1+\alpha)$-approximate solution for a restricted instance derived from J. So, we use $\varepsilon = \frac{\alpha}{7}$ and $\delta = \frac{\alpha}{7}$ in the application of Lemma 6. To apply Lemma 6, we also need the value for β and promised bounds on the sizes of the clusters for which we are seeking center vectors. Towards that, we give a detailed explanation of how to use Lemma 6. Notice that (C_1, S, δ_1) is a partial solution already computed and we assumed that $(C_1, X \setminus S)$ is δ_1-extendable for J. Let C_2 be a good δ-extension of $(C_1, X \setminus S)$. Since C_2 is a good δ-extension of $(C_1, X \setminus S)$, there exist $I \in \binom{[k]}{|C_1|}$ and a partition $Z_1 \uplus Z_2$ of S such that C_2 is an optimum solution to $J' = (Z_2, \ell = k - |C_1|, \mathcal{R}(I, C_1))$ and $\text{cost}(S, C_1 \cup C_2) = \text{cost}(Z_1, C_1) + \text{cost}(Z_2, C_2)$ (see Lemma 9). Let B' be the set defined in Lemma 8. The application of Lemma 6 is important for the partial solution (C_1, S, δ_1) only when the size of B' is at most $\frac{|S|}{2}$. Then, by Lemma 8(iii), we know that $|Z_2| \geq (\frac{\delta'}{2(1+\delta')})|S|$. Therefore, the largest cluster corresponding to the solution C_2 of J' is at least $(\frac{\delta'}{2k(1+\delta')})|S|$. Let $X_1, \cdots, X_\ell$ be the clusters corresponding to C_2, where $\ell = |C_2|$. For the ease of presentation, assume that $|X_1| \geq |X_2| \ldots \geq |X_\ell|$. Let k' be the smallest integer such that $|X_{k'}| > \frac{k}{\alpha}|X_{k'+1}|$. Therefore, we have that $|X_1| \leq \frac{k}{\alpha}|X_2| \leq \cdots \leq (\frac{k}{\alpha})^{k'-1}|X_{k'}|$. (Here the set $[k']$ plays the role of I' in Lemma 10, i.e., $[k']$ is a set of indices of $\frac{k}{\alpha}$-heavy clusters.) This implies that the size of X_j is at least $(\frac{\alpha}{k})^{k'-1}|X_1| \geq (\frac{\alpha}{k})^{k-1}\frac{\delta'}{2k(1+\delta')}|S|$ (because $\frac{\alpha}{k} \leq 1$) for any $j \in [k']$. So, we set $\beta = (\frac{\alpha}{k})^{k-1}\frac{\delta'}{k(2+\delta')}$. We set the value c in Lemma 6 to be $|X_{k'}|$. Notice that we do not have to know the value of c explicitly, but instead, we need to know the weights w_is within a promised bound. From the value of c, it is clear that for any $j \in [k']$, $\frac{|X_j|}{c} \leq (\frac{k}{\alpha})^{k'-j} \leq (\frac{k}{\alpha})^{k'-1}$. Let $h = (\frac{k}{\alpha})^{k'-1}$. Then there exists $w_j \in \{(1+\delta)^0, (1+\delta)^1 \ldots, (1+\delta)^{\log_{1+\delta} h}\}$ such that $\frac{|X_j|}{c} \leq w_j \leq \frac{(1+\delta)|X_j|}{c}$. Thus, we apply Lemma 6 for all possible values $w_1, \ldots, w_{k'} \in \{(1+\delta)^0, (1+\delta)^1 \ldots, (1+\delta)^{\log_{1+\delta} h}\}$ and extend the set C_1 in each case.

ALGORITHM 1: Algorithm for Binary Constrained Clustering assuming the input instance is $(k, \frac{\varepsilon}{8k^*})$-irreducible, where k is the number of vectors allowed in the solution, ε is the approximation factor, and $k^* \geq k$.

Input: An instance $J = (X, k, \mathcal{R} = \{R_1, \ldots, R_m\})$ of Binary Constrained Clustering and $\varepsilon > 0$.
Output: A solution $C \subseteq \{0,1\}^m$ for J.

1 $\mathcal{P} \leftarrow \{(\emptyset, X, 0)\}$
2 $\delta' \leftarrow \frac{\varepsilon}{40k^*}$, $\alpha \leftarrow \frac{\delta'}{(5^k-1)}$, $\beta \leftarrow \left(\frac{\alpha}{k}\right)^{k-1} \cdot \frac{\delta'}{2k(1+\delta')}$, $\varepsilon \leftarrow \frac{\alpha}{7}$, and $\delta \leftarrow \frac{\alpha}{7}$.
3 **for** $(C_1, S, \delta_1) \in \mathcal{P}$ *such that* $|C_1| < k$ *and* $S \neq \emptyset$ **do**
4 $\quad \mathcal{P} \leftarrow \mathcal{P} \setminus \{(C_1, S, \delta_1)\}$
5 $\quad$ Let π be a linear order of S according to the non-decreasing distance $d_H(\mathbf{v}, C_1)$, where $\mathbf{v} \in S$.
6 $\quad$ Let S' be the last $\lceil \frac{|S|}{2} \rceil$ vectors in the order π.
7 $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{(C_1, S', \delta_1)\}$
8 $\quad$ Guess $I \in \binom{[k]}{|C_1|}$ such that $(i) < C_1, \text{proj}_I(\mathcal{R}) >$ (for a proper ordering of the vectors in C_1), (ii) there is a good δ_1-extension C_2 of $(C_1, X \setminus S)$, and $(iii) < C_2, \mathcal{R}(I, C_1) >$.
9 $\quad \mathcal{R}' \leftarrow \mathcal{R}(I, C_1)$.
10 $\quad \ell \leftarrow k - |C_1|$
11 $\quad$ **for** $k' \in \{1, \ldots, k - |C_1|\}$ *and* $I' \in \binom{[\ell]}{k'}$ **do**
12 $\quad\quad h \leftarrow \left(\frac{k}{\alpha}\right)^{k'-1}$
13 $\quad\quad$ **for** $w_1, \ldots, w_{k'} \in \{(1+\delta)^0, (1+\delta)^1 \ldots, (1+\delta)^{\log_{1+\delta} h}\}$ **do**
14 $\quad\quad\quad C' \leftarrow$ the output of Algorithm $\mathcal{A}$ from Lemma 6 on input $S, k', \text{proj}_{I'}(\mathcal{R}'), \beta, \delta, \varepsilon$, and $w_1, \ldots, w_{k'}$
15 $\quad\quad\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{(C_1 \cup C', S, 5\delta_1 + 4\alpha)\}$
16 Let C be such that there is $(C, S^*, \delta^*) \in \mathcal{P}$ for some $S^* \subseteq X, \delta^* \geq 0$ and $\text{cost}(X, C) = \min\{\text{cost}(X, C') :$ there is $S' \subseteq X, \delta_1 \geq 0$ such that $(C', S', \delta_1) \in \mathcal{P}\}$.
17 **return** a solution $D \supseteq C$ using Proposition 3.1.

Now, we are ready to give the formal proof of the theorem.

Proof of Theorem 3. The pseudocode of our algorithm is given in Algorithm 1. First, we define an *iteration* of the algorithm to be the execution of one step of the **for loop** at Line 3. That is, at the beginning of an iteration, one partial solution will be deleted from $\mathcal{P}$ and later during the execution of the iteration many partial solutions will be added to $\mathcal{P}$ (see Lines 7 and 15). Next, we prove that the algorithm terminates. Notice that when there is no partial solution $(C_1, S, \delta_1) \in \mathcal{P}$ with $|C_1| < k$ and $S \neq \emptyset$, then the algorithm terminates. When there is a partial solution $(C_1, S, \delta_1) \in \mathcal{P}$ with $|C_1| < k$ and $S \neq \emptyset$, then there is an iteration of the algorithm (Line 3) where in the **for** loop, we consider (C_1, S, δ_1), delete (C_1, S, δ_1) from $\mathcal{P}$, and add many partial solutions to $\mathcal{P}$. For each such partial solution (C_1', S', δ_2), either $|C_1'| > |C_1|$ or $|S'| < |S|$. This implies that Algorithm 1 will terminate.

Now, we prove the correctness of the algorithm. That is, we prove that Algorithm 1 will return a $(1 + \frac{\varepsilon}{40k^*})$-approximate solution with probability at least $p(k^*, k, \varepsilon)$.

For convenience, we let the $0th$ iteration to be the initial assignments before any of the execution of Line 3. That is, at the end of the $0th$ iteration, we have that $\mathcal{P} = \{(\emptyset, X, 0)\}$. When we apply in Line 14 Algorithm $\mathcal{A}$ from Lemma 6 on input $S, k', \text{proj}_{I'}(\mathcal{R}')$, β, δ, ε, and $w_1, \ldots, w_{k'}$, we know that by Lemma 6, it outputs a solution with probability at least $\frac{\varepsilon \cdot \beta^{r \cdot k}}{1+\varepsilon}$, where $r = \Theta(\frac{k}{\varepsilon^2} \log \frac{1}{\varepsilon})$. Let c' be a constant such that $r = c' \cdot \frac{k}{\varepsilon^2} \log \frac{1}{\varepsilon}$.

Correctness Invariant: At the end of every iteration of the algorithm, there is a partial solution $(C, S, \delta_1) \in \mathcal{P}$ such that $(C, X \setminus S)$ is δ_1-extendable for J with probability at least $(\frac{\varepsilon\beta^r}{1+\varepsilon})^q$ where $q = |C|$.

We need the following observation to prove the correctness invariant:

Observation 8.1. *At any step of the algorithm, for any partial solution $(C, S, \delta_1) \in \mathcal{P}$, $\delta_1 \leq \delta'$.*

Proof Sketch. By induction on the number of iteration of the algorithm one can prove that at the end of each iteration, for any partial solution $(C, S, \delta_1) \in \mathcal{P}$, $\delta_1 \leq \gamma(|C|)$. Recall that $\gamma(k') = (5^{k'} - 1)\alpha$ and $\gamma(k') \leq \delta'$ for all $k' \in \{0, 1, \ldots, k\}$. Thus, the observation follows. □

Claim 7. *Correctness invariant is maintained at the end of every iteration.*

Proof. We prove the claim using induction on the number of iterations. The base case is for the $0th$ iteration. Since at the end of $0th$ iteration $(\emptyset, X, 0) \in \mathcal{P}$ and the fact that $(\emptyset, \emptyset)$ is 0-extendable for J with probability 1, the base case follows. Now, we consider the induction step. That is, consider the iteration i of the algorithm where $i > 0$. Let (C_1, S, δ^*) be the partial solution for which the iteration i corresponds to. At the beginning of iteration i, $(C_1, S, \delta^*) \in \mathcal{P}$ and by induction hypothesis there is a partial solution $(C', S', \delta_1) \in \mathcal{P}$ satisfying the properties mentioned in the correctness invariant. Notice that during the iteration i, we will delete (C_1, S, δ^*) from $\mathcal{P}$ and add many partial solutions to $\mathcal{P}$. If $(C', S', \delta_1) \neq (C_1, S, \delta^*)$, then at the end of iteration i, $(C', S', \delta_1) \in \mathcal{P}$ and the invariant follows.

So now, we assume that $(C_1, S, \delta^*) = (C', S', \delta_1)$. This implies that $(C_1, X \setminus S)$ is δ^*-extendable for J with probability at least $(\frac{\varepsilon\beta^r}{1+\varepsilon})^q$, where $q = |C_1|$. By Lemma 9, there is a good δ^*-extension C_2 of $(C_1, X \setminus S)$, a partition $Z_1 \uplus Z_2$ of $X \setminus S$, and $I \in \binom{k}{|C_1|}$ such that $C_1 \in \text{proj}_I(\mathcal{R})$ and C_2 is an optimum solution to $(Z_2, k - |C_1|, \mathcal{R}(I, C_1))$ and $\text{cost}(S, C_1 \cup C_2) = \text{cost}(Z_1, C_1) + \text{cost}(Z_2, C_2)$. Let $t = \min_{\mathbf{c} \in C_1, \mathbf{c}' \in C_2} d_H(\mathbf{c}, \mathbf{c}')$. Let $B' = \bigcup_{\mathbf{c} \in C_1} \mathcal{B}(\mathbf{c}, t'/2) \cap S$ and $S^* = S \setminus B'$. Then by Lemma 8, we get the following:

(*i*) $B' \subseteq Z_1$ and B' consists of the first $|B'|$ vectors of S in the ordering according to the non-decreasing distance $d_H(\mathbf{x}, C_1)$ where $\mathbf{x} \in S$.
(*ii*) $\text{cost}(S^*, C_1 \cup C_2) = \text{cost}(Z_1 \setminus B', C_1) + \text{cost}(Z_2, C_2)$. Moreover, $(C_1, X \setminus S^*)$ is δ^*-extendable for J and C_2 is a δ^*-extension of $(C_1, X \setminus S^*)$.
(*iii*) Since $\delta^* \leq \delta'$ (by Observation 8.1) and the fact that J is $(k, 5\delta')$-irreducible (because of our assumption), we have that $|Z_2| \geq (\frac{\delta'}{1+\delta'})|S^*|$. If $|B'| \leq \frac{|S|}{2}$, then $|Z_2| \geq (\frac{\delta'}{2(1+\delta')})|S|$.

Suppose $|B'| > \frac{|S|}{2}$. Recall the definition of S' from Line 6. Notice that $S \setminus S' \subseteq B'$. Therefore, by the assumption that $(C_1, X \setminus S)$ is δ^*-extendable for J and by Lemma 8(*ii*) and Observation 7.1, $(C_1, X \setminus S')$ is δ^*-extendable for J. Since $(C_1, X \setminus S)$ is δ^*-extendable with probability at least $(\frac{\varepsilon\beta^r}{1+\varepsilon})^q$, $(C_1, X \setminus S')$ is δ^*-extendable with probability at least $(\frac{\varepsilon\beta^r}{1+\varepsilon})^q$.

Now consider the case $|B'| \leq \frac{|S|}{2}$. We know that C_2 is an optimum solution to $(Z_2, \ell = k - |C_1|, \mathcal{R}(I, C_1))$. Let $C_2 = \{\mathbf{c}_1, \ldots, \mathbf{c}_\ell\}$ and $Y_1, \ldots Y_\ell$ be the clusters corresponding to the solution C_2 of $(Z_2, \ell = k - |C_1|, \mathcal{R}(I, C_1))$. Let π be a permutation of $[\ell]$ such that $|Y_{\pi(1)}| \geq \cdots \geq |Y_{\pi(\ell)}|$. Let $X_j = Y_{\pi(j)}$ and $\mathbf{c}^*_j = \mathbf{c}_{\pi(j)}$ for all $j \in [\ell]$. Let k' be the smallest integer in $[\ell]$ such that $|X_{k'}| > (\frac{k}{\alpha})|X_{k'+1}|$. This implies that $|X_1| \leq (\frac{k}{\alpha})|X_2| \leq \cdots \leq (\frac{k}{\alpha})^{k'-1}|X_{k'}|$. Thus, by the fact that $|X_1| \geq \frac{|Z_2|}{\ell} \geq \frac{|Z_2|}{k}$, we have that for any $j \in [k']$,

$$|X_j| \geq \left(\frac{\alpha}{k}\right)^{k'-1} \frac{|Z_2|}{k}. \tag{36}$$

Let $I' = \{\pi(j) : j \in [k']\}$. By statement (iii) above and Equation (36), for all $j \in I'$,

$$|Y_j| \geq \left(\frac{\alpha}{k}\right)^{k'-1} \cdot \frac{\delta'}{2k(1+\delta')}|S| \geq \left(\frac{\alpha}{k}\right)^{k-1} \cdot \frac{\delta' \cdot |S|}{2k(1+\delta')} \qquad \left(\text{Since } \frac{\alpha}{k} \leq 1\right). \tag{37}$$

This implies that for all $j \in I'$, $|Y_j| \geq \beta|S|$. Let $C^* = \{\mathbf{c}_j : j \in I'\}$. Let $V = \sum_{j\in I'} \text{cost}(Y_j, \mathbf{c}_j)$. Moreover, (a) $\{Y_j : j \in I'\}$ is a set of $\frac{k}{\alpha}$-heavy clusters (see Definition 12). Let $c = |Y_{k'}|$. Notice that (b) for any $j \in I'$, $\frac{Y_j}{c} \leq (\frac{k}{\alpha})^{k'-1}$. Let $\mathcal{R}' = \mathcal{R}(I, C_1)$ and J' be the instance $(Z = \bigcup_{j\in I'} Y_j, k', \text{proj}_{I'}(\mathcal{R}'))$. Now in the iteration i, consider the execution of **for loop** at Line 11 for k' and I'. Notice that for any $j \in I'$, there exists $w \in \{(1+\delta)^0, (1+\delta)^1 \ldots, (1+\delta)^{\log_{1+\delta} h}\}$, where $h = (\frac{k}{\alpha})^{k'-1}$, such that $\frac{|Y_j|}{c} \leq w \leq \frac{(1+\delta)|Y_j|}{c}$. Now consider the **for loop** at Line 13 for values $w_1, \ldots, w_{k'}$ such that $\frac{|Y_{\pi(j)}|}{c} \leq w_j \leq \frac{(1+\delta)|Y_{\pi(j)}|}{c}$ for all $j \in [k']$. Then, by Lemma 6, in Line 14, we get a set $C' = \{\mathbf{c}'_i : i \in I'\}$ of k' cluster centers such that with probability at least $\frac{\varepsilon\cdot\beta^{r\cdot k'}}{1+\varepsilon}$,

$$\sum_{i\in I'} \text{cost}(Y_i, \{\mathbf{c}'_i\}) \leq (1+\varepsilon)^2(1+\delta)V \leq (1+\alpha) \cdot \sum_{j\in I'} \text{cost}(Y_j, \mathbf{c}_j). \tag{38}$$

Therefore, $(C_1 \cup C', S, 5\delta^* + 4\alpha)$ belongs to $\mathcal{P}$ at the end of the iteration i with probability at least $(\frac{\varepsilon\cdot\beta^r}{1+\varepsilon})^{(q+k')} = (\frac{\varepsilon\cdot\beta^r}{1+\varepsilon})^{|C_1\cup C'|}$. Then by Lemma 10, $(C_1 \cup C', X \setminus S)$ is $(5\delta^* + 4\alpha)$-extendable for J and this completes the proof of the claim. □

Now for the proof of the correctness of the algorithm consider the invariant at the end of the last iteration. At the end of last iteration for all $(C, S, \delta_1) \in \mathcal{P}$, we have that either $|C| = k$ or $S = \emptyset$. Then by Claim 7, at the end of the last iteration, $\mathcal{P}$ contains a partial solution (C, S, δ_1) that is δ_1-extendable with probability at least

$$p = \left(\frac{\varepsilon\beta^r}{1+\varepsilon}\right)^k. \tag{39}$$

Then the output set $D \supseteq C$ is a $(1+\delta_1)$-approximate solution of J with probability at least p, by Proposition 3.1. By Observation 8.1, D is a $(1+\frac{\varepsilon}{40k^*})$-approximate solution of J. By substituting the values of β and r into Equation (39), we bound the value of p as follows:

$$\begin{aligned}
p &= \left(\frac{\varepsilon}{1+\varepsilon}\right)^k \cdot \left(\left(\frac{\delta'}{k(5^k-1)}\right)^{k-1} \frac{\delta'}{2k(1+\delta')}\right)^{r\cdot k} \\
&= \left(\frac{\varepsilon}{1+\varepsilon}\right)^k \cdot \left(\frac{\left(\frac{\varepsilon}{40k^*}\right)^k}{\left(k(5^k-1)\right)^{k-1} \cdot 2k \cdot (1+\frac{\varepsilon}{40k^*})}\right)^{r\cdot k} \\
&= \left(\frac{\varepsilon}{1+\varepsilon}\right)^k \cdot \left(\frac{\varepsilon^k}{\left(40k\cdot k^*\cdot(5^k-1)\right)^{k-1} \cdot 2k \cdot (40k^*+\varepsilon)}\right)^{r\cdot k} \\
&= \left(\frac{\varepsilon}{1+\varepsilon}\right)^k \cdot \left(\frac{\varepsilon^k}{\left(40k\cdot k^*\cdot(5^k-1)\right)^{k-1} \cdot 2k \cdot (40k^*+\varepsilon)}\right)^{\frac{c'\cdot k^2}{\varepsilon^2}\log\frac{1}{\varepsilon}}.
\end{aligned}$$

Running time. Now, we analyze the running time of the algorithm. Notice that in the algorithm, initially, we have $\mathcal{P} = \{(\emptyset, X, 0)\}$ and in each step, we delete a partial solution from $\mathcal{P}$ and add many partial solutions. Towards analyzing the running time, we define a node-labelled rooted tree T as follows: The root is labelled with $(\emptyset, X, 0)$. Each node of the tree is labelled with a partial solution

(C, S, δ_1), which corresponds an execution of **for loop** at Line 3. For a node labelled with (C, S, δ_1), let $S_1, \ldots, S_\ell$ be the tuples added to $\mathcal{P}$ during the iteration corresponding to (C, S, δ_1). Then the node labelled (C, S, δ_1) will have ℓ children and they are labelled with $S_1, \ldots, S_\ell$, respectively. Therefore, the number of iterations in the algorithm is equal to the number of nodes in the tree T. In each iteration (corresponding to a partial solution (C, S, δ_1)), we sort the set S of vectors according to the Hamming distance to C (see Line 5). This can be done in time $O(kn'm)$, where $n' = |S|$. Then, we add (C, S', δ_1) to $\mathcal{P}$, where $|S'| \leq \frac{|S|}{2}$. Then, because of Lines 8, 11, and 13, we add at most $L = 2^{2k} \cdot (\log_{1+\delta} \frac{k}{\alpha})^k$ tuples to $\mathcal{P}$ (in Line 14) where the cardinality of the first entry of each tuple is strictly more than $|C|$. The time required to execute the Line 14 is at most $O(m2^k(\frac{k}{\varepsilon})^2 \log \frac{1}{\varepsilon})$ (by Lemma 6). Therefore, the time spent in one iteration of the algorithm is at most

$$O\left(L \cdot 2^k n' \cdot m \cdot \left(\frac{k}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}\right).$$

For any node v labelled with (C, S, δ_1), let $N(k - |C|, |S|)$ be the time taken by the iterations that are labelled by the nodes of the subtree of T, rooted at the node v. The value of $N(k - |C|, |S|)$ can be upper bounded using the following recurrence formula: There is a constant c such that for any $0 \leq k' \leq k$ and $0 \leq n' \leq n$,

$$N(k', n') \leq \begin{cases} c & \text{if } k' = 0 \text{ or } n' = 0 \\ N(k' + \frac{n'}{2}) + L \cdot N(k' - 1, n') + \left(cL \cdot 2^k n' m \left(\frac{k}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}\right) & \text{if } \quad k', n' > 0 \end{cases}. \quad (40)$$

Clearly, the running time of the algorithm will be upper bounded by $N(k, n)$. We claim that

$$N(k, n) \leq c \cdot (2L)^k 2^{3k^2} \cdot n \cdot m \cdot \left(\frac{k}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}.$$

Towards that, we prove that for any $0 \leq k' \leq k$ and $0 \leq n' \leq n$, $N(k', n') \leq c \cdot 2^k \cdot L^{k'} 2^{3k'^2} \cdot n' \cdot m(\frac{k}{\varepsilon})^2 \log \frac{1}{\varepsilon}$ using induction. The base case is when $k' = 0$ or $n' = 0$ and it holds by Equation (40). By induction hypothesis, we have that

$$\begin{aligned} N(k', n') \leq \left(c \cdot 2^k L^{k'} 2^{3k'^2} \frac{n'm}{2} \left(\frac{k}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}\right) &+ \left(L \cdot c2^k L^{k'-1} 2^{3(k'-1)^2} n'm \left(\frac{k}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}\right) \\ &+ \left(cL \cdot 2^k n' \cdot m \left(\frac{k}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}\right). \end{aligned}$$

To prove that $N(k', n') \leq c2^k \cdot L^{k'} 2^{3k'^2} \cdot n' \cdot m \cdot (\frac{k}{\varepsilon})^2 \log \frac{1}{\varepsilon}$, it is enough to prove that $2^{3k'^2-1} + 2^{3(k'-1)^2} + 1 \leq 2^{3k'^2}$, which is true for any $k' \geq 1$.

Therefore, we upper bound the running time of the algorithm as follows:

$$\begin{aligned} N(k, n) &\leq \left(c2^k 2^{3k^2} \cdot n \cdot m \left(\frac{k}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}\right) \cdot L^k \\ &\leq \left(2^{O(k^2)} \cdot n \cdot m \left(\frac{1}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}\right) \cdot 2^{2k} \cdot \left(\log_{1+\delta} \frac{k}{\alpha}\right)^k \\ &\leq \left(2^{O(k^2)} \cdot n \cdot m \left(\frac{1}{\varepsilon}\right)^2 \log \frac{1}{\varepsilon}\right) \cdot \left(\frac{\ln \frac{k}{\alpha}}{\ln(1+\delta)}\right)^k \end{aligned}$$

$$\leq \left(2^{O(k^2+k\log(k^*))}\cdot n\cdot m\cdot\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)\cdot\left(\frac{\ln\frac{1}{\varepsilon}}{\ln(1+\delta)}\right)^k$$

$$\leq \left(2^{O(k^2+k\log(k^*))}\cdot n\cdot m\cdot\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)\cdot\left(\frac{(1+\delta)\ln\frac{1}{\varepsilon}}{\delta}\right)^k$$

$$\leq \left(2^{O(k^2+k\log(k^*))}\cdot n\cdot m\right)\cdot\left(\frac{1}{\varepsilon}\right)^{O(k)}.$$

This completes the proof of the theorem. □

8.2 The Final Step

Now, we are ready to prove Theorem 2. Let $J=(X,k,\mathcal{R})$ be the input instance. Let

$$p(k^*,k,\varepsilon)=\left(\frac{\varepsilon}{1+\varepsilon}\right)^k\cdot\left(\frac{\varepsilon^k}{\left(40k\cdot k^*\cdot(5^k-1)\right)^{k-1}\cdot 2k\cdot(40k^*+\varepsilon)}\right)^{\frac{c'\cdot k^2}{\varepsilon^2}\log\frac{1}{\varepsilon}}$$

be the success probability from Theorem 3, where c' is a constant. For each $I\subseteq[k]$, we apply Theorem 3 on $(X,|I|,\mathrm{proj}_I(\mathcal{R}))$ and $\frac{\varepsilon}{4}$ (where we substitute $k=|I|$ and $k^*=k$) $\frac{1}{p(k,|I|,\frac{\varepsilon}{4})}$ times. Let C'_I be the best solution obtained by the above process. By using Proposition 3.1, let C_I be the solution for J obtained from C'_I. Then, we output the best solution among $\{C_I : I\subseteq[k]\}$. The running time of the algorithm mentioned in Theorem 3 is $2^{O(k^2+k\log(k^*))}\cdot n\cdot m\cdot(\frac{1}{\varepsilon})^{O(k)}$, and the running time of the algorithm mentioned in Proposition 3.1 is linear in the input size. Thus, the running time of our algorithm is at most $\frac{1}{p(k,k,\frac{\varepsilon}{4})}\cdot 2^{O(k^2)}\cdot(\frac{1}{\varepsilon})^{O(k)}\cdot n\cdot m$. The value of $\frac{1}{p(k,k,\frac{\varepsilon}{4})}$ is upper bounded as follows:

$$\begin{aligned}\frac{1}{p(k,k,\frac{\varepsilon}{4})}&=\left(\frac{4+\varepsilon}{\varepsilon}\right)^k\cdot\left(\frac{\left(40k^2\cdot(5^k-1)\right)^{k-1}\cdot 2k\cdot(40k+\frac{\varepsilon}{4})}{\left(\frac{\varepsilon}{4}\right)^k}\right)^{\frac{c'\cdot k^2}{\varepsilon^2}\log\frac{1}{\varepsilon}}\\&=\left(\frac{1}{\varepsilon}\right)^{O(k)}\cdot 2^{O\left(\frac{k^4}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)}\cdot\left(\frac{1}{\varepsilon}\right)^{O\left(\frac{k}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)}\\&=2^{O\left(\frac{k^4}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)}\cdot\left(\frac{1}{\varepsilon}\right)^{O\left(\frac{k}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)}.\end{aligned}$$

Thus, the running time of our algorithm is $2^{O(\frac{k^4}{\varepsilon^2}\log\frac{1}{\varepsilon})}\cdot(\frac{1}{\varepsilon})^{O(\frac{k}{\varepsilon^2}\log\frac{1}{\varepsilon})}n\cdot m$.

Now, we prove the correctness of the algorithm. Let $\widehat{k}\in[k]$ be the integer defined in Lemma 7. We know that $\mathrm{OPT}_{\widehat{k}}(J)\leq(1+\frac{\varepsilon}{4})\mathrm{OPT}(J)$. This implies that there is $I\in\binom{[k]}{\widehat{k}}$ such that the $\mathrm{OPT}(J')$ is at most $(1+\frac{\varepsilon}{4})\mathrm{OPT}(J)$, where $J'=(X,\widehat{k},\mathrm{proj}_I(\mathcal{R}))$. In the iteration of our algorithm corresponding to I, we get a solution C'_I to J' of cost at most $(1+\frac{\varepsilon}{160k})\mathrm{OPT}(J')$ with probability at least $1-(1-p(k^*,|I|,\frac{\varepsilon}{4})^{1/p(k^*,|I|,\frac{\varepsilon}{4})}\geq 1-\frac{1}{e}$. By Proposition 3.1, C_I is a solution to J of cost at most $(1+\frac{\varepsilon}{160k})\mathrm{OPT}(J')$. Thus, by Lemma 7, the cost of C_I is $(1+\frac{\varepsilon}{4})(1+\frac{\varepsilon}{160k})\mathrm{OPT}(J)\leq(1+\varepsilon)\mathrm{OPT}(J)$. This completes the proof of the theorem.

9 PROOFS OF LEMMATA 1, 2, AND 3

In this section, we provide the missing proofs of Lemmata 1, 2, and 3. Recall that in Lemma 1, we claim the following:

For any instance $(\mathbf{A}, r)$ *of* Low GF(2)-Rank Approximation *, one can construct in time* $O(m + n + 2^{2r})$ *an instance* $(X, k = 2^r, \mathcal{R})$ *of* Binary Constrained Clustering *with the following property: Given any* α*-approximate solution* C *of* $(X, k, \mathcal{R})$*, an* α*-approximate solution* $\mathbf{B}$ *of* $(\mathbf{A}, r)$ *can be constructed in time* $O(rmn)$ *and vice versa.*

Proof of Lemma 1. Observe that if GF(2)-rank($\mathbf{B}$) $\leq r$, then $\mathbf{B}$ has at most 2^r distinct columns, because each column is a linear combination of at most r vectors of a basis of the column space of $\mathbf{B}$. Also, the task of Low GF(2)-Rank Approximation can equivalently be stated as follows: Find vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r \in \{0,1\}^m$ over GF(2) such that

$$\sum_{i=1}^{n} \min\{d_H(\mathbf{s}, \mathbf{a}_i) \mid \mathbf{s} \text{ is a linear combination of } \mathbf{s}_1, \ldots, \mathbf{s}_r\}$$

is minimum, where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are the columns of $\mathbf{A}$. To encode an instance of Low GF(2)-Rank Approximation as an instance of Binary Constrained Clustering, we construct the following relation R: Set $k = 2^r$. Let $\Lambda = (\lambda_1, \ldots, \lambda_k)$ be the k-tuple composed by all distinct vectors of $\{0,1\}^r$. Thus, each element $\lambda_i \in \Lambda$ is a binary r-vector. We define $R = \{(x^\intercal \lambda_1, \ldots, x^\intercal \lambda_k) \mid x \in \{0,1\}^r\}$. Thus, R consists of $k = 2^r$ k-tuples and every k-tuple in R is a row of the matrix $\Lambda^\intercal \cdot \Lambda$. Now, we define X to be the set of columns of $\mathbf{A}$ and for each $i \in [m]$, $R_i = R$. Note that, since all R_i are equal, we can construct and keep just one copy of R.

To show that the instance $(\mathbf{A}, r)$ of Low GF(2)-Rank Approximation is equivalent to the constructed instance $(X, k, \mathcal{R})$, assume first that the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r \in \{0,1\}^m$ over GF(2) compose an (approximate) solution of Low GF(2)-Rank Approximation. For every $i \in [k]$, we consider $\lambda_i^\intercal = (\lambda_i[1], \ldots, \lambda_i[r])$ and define vector

$$\mathbf{c}_i = \lambda_i[1]\mathbf{s}_1 \oplus \cdots \oplus \lambda_i[r]\mathbf{s}_r,$$

where $\oplus$ denotes the sum over GF(2), and set $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$. Observe that C contains all linear combinations of $\mathbf{s}_1, \ldots, \mathbf{s}_r$. For every $i \in [k]$ and $j \in [m]$, we have that $\mathbf{c}_i[j] = (\mathbf{s}_1[j], \ldots, \mathbf{s}_r[j])\lambda_i$. Therefore, $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) \in R$ for $j \in [m]$. Since for every $i \in [n]$,

$$\min\{d_H(\mathbf{s}, \mathbf{a}_i) \mid \mathbf{s} \text{ is a linear combination of } \mathbf{s}_1, \ldots, \mathbf{s}_r\} = \min\{d_H(\mathbf{c}_j, \mathbf{a}_i) \mid j \in [k]\},$$

we have that

$$\sum_{i=1}^{n} \min\{d_H(\mathbf{s}, \mathbf{a}_i) \mid \mathbf{s} \text{ is a linear combination of } \mathbf{s}_1, \ldots, \mathbf{s}_r\} = \sum_{i=1}^{n} d_H(\mathbf{a}_i, C) = \sum_{\mathbf{x} \in X} d_H(\mathbf{x}, C).$$

For the opposite direction, assume that $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ is an (approximate) solution for $(X, k, \mathcal{R})$. We construct the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r$ as follows: Let $j \in [m]$. We have that $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) \in R$. Therefore, there is $\mathbf{x} \in \{0,1\}^r$ such that $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) = (\mathbf{x}^\intercal \lambda_1, \ldots, \mathbf{x}^\intercal \lambda_k)$. We set $\mathbf{s}_i[j] = \mathbf{x}[i]$ for $i \in [r]$. Observe that C is the set of all linear combinations of the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r$. Hence,

$$\begin{aligned}\sum_{\mathbf{x} \in X} d_H(\mathbf{x}, C) &= \sum_{\mathbf{x} \in X} \min\{d_H(\mathbf{s}, \mathbf{x}) \mid \mathbf{s} \text{ is a linear combination of } \mathbf{s}_1, \ldots, \mathbf{s}_r\} \\ &= \sum_{i=1}^{n} \min\{d_H(\mathbf{s}, \mathbf{a}_i) \mid \mathbf{s} \text{ is a linear combination of } \mathbf{s}_1, \ldots, \mathbf{s}_r\}.\end{aligned}$$

Notice that to find vectors $\mathbf{x}$, formally, we have to solve m systems of linear equations with r variables such that each system contains 2^r equations. But, since Λ contains all pairwise distinct vectors of $\{0,1\}^r$, we can find the solution by checking the equations corresponding to the vectors containing unique non-zero elements. This immediately implies that for any α-approximate solution C of $(X, k, \mathcal{R})$ an α-approximate solution $\mathbf{B}$ of $(\mathbf{A}, r)$ can be constructed in time $O(rm)$. □

Recall that in Lemma 2, we claim that *for any instance* $(\mathbf{A}, r)$ *of* Low Boolean-Rank Approximation, *one can construct in time* $O(m + n + 2^{2r})$ *an instance* $(X, k = 2^r, \mathcal{R})$ *of* Binary Constrained Clustering *with the following property: Given any* α*-approximate solution* C *of* $(X, k, \mathcal{R})$ *an* α*-approximate solution* $\mathbf{B}$ *of* $(\mathbf{A}, r)$ *can be constructed in time* $O(rmn)$ *and vice versa.* The proof of Lemma 2 essentially repeats the proof of Lemma 1. While we are working now with the Boolean semi-ring $(0, 1, \wedge, \vee)$, we still can use exactly the same trick to reduce Low Boolean-Rank Approximation to Binary Constrained Clustering. The only difference is that GF(2) summations and products are replaced by $\vee$ and $\wedge$, respectively, in the definition of the relation R. Thus, every k-tuple in R is a row of the matrix $\Lambda^\intercal \wedge \Lambda$.

Next, we give a proof sketch of Lemma 3.

Proof sketch of Lemma 3. The task of Binary Projective Clustering can equivalently be stated as follows: Find vectors $\mathbf{s}_{1,1}, \ldots, \mathbf{s}_{1,r}, \ldots, \mathbf{s}_{k,r} \in \{0,1\}^m$ over GF(2) such that

$$\sum_{\mathbf{x} \in X} \min\{d_H(\mathbf{s}, \mathbf{x}) \mid i \in [k] \text{ and } \mathbf{s} \text{ is a linear combination of } \mathbf{s}_{i,1}, \ldots, \mathbf{s}_{i,r}\}$$

is minimum. Now the construction of relation $\mathcal{R}$ is analogous to that of Lemma 1. Set $q = 2^r$. Let $\Lambda = (\lambda_1, \ldots, \lambda_q)$ be the q-tuple composed by all distinct vectors of $\{0,1\}^r$. Thus, each element $\lambda_i \in \Lambda$ is a binary r-vector. We define $R' = \{(x^\intercal \lambda_1, \ldots, x^\intercal \lambda_q) \mid x \in \{0,1\}^r\}$ and $R = \{(b_{1,1}, \ldots, b_{1,q}, b_{2,1}, \ldots, b_{2,q}, \ldots, b_{k,1}, \ldots, b_{k,q}) \mid \text{for all } i \in [k], (b_{i,1}, \ldots, b_{i,q}) \in R'\}$. Thus, R consists of $k' = 2^{k \cdot r}$ tuples of length $k2^r$ each. Now, we define $R_i = R$ for all $i \in [m]$. This completes the construction of instance $(X, k', \mathcal{R})$ of Binary Constrained Clustering.

The proof of correctness of the lemma is similar to the proof of Lemma 1. □

REFERENCES

[1] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. 2004. Approximating extent measures of points. *J. ACM* 51, 4 (2004), 606–635. DOI: https://doi.org/10.1145/1008731.1008736

[2] Noga Alon and Benny Sudakov. 1999. On two segmentation problems. *J. Alg.* 33, 1 (1999), 173–184. DOI: https://doi.org/10.1006/jagm.1999.1024

[3] Noga Amit. 2004. *The Bicluster Graph Editing Problem.* Master's thesis. Tel Aviv University.

[4] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. 2012. Computing a nonnegative matrix factorization—provably. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC'12)*. ACM, 145–162.

[5] Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. 2002. Approximate clustering via core-sets. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC'02)*. ACM, 250–257. DOI: https://doi.org/10.1145/509907.509947

[6] F. Ban, V. Bhattiprolu, K. Bringmann, P. Kolev, E. Lee, and D. P. Woodruff. 2019. A PTAS for ℓ_p -low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*. 747–766.

[7] Eduard Bartl, Radim Belohlávek, and Jan Konecny. 2010. Optimal decompositions of matrices with grades into binary and graded matrices. *Ann. Math. Artific. Intell.* 59, 2 (June 2010), 151–167. DOI: https://doi.org/10.1007/s10472-010-9185-y

[8] Radim Belohlávek and Vilém Vychodil. 2010. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. Syst. Sci.* 76, 1 (2010), 3–20. DOI: https://doi.org/10.1016/j.jcss.2009.05.002

[9] Karl Bringmann, Pavel Kolev, and David P. Woodruff. 2017. Approximation algorithms for ℓ_0-low rank approximation. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS'17)*. 6651–6662. Retrieved from http://papers.nips.cc/paper/7242-approximation-algorithms-for-ell_0-low-rank-approximation.

[10] L. Sunil Chandran, Davis Issac, and Andreas Karrenbauer. 2016. On the parameterized complexity of biclique cover and partition. In *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC'16) (LIPIcs)*, Vol. 63. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, 11:1–11:13. DOI: https://doi.org/10.4230/LIPIcs.IPEC.2016.11

[11] Kenneth L. Clarkson and David P. Woodruff. 2015. Input sparsity and hardness for robust subspace approximation. In *Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS'15)*. IEEE Computer Society, 310–329.

[12] Chen Dan, Kristoffer Arnsfelt Hansen, He Jiang, Liwei Wang, and Yuchen Zhou. 2015. On low rank approximation of binary matrices. *CoRR* abs/1511.01699 (2015). Retrieved from http://arxiv.org/abs/1511.01699.

[13] Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. 2018. Parameterized low-rank binary matrix approximation. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP'18). (LIPIcs)*,

Vol. 107. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, 53:1–53:16. DOI: https://doi.org/10.4230/LIPIcs.ICALP.2018.53

[14] Yinghua Fu, Nianping Jiang, and Hong Sun. 2010. Binary matrix factorization and consensus algorithms. In *Proceedings of the International Conference on Electrical and Control Engineering (ICECE'10)*. IEEE, 4563–4567.

[15] Nicolas Gillis and Stephen A. Vavasis. 2015. On the complexity of robust PCA and ℓ_1-norm low-rank matrix approximation. *CoRR* abs/1509.09236 (2015). Retrieved from http://arxiv.org/abs/1509.09236.

[16] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. 2008. Data reduction and exact algorithms for clique cover. *ACM J. Exper. Alg.* 13 (2008). DOI: https://doi.org/10.1145/1412228.1412236

[17] David A. Gregory, Norman J. Pullman, Kathryn F. Jones, and J. Richard Lundgren. 1991. Biclique coverings of regular bigraphs and minimum semiring ranks of regular matrices. *J. Combin. Theor. Ser. B* 51, 1 (1991), 73–89. DOI: https://doi.org/10.1016/0095-8956(91)90006-6

[18] Harold W. Gutch, Peter Gruber, Arie Yeredor, and Fabian J. Theis. 2012. ICA over finite fields—Separability and algorithms. *Sig. Proc.* 92, 8 (2012), 1796–1808. DOI: https://doi.org/10.1016/j.sigpro.2011.10.003

[19] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* 58, 301 (1963), 13–30. DOI: https://doi.org/10.1080/01621459.1963.10500830

[20] Peng Jiang and Michael T. Heath. 2013. Mining discrete patterns via binary matrix factorization. In *Proceedings of the Industrial Conference on Data Mining Workshops (ICDM'13)*. IEEE Computer Society, 1129–1136.

[21] Peng Jiang, Jiming Peng, Michael Heath, and Rui Yang. 2014. A clustering approach to constrained binary matrix factorization. In *Data Mining and Knowledge Discovery for Big Data: Methodologies, Challenge and Opportunities*. Springer Berlin, 281–303.

[22] Ravindran Kannan and Santosh Vempala. 2009. Spectral algorithms. *Found. Trends Theor. Comput. Sci.* 4, 3–4 (2009), 157–288. DOI: https://doi.org/10.1561/0400000025

[23] Jon Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. 2004. Segmentation problems. *J. ACM* 51, 2 (2004), 263–280. https://doi.org/10.1145/972639.972644

[24] Mehmet Koyutürk and Ananth Grama. 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'03)*. ACM, New York, NY, 147–156. DOI: https://doi.org/10.1145/956750.956770

[25] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. 2010. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM* 57, 2 (2010), 5:1–5:32. DOI: https://doi.org/10.1145/1667053.1667054

[26] Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, and Yuan Hong. 2012. Constraint-aware role mining via extended Boolean matrix decomposition. *IEEE Trans. Depend. Sec. Comput.* 9, 5 (2012), 655–669. DOI: https://doi.org/10.1109/TDSC.2012.21

[27] Michael W. Mahoney. 2011. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.* 3, 2 (2011), 123–224. Retrieved from http://dx.doi.org/10.1561/2200000035.

[28] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. 2008. The discrete basis problem. *IEEE Trans. Knowl. Data Eng.* 20, 10 (2008), 1348–1362. DOI: https://doi.org/10.1109/TKDE.2008.53

[29] Pauli Miettinen and Jilles Vreeken. 2011. Model order selection for Boolean matrix factorization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*. ACM, 51–59. DOI: https://doi.org/10.1145/2020408.2020424

[30] Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. 2016. A survey of role mining. *ACM Comput. Surv.* 48, 4, Article 50 (Feb. 2016), 37 pages. DOI: https://doi.org/10.1145/2871148

[31] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY.

[32] Ankur Moitra. 2016. An almost optimal algorithm for computing nonnegative rank. *SIAM J. Comput.* 45, 1 (2016), 156–173. DOI: https://doi.org/10.1137/140990139

[33] Rafail Ostrovsky and Yuval Rabani. 2002. Polynomial-time approximation schemes for geometric min-sum median clustering. *J. ACM* 49, 2 (2002), 139–156. DOI: https://doi.org/10.1145/506147.506149

[34] Amichai Painsky, Saharon Rosset, and Meir Feder. 2016. Generalized independent component analysis over finite alphabets. *IEEE Trans. Inform. Theor.* 62, 2 (2016), 1038–1053. DOI: https://doi.org/10.1109/TIT.2015.2510657

[35] Ilya P. Razenshteyn, Zhao Song, and David P. Woodruff. 2016. Weighted low rank approximations with provable guarantees. In *Proceedings of the 48th ACM Symposium on Theory of Computing (STOC'16)*. ACM, 250–263. DOI: https://doi.org/10.1145/2897518.2897639

[36] Bao-Hong Shen, Shuiwang Ji, and Jieping Ye. 2009. Mining discrete patterns via binary matrix factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*. ACM, New York, NY, 757–766. DOI: https://doi.org/10.1145/1557019.1557103

[37] Jaideep Vaidya. 2012. Boolean matrix decomposition problem: Theory, variations, and applications to data engineering. In *Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE'12)*. IEEE Computer Society, 1222–1224. DOI: https://doi.org/10.1109/ICDE.2012.144

[38] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. 2007. The role mining problem: Finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT'07)*. 175–184. DOI: https://doi.org/10.1145/1266840.1266870

[39] David P. Woodruff. 2014. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.* 10, 1–2 (2014), 1–157. DOI: https://doi.org/10.1561/0400000060

[40] Sharon Wulff, Ruth Urner, and Shai Ben-David. 2013. Monochromatic bi-clustering. In *Proceedings of the 30th International Conference on Machine Learning (ICML'13) (JMLR Workshop and Conference Proceedings)*, Vol. 28. JMLR.org, 145–153. Retrieved from http://jmlr.org/proceedings/papers/v28/.

[41] Arie Yeredor. 2011. Independent component analysis over Galois fields of prime order. *IEEE Trans. Inform. Theor.* 57, 8 (2011), 5342–5359. DOI: https://doi.org/10.1109/TIT.2011.2145090

Received November 2018; revised August 2019; accepted August 2019