



# Binary $k$ -Center with Missing Entries: Structure Leads to Tractability

Tobias Friedrich  

Hasso Plattner Institute (HPI), University of Potsdam, Germany

Kirill Simonov  

Department of Informatics, University of Bergen, Norway

Farehe Soheil  

Hasso Plattner Institute (HPI), University of Potsdam, Germany

---

## Abstract

$k$ -Center clustering is a fundamental classification problem, where the task is to categorize the given collection of entities into  $k$  clusters and come up with a representative for each cluster, so that the maximum distance between an entity and its representative is minimized. In this work, we focus on the setting where the entities are represented by binary vectors with missing entries, which model incomplete categorical data. This version of the problem has wide applications, from predictive analytics to bioinformatics.

Our main finding is that the problem, which is notoriously hard from the classical complexity viewpoint, becomes tractable as soon as the known entries are sparse and exhibit a certain structure. Formally, we show fixed-parameter tractable algorithms for the parameters vertex cover, fracture number, and treewidth of the row-column graph, which encodes the positions of the known entries of the matrix. Additionally, we tie the complexity of the 1-cluster variant of the problem, which is famous under the name Closest String, to the complexity of solving integer linear programs with few constraints. This implies, in particular, that improving upon the running times of our algorithms would lead to more efficient algorithms for integer linear programming in general.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Fixed parameter tractability

**Keywords and phrases** Clustering, Missing Entries,  $k$ -Center, Parameterized Algorithms

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2025.8

**Related Version** *Full Version*: <https://arxiv.org/abs/2503.01445>

**Funding** *Farehe Soheil*: Supported by the HPI Research School on Foundations of AI (FAI).

**Acknowledgements** We thank Nikolai Karpov for pointing out the applications in bioinformatics.

## 1 Introduction

Clustering is a fundamental problem in computer science with a wide range of applications [28, 30, 49, 25, 51], and has been thoroughly explored [39, 3, 31, 9, 52, 4]. In its most general formulation, given  $n$  data points, the aim of a clustering algorithm is to partition these points into groups, called clusters, based on the similarity. The degree of similarity between points is modeled by a given distance function. Depending on the representation of the data points, several computationally different variants of the clustering problem arise. Commonly, the points are embedded in the  $d$ -dimensional Euclidean space  $\mathbb{R}^d$  with the standard Euclidean distance or the distance given by  $L_p$  norms, or in the space of binary strings equipped with the Hamming distance, or in a general metric space with an explicit given distance function.

Additionally, there exist different approaches to how the similarity is aggregated. For the purpose of this work, we focus on the classical center-based clustering objectives. In  $k$ -Center clustering, given a set of data points and the parameter  $k$ , the objective is to partition the points into  $k$  clusters and identify for each cluster a point called the *center*, so



© Tobias Friedrich, Kirill Simonov, and Farehe Soheil;  
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Parameterized and Exact Computation (IPEC 2025).

Editors: Akanksha Agrawal and Erik Jan van Leeuwen; Article No. 8; pp. 8:1–8:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

that the *maximum* distance between the center and any point within its cluster is minimized.  $k$ -Median clustering is defined in the same way, except that the *sum* of distances between the data points and their respective centers is minimized, and  $k$ -Means clustering aims to minimize the sum of *squared* distances instead.

Unfortunately, virtually all versions of clustering are computationally hard, in the classical sense.  $k$ -Means in  $\mathbb{R}^d$  is NP-hard even on the plane (dimension  $d = 2$ ) [42], and it is also NP-hard for  $k = 2$  clusters even when the vectors have binary entries [2, 18].  $k$ -Median and  $k$ -Center are NP-hard in  $\mathbb{R}^2$  as well [44]. Moreover,  $k$ -Center on binary strings under the Hamming distance is NP-hard even for  $k = 1$  cluster; that is, the problem of finding the binary string that minimizes the maximum distance to a given collection of strings is already NP-hard [19, 37]. The latter problem is well-studied in the literature under the name of CLOSEST STRING, given its important applications in coding theory [35] to bioinformatics [50].

In order to circumvent the general hardness results and simultaneously increase the modeling power of the problem, we consider the following variant of clustering with missing entries. We assume that the data points are represented by vectors, where each entry is either an element of the original domain (e.g., in  $\mathbb{R}$  or  $\{0, 1\}$ ), or the special element “?”, which corresponds to an unknown entry. For the clustering objective, the distance is computed normally between the known entries, but the distance to an unknown entry is always zero. In this way, we can define the problems  $k$ -CENTER WITH MISSING ENTRIES and  $k$ -MEANS WITH MISSING ENTRIES. For formal definitions, we refer to Section 2.

These problems have a wide range of applications. For an example in predictive analytics, consider the setting of the well-known Netflix Prize challenge<sup>1</sup>. The input is a collection of user-movie ratings, and the task is to predict unknown ratings. The data can be represented in the matrix form, where the rows correspond to the users and the columns to the movies, and naturally *most* of the entries in this matrix would be unknown [46]. Clustering in this setting is then an important tool for grouping/labelling similar users or similar movies, based on the available data.

Clustering with missing entries is also closely related to string problems that arise in bioinformatics applications. In the fundamental genome phasing problem (known also as “haplotype assembly”), the input is a collection of *reads*, i.e., short subsequences of the two copies of the genome, and the task is to reconstruct both of the original sequences. Finding the best possible reconstruction in the presence of errors is then naturally modeled as an instance of  $k$ -MEANS WITH MISSING ENTRIES with  $k = 2$ , where the data points correspond to the individual reads, using missing entries to mark the unknown parts of each read; the target centers represent the desired complete genomic sequences; and the clustering objective represents the total number of errors between the known reads and the desired sequences, which needs to be minimized. In fact, [47] use exactly this formalization of the phasing problem (under the name of “Weighted Minimum Error Correction”) as the algorithmic core of their WhatsHap phasing software. Note that in both examples above, the known entries lie in a finite, small domain. For the technical results in this work, we focus on vectors where the vectors have binary values, e.g., in  $\{0, 1\}$ ; the results however can be easily extended to the bounded domain setting.

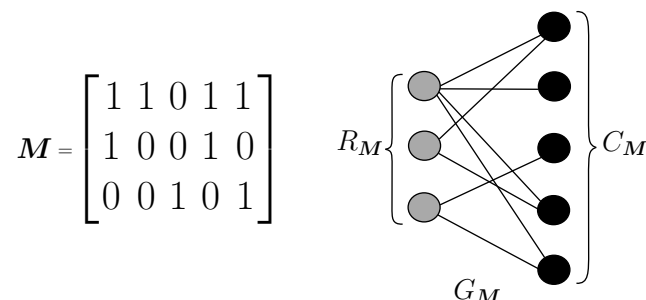
Generally speaking,  $k$ -CENTER WITH MISSING ENTRIES and  $k$ -MEANS WITH MISSING ENTRIES cannot be easier than their fully-defined counterparts. While greatly increasing the modeling power of the problem, the introduction of missing entries poses also additional

---

<sup>1</sup> The problem description and the dataset is available at <https://www.kaggle.com/netflix-inc/netflix-prize-data>.

technical challenges. In particular, most of the methods developed for the standard, full-information versions of clustering are no longer applicable, since the space formed by vectors with missing entries is not necessarily metric: the distances may violate triangle inequality. On the positive side, one can observe that in practical applications the structure of the missing entries is not completely arbitrary. In particular, the known entries are often *sparse* – for example, in the above-mentioned Netflix Prize challenge, only about 1% of the user-movie pairs have a known rating [46]. Therefore, the “hard” cases coming from the standard fully-defined versions of  $k$ -Center/ $k$ -Means are quite far from the instances arising in applications of clustering with missing entries. This motivates the aim to identify tractable cases of  $k$ -CENTER WITH MISSING ENTRIES based on the structure of the missing entries, since the general hardness results for  $k$ -Center are not applicable in this setting. Formally, we use the framework of *parameterized complexity* in order to characterize such cases. We refer to standard textbooks on parameterized complexity for a more thorough introduction to the subject [11, 10].

In order to apply the existing machinery and to put the parameters we consider into perspective, we encode the arrangement of the missing entries into a graph. We say that the *incidence graph* of a given instance is the following bipartite graph: the vertices are the data points and the coordinates, and the edge between a point and a coordinate is present when the respective entry is *known*. When interpreting the input as a matrix, where the data points are the rows, replacing the known entries by “1” and missing entries by “0” results exactly in the biadjacency matrix of the incidence graph. We call this matrix the *mask matrix* of the instance, denoted by  $M$ , and denote the incidence graph of the instance by  $G_M$ ; see Figure 1 for an illustration.



■ **Figure 1** Mask matrix  $M$ , and its corresponding incidence graph, on the right. Vertices corresponding to *rows* of  $M$ , denoted by  $R_M$ , are illustrated in gray. Vertices corresponding to *columns* of  $M$ , denoted by  $C_M$ , are in black.

We mainly consider the following three fundamental sparsity parameters of the incidence graph  $G_M$ :

- Vertex cover number  $\text{vc}(G_M)$ : the smallest number of vertices that are necessary to cover all edges of the graph;
- Fracture number  $\text{fr}(G_M)$ : the smallest number of vertices one needs to remove so that the connected components of the remaining graph are small, i.e., their size is bounded by the same number;
- Treewidth  $\text{tw}(G_M)$ : the classical decomposition parameter measuring how “tree-like” the graph is.

See Figures 2 and 3 for a visual representation of instances with small vertex cover and fracture number, respectively. Treewidth is the most general parameter out of the three, and it encompasses a wide range of instances. For example, the main algorithmic ingredient in the

WhatsHap genomic phasic software [47] is the FPT algorithm for  $k$ -MEANS WITH MISSING ENTRIES parameterized by the maximum number of known entries per column<sup>2</sup>, in the special case where  $k = 2$  and the known entries in each row form a continuous subinterval. In this setting, the treewidth of the incidence graph is never larger than this parameter. Vertex cover, on the other hand, is the most restrictive of the three; however, comparatively small vertex cover may be a feasible model in settings such as the Netflix Prize challenge, where most users would only have ratings for a relatively small collection of the most popular movies. The fracture number aims to generalize the setting of small vertex cover, to also allow for an arbitrary number of small local “information patches”, outside of the few “most popular” rows and columns. Note that fracture number is a strictly more general parameter than vertex cover number, since removing any vertex cover from the graph results in connected components of size one; that is, for any instance,  $\text{fr}(G_M) \leq \text{vc}(G_M)$ . On the other hand, it holds that  $\text{tw}(G_M) \leq 2 \text{fr}(G_M)$ , see Section 6 for the details.

Previously, the perspective outlined above has been successfully applied to  $k$ -MEANS WITH MISSING ENTRIES. Ganian et al. [20] show that the problem admits an FPT algorithm parameterized by treewidth of the incidence graph in case of the bounded domain, as well as further FPT algorithms for real-valued vectors in more restrictive parametrization. However, similar questions for  $k$ -CENTER WITH MISSING ENTRIES remain widely open.

In this work, we aim to close this gap and investigate parameterized algorithms for the  $k$ -Center objective on classes of instances where the known entries are “sparse”, in the sense of the structural parameters above. Our motivation stems from the following. First,  $k$ -Center is a well-studied and widely applicable similarity objective, which in certain cases might be preferable over  $k$ -Means; specifically, whenever the cost of clustering is associated with each individual data point and has to be equally small, as opposed to minimizing the total, “social”, cost spread out over all data points. Second,  $k$ -Center is interesting from a theoretical perspective, being a natural optimization target that on the technical level behaves very differently from  $k$ -Means. Our findings, as described next, show that the  $k$ -Center objective is in fact more challenging than  $k$ -Means in this context, and we prove that  $k$ -CENTER WITH MISSING ENTRIES is as general as a wide class of integer linear programs.

**Our contribution.** We present several novel parameterized algorithms for  $k$ -CENTER WITH MISSING ENTRIES. First, we show that  $k$ -CENTER WITH MISSING ENTRIES is FPT when parameterized by the vertex cover number  $\text{vc}(G_M)$  of the incidence graph plus  $k$ . Specifically, we prove the following result. Here and next,  $n$  is the number of data points in the instance and  $m$  is their dimension.

► **Theorem 1.**  *$k$ -CENTER WITH MISSING ENTRIES admits an algorithm with running time*

$$2^{\mathcal{O}(k \cdot \text{vc}(G_M) + \text{vc}(G_M)^2 \cdot \log \text{vc}(G_M))} \text{poly}(nm).$$

This result can be compared to the result of [15], who considered the complementary parametrization of the same problem (under the name of ANY-CLUSTERING-COMPLETION). That is, they consider the minimum number of rows and columns that are needed to cover all *missing* entries. Interestingly, for their FPT algorithm, it was also necessary to include the target distance  $d$  in the parameter – we do not need this restriction in our setting, which highlights the property that instances of  $k$ -CENTER WITH MISSING ENTRIES, where missing entries are dense, are, in a sense, easier.

---

<sup>2</sup> Called *coverage* in their work.

Moving further, we extend the result of Theorem 1 to the more general setting where the parameter is the fracture number of the incidence graph  $G_M$ . We show that one can achieve the running time of Theorem 1 even for fracture number, which is the most technical result of this paper.

► **Theorem 2.**  *$k$ -CENTER WITH MISSING ENTRIES admits an algorithm with running time*

$$2^{\mathcal{O}(k \cdot \text{fr}(G_M) + \text{fr}(G_M)^2 \cdot \log \text{fr}(G_M))} \text{poly}(nm).$$

To the best of our knowledge, no previous work on clustering problems considers the fracture number as the parameter; however it has been successfully applied to other fundamental problems such as Integer Linear Programs (ILPs) [24] and Edge Disjoint Paths [22]. Furthermore, a very similar parameter, equivalent to fracture number, has been studied in the literature under the name *vertex integrity*, for example in the context of Subgraph Isomorphism [5] and algorithmic metatheorems [36].

In order to prove Theorem 2, we also need an algorithm with respect to the treewidth of the incidence graph  $G_M$ , stated in the next theorem. We denote by  $d$  the target radius of the cluster, i.e., the maximum distance between a point and its cluster center.

► **Theorem 3.**  *$k$ -CENTER WITH MISSING ENTRIES admits an algorithm with running time*

$$d^{\mathcal{O}(\text{tw}(G_M))} 2^{\mathcal{O}(k \cdot \text{tw}(G_M))} \text{poly}(nm).$$

In other words, the problem is FPT when parameterized by  $\text{tw}(G_m) + d + k$ , or XP when parameterized by  $\text{tw}(G_m) + k$ . Note that, as opposed to Theorem 1 and Theorem 2, here we need the dependence on  $d$  in the exponential part of the running time. This is, however, still sufficient to enable the algorithm claimed by Theorem 2.

While we are not aware of matching hardness results based on standard complexity assumptions, we can nevertheless argue that improving the running time in Theorems 1 and 2 resolves a fundamental open question. Specifically, we show a parameterized equivalence between CLOSEST STRING, parameterized by the number of strings, and Integer Linear Program (ILP) with bounded variables, parameterized by the number of constraints. While we use the reduction from CLOSEST STRING to ILP as a building block in the algorithm of Theorem 1, the reduction in the other direction, i.e., from ILP to CLOSEST STRING, is most relevant here. Formally, it yields the following statement:

► **Theorem 4.** *For any  $\alpha > 0$ , assume that CLOSEST STRING admits an algorithm with running time  $2^{\mathcal{O}(n^{1+\alpha})} \cdot \text{poly}(n\ell)$ , where  $n$  is the number of strings and  $\ell$  is their length. Then the ILP  $\{\mathbf{Ax} = \mathbf{b} : \forall i, \ell_i \leq x_i \leq u_i\}$ , where  $\mathbf{A} \in \mathbb{Z}^{r \times c}$ ,  $\mathbf{b} \in \mathbb{Z}^r$ , and  $\ell_i \leq u_i \in \mathbb{Z}$ , can be solved in time  $2^{\mathcal{O}(r^{1+\alpha+o(1)})} \cdot \text{poly}(rc)$ , assuming that  $\|\mathbf{A}\|_\infty = \mathcal{O}(r)$  and  $\delta = 2^{\mathcal{O}(r)}$ , where  $\delta = \max_{i \in [c]}(u_i - \ell_i)$ .*

Note that CLOSEST STRING is a special case of  $k$ -CENTER WITH MISSING ENTRIES with  $k = 1$  and no missing entries, where also the number of strings matches with the vertex cover of the known entries. Therefore, we immediately get that improving the  $\text{vc}(G_M)^2$  term in the exponent of Theorem 1 to  $\text{vc}(G_M)^{1+\alpha}$  implies the same improvement for this class of ILP instances, and the same holds for the result of Theorem 2. Notably, the result of Theorem 4 was independently discovered by Rohwedder and Węgrzycki in a recent paper [48]; they also provide further examples of problems, where such an improvement implies breaking long-standing barriers in terms of the best-known running time, and conjecture that this might be impossible.

**Related work.** Clustering problems are also extensively studied from the approximation perspective. The  $k$ -Center problem classically admits 2-approximation in poly-time [26, 17]. On the other hand, approximating  $k$ -CENTER in  $\mathbb{R}^2$  within a factor of 1.82 is NP-hard [45, 7]. Katsikarelis et al. [32] study the  $(k, r)$ -center problem, where the task is to select at most  $k$  vertices (centers) in an edge-weighted graph, so that all the other vertices are at distance at most  $r$  from a center. They establish tight complexity bounds with respect to standard structural graph parameters (clique-width, vertex cover, tree-depth), including matching SETH/ETH lower bounds. Moreover they introduce parameterized approximation schemes that for any  $\epsilon > 0$  efficiently compute a  $(k, (1 + \epsilon)r)$ -center when parameterized by treewidth or clique-width of the graph.

CLOSEST STRING is well-studied under various parameters [38, 27, 40, 8] and in the area of approximation algorithms [23, 38, 41, 43]. Abboud et al. [1] have shown that CLOSEST STRING on binary strings of length  $\ell$  can not be solved in time  $(2 - \epsilon)^\ell \cdot \text{poly}(n\ell)$  under the SETH, for any  $\epsilon > 0$ . This can be seen as a tighter analogue of Theorem 4 for the parametrization of CLOSEST STRING by the lengths of the strings. A version of the CLOSEST STRING with wildcards was studied from the viewpoint of parameterized complexity [29]; the wildcards behave exactly like missing entries in our definition, therefore this problem is equivalent to  $k$ -CENTER WITH MISSING ENTRIES for  $k = 1$ .

Knop et al. [33] studied combinatorial  $n$ -fold integer programming, yielding in particular  $n^{\mathcal{O}(n^2)} \text{poly}(m)$  algorithms for CLOSEST STRING, CLOSEST STRING with wildcards, and  $k$ -CENTER (with binary entries), where  $n$  is the number of strings/rows and  $m$  is the length of the strings/number of coordinates. These results can be seen as special cases of our Theorem 2.

The versions of  $k$ -CENTER WITH MISSING ENTRIES/ $k$ -MEANS WITH MISSING ENTRIES with zero target cost of clustering is considered in the literature under the class of “matrix completion” problems. There, given a matrix with missing entries, the task is to complete it to achieve certain structure, such as few distinct rows (resulting in the same objective as in the clustering problems) or small rank. Parameterized algorithms for matrix completion problems were also studied in [21]. Moreover, Eiben et al. [14] study the Independent Set problem on induced subgraphs of powers of the hypercube, which corresponds to analyzing the “diversity” of a set of vectors, in contrast to minimizing the number of clusters. They also introduce the *Pow-Hyp-IS-Completion* variant, where given Boolean vectors with missing entries and integers  $k$  and  $r$ , the goal is to complete the vectors so that a subset of  $k$  vectors has pairwise Hamming distance at least  $r + 1$ , or determine that no such subset exists.

Finally,  $k$ -MEANS WITH MISSING ENTRIES has been studied in  $\mathbb{R}^d$  from a parameterized approximation perspective. Geometrically, each data point with missing entries can be seen as an axis-parallel linear subspace of  $\mathbb{R}^d$ , and the task is to identify  $k$  centers as points in  $\mathbb{R}^d$  that minimize the total squared distance to the assigned subspaces. [13] show that this problem admits a  $(1 + \epsilon)$  approximation in time  $2^{\text{poly}(k, \epsilon, \Delta)} \cdot \text{poly}(nd)$ , where  $\Delta$  is the maximum number of missing entries per row, meaning that the subspaces have dimension at most  $\Delta$ .

**Paper organization.** We define the necessary preliminaries in Section 2. Then we show the reduction from ILP to CLOSEST STRING in Section 3. Sections 4–6 are dedicated to the respective FPT results for the parameters vertex cover, treewidth and fracture number. We provide a conclusion in Section 7. Due to space constraints, most of the technical proofs are deferred to the full version.

## 2 Preliminaries

In this section, we introduce key definitions and notations used throughout the paper. For an integer  $n$  we write  $[n]$  to denote the set  $\{1, \dots, n\}$ . We use  $\mathbb{Z}_+$  to denote the set of non-negative integers. For a vector of real numbers  $\mathbf{v} \in \mathbb{R}^\ell$  with length  $\ell$ , its  $i$ -th entry is denoted by  $\mathbf{v}[i]$  and  $v_i$  interchangeably. By  $\mathbf{v}^T$ , we denote the transpose of the vector. Similarly, for a binary string  $s \in \{0, 1\}^\ell$  of length  $\ell$ , its  $i$ -th bit is denoted by  $s[i]$ , and  $s_i$  and we write  $s = [s_1, \dots, s_\ell]$  to denote the complete string. For a set  $I \subseteq [\ell]$ , the substring of  $s$  restricted to the indices in  $I$  is written as  $s[I] \in \{0, 1\}^{|I|}$ . For two indices  $i, j \in [\ell]$ , the substring from  $s[i]$  throughout  $s[j]$  (inclusive) is denoted by  $s[i, j] \in \{0, 1\}^{j-i+1}$ . The complement of  $s$  is represented by  $\bar{s}$ . The Hamming distance between two binary strings  $s_1, s_2 \in \{0, 1\}^\ell$  is denoted by  $d_H : \{0, 1\}^\ell \times \{0, 1\}^\ell \rightarrow \mathbb{Z}_+$  with  $d_H(s_1, s_2) = |\{i \in [\ell] : s_1[i] \neq s_2[i]\}|$ . For a graph  $G$ , the set of vertices and edges are denoted by  $V(G)$  and  $E(G)$  respectively.

Consider a matrix  $\mathbf{A}$  with  $n$  rows and  $m$  coordinates. The sets of rows and coordinates of  $\mathbf{A}$  are denoted by  $R_{\mathbf{A}} = \{r_1, r_2, \dots, r_n\}$  and  $C_{\mathbf{A}} = \{c_1, c_2, \dots, c_m\}$  respectively. For  $i \in [n]$  and  $j \in [m]$ , the  $i$ -th row of  $\mathbf{A}$  is denoted by  $\mathbf{A}[i]$  and the  $j$ -th entry of  $\mathbf{A}[i]$  is written as  $\mathbf{A}[i][j]$ . We may also use  $a_i$  and  $a_{ij}$  respectively to refer to  $\mathbf{A}[i]$  and  $\mathbf{A}[i][j]$ , when clear from context. For matrices  $\mathbf{A}$  and  $\mathbf{B}$ , with the same number of rows and columns, their *entry-wise* subtraction is written as  $\mathbf{A} - \mathbf{B}$ , meaning  $(\mathbf{A} - \mathbf{B})[i][j] = \mathbf{A}[i][j] - \mathbf{B}[i][j]$ . Similarly, their *entry-wise* product is denoted by  $\mathbf{A} \circ \mathbf{B}$ , where  $(\mathbf{A} \circ \mathbf{B})[i][j] = \mathbf{A}[i][j] \cdot \mathbf{B}[i][j]$ . For two binary matrices  $\mathbf{A}$  and  $\mathbf{B}$ , we define their row-wise Hamming distance as a column vector  $d_H(\mathbf{A}, \mathbf{B})$  of length  $n$ , where,  $d_H(\mathbf{A}, \mathbf{B})[i] = d_H(\mathbf{A}[i], \mathbf{B}[i])$ . For two vectors  $\mathbf{a}$  and  $\mathbf{b}$  of the same length,  $\mathbf{a} \leq \mathbf{b}$  if for all coordinate  $i$ ,  $\mathbf{a}[i] \leq \mathbf{b}[i]$ .

Moreover, a missing entry is denoted by “?”. We define the Hamming distance between a known entry (0 or 1) and an unknown entry (?) as zero, i.e.,  $d_H(0, ?) = d_H(1, ?) = 0$ . We remark here that assuming the distance to a missing entry to be any other constant (e.g., 1 instead of 0) does not introduce additional complexity, as the number of missing entries in any given row is fixed by the input and their contribution to the cost of clustering does not depend on the solution. It is easy to observe that all the algorithms and techniques presented here would still apply in a straightforward manner to the problem with arbitrary cost of a missing entry, by adjusting the target radius of each row. For clarity of the exposition and consistency with existing literature on missing entries, we define the cost of a missing entry to be zero, as above.

For a binary matrix  $\mathbf{M} \in \{0, 1\}^{n \times m}$ , its *incidence graph* is an undirected bipartite graph defined as  $G_{\mathbf{M}} = (V_{\mathbf{M}}, E_{\mathbf{M}})$ , where  $V_{\mathbf{M}} = (R_{\mathbf{M}} \cup C_{\mathbf{M}})$  and  $E_{\mathbf{M}} = \{(u, v) : u \in R_{\mathbf{M}}, v \in C_{\mathbf{M}}, \mathbf{M}[u][v] = 1\}$ . We use *row* (resp. *coordinate*) vertices to represent the vertices in  $G_{\mathbf{M}}$  that correspond to the rows (coordinates) of  $\mathbf{M}$ . For a visual representation of the incidence graph, refer to Figure 1.

With these definitions, we can now proceed to formalize the main problem in question. Note that we assume the input matrix  $\mathbf{A}$  to only contain  $\{0, 1\}$  entries, as the missing entries are encoded by the matrix  $\mathbf{M}$ , and when  $\mathbf{M}[i][j] = 0$ , its product with  $\mathbf{A}[i][j]$  is always 0.

► **Definition 5** (*k-CENTER WITH MISSING ENTRIES*). *Given matrices  $\mathbf{A} \in \{0, 1\}^{n \times m}$ , and  $\mathbf{M} \in \{0, 1\}^{n \times m}$ , integers  $k$  and  $d$ . The task is to determine if there exists a binary matrix  $\mathbf{B} \in \{0, 1\}^{n \times m}$  with at most  $k$  distinct rows such that:*

$$d_H((\mathbf{M} \circ \mathbf{A}), (\mathbf{M} \circ \mathbf{B})) \leq \mathbf{d},$$

where  $\mathbf{d}$  is the  $n$ -dimensional column vector  $(d, d, \dots, d)$ .



Note that in the context of a  $k$ -CENTER WITH MISSING ENTRIES instance, we use the term “point” interchangeably with “row” and “coordinate” interchangeably with “column”, since the points to be clustered are represented as rows of the matrices  $\mathbf{A}$  and  $\mathbf{M}$ . We also say that an entry of  $\mathbf{A}$  is a missing entry or “?” whenever the corresponding entry of  $\mathbf{M}$  is zero.

Next, we define the related problems.

► **Definition 6** (CLOSEST STRING). *Given a set of  $n \in \mathbb{Z}_+$  binary strings  $S = \{s_1, \dots, s_n\}$  each of length  $\ell$  and a non-negative integer  $d \in \mathbb{Z}_+$ , determine whether there exist a binary string  $s^*$  of length  $\ell$  such that for all  $i \in [n]$ :*

$$d_H(s^*, s_i) \leq d.$$

► **Definition 7** (NON-UNIFORM CLOSEST STRING). *Given a set of  $n \in \mathbb{Z}_+$  binary strings  $S = \{s_1, \dots, s_n\}$  each of length  $\ell$  and a distance vector  $\mathbf{d} = (d_1, \dots, d_n) \in \mathbb{Z}_+^n$ , determine whether there exist a binary string  $s^*$  of length  $\ell$  such that for all  $i \in [n]$ ,*

$$d_H(s^*, s_i) \leq d_i.$$

► **Definition 8** (ILP FEASIBILITY). *Given a constraint matrix  $\mathbf{A} \in \mathbb{Z}^{r \times c}$ , a column vector  $\mathbf{b} \in \mathbb{Z}^r$ , and integers  $\ell_i \leq u_i$  for each  $i \in [c]$ , determine whether there exists a column vector  $\mathbf{x} \in \mathbb{Z}^c$  such that*

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

*and  $\ell_i \leq x_i \leq u_i$  for each  $i \in [c]$ .*

Finally, we recall the known algorithms for CLOSEST STRING/ILP FEASIBILITY with respect to the number of strings/number of constraints. Knop et al. [33] designed a general approach to solve ILP formulations of various problems, including CLOSEST STRING and NON-UNIFORM CLOSEST STRING, which is also applicable to versions with missing entries.

► **Theorem 9** (Knop et al. [33]). *An instance of NON-UNIFORM CLOSEST STRING with  $n$  binary strings of length  $\ell$  can be solved in time  $n^{\mathcal{O}(n^2)} \cdot \ell$ . The running time also holds if the strings have missing entries.*

For ILP FEASIBILITY where the coefficients are bounded, Eisenbrand and Weismantel [16] show a fast dynamic programming algorithm that has a similar running time.

► **Theorem 10** (Eisenbrand and Weismantel [16]). *An ILP of form  $\max\{\mathbf{c}^T \mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, 0 \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^n\}$ , where  $\mathbf{A} \in \mathbb{Z}^{m \times n}$  with  $\|\mathbf{A}\|_\infty \leq \Delta$ ,  $\mathbf{b} \in \mathbb{Z}^m$ ,  $\mathbf{u} \in \mathbb{Z}_+^n$ , and  $\mathbf{c} \in \mathbb{Z}^n$ , can be solved in time  $n \cdot \mathcal{O}(m)^{(m+1)^2} \cdot \mathcal{O}(\Delta)^{m \cdot (m+1)} \cdot \log^2 m \cdot \Delta$ .*

### 3 Reduction from ILP Feasibility to Closest String

This section is dedicated to the reduction from ILP FEASIBILITY to CLOSEST STRING that preserves the number of rows up to a factor of  $\mathcal{O}(\log \|\mathbf{A}\|_\infty)$ , where  $\mathbf{A}$  is the constraint coefficient matrix in the ILP FEASIBILITY instance. We show the sequence of reductions that lead to the proof of Theorem 11, which is then used to establish Theorem 4. While complete proofs of Lemma 12 to Lemma 18 are provided in the full version, we state the results formally next.



► **Theorem 11.** *Let  $\{\mathbf{Ax} = \mathbf{b} : \forall i, \ell_i \leq x_i \leq u_i\}$  be an ILP FEASIBILITY instance, with  $\mathbf{A} \in \mathbb{Z}^{r \times c}$ ,  $\mathbf{b} \in \mathbb{Z}^r$ , and  $\ell_i, u_i \in \mathbb{Z}$  for each  $i \in [c]$ . In polynomial time, one can construct an equivalent instance of CLOSEST STRING with  $n = \mathcal{O}(\log \|\mathbf{A}\|_\infty \cdot r)$ ,  $\ell = \mathcal{O}(\|\mathbf{A}\|_\infty \cdot \log^2 \|\mathbf{A}\|_\infty \cdot r \cdot c \cdot \delta)$ , and  $d = \mathcal{O}(\|\mathbf{A}\|_\infty \cdot \log \|\mathbf{A}\|_\infty \cdot c \cdot \delta)$ ; here,  $\delta = \max_{i \in [c]}(u_i - \ell_i)$ .*

We start with a lemma showing that a general ILP FEASIBILITY instance, where the coefficients may be negative, and variables may have negative lower bounds, can be transformed, in linear time with respect to the number of variables, into an equivalent instance with non-negative variable domains, i.e., of form  $0 \leq y_i \leq u'_i$ , non-negative coefficient matrix and non-negative right-hand side, while mostly preserving the number of constraints, variables and the magnitude of the entries, by a constant factor.

► **Lemma 12.** *Let  $\{\mathbf{Ax} = \mathbf{b} : \forall i, \ell_i \leq x_i \leq u_i\}$  be an ILP FEASIBILITY instance with  $\mathbf{A} \in \mathbb{Z}^{r \times c}$ , and  $\mathbf{b} \in \mathbb{Z}^r$ , an integer column vector. In time  $\mathcal{O}(r \cdot (r + c))$ , we can construct an equivalent instance of ILP FEASIBILITY  $\{\mathbf{A}'\mathbf{y} = \mathbf{b}' : \forall i, 0 \leq y_i \leq u'_i\}$ , with  $\mathbf{b}' \in \mathbb{Z}_+^{2r}$ ,  $\mathbf{A}' \in \mathbb{Z}_+^{2r \times (r+c)}$  such that  $\|\mathbf{A}'\|_\infty = \|\mathbf{A}\|_\infty$ ,  $\|\mathbf{b}'\|_\infty = \mathcal{O}(c \cdot \delta \cdot \|\mathbf{A}\|_\infty)$  and  $\max\{u'_i\} = \mathcal{O}(c \cdot \delta \cdot \|\mathbf{A}\|_\infty)$ , where  $\delta = \max_{i \in [c]}(u_i - \ell_i)$ .*

By Lemma 12, we observe that a general ILP FEASIBILITY instance is equivalent to an instance where everything is non-negative, with a bounded blow-up in the number of constraints, variables and the magnitude of values. Thus, we continue our chain of reductions from a non-negative ILP. In the next step, we reduce from arbitrary bounds on the variables to binary variables, as stated in the following lemma.

► **Lemma 13.** *Let  $\{\mathbf{Ax} = \mathbf{b} : \forall i, 0 \leq x_i \leq u_i\}$  be an ILP FEASIBILITY instance with  $\mathbf{A} \in \mathbb{Z}_+^{r \times c}$ , a non-negative integer matrix with  $r$  rows and  $c$  columns and  $\mathbf{b} \in \mathbb{Z}_+^r$ , a column vector. In polynomial time, this instance can be reduced to an equivalent instance  $\{\mathbf{A}'\mathbf{y} = \mathbf{b}' : \forall i, y_i \in \{0, 1\}\}$  of ILP FEASIBILITY where  $\mathbf{A}' \in \mathbb{Z}_+^{r \times c'}$  is a matrix with  $r$  rows,  $c' \leq c \cdot \max\{u_i + 1\}$  columns and  $\|\mathbf{A}'\|_\infty = \|\mathbf{A}\|_\infty$ , and  $\mathbf{b}' \in \mathbb{Z}_+^{r'}$  is a vector with  $\|\mathbf{b}'\|_\infty \leq \|\mathbf{b}\|_\infty$ .*

Next, we show that an instance of ILP FEASIBILITY with non-negative coefficients and binary variables can be reduced to an instance with binary coefficients. The proof is similar in spirit to the proof of Lemma 8 in [34], however we are interested in ILP FEASIBILITY instances where variables are only allowed to take values in  $\{0, 1\}$ , which is the main technical difference.

► **Lemma 14.** *Let  $\{\mathbf{Ax} = \mathbf{b} : \forall i, x_i \in \{0, 1\}\}$  be an ILP FEASIBILITY instance with  $\mathbf{A} \in \mathbb{Z}_+^{r \times c}$ , a non-negative integer matrix with  $r$  rows and  $c$  columns and  $\mathbf{b} \in \mathbb{Z}_+^r$ , a column vector. In polynomial time, this instance can be reduced to an equivalent instance  $\{\mathbf{A}'\mathbf{y} = \mathbf{b}' : \forall i, y_i \in \{0, 1\}\}$  of ILP FEASIBILITY where  $\mathbf{A}'$  is a  $\{0, 1\}$ -matrix with  $r' = \mathcal{O}(\log \|\mathbf{A}\|_\infty \cdot r)$  rows and  $c' = \mathcal{O}(\log \|\mathbf{A}\|_\infty \cdot (c + \|\mathbf{b}\|_\infty))$  columns, where  $\mathbf{b}' \in \mathbb{Z}_+^{r'}$  is a vector with  $\|\mathbf{b}'\|_\infty = \mathcal{O}(\|\mathbf{b}\|_\infty)$ .*

Now we introduce a lemma that allows to reduce a  $\{0, 1\}$ -coefficient matrix to a matrix with values only in  $\{-1, 1\}$ .

► **Lemma 15.** *Let  $\{\mathbf{Ax} = \mathbf{b} : x_i \in \{0, 1\}\}$  be an ILP FEASIBILITY instance with  $\mathbf{A} \in \{0, 1\}^{r \times c}$  and  $\mathbf{b} \in \mathbb{Z}_+^r$ . In polynomial time, this instance can be reduced to an equivalent instance  $\{\mathbf{A}'\mathbf{y} = \mathbf{b}' : \forall i, y_i \in \{0, 1\}\}$  of ILP FEASIBILITY where  $\mathbf{A}'$  is a  $\{-1, 1\}$ -matrix with  $r' = r + 1$  rows and  $c' = 2c$  columns and  $\mathbf{b}' \in \mathbb{Z}_+^{r'}$  is a vector with  $\|\mathbf{b}'\|_\infty = 2\|\mathbf{b}\|_\infty$ .*

It will be more convenient to work with ILP instances of form  $\mathbf{Ax} \leq \mathbf{b}$ , which motivates the following lemma.

► **Lemma 16.** *Every ILP FEASIBILITY instance  $\{\mathbf{Ax} = \mathbf{b} : \forall i, x_i \in \{0, 1\}\}$  with  $\mathbf{A} \in \{-1, +1\}^{r \times c}$  and  $\mathbf{b} \in \mathbb{Z}_+^r$  can be reduced to an equivalent ILP instance  $\{\mathbf{A}'\mathbf{y} \leq \mathbf{b}' : \forall i, y_i \in \{0, 1\}\}$  with  $\mathbf{A}' \in \{-1, +1\}^{2r \times c}$  and  $\mathbf{b}' \in \mathbb{Z}^{2r}$ , where  $\|\mathbf{b}'\|_\infty = \|\mathbf{b}\|_\infty$ .*

Before reducing to CLOSEST STRING itself, we first show a reduction to a slightly more general version of the problem, NON-UNIFORM CLOSEST STRING (Definition 7), where each string has its own upper bound on the distance to the closest string.

► **Lemma 17.** *Every ILP instance  $\{\mathbf{Ax} \leq \mathbf{b} : \forall i, x_i \in \{0, 1\}\}$  with  $\mathbf{A} \in \{-1, +1\}^{r \times c}$  and  $\mathbf{b} \in \mathbb{Z}^r$  can be reduced, in polynomial time, to a NON-UNIFORM CLOSEST STRING instance with  $r$  strings of length  $c$ , where  $d_i \leq (c + \|\mathbf{b}\|_\infty)$  for each  $i \in [n]$ .*

Finally, we show a reduction from NON-UNIFORM CLOSEST STRING to CLOSEST STRING.

► **Lemma 18.** *Every NON-UNIFORM CLOSEST STRING instance with strings  $S = \{s_1, \dots, s_n\}$  of length  $\ell \in \mathbb{N}$  and distance vector  $\mathbf{d} = (d_1, \dots, d_n)$  can be reduced to a CLOSEST STRING instance with  $2n$  strings  $S' = \{s'_1, \dots, s'_{2n}\}$  and distance  $d = \max_{i=1}^n d_i$  in time  $\mathcal{O}(n \cdot \ell)$ , where length of the constructed strings is at most  $\ell + 2n \cdot d$ .*

With the reductions above, we can conclude with the proof of Theorem 11 as outlined below.

**Proof of Theorem 11.** We start by applying Lemma 12, to obtain an equivalent ILP FEASIBILITY instance where the coefficients, constraints and variable domains are all non-negative integers. Next, by applying Lemma 13, we get an equivalent instance of ILP FEASIBILITY with binary variables. Then, with the help of Lemma 14, we get that also the coefficients in the constraints are binary. After applying Lemma 15, we get that all coefficients are either  $-1$  or  $+1$ , and additionally that the ILP instance is of the form  $\mathbf{Ax} \leq \mathbf{b}$  by applying Lemma 16. Finally, we reduce to NON-UNIFORM CLOSEST STRING via Lemma 17, and then to CLOSEST STRING via Lemma 18.

We now argue about the size of the constructed instance. The number of rows is increased by a factor of  $\mathcal{O}(\log \|\mathbf{A}\|_\infty)$  by the reduction of Lemma 14, and by at most a constant factor in all other reductions. Therefore,  $n = \mathcal{O}(\log \|\mathbf{A}\|_\infty \cdot r)$ . The number of columns is increased by the original number of rows in Lemma 12, by a factor of  $\delta$  in Lemma 13, where  $\delta = \max u_i - \ell_i + 1$ , and then  $\mathcal{O}(c \cdot \delta \cdot \|\mathbf{A}\|_\infty)$  additional columns are added in Lemma 14, along with an additional factor of  $\mathcal{O}(\log \|\mathbf{A}\|_\infty)$ . Lemma 15 only increases the number of columns by at most a constant factor, and Lemmata 16, 17 do not change the number of columns. Finally, in Lemma 18,  $d$  is set to at most  $\mathcal{O}(\|\mathbf{A}\|_\infty \cdot \log \|\mathbf{A}\|_\infty \cdot c \cdot \delta)$  and additional  $2n \cdot d$  columns are attached, where  $n = \mathcal{O}(\log \|\mathbf{A}\|_\infty \cdot r)$ . Therefore, the final length of the strings  $\ell$  is at most  $\mathcal{O}(\|\mathbf{A}\|_\infty \cdot \log^2 \|\mathbf{A}\|_\infty \cdot r \cdot c \cdot \delta)$ . ◀

As mentioned in the previous section, ILP FEASIBILITY admits an algorithm with running time  $(r\Delta)^{(r+1)^2} \cdot \text{poly}(rc)$  [16], where  $\Delta = \|\mathbf{A}\|_\infty$ . On the other hand, a version of ILP FEASIBILITY without upper bounds on the variables can be solved in time  $\mathcal{O}(c \cdot (r\Delta)^{2r} \cdot \|\mathbf{b}\|_1^2)$  [16], and this is tight under the ETH [34]. This raises a natural question whether the gap in the running time between the two versions is necessary. As CLOSEST STRING admits a straightforward reduction to ILP FEASIBILITY, it also makes sense to ask whether running time better than  $n^{\mathcal{O}(n^2)} \cdot \text{poly}(n\ell)$  can be achieved for CLOSEST STRING. From Theorem 11,

it follows that an improved CLOSEST STRING algorithm would automatically improve the best-known running time for ILP FEASIBILITY (with upper bounds). Formally, we prove Theorem 4, as stated in Section 1.

► **Theorem 4.** *For any  $\alpha > 0$ , assume that CLOSEST STRING admits an algorithm with running time  $2^{\mathcal{O}(n^{1+\alpha})} \cdot \text{poly}(n\ell)$ , where  $n$  is the number of strings and  $\ell$  is their length. Then the ILP  $\{\mathbf{Ax} = \mathbf{b} : \forall i, \ell_i \leq x_i \leq u_i\}$ , where  $\mathbf{A} \in \mathbb{Z}^{r \times c}$ ,  $\mathbf{b} \in \mathbb{Z}^r$ , and  $\ell_i \leq u_i \in \mathbb{Z}$ , can be solved in time  $2^{\mathcal{O}(r^{1+\alpha+o(1)})} \cdot \text{poly}(rc)$ , assuming that  $\|\mathbf{A}\|_\infty = \mathcal{O}(r)$  and  $\delta = 2^{\mathcal{O}(r)}$ , where  $\delta = \max_{i \in [c]}(u_i - \ell_i)$ .*

**Proof.** We construct a CLOSEST STRING instance following Theorem 11, and then by applying the assumed algorithm for CLOSEST STRING on the obtained instance, we obtain an algorithm for the original ILP FEASIBILITY with running time:

$$2^{\mathcal{O}(n^{1+\alpha})} \cdot \text{poly}(n\ell),$$

where  $n = \mathcal{O}(\log \|\mathbf{A}\|_\infty \cdot r)$ ,  $\ell = \mathcal{O}(\|\mathbf{A}\|_\infty \cdot \log^2 \|\mathbf{A}\|_\infty \cdot r \cdot c \cdot \delta)$ . This can be expressed as

$$\begin{aligned} 2^{\mathcal{O}(n^{1+\alpha})} \text{poly}(n\ell) &= 2^{\mathcal{O}((\log \|\mathbf{A}\|_\infty \cdot r)^{1+\alpha})} \cdot (\|\mathbf{A}\|_\infty \cdot r \cdot c \cdot \delta)^{\mathcal{O}(1)} \\ &= 2^{\mathcal{O}(r^{1+\alpha+o(1)})} \cdot r^{\mathcal{O}(1)} \cdot c^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(r)} \\ &= 2^{\mathcal{O}((r^{1+\alpha+o(1)})+1)} \cdot \text{poly}(rc). \\ &= 2^{\mathcal{O}(r^{1+\alpha+o(1)})} \cdot \text{poly}(rc). \end{aligned}$$

From Theorem 1 in [34], and the chain of reductions in the proof of Theorem 11, we can also get the following hardness result for CLOSEST STRING, that at least the running time of  $2^{\mathcal{O}(n \log n)} \cdot \text{poly}(n\ell)$  is necessary, similarly to ILP FEASIBILITY with unbounded variables.

► **Corollary 19.** *Assuming ETH, there is no algorithm that solves CLOSEST STRING in time  $2^{o(n \log n)} \cdot \text{poly}(n\ell)$ .*

## 4 Vertex Cover

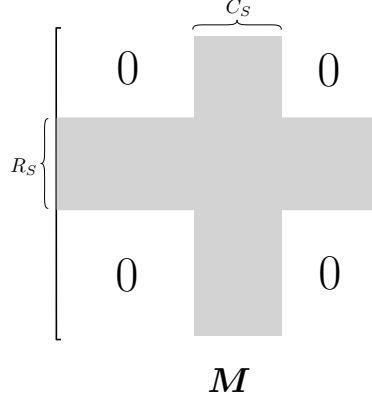
In this section, we describe an FPT algorithm for  $k$ -CENTER WITH MISSING ENTRIES parameterized by the vertex cover number of the incidence graph,  $G_M$ , derived from mask matrix  $M$ . More formally, we prove the following:

► **Theorem 1.**  *$k$ -CENTER WITH MISSING ENTRIES admits an algorithm with running time*

$$2^{\mathcal{O}(k \cdot \text{vc}(G_M) + \text{vc}(G_M)^2 \cdot \log \text{vc}(G_M))} \text{poly}(nm).$$

**Proof.** Let  $S$  be a minimum vertex cover of the incidence graph  $G_M$ . Define sets  $R_S = (S \cap R_M)$  and  $C_S = (S \cap C_M)$ , to represent rows and coordinates of  $M$  that are in the vertex cover  $S$ . Respectively, define  $\overline{C_S} = C_M \setminus C_S$ , and  $\overline{R_S} = R_M \setminus R_S$ , as their complements. Note that, by definition of  $G_M$ , for every  $r \in \overline{R_S}$  and  $c \in \overline{C_S}$ , it follows that  $M[r][c] = 0$ . To see this, assume there exist  $r_0 \in \overline{R_S}$ , and  $c_0 \in \overline{C_S}$  such that  $M[r_0][c_0] = 1$ . Then, the edge  $(r_0, c_0)$  exists in  $G_M$  that is not covered by  $S$ , as neither  $r_0$  nor  $c_0$  belong to  $S$ , which contradicts that  $S$  is a vertex cover of  $G_M$ . Thus, the cost of clustering only depends on the rows and coordinates in  $S$ , i.e.  $(R_S \cap C_S)$ . We refer  $R_S$ , and  $\overline{R_S}$  as *long rows*, and *short rows*

respectively, as the latter has known values only along the coordinates in  $C_S$ . See Figure 2 for an illustration of the instance structure; rows and coordinates may need to be reordered to reflect this structure, but this does not affect the problem's objective or the algorithm. We also define the following important mappings:



■ **Figure 2** The mask matrix  $M$ . The non-zero entries are distributed only within  $R_S$  and  $C_S$ . For every  $r \in \overline{R_S}$  and  $c \in \overline{C_S}$  it holds that  $M[r][c] = 0$ .

- Partial cluster assignment;  $\mathbf{part} : R_S \rightarrow [k]$ ,
- Partial center assignment;  $\mathbf{cent} : [k] \times C_S \rightarrow \{0, 1\}$

Intuitively,  $\mathbf{part}$  assigns each long row  $r \in R_S$  to one of the  $k$  clusters, and  $\mathbf{cent}$  assigns value to the centers of *all* clusters, along the coordinates in  $C_S$ . We call a pair  $(\mathbf{part}, \mathbf{cent})$ , a *partial assignment* and say it is *valid* if,  $d_H((\mathbf{A} \circ \mathbf{M})[r][C_S], \mathbf{cent}[\mathbf{part}[r], C_S]) \leq d$ , for every  $r \in R_S$ . Let  $P$  denote the set of all possible partial assignments. For a valid  $(\mathbf{part}, \mathbf{cent}) \in P$ , and cluster  $j \in [k]$ , define  $T_{\mathbf{part}, j} = \{r \in R_S \mid \mathbf{part}[r] = j\}$ , as the set of *long* rows assigned to cluster  $j$  by  $\mathbf{part}$ . We denote the cardinality of  $T_{\mathbf{part}, j}$  by  $p_j$ .

With the necessary definitions set, we now proceed to outline Algorithm 1. Note that, in order to ensure a correct cluster assignment, we must ensure that each long row lies within distance  $d$  of its cluster center along coordinates  $C_M$ , and similarly that each short row lies within distance  $d$  along  $C_S$ . Broadly, Algorithm 1 can be divided into the following key steps:

1. First, it establishes a valid partial assignment for the *long* rows along coordinates in  $C_S$ .
2. Then, for the clusters that have been assigned to the *long* rows, it tries to determine if there is a valid center assignment along the remaining coordinates,  $\overline{C_S}$ .
3. Finally, it decides whether a valid cluster assignment is also possible for the *short* rows.

We start by obtaining the minimum vertex cover of  $G_M$ . Then we establish a valid partial assignment  $(\mathbf{part}, \mathbf{cent})$  for the *long* rows along coordinates in  $C_S$  (*lines 1-4*). Consequently, the remaining distance that each long row  $r_i \in T_{\mathbf{part}, j}$  can have to its respective cluster center along  $\overline{C_S}$  is bounded by  $d'_i = d - d_H((\mathbf{A} \circ \mathbf{M})[r_i][C_S], \mathbf{cent}[j, C_S])$  (*lines 5-12*).

Next, for clusters that have been assigned to the *long* rows, we try to determine if there exists a valid center assignment along the remaining coordinates,  $\overline{C_S}$ . To do so, we solve a binary NON-UNIFORM CLOSEST STRING instance (denoted as *NUCS* on line 14) for each *non-empty* cluster  $j \in [k]$ , with the distance vector  $\mathbf{d}'_j = (d'_1, \dots, d'_{p_j})$ , and the strings  $r'_j = \mathbf{A}[r_i][\overline{C_S}]$ , for  $i \in [p_j]$  (*lines 13-16*). At this stage, we have a valid cluster assignment for the long rows, and the final task is to assign the short rows to appropriate clusters.

Finally, we decide whether a valid cluster assignment is also possible for the *short* rows. So, we assign each  $r \in \overline{R_S}$  to the smallest  $j \in [k]$  such that  $d_H((\mathbf{A} \circ \mathbf{M})[r][C_S], \text{cent}[j, C_S]) \leq d$  (lines 17-24). If at this step, we can assign all the *short* rows ( $\overline{R_S}$ ) to some cluster center defined by  $\text{cent}$ , then the given  $k$ -CENTER WITH MISSING ENTRIES instance is feasible (lines 25-26). Otherwise, we repeat the process for another valid partial assignment.

**For correctness:** First assume there is a solution, we show that our algorithm correctly finds it. Let  $\text{part}^* : R_M \rightarrow [k]$  be the solution cluster assignment and  $\text{cent}^* : [k] \times C_M \rightarrow \{0, 1\}$  be the solution center assignment. Then for each cluster  $i \in [k]$  and each long row  $r \in R_S$  assigned to it, that is  $\text{part}^*[r] = i$ , it holds that

$$\begin{aligned} d_H((\mathbf{A} \circ \mathbf{M})[r], \text{cent}^*[i]) &= d_H((\mathbf{A} \circ \mathbf{M})[r][C_S], \text{cent}^*[i][C_S]) \\ &\quad + d_H((\mathbf{A} \circ \mathbf{M})[r][\overline{C_S}], \text{cent}^*[i][\overline{C_S}]) \leq d \end{aligned}$$

Since we consider all the possible valid partial assignments for long rows in our algorithm,  $\text{cent}^*[\text{part}^*[r]][C_S]$  is captured by at least one of the partial assignments  $(\text{part}, \text{cent}) \in P$ . In other words, there is at least one partial assignments  $(\text{part}, \text{cent}) \in P$  such that  $\text{part} = \text{part}^*|_{R_S}$  and  $\text{cent} = \text{cent}^*|_{C_S}$ , where  $|$  denotes restriction to a set. Thus, by definition, it holds that  $d_H((\mathbf{A} \circ \mathbf{M})[r][\overline{C_S}], \text{cent}^*[\text{part}^*[r]][\overline{C_S}]) \leq d_r$ . Consequently, the NON-UNIFORM CLOSEST STRING instance for cluster  $i$  and the long rows assigned to it, will correctly output feasible. As a result, for the long rows, we correctly output that a  $k$ -cluster with radius at most  $d$  is feasible. For the short rows, note that they have known entries only along coordinates  $C_S$ , so for each short row  $r \in R_S$  assigned to cluster  $i \in [k]$ , that is  $\text{part}^*[r] = i$ , we have:

$$\begin{aligned} d_H((\mathbf{A} \circ \mathbf{M})[r], \text{cent}^*[i]) &= d_H((\mathbf{A} \circ \mathbf{M})[r][C_S], \text{cent}^*[i, C_S]) \\ &= d_H((\mathbf{A} \circ \mathbf{M})[r][C_S], \text{cent}[i, C_S]) \leq d. \end{aligned}$$

Hence,  $r$  will at least be assigned to cluster  $i$ . Therefore if there is a solution to this instance of  $k$ -CENTER WITH MISSING ENTRIES, then our algorithm will successfully output feasible. The other direction can similarly be showed.

As of the running time, in the first step, we obtain the minimum vertex cover of  $G_M$  can be computed in time  $\mathcal{O}(1.2738^{\text{vc}(G_M)} + (\text{vc}(G_M) \cdot nm))$  by the state-of-the-art algorithm of [6]. For the next steps, note that there are at most  $k^{|R_S|} = k^{\mathcal{O}(\text{vc}(G_M))}$  possible mappings for  $\text{part}$  and  $2^{(|C_S| \cdot |J|)} = 2^{\mathcal{O}(\text{vc}(G_M) \cdot k)}$  for  $\text{cent}$ . The NON-UNIFORM CLOSEST STRING instances are solved via Theorem 9 in time  $m \cdot \text{vc}(G_M)^{\mathcal{O}(\text{vc}(G_M))}$ . Note that because the distance of each short row  $r \in \overline{R_S}$  to a cluster center can be checked in time  $(k' \cdot |C_S|) = \mathcal{O}(k \cdot \text{vc}(G_M))$ , the final step can be computed in time  $\mathcal{O}(m \cdot k \cdot \text{vc}(G_M) \cdot 2^{(\text{vc}(G_M) \cdot k)})$ . In total, the running time of the algorithm is dominated by  $2^{\mathcal{O}(k \cdot \text{vc}(G_M) + \text{vc}(G_M)^2 \cdot \log \text{vc}(G_M))} \text{poly}(nm)$ . ◀

## 5 Treewidth

This section is dedicated to a fixed-parameter algorithm for the problem parameterized by the treewidth of the incidence graph  $G_M$ , the number of clusters  $k$  and the maximum permissible radius of the cluster,  $d$ . We restate the formal result next for convenience.

► **Theorem 3.**  $k$ -CENTER WITH MISSING ENTRIES admits an algorithm with running time

$$d^{\mathcal{O}(\text{tw}(G_M))} 2^{\mathcal{O}(k \cdot \text{tw}(G_M))} \text{poly}(nm).$$

■ **Algorithm 1**  $k$ -CENTER WITH MISSING ENTRIES parameterized by  $\text{vc}(G_M)$ .

---

```

1  $S \leftarrow$  Minimum Vertex Cover of  $G_M$ 
2  $P \leftarrow \{(\text{part}, \text{cent}) \in (R_S \rightarrow [k]) \times ([k] \times C_S \rightarrow \{0, 1\})\}$ 
3 for  $(\text{part}, \text{cent}) \in P$  and  $\text{isValid}(\text{part}, \text{cent})$  do
4    $\text{soln} \leftarrow \text{true}$ 
5   for  $j \in [k]$  do
6      $T_j \leftarrow \{r \in R_S \mid \text{part}[r] = j\}$ 
7     if  $T_j = \emptyset$  then
8       continue
9     for  $r_i \in T_j$  do
10       $d'_i = d - d_H((A \circ M)[r_i][C_S], \text{cent}[j, C_S])$ 
11       $r'_i = A[r_i][C_S]$ 
12     $\mathbf{d}' \leftarrow (d'_1, \dots, d'_{p_j})$ 
13     $\mathbf{r}' \leftarrow (r'_1, \dots, r'_{p_j})$ 
14     $\text{soln} \leftarrow \text{soln} \wedge \text{NUCS}(k, \mathbf{d}', \mathbf{r}')$ 
15    if  $\neg \text{soln}$  then
16      break
17  if  $\text{soln}$  then
18     $\text{counter} \leftarrow 0$ 
19    for  $r \in \overline{R_S}$  do
20      for  $j \in [k]$  do
21        if  $d_H((A \circ M)[r][C_S], \text{cent}[j, C_S]) \leq d$  then
22           $\text{counter}++$ 
23          break
24      if  $\text{counter} = |\overline{R_S}|$  then
25        return YES
26 return NO

```

---

We briefly sketch the intuition of our approach. For a complete proof and discussion, refer to the full version. Informally, the fact that  $G_M$  has treewidth at most  $t$  means that the graph can be constructed in a tree-like fashion, where at each point only a vertex subset of size at most  $t$  is “active”. This small subset is called a *bag*, and in particular is a separator for the graph: there are no edges between the “past”, already constructed part of the graph and the “future”, not encountered yet part of the graph; all connections between these parts are via the bag itself. In terms of the  $k$ -CENTER WITH MISSING ENTRIES instance, this means that for each “past” row, all its entries that correspond to “future” columns are missing (as the respective entry of the mask matrix  $M$  has to be 0), and the same holds for “past” columns and “future” rows.

Our algorithm performs dynamic programming over this decomposition, supporting a collection of records for the current bag. For the rows of the current bag, we store the partition of the rows into clusters, and additionally for each row the distance to its center vector among the already encountered columns. For each cluster and for each column of the bag, we store the value of the cluster center in this column. These three characteristics (called together a *fragment*) act as a “trace” of a potential solution on the current bag. In

our DP table, we store whether there exists a partial solution for each choice of the fragment. Because of the separation property explained above, only knowing the fragments is sufficient for computing the records for every possible update on the bag. The claimed running time follows from upper-bounding the number of possible fragments for a bag of size at most  $t$ .

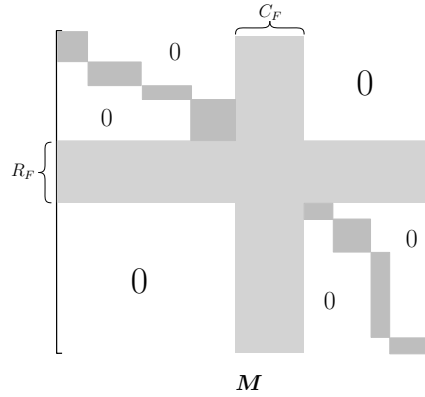
## 6 Fracture number

In this section, we present a fixed-parameter algorithm for the  $k$ -CENTER WITH MISSING ENTRIES, where the parameter is the fracture number of the incidence graph  $G_{\mathbf{M}}$ . To provide context, we first define the key concepts of *fracture modulator* and *fracture number*. Following this, we exploit the relevant results from Theorem 1 and Theorem 3, which lead to the proof of Theorem 2.

► **Theorem 2.**  *$k$ -CENTER WITH MISSING ENTRIES admits an algorithm with running time*

$$2^{\mathcal{O}(k \cdot \text{fr}(G_{\mathbf{M}}) + \text{fr}(G_{\mathbf{M}})^2 \cdot \log \text{fr}(G_{\mathbf{M}}))} \text{poly}(nm).$$

For a graph  $G = (V, E)$ , a *fracture modulator* is a subset of vertices  $F \subseteq V$  such that, after removing the vertices in  $F$ , each remaining connected component contains at most  $|F|$  vertices; the size of the smallest fracture modulator is denoted by  $\text{fr}(G)$ . Consider the incidence graph  $G_{\mathbf{M}}$  corresponding to the mask matrix  $\mathbf{M}$ , and let  $F$  represent the fracture modulator of  $G_{\mathbf{M}}$  with the smallest cardinality. We note that  $F$  can be computed in time  $\mathcal{O}((\text{fr}(G) + 1)^{\text{fr}(G)} nm)$  by the algorithm of [12]. Let  $R_F = (R_{\mathbf{M}} \cap F)$  and  $C_F = (C_{\mathbf{M}} \cap F)$  represent the row and column vertices of  $G_{\mathbf{M}}$  that belong to the fracture modulator  $F$ , respectively. Additionally, define  $\overline{R}_F = R_{\mathbf{M}} \setminus R_F$  and  $\overline{C}_F = C_{\mathbf{M}} \setminus C_F$ . We refer to the rows in  $R_F$  as *long* rows while the rows corresponding to  $\overline{R}_F$  are called *short* rows. See Figure 3 for an illustration of the structure of the instance. Now, the algorithm considers two cases.



■ **Figure 3** The rows  $R_F$  and columns  $C_F$  of the fracture modulator are in light gray, the “blocks” induced by the connected components of  $G_{\mathbf{M}} \setminus F$  are in dark gray. All entries of  $\mathbf{M}$  outside of the gray areas are 0.

$2 \text{fr}(G_{\mathbf{M}}) \leq d$ . In this case, each of the short rows has distance at most  $d$  to any fixed vector. This holds since for each row  $i \in \overline{R}_F$ ,  $\mathbf{M}[i][j] = 1$  only for  $j \in C_F$  or  $j$  in the same connected component of  $G_{\mathbf{M}} \setminus F$  as  $i$ ; this is at most  $2 \text{fr}(G_{\mathbf{M}}) \leq d$  entries, and all other entries in the row are missing. Therefore, the short rows are essentially irrelevant for the solution, as they can be assigned to any cluster with any center vector. We run the algorithm



of Theorem 1 on the instance restricted to the long rows  $R_F$ , and complement the resulting solution with an arbitrary assignment of the short rows to the  $k$  clusters. Since the vertex cover of the restricted instance is at most  $|R_F| \leq \text{fr}(G_M)$ , the running time bound holds as desired.

**$2 \text{fr}(G_M) > d$ .** Here, we use the algorithm of Theorem 3 to solve the given instance. We observe that a tree decomposition of width at most  $2 \text{fr}(G_M)$  can be constructed for  $G_M$  in a straightforward fashion: create a bag for each connected component of  $G_M \setminus F$  containing all vertices of this component together with  $F$ , and arrange these bags on a path in arbitrary order. Since  $d < 2 \text{fr}(G_M)$ , the running time of Theorem 3 gives the desired bound.

## 7 Conclusion

We have investigated the algorithmic complexity of  $k$ -CENTER WITH MISSING ENTRIES in the setting where the missing entries are sparse and exhibit certain graph-theoretic structure. We have shown that the problem is FPT when parameterized by  $\text{vc} + k$ ,  $\text{fr} + k$  and  $\text{tw} + k + d$ , where  $\text{vc}$  is the vertex cover number,  $\text{fr}$  is the fracture number, and  $\text{tw}$  is the treewidth of the incidence graph. In fact, it is not hard to get rid of  $k$  in the parameter; for example, in the enumeration of partial center assignments in the algorithm of Theorem 1 it can be assumed that  $k \leq 2^{\text{vc}}$ , as multiple centers that have the same partial assignment are redundant. However, this would increase the running time as a function of the parameter to doubly exponential, therefore we state the upper bounds with explicit dependence on  $k$ . It is, on the other hand, an interesting open question, whether  $d$  in the parameter is necessary for the algorithm parameterized by  $\text{tw}$ . As shown by [20], this is not necessary for  $k$ -MEANS WITH MISSING ENTRIES, since the problem admits an FPT algorithm when parameterized by treewidth alone. Yet, it does not seem that the dynamic programming approaches used in their work and in our work, can be improved to avoid the factor of  $d^{\mathcal{O}(\text{tw})}$  in the case of  $k$ -CENTER WITH MISSING ENTRIES. Therefore, it is natural to ask whether it can be shown that  $k$ -CENTER WITH MISSING ENTRIES is W[1]-hard in this parameterization.

Another intriguing open question is the tightness of our algorithm in the parameterization by the vertex cover number (and fracture number). On the one hand, the running time we show is  $2^{\text{vc}^{2+o(1)}} \cdot \text{poly}(nm)$  (for values of  $k$  in  $\mathcal{O}(\text{vc})$ ), exceeding the “natural” single-exponential in  $\text{vc}$  time, and we are not aware of a matching lower bound that is based on standard complexity assumptions. On the other hand, we show that improving this running time improves also the best-known running time for ILP FEASIBILITY when parameterized by the number of constraints and CLOSEST STRING parameterized by the number of strings. The latter are major open questions; [48] show also that several other open problems are equivalent to these. On the positive side, our algorithm in fact reduces  $k$ -CENTER WITH MISSING ENTRIES to just  $2^{\text{vc}}$  instances of ILP FEASIBILITY. Since practical ILP solvers are quite efficient, coupling our reduction with an ILP solver is likely to result in the running time that is much more efficient than prescribed by the upper bound of Theorem 1.

---

## References

- 1 Amir Abboud, Nick Fischer, Elazar Goldenberg, Karthik C. S., and Ron Safier. Can you solve closest string faster than exhaustive search? In *31st Annual European Symposium on Algorithms (ESA 2023)*, volume 274 of *LIPIcs*, pages 3:1–3:17, 2023. doi:10.4230/LIPIcs.ESA.2023.3.

- 2 Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, 2009. doi:10.1007/S10994-009-5103-0.
- 3 Daniel N. Baker, Vladimir Braverman, Lingxiao Huang, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in graphs of bounded treewidth. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119 of *Proceedings of Machine Learning Research*, pages 569–579, 2020. URL: <http://proceedings.mlr.press/v119/baker20a.html>.
- 4 Sayan Bandyapadhyay, Fedor V. Fomin, and Kirill Simonov. On coresets for fair clustering in metric and euclidean spaces and their applications. *J. Comput. Syst. Sci.*, 142:103506, 2024. doi:10.1016/J.JCSS.2024.103506.
- 5 Hans L. Bodlaender, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Yoshio Okamoto, Yota Otachi, and Tom C. van der Zanden. Subgraph isomorphism on graph classes that exclude a substructure. *Algorithmica*, 82:3566–3587, 2020. doi:10.1007/S00453-020-00737-Z.
- 6 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *Mathematical Foundations of Computer Science 2006*, pages 238–249. Springer Berlin Heidelberg, 2006. doi:10.1007/11821069\_21.
- 7 Raymond Chen. On mentzer’s hardness of the k-center problem on the euclidean plane. Technical Report 383, Dartmouth College, 2021. Technical Report. URL: [https://digitalcommons.dartmouth.edu/cs\\_tr/383/](https://digitalcommons.dartmouth.edu/cs_tr/383/).
- 8 Zhi-Zhong Chen, Bin Ma, and Lusheng Wang. Randomized and parameterized algorithms for the closest string problem. In *Combinatorial Pattern Matching - 25th Annual Symposium, CPM 2014, Moscow, Russia, June 16-18, 2014. Proceedings*, volume 8486, pages 100–109. Springer, 2014. doi:10.1007/978-3-319-07566-2\_11.
- 9 Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for k-median and k-means coresets. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*, pages 1038–1051, Rome, Italy, 2022. doi:10.1145/3519935.3519946.
- 10 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 11 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
- 12 Pavel Dvorák, Eduard Eiben, Robert Ganian, Dusan Knop, and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP: programs with few global variables and constraints. *Artif. Intell.*, 300:103561, 2021. doi:10.1016/J.ARTINT.2021.103561.
- 13 Eduard Eiben, Fedor V. Fomin, Petr A. Golovach, William Lochet, Fahad Panolan, and Kirill Simonov. EPTAS for  $k$ -means clustering of affine subspaces. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2649–2659, 2021. doi:10.1137/1.9781611976465.157.
- 14 Eduard Eiben, Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. From Data Completion to Problems on Hypercubes: A Parameterized Analysis of the Independent Set Problem. In *18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*, volume 285, pages 16:1–16:14, 2023. doi:10.4230/LIPIcs.IPEC.2023.16.
- 15 Eduard Eiben, Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. On the parameterized complexity of clustering problems for incomplete data. *J. Comput. Syst. Sci.*, 134:1–19, 2023. doi:10.1016/J.JCSS.2022.12.001.
- 16 Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. *ACM Trans. Algorithms*, 16:5:1–5:14, 2020. doi:10.1145/3340322.

- 17 Tomás Feder and Daniel H. Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 434–444, 1988. doi:10.1145/62212.62255.
- 18 Uriel Feige. Np-hardness of hypercube 2-segmentation. *CoRR*, abs/1411.0821, 2014. arXiv:1411.0821.
- 19 Moti Frances and Ami Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997. doi:10.1007/S002240000044.
- 20 Robert Ganian, Thekla Hamm, Viktoriia Korchemna, Karolina Okrasa, and Kirill Simonov. The complexity of k-means clustering when little is known. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162, pages 6960–6987, 2022. URL: <https://proceedings.mlr.press/v162/ganian22a.html>.
- 21 Robert Ganian, Iyad A. Kanj, Sebastian Ordyniak, and Stefan Szeider. Parameterized algorithms for the matrix completion problem. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80, pages 1642–1651, 2018. URL: <http://proceedings.mlr.press/v80/ganian18a.html>.
- 22 Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. On structural parameterizations of the edge disjoint paths problem. *Algorithmica*, 83:1605–1637, 2021. doi:10.1007/S00453-020-00795-3.
- 23 Leszek Gasieniec, Jesper Jansson, and Andrzej Lingas. Efficient approximation algorithms for the hamming center problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1999)*, pages 905–906. ACM/SIAM, 1999. URL: <http://dl.acm.org/citation.cfm?id=314500.315081>.
- 24 Tomáš Gavenčík, Martin Koutecký, and Dušan Knop. Integer programming in parameterized complexity: Five miniatures. *Discrete Optimization*, 44:100596, 2022. doi:10.1016/j.disopt.2020.100596.
- 25 Rong Ge, Martin Ester, Byron J. Gao, Zengjian Hu, Binay Bhattacharya, and Boaz Ben-Moshe. Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. *ACM Transactions on Knowledge Discovery from Data*, 2(2):1–35, 2008. doi:10.1145/1376815.1376816.
- 26 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.
- 27 Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for CLOSEST STRING and related problems. *Algorithmica*, pages 25–42, 2003. doi:10.1007/S00453-003-1028-3.
- 28 Pierre Hansen and Brigitte Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79(1):191–215, 1997. doi:10.1007/BF02614317.
- 29 Danny Hermelin and Liat Rozenberg. Parameterized complexity analysis for the closest string with wildcards problem. *Theoretical Computer Science*, pages 11–18, 2015. doi:10.1016/j.tcs.2015.06.043.
- 30 Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979. doi:10.1016/0166-218X(79)90044-1.
- 31 Debajyoti Kar, Mert Kusan, Debmalya Mandal, Sourav Medya, Arlei Silva, Palash Dey, and Swagato Sanyal. Feature-based individual fairness in k-clustering. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*, pages 2772–2774, 2023. doi:10.5555/3545946.3599073.
- 32 Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structural parameters, tight bounds, and approximation for (k,r)-center. *Discrete Applied Mathematics*, pages 90–117, 2019. doi:10.1016/j.dam.2018.11.002.
- 33 Dusan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Math. Program.*, 184:1–34, 2020. doi:10.1007/S10107-019-01402-2.

- 34 Dusan Knop, Michal Pilipczuk, and Marcin Wrochna. Tight complexity lower bounds for integer linear programming with few constraints. *ACM Trans. Comput. Theory*, 12:19:1–19:19, 2020. doi:10.1145/3397484.
- 35 Yuval Kochman, Arya Mazumdar, and Yury Polyanskiy. The adversarial joint source-channel problem. In *Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012*, pages 2112–2116. IEEE, 2012. doi:10.1109/ISIT.2012.6283735.
- 36 Michael Lampis and Valia Mitsou. Fine-grained meta-theorems for vertex integrity. *Log. Methods Comput. Sci.*, 20, 2024. doi:10.46298/LMCS-20(4:18)2024.
- 37 J. Kevin Lanctôt, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, 2003. doi:10.1016/S0890-5401(03)00057-9.
- 38 Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002. doi:10.1145/506147.506150.
- 39 Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982. doi:10.1109/TIT.1982.1056489.
- 40 Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. In *Research in Computational Molecular Biology, 12th Annual International Conference, RECOMB 2008, Singapore, March 30 - April 2, 2008. Proceedings*, volume 4955, pages 396–409. Springer, 2008. doi:10.1007/978-3-540-78839-3\_33.
- 41 Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. *SIAM J. Comput.*, pages 1432–1443, 2009. doi:10.1137/080739069.
- 42 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar  $k$ -means problem is np-hard. In *Proceedings of the 3rd International Workshop on Algorithms and Computation (WALCOM)*, pages 274–285, 2009. doi:10.1007/978-3-642-00202-1\_24.
- 43 Arya Mazumdar, Yury Polyanskiy, and Barna Saha. On chebyshev radius of a set in hamming space and the closest string problem. In *Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey*, pages 1401–1405. IEEE, 2013. doi:10.1109/ISIT.2013.6620457.
- 44 Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984. doi:10.1137/0213014.
- 45 Stuart G. Mentzer. Approximability of metric clustering problems, 2016. Unpublished manuscript, March.
- 46 Jacob Munson, Breschine Cummins, and Dominique Zosso. An introduction to collaborative filtering through the lens of the netflix prize. *Knowledge and Information Systems*, 67:3049–3098, 2025. doi:10.1007/s10115-024-02315-z.
- 47 Murray Patterson, Tobias Marschall, Nadia Pisanti, Leo van Iersel, Leen Stougie, Gunnar W. Klau, and Alexander Schönhuth. WhatsHap: Weighted haplotype assembly for future-generation sequencing reads. *Journal of Computational Biology*, 22(6):498–509, February 2015. doi:10.1089/CMB.2014.0157.
- 48 Lars Rohwedder and Karol Węgrzycki. Fine-Grained Equivalence for Problems Related to Integer Linear Programming. In *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325, pages 83:1–83:18, 2025. doi:10.4230/LIPIcs.ITCS.2025.83.
- 49 Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. doi:10.1109/34.868688.
- 50 Nikola Stojanovic, Piotr Berman, Deborah Gumucio, Ross Hardison, and Webb Miller. A linear-time algorithm for the 1-mismatch problem. In *Proceedings of the 5th International Workshop on Algorithms and Data Structures, WADS '97*, pages 126–135. Springer-Verlag, 1997. doi:10.1007/3-540-63307-3\_53.
- 51 Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Data mining cluster analysis: basic concepts and algorithms. *Introduction to Data Mining*, 487:533, 2013.
- 52 Xiaoliang Wu, Qilong Feng, Ziyun Huang, Jinhui Xu, and Jianxin Wang. New algorithms for distributed fair  $k$ -center clustering: Almost accurate as sequential algorithms. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024)*, pages 1938–1946, 2024. doi:10.5555/3635637.3663057.