Finding a Cluster in Incomplete Data

Eduard Eiben
Department of Computer Science, Royal Holloway, University of London, Egham, UK

Robert Ganian
Algorithms and Complexity Group, TU Wien, Austria

Iyad Kanj
School of Computing, DePaul University, Chicago, IL, USA

Sebastian Ordyniak
University of Leeds, School of Computing, Leeds, UK

Stefan Szeider
Algorithms and Complexity Group, TU Wien, Austria

Abstract

We study two variants of the fundamental problem of finding a cluster in incomplete data. In the problems under consideration, we are given a multiset of incomplete d-dimensional vectors over the binary domain and integers k and r, and the goal is to complete the missing vector entries so that the multiset of complete vectors either contains (i) a cluster of k vectors of radius at most r, or (ii) a cluster of k vectors of diameter at most r. We give tight characterizations of the parameterized complexity of the problems under consideration with respect to the parameters k, r, and a third parameter that captures the missing vector entries.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized complexity, incomplete data, clustering

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.47

Funding *Robert Ganian*: Project No. Y1329 of the Austrian Science Fund (FWF).
Sebastian Ordyniak: Project EP/V00252X/1 of the Engineering and Physical Sciences Research Council (EPSRC).
Stefan Szeider: Project No. P32441 of the Austrian Science Fund (FWF) and Project No. ICT19-065 of the Vienna Science and Technology Fund (WWTF).

1 Introduction

We consider two formulations of the fundamental problem of finding a sufficiently large cluster in incomplete data [2, 3, 18, 24]. In the setting under consideration, the input is a multiset M of d-dimensional Boolean vectors – regarded as the rows of a matrix, some of whose entries might be missing – and two parameters $k, r \in \mathbb{N}$. In the first problem under consideration, referred to as Diam-Cluster-Completion, the goal is to decide whether there is a completion of M that admits a multiset of k vectors (which we call a k-cluster) of diameter at most r; that is, a k-cluster such that the Hamming distance between any two cluster-vectors is at most r. In the second problem, referred to as Rad-Cluster-Completion, the goal is to decide whether there is a completion of M that admits a k-cluster of radius at most r; that is, a k-cluster such that there is a *center* vector $\vec{s} \in \{0, 1\}^d$ with Hamming distance at most r to each cluster-vector.

The cluster-diameter and cluster-radius are among the most widely used measures for intra-cluster similarity [5, 8, 12, 15, 16, 17, 18]. Our interest in studying these problems stems from the recent relevant research within the theory community [9, 14, 19, 20, 21], as well as the ubiquitous presence of incomplete data in relevant areas, such as recommender systems, machine learning, computer vision, and data science [1, 10, 11, 28].

We study the parameterized complexity of the above problems with respect to the two parameters k, r, and a third parameter that captures the occurrence of missing vector entries. Naturally, parameterizing by the number of missing entries alone is not desirable since one would expect that number to be rather large. In their recent related works on clustering problems, Koana, Froese, and Niedermeier [20, 21] restricted the occurrence of missing entries by using the maximum number of missing entries per row as the parameter. Another parameter for restricting the occurrence of missing entries is the minimum number of vectors plus coordinates needed to "cover" all missing entries, which was proposed and used by Eiben et al. [9] and Ganian et al. [14], who studied various data completion and clustering problems. In this paper, we propose and use a parameter that unifies and subsumes both previous parameterizations: the "*deletion distance to near-completion*", denoted $\lambda(M)$, which is the minimum integer p such that at most p vectors can be removed from M so that every remaining vector contains at most p missing entries. Clearly, the parameter $\lambda(M)$ is computable in polynomial time and is not larger than any of (and hence subsumes) the two parameters considered by Koana et al. [20, 21], Eiben et al. [9], and Ganian et al. [14].

Results and Techniques. We perform an in-depth analysis of the two considered data completion problems w.r.t. the aforementioned parameterizations. We obtain results that provide a nearly complete complexity landscape of these problems. An overview of our results is provided in Table 1. As a byproduct, our results establish that both problems under consideration are fixed-parameter tractable parameterized by $k + r$ when the data is complete, which answers an open question in the literature [2, 3].

Table 1 Overview of the results obtained in this paper.

	k	r	$k+r$	$k+\lambda$	$r+\lambda$	$k+r+\lambda$
DIAM-CLUSTER-C.	W[1]-h/XP	paraNP-c	W[1]-h/XP	W[1]-h/XP	FPT	FPT
RAD-CLUSTER-C.	W[1]-h/XP	paraNP-c	W[1]-h/XP	?/XP	?/XP	FPT

We summarize the new results obtained in this paper below.

1. We show that DIAM-CLUSTER-COMPLETION is fixed-parameter tractable (FPT) parameterized by $r + \lambda(M)$ (Theorem 9). The significance of the above result is in removing the dependency on the cluster size k in the running time of the algorithm, thus showing that finding a large cluster in incomplete data can be feasible when both the cluster diameter and the parameter $\lambda(M)$ are small. This result is the pinnacle of our technical contributions and relies on two ingredients: a fixed-parameter algorithm for the same problem parameterized by $k + r + \lambda(M)$ (Theorem 1), which is then used as a subroutine in the main algorithm, and a new technique that we dub *iterative sunflower harvesting*. Crucial to this new technique is a general structural lemma, allowing us to represent a family of sets in a succinct manner in terms of sunflower cores, which we believe to be interesting in its own right. We note that the use of sunflowers to obtain a succinct set representation (leading to a kernel) is not uncommon in such settings [20, 21, 22, 25]. What makes the sunflower harvesting technique novel is that it allows us to (1) show

that each solution can be covered by a small number of sunflowers, and to (2) iteratively "harvest" these sunflowers to obtain a solution in boundedly-many (in the parameter) branching steps.

2. We give an XP-algorithm for Diam-Cluster-Completion parameterized by k alone (Theorem 10). Together with Theorem 11 (showing the W[1]-hardness of Diam-Cluster w.r.t. k) and Theorem 12 (showing the W[1]-hardness for Diam-Cluster-Completion w.r.t. k even for $r = 0$), this gives a complete complexity landscape for Diam-Cluster-Completion parameterized by any combination of the parameters k, r, and $\lambda(M)$.
3. We show that Rad-Cluster-Completion is FPT parameterized by $k + r + \lambda(M)$ (Theorem 15); this result answers open questions in the literature [2, 3], which asked about the fixed-parameter tractability of the easier complete version of the problem (i.e., when $\lambda(M) = 0$).
4. We provide an XP-algorithm for Rad-Cluster-Completion parameterized by $r+\lambda(M)$ in which the degree of the polynomial in the runtime has only a logarithmic dependence on r (Theorem 16). We remark that the problem is in XP parameterized by k alone (Observation 13).
5. We provide an accompanying W[1]-hardness result for Rad-Cluster-Completion that rules out its fixed-parameter tractability when parameterized by $k + r$ (Theorem 12). Since the problem is NP-hard for fixed r (as also follows from Theorem 12), this leaves only two questions open for the considered parameterizations: whether Rad-Cluster-Completion is FPT when parameterized by either $k + \lambda(M)$ or by $r + \lambda(M)$.
6. We give an FPT-approximation scheme for the optimization version (w.r.t. the cluster size) of Rad-Cluster-Completion.

Related Work

The Rad-Cluster problem (i.e., Rad-Cluster-Completion for complete data) and variants of it were studied as early as the 1980's, albeit under different names. Dyer and Frieze presented a heuristic algorithm for approximating a variant of Rad-Cluster, referred to as the p-Center problem, where the goal is to compute $p \in \mathbb{N}$ clusters, each of radius at most r, that contain all vectors of M; hence, Rad-Cluster corresponds to the case of p-Center where $p = 1$ and $k = |M|$ (i.e., when the cluster contains all vectors in M). Cabello et al. [4] studied the parameterized complexity of the geometric p-Center Problem in $\mathbb{R}^d$.

Frances and Litman [13] studied the complexity of Rad-Cluster with $k = |M|$, in the context of computing the radius of a binary code; they referred to it as the Covering Radius problem and showed it to be NP-hard. Gąsieniec et al. [15, 16] studied (the optimization versions of) Rad-Cluster and Diam-Cluster with $k = |M|$ and obtained polynomial-time algorithms as well as lower bounds for a number of cases. They also obtained 2-approximation algorithms for these problems by extending an earlier algorithm by Gonzalez [17].

The Rad-Cluster problem restricted to the subcase of $k = |M|$ was also extensively studied under the nomenclature Closest String. Li et al. [24] showed that the problem admits a polynomial time approximation scheme if the goal is to minimize r. Gramm et al. [18] studied Closest String from the parameterized complexity perspective and showed it to be fixed-parameter tractable parameterized by r. Following this naming convention, Boucher and Ma [2], and Bulteau and Schmid [3] studied the parameterized complexity of Rad-Cluster under the nomenclature Closest String with Outliers. They considered several parameters, including some of the parameters under consideration in this paper. Notably, the restriction of our fixed-parameter algorithm for Rad-Cluster parameterized by $k + r + \lambda$ to the subcase where $\lambda = 0$ answers an open question in [2, see, e.g., Table 1]. Moreover, our XP algorithm for Rad-Cluster-Completion provided in Theorem 16 that has a run-time in which the degree of the polynomial has only a logarithmic dependence on r, immediately implies an algorithm of the same running time for Closest String with Outliers, as a special case.

For incomplete data, Hermelin and Rozenberg [19] studied the parameterized complexity of Rad-Cluster-Completion for $k = |M|$ under the nomenclature Closest String with Wildcards problem, with respect to several parameterizations. Very recently, Koana et al. [20] revisited the earlier work of Hermelin and Rozenberg [19] and obtained, among other results, a fixed-parameter algorithm for the problem parameterized by r plus the maximum number of missing entries per row. Even more recently, the same group [21] also studied a problem related to Diam-Cluster-Completion for $k = |M|$. They obtain a classical-complexity classification w.r.t. constant lower and upper bounds on the diameter and the maximum number of missing entries per row.

2 Preliminaries

We assume basic familiarity with parameterized complexity, including the classes W[1], FPT, XP, as well as Turing kernelization and FPT-approximation schemes [7, 6, 27].

Vector Terminology. Let $\vec{a}$ and $\vec{b}$ be two vectors in $\{0, 1, \square\}^d$, where $\square$ is used to represent coordinates whose value is unknown (i.e., missing entries). We denote by $\Delta(\vec{a}, \vec{b})$ the set of coordinates in which $\vec{a}$ and $\vec{b}$ are guaranteed to differ, i.e., $\Delta(\vec{a}, \vec{b}) = \{\, i \mid (\vec{a}[i] = 1 \wedge \vec{b}[i] = 0) \vee (\vec{a}[i] = 0 \wedge \vec{b}[i] = 1) \,\}$, and we denote by $\delta(\vec{a}, \vec{b})$ the *Hamming distance* between $\vec{a}$ and $\vec{b}$ measured only between known entries, i.e., $|\Delta(\vec{a}, \vec{b})|$. Moreover, for a subset $D' \subseteq [d]$ of coordinates, where $[d] = \{1, \ldots, d\}$, we denote by $\vec{a}[D']$ the vector $\vec{a}$ restricted to the coordinates in D'.

There is a one-to-one correspondence between vectors in $\{0, 1\}^d$ and subsets of coordinates, i.e., for every vector, we can associate the unique subset of coordinates containing all its one-coordinates and vice-versa. It will be useful to represent a vector by the set of coordinates where the vector has the value 1. We introduce the following notation for vectors to switch between their set-representation and vector-representation. We denote by $\Delta(\vec{a})$ the set $\Delta(\vec{0}, \vec{a})$. We extend this notation to sets of vectors as follows: for a set N of vectors in $\{0, 1, \square\}^d$, we denote by $\Delta(N)$ the set $\{\, \Delta(\vec{v}) \mid \vec{v} \in N \,\}$. We say that a vector $\vec{a} \in \{0, 1, \square\}^d$ is a *t-vector* if $|\Delta(\vec{a})| = t$ and we say that $\vec{a}$ *contains* a subset S of coordinates if $S \subseteq \Delta(\vec{a})$.

We say that a multiset[1] $M^* \subseteq \{0, 1\}^d$ is a *completion* of a multiset $M \subseteq \{0, 1, \square\}^d$ if there is a bijection $\alpha : M \to M^*$ such that for all $\vec{a} \in M$ and all $i \in [d]$ it holds that either $\vec{a}[i] = \square$ or $\alpha(\vec{a})[i] = \vec{a}[i]$. For a multiset M of vectors over $\{0, 1, \square\}^d$, we let *the deletion distance to near-completion*, $\lambda(M)$, denote the minimum integer such that there exists a subset $D_M \subseteq M$ with the following properties: (a) $|D_M| \leq \lambda(M)$, and (b) every vector in $M \setminus D_M$ contains at most $\lambda(M)$ missing entries. We call D_M the *deletion (multi-)set*, and observe that $\lambda(M)$ along with a corresponding deletion set can be trivially computed from M in linear time.

A *sunflower* in a set family $\mathcal{F}$ is a subset $\mathcal{F}' \subseteq \mathcal{F}$ such that all pairs of elements in $\mathcal{F}'$ have the same intersection. We will say that a multiset P is a *DIAM-Cluster* (or $|P|$-*DIAM-Cluster*) if $\delta(\vec{p}, \vec{q}) \leq r$ for every pair $\vec{p}, \vec{q} \in P$. Similarly, P is a *RAD-Cluster* (or $|P|$-*RAD-Cluster*) if there exists a vector $\vec{c} \in \{0, 1\}^d$ such that $\delta(\vec{c}, \vec{p}) \leq r$ for every $\vec{p} \in P$.

[1] We remark that, in the interest of brevity and when clear from context, we will sometimes use standard set notation such as $A \subseteq B$ in conjunction with multisets.

In all problems under consideration, the input size is considered to be $|M|$, i.e., the size of the matrix including multiplicities.

3 Finding a DIAM-Cluster in Incomplete Data

In this section, we present our results for DIAM-CLUSTER-COMPLETION. Our main algorithmic results are that DIAM-CLUSTER-COMPLETION is FPT parameterized by $r + \lambda(M)$ and is in XP parameterized by k alone. Together with Theorem 11 (showing the W[1]-hardness of DIAM-CLUSTER parameterized by k) and Theorem 12 (showing the W[1]-hardness of DIAM-CLUSTER-COMPLETION parameterized by k even for $r = 0$), this gives a complete complexity landscape for DIAM-CLUSTER-COMPLETION parameterized by any combination of the parameters k, r, and $\lambda(M)$.

3.1 DIAM-CLUSTER-COMPLETION Parameterized by $k + r + \lambda(M)$

We start by showing that DIAM-CLUSTER-COMPLETION parameterized by $k + r + \lambda(M)$ is FPT. We will later show a stronger result, namely that the same result already holds if we only parameterize by $r + \lambda(M)$. Showing the weaker result here is important for the following reasons: (1) we use the algorithm presented here as a subroutine in our result for the parameterization $r + \lambda(M)$, (2) the techniques developed here can also be employed for RAD-CLUSTER-COMPLETION, and (3) we obtain a Turing kernel of size polynomial in k.

The main approach behind the Turing kernel is to guess two vectors of maximum distance in the desired cluster. This will allow us to pre-process the instance such that if the resulting instance contains too many vectors, then it has a solution. Note that this approach only works for the case that a solution contains at least two vectors from $M \setminus D_M$ (recall that D_M denotes the deletion set); otherwise, we can guess the at most one vector from $M \setminus D_M$ that is in the solution and remove all the other vectors from $M \setminus D_M$. Therefore, in all cases, we end up with a reduced instance with boundedly many vectors and we will then show that we can remove all but boundedly many coordinates while preserving solutions.

▶ **Theorem 1.** DIAM-CLUSTER-COMPLETION *parameterized by* $k + r + \lambda(M)$ *has a Turing-kernel containing at most* $n = k3^{2\lambda(M)+r} + \lambda(M) + 2$ *vectors, each having at most* $\max\{r(n-1) + \lambda(M), \binom{\lambda(M)}{2}(r+1)\}$ *coordinates.*

3.2 DIAM-CLUSTER-COMPLETION Parameterized by $r + \lambda(M)$

With Theorem 1 in hand, we can move on to establishing the fixed-parameter tractability of DIAM-CLUSTER-COMPLETION parameterized by $r + \lambda(M)$. At the heart of our approach lies a new technique for analyzing the structure of vectors through sunflowers in their set representations, which we dub *iterative sunflower harvesting.* We first preprocess the instance to establish some basic properties. We then show a general result about sunflowers that allows us to derive a succinct representation of the solution cluster – in particular, this guarantees that the hypothetical solution can be described by a bounded number of sunflower cores. Finally, we proceed to "harvesting" these sunflowers cores using a branching procedure, thus computing the solution cluster.

3.2.1 Preprocessing the Instance

Let (M, k, r) be an instance of DIAM-CLUSTER-COMPLETION, let M^* be a completion of M that contains a maximum size DIAM-Cluster, and fix P^* to be such a maximum size DIAM-Cluster in M^*. We also fix D_M to be a $\lambda(M)$-deletion set. The goal of the algorithm is to find M^* and P^*.

If $k < \lambda(M) + 2$, then we can use the algorithm in Theorem 1 to obtain a Turing kernel whose size is a function of $r + \lambda(M)$, which, in turn, would imply that DIAM-CLUSTER-COMPLETION if FPT parameterized by $r + \lambda(M)$, thus giving the desired result. We assume henceforth that P^* contains the completion of at least two vectors in $M \setminus D_M$. We can guess the subset P_R of D_M that will be completed to vectors in P^*, and restrict our attention to finding a DIAM-Cluster in $M \setminus D_M$ of size $|P^* \setminus D_M|$. We will do so by enumerating all such DIAM-Cluster's in $M \setminus D_M$ and at the end check whether one of them, together with P_R, can be extended into a DIAM-Cluster.

Therefore, we will focus on what follows on finding a cluster in $M \setminus D_M$. We first guess two vectors $\vec{v}$ and $\vec{u}$ in $M \setminus D_M$, together with their completions $\vec{v}^*$ and $\vec{u}^*$, respectively, such that $\vec{v}^*$ and $\vec{u}^*$ are both in P^* and are the farthest vectors apart in $P^* \setminus D_M$; fix $r_{\max} = \delta(\vec{v}^*, \vec{u}^*)$. We remove all other vectors that can be completed into $\vec{v}^*$ or $\vec{u}^*$, and reduce k accordingly; hence, we do not keep duplicates of the two vectors that we already know to be in P^*. We then normalize all vectors in M so that $\vec{v}^*$ becomes the all-zero vector, i.e., we replace $\vec{v}^*$ by the all-zero vector, and for every other vector $\vec{w} \neq \vec{v}$, we replace it with the vector $\vec{w}'$ such that $\vec{w}'[i] = 0$ if $\vec{v}^*[i] = \vec{w}[i]$, $\vec{w}'[i] = \square$ if $\vec{w}[i] = \square$, and $\vec{w}'[i] = 1$, otherwise. Finally, for each vector $\vec{w} \in M \setminus D_M$, we compute the set $\Lambda(\vec{w})$ of all completions of $\vec{w}$ at distance at most $r_{\max}$ from both $\vec{v}^*$ and $\vec{u}^*$. Note that $\Lambda(\vec{w})$ can be computed in $\mathcal{O}(2^{\lambda(M)} \cdot d)$ time for each vector in $M \setminus D_M$, where d is the dimension of the vectors in M. We then remove all vectors $\vec{w}$ with $\Lambda(\vec{w}) = \emptyset$ from $M \setminus D_M$.

We will extend the notation $\Lambda(\vec{w})$ to $\Lambda_C(\vec{w})$, for a multiset C of vectors in $\{0, 1\}^d$, such that $\Lambda_C(\vec{w})$ is the set of all completions of a vector $\vec{w}$ at distance at most $r_{\max}$ to all vectors in C. We are now ready to show that after normalizing the vectors in P^*, the multiset $P^* \setminus D_M$ satisfies certain structural properties that we refer to as an r-saturated subset (of M^*); these structural properties allow for a succinct representation of P^*.

3.2.2 Sunflower Fields or Representing Sets by Cores of Sunflowers

In this subsection, we provide the central component for our algorithm based on the iterative sunflower harvesting technique. Crucial to this component is a general structural lemma that allows us to represent a family of sets in a succinct manner in terms of sunflower cores, which we believe to be interesting in its own right. We first state the result in its most general form (for sets), and then show how to adapt it to our setting.

▶ **Definition 2.** *Let U be a universe, $\mathcal{B}$ a family of subsets of U and $\mathcal{A} \subseteq \mathcal{B}$. We say that $\mathcal{A}$ is an r-*saturated *subfamily of $\mathcal{B}$ (for $r \in \mathbb{N}$) if the following holds for every $t \in \mathbb{N}$ and every sunflower $S \subseteq \mathcal{A}$ containing at least $r + 1$ sets of cardinality t with core C: $\mathcal{A}$ contains every set $B \in \mathcal{B}$ of cardinality t such that $C \subseteq B$.*

Intuitively, this property states that $\mathcal{A}$ contains all sets in $\mathcal{B}$ which are super-sets of cores of every sufficiently-large sunflower in $\mathcal{A}$ (with sets of the same cardinality).

The connection of this set property to clusters is as follows. We will show that every maximal cluster is an r-saturated subset of M^*. Since P^* contains the all-zero vector $\vec{v}^*$, any vector in P^* contains at most r ones. Fix $t \in [r]$, and consider the set of all vectors in P^* containing exactly t ones. The above notion will allow to draw the following assumption: if the aforementioned set of vectors is large, then it must contain a large sunflower and all the vectors in M whose completions share the core of this sunflower *must be* in P. This property will subsequently allow us to represent every hypothetical solution using a bounded number of sunflowers, as we show next.

The crucial insight now is that every r-saturated subfamily containing only sets of bounded size admits a succinct representation, where we can completely describe the set via a bounded number of sunflower cores. This is made precise in the following lemma.

▶ **Lemma 3.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two families of sets of cardinality at most r' over universe U such that $\mathcal{A} \subseteq \mathcal{B}$. If $\mathcal{A}$ is an r-saturated subset of $\mathcal{B}$, then there is a set $\mathcal{S}$ of at most $(rr')^{r'}$ subsets of U such that $\mathcal{A}$ is equal to the set of all sets B in $\mathcal{B}$ satisfying that $S \subseteq B$ for some $S \in \mathcal{S}$. Moreover, for each set $S \in \mathcal{S}$, S is either the core of a sunflower in $\mathcal{A}$ with at least $r+1$ petals, or $|S| = r'$.*

We will now show how Lemma 3 can be employed in our setting. Let $M, M' \subseteq \{0,1\}^d$ with $M' \subseteq M$. Then M' is an *r-saturated subset* of M if $\Delta(M')$ is an r-saturated subset of $\Delta(M)$. Herein, one can think of M as being (a part of) the input matrix and M' as being an inclusion-wise maximal cluster; we will later show that this property guarantees that M' is an r-saturated subset of M. Using this definition, the following corollary now follows immediately from Lemma 3.

▶ **Corollary 4.** *Let $r', r \in \mathbb{N}$ and $M, M' \subseteq \{0,1\}^d$ be sets of r'-vectors such that M' is an r-saturated subset of M. There is a set $\mathcal{S}$ of at most $(rr')^{r'}$ subsets of $[d]$ such that M' is equal to the set of all vectors $\vec{m}$ in M satisfying $S \subseteq \Delta(\vec{m})$ for some $S \in \mathcal{S}$. Moreover, for each set $S \in \mathcal{S}$, S is either a core of a sunflower in $\Delta(M')$ with at least $r+1$ petals, or $|S| = r'$.*

We now can show that after normalizing the vectors in P^*, the multiset $P^* \setminus D_M$ is an r-saturated subset of M^*.

▶ **Lemma 5.** *Let (M, k, r) be an instance of* Diam-Cluster-Completion, *let M^* be a completion of M and let P^* be a DIAM-Cluster in M^* of maximum size such that $\vec{0} \in P^*$. Then for every $N \subseteq M^*$, $P^* \setminus N$ is an r-saturated subset of $M^* \setminus N$.*

Since $P^* \setminus N$ is an r-saturated subset of $M^* \setminus N$, by Corollary 4, applied separately for each $r' \in [r]$, there exists a set $\mathcal{S} = \{(S_1, r_1), \dots, (S_\ell, r_\ell)\}$, with $\ell \leq \sum_{r' \in [r]} (rr')^{r'} \leq r^{2r+1}$, such that $P^* \setminus N$ contains precisely all the vectors $\vec{w}$ in M^*, such that for some (S_i, r_i), $i \in [\ell]$, $S_i \in \Delta(\vec{w})$ and $|\Delta(\vec{w})| = r_i$.

We call the pair (S_i, r_i) an *r_i-center* (of $P^* \setminus N$ in $M^* \setminus N$). We say that a vector $\vec{w} \in \{0,1,\square\}^d$ is *compatible* with r_i-center (S_i, r_i) if there is a completion $\vec{w}^* \in \{0,1\}^d$ of $\vec{w}$, called *witness of compatibility*, such that $S_i \subseteq \Delta(\vec{w}^*)$ and $|\Delta(\vec{w}^*)| = r_i$. We say that $\vec{w} \in \{0,1,\square\}^d$ is *compatible* with $\mathcal{S}$ if it is compatible with some $(S_i, r_i) \in \mathcal{S}$.

The *size* of an r_i-center is the number of vectors that are compatible with it in M. Moreover, for a set $\mathcal{S} = \{(S_1, r_1), \dots, (S_\ell, r_\ell)\}$ and a multiset C of vectors from $\{0,1\}^d$, we say that $\mathcal{S}$ *defines* C, if every vector $\vec{c} \in C$ is compatible with $\mathcal{S}$ and for every $(S_i, r_i) \in \mathcal{S}$ there is a vector in C compatible with (S_i, r_i). We say that $\mathcal{S}$ *properly* defines C, if $|\mathcal{S}| \leq r^{2r+1}$, $\mathcal{S}$ defines C, and for every $(S_i, r_i) \in \mathcal{S}$ either:

- $|S_i| = r_i$ and the unique vector that is compatible with (S_i, r_i) is in C; or
- $|S_i| < r_i$ and C contains a set N of $r+1$ r_i-vectors such that $\Delta(N)$ forms a sunflower with core S_i.

Note that if every vector in C has at most $r_{\max}$ 1's, then since C is an r-saturated subset of C, it follows from Corollary 4 that there always exists a set $\mathcal{S}$ that properly defines C.

▶ **Observation 6.** *Let C be a multiset of vectors from $\{0,1\}^d$, with $\max_{\vec{c} \in C} |\Delta(\vec{c})| \leq r$. Then there exists a set $\mathcal{S}$ of at most r^{2r+1} r_i-centers that properly defines C.*

Suppose that we have a correct guess for $\mathcal{S}$, then we can already solve the problem as follows. For each $(S_i, r_i) \in \mathcal{S}$ such that $|S_i| = r_i$, there is only one possible r_i-vector that contains S_i. If our guess is correct, then P^* contains at least one vector that can be completed to this particular r_i-vector, and by maximality of P^*, P^* has to contain all such vectors. If $(S_i, r_i) \in \mathcal{S}$ such that $|S_i| < r_i$, then P^* contains a sunflower containing r_i-vectors of size at least $r+1$ whose core is S_i. Clearly, P^* contains all the vectors that can be completed to an r_i-vector containing S_i, since Lemma 5 holds for any completion M^* of M that contains P^* as a subset. The following lemma shows that all such vectors can be completed arbitrarily since all that matters is that their completion is compatible with $\mathcal{S}$.

▶ **Lemma 7.** *Let (M, k, r) be an instance of* Diam-Cluster-Completion*, M^* a completion of M, P^* a DIAM-Cluster in M^* of maximum size with $\vec{0} \in P^*$, and $N \subseteq M^*$. If $\mathcal{S}$ properly defines $P^* \setminus N$, then for every pair of vectors $\vec{w}_1, \vec{w}_2 \in \{0,1\}^d$ compatible with $\mathcal{S}$ it holds that $\delta(\vec{w}_1, \vec{w}_2) \le r$.*

It is easy to see that there are at most d^{r+1} choices for a pair (S_i, r_i) (i.e., a r_i-center), and hence we have at most $(d^{r+1})^{r^{2r+1}} = d^{\mathcal{O}(r^{2r+2})}$ possible choices for the set $\mathcal{S}$ that properly defines P^*. This bound already implies an XP-algorithm. To obtain a fixed-parameter algorithm, it suffices to find the correct guess for $\mathcal{S}$ in FPT-time, which is our next goal.

3.2.3 Iterative Sunflower Harvesting

We are now ready to describe the iterative sunflower harvesting procedure, which allows us to obtain the desired FPT-algorithm. Namely, we show that instead of enumerating all $d^{r^{2r+2}}$ possible sets $\mathcal{S}$ of r_i-centers to find the one that properly defines $P^* \setminus D_M$, it suffices to enumerate only $f(r, \lambda(M))$-many "important" r_i-centers for each choice of $\vec{v}^*$ and $\vec{u}^*$ (recall that $\vec{v}^*$ and $\vec{u}^*$ are the two fixed vectors in $P^* \setminus D_M$ that were guessed), where f is some function that depends only on r and $\lambda(M)$. Moreover, we can enumerate these possibilities in FPT-time.

We compute $\mathcal{S}$ by iteratively adding r_i-centers one by one. The main idea is to show that, for any partial solution $\mathcal{S}'$, there is a bounded number of choices for the next r_i-center to add. As a first step in this direction, the following lemma shows that for $\mathcal{S}'$, there is always a "large" r_i-center (S_i, r_i) that can be added to $\mathcal{S}'$, i.e., of size at least a $(2^r r^{2r+1})$-fraction of the remaining vectors. Before we state the lemma, we introduce the following notations. If $\vec{w}$ is compatible with $\mathcal{S}$, we will denote by $\zeta^{\mathcal{S}}(\vec{w})$ the set of witnesses of compatibility for $\vec{w}$ and $\mathcal{S}$. Recall that, for a vector $\vec{w} \in \{0, 1, \square\}^d$ and multiset C of vectors from $\{0,1\}^d$, $\Lambda_C(\vec{w})$ denotes the set of all completions of vector $\vec{w}$ at distance at most $r_{\max}$ to all vectors in C, i.e., $\max_{\vec{c} \in C}\{\delta(\vec{c}, \vec{c}_w)\} \le r_{\max}$.

▶ **Lemma 8.** *Let P^* be a maximum DIAM-Cluster in (M, k, r), $\mathcal{S}$ the set of r_i-centers that properly define $P^* \setminus D_M$, and $\mathcal{S}' \subseteq \mathcal{S}$. Moreover, let C' be the multiset of vectors $\vec{w}$ in $M \setminus D_M$ with $\zeta^{\mathcal{S}'}(\vec{w}) \neq \emptyset$ and C the multiset containing a vector $\vec{w}_c \in \zeta^{\mathcal{S}'}(\vec{w})$ for every $\vec{w} \in C'$. Finally, let M' be the multiset consisting of all the vectors $\vec{w} \in M \setminus (C \cup D_M)$ with $\Lambda_C(\vec{w}) \neq \emptyset$. Then there exists $(S_i, r_i) \subseteq \mathcal{S} \setminus \mathcal{S}'$ such that at least $(|M'|/2^{r_{\max}} - |D_M|)/r^{2r+1}$ vectors in M' are compatible with (S_i, r_i).*

Note that each normalised vector $\vec{w}$ can be compatible with at most $2^{r+\lambda(M)}$ r_i-centers (S_i, r_i), since $S_i \subseteq \Delta(\vec{w}^*)$ for some completion $\vec{w}^*$ of $\vec{w}$. Now it follows from a counting argument that the number of large r_i-centers is at most $2^{r+\lambda(M)}(2^r r^{2r+1}) = 2^{2r+\lambda(M)} r^{2r+1}$ and those can be enumerated in time $\mathcal{O}(2^{r+\lambda(M)}|M|)$. By Observation 6, $|\mathcal{S}|$ and hence the depth of the branching algorithm, is at most r^{2r+1}, which implies the following theorem.

▶ **Theorem 9.** Diam-Cluster-Completion *is fixed-parameter tractable parameterized by $r + \lambda(M)$.*

3.3 Diam-Cluster-Completion Parameterized by k

Here, we use an Integer Linear Programming subroutine to show that Diam-Cluster-Completion parameterized by k is in XP.

Moreover, we also observe in Theorem 11 that, unless W[1]=FPT, this cannot be improved to an FPT-algorithm even for complete data.

▶ **Theorem 10.** Diam-Cluster-Completion *is in* XP *parameterized by* k.

Proof. Let (M, k, r) be an instance of Diam-Cluster-Completion. The algorithm works by enumerating all potential clusters C of size exactly k, and then uses a reduction to an ILP instance with $f(k)$ variables to check whether C can be completed into a cluster. Since there are at most $|M|^k$ many potential clusters of size exactly k, it only remains to show how to decide whether a given set C of exactly k vectors in M can be completed into a DIAM-Cluster. Let M_C be the submatrix of M containing only the vectors in C. Then M_C has at most 3^k distinct columns, and moreover, each of those columns can be completed in at most 2^k possible ways. Let T be the set of all columns occurring in M_C and for a column $\vec{t} \in T$, let $F(\vec{t})$ be the set of all possible completions of $\vec{t}$, and let $\#(\vec{t}\,)$ denote the number of columns in M_c equal to $\vec{t}$. For a vector $\vec{f} \in \{0,1\}^k$ (representing the completion of a column), let $T(\vec{f})$ denote the subset of T containing all columns $\vec{t}$ with $\vec{f} \in F(\vec{t})$. Moreover, for every i and j with $1 \leq i < j \leq k$ (representing the i-th and the j-th vectors in C), we denote by $FD(i,j)$ the set of all vectors (completions of columns) $\vec{f} \in \{0,1\}^k$ such that $\vec{f}[i] \neq \vec{f}[j]$.

We are now ready to construct an ILP instance $\mathcal{I}$ with at most $3^k 2^k$ variables that is feasible if and only if C can be completed into a DIAM-Cluster. $\mathcal{I}$ has one variable $x_{\vec{t},\vec{f}}$ for every $\vec{t} \in T$ and every $\vec{f} \in F(\vec{t})$ whose value (in a feasible assignment) represents how many columns of type $\vec{t}$ in M_C will be completed to $\vec{f}$. Moreover, $\mathcal{I}$ has the following constraints:

- One constraint for every $\vec{t} \in T$ stipulating that every column of type $\vec{t}$ in M_C is completed in some manner:

$$\sum_{\vec{f} \in F(\vec{t})} x_{\vec{t},\vec{f}} = \#(\vec{t}\,).$$

- For every i and j with $1 \leq i < j \leq k$ (representing the i-th and the j-th vectors in C), one constraint stipulating that the Hamming distance between the i-th and the j-th vectors in C does not exceed r:

$$\sum_{\vec{f} \in FD(i,j) \vec{t} \in T(\vec{f})} x_{\vec{t},\vec{f}} \leq r.$$

This completes the construction of $\mathcal{I}$ and it is straightforward to verify that $\mathcal{I}$ has a feasible assignment if and only if C can be completed to a DIAM-Cluster. Since $\mathcal{I}$ has at most $3^k 2^k$ variables, and since it is well known that ILP can be solved in FPT-time w.r.t. the number of variables [23], $\mathcal{I}$ can be solved in FPT-time w.r.t. k. ◀

▶ **Theorem 11.** Diam-Cluster-Completion *is* W[1]*-hard parameterized by* k *even if* $\lambda(M) = 0$.

We note that our second result also establishes the W[1]-hardness of Rad-Cluster-Completion (since both problems coincide when $r = 0$).

▶ **Theorem 12.** Diam-Cluster-Completion *and* Rad-Cluster-Completion *are both* W[1]*-hard parameterized by* k *even if* $r = 0$.

4 Finding a RAD-Cluster in Incomplete Data

In this section, we present our results for RAD-CLUSTER-COMPLETION. We will show that RAD-CLUSTER-COMPLETION is FPT parameterized by $k + r + \lambda(M)$, and is in XP parameterized by $r + \lambda$ alone. Notably, the degree of the polynomial in the run-time of our XP algorithm grows only logarithmically in r and the algorithm can be employed to solve the CLOSEST STRING WITH OUTLIERS problem [2, 3].

Before proceeding to the main contributions of this section, we observe that, by combining the trivial branching procedure, in which we branch over all sets of k vectors from M (where in each branch we proceed under the assumption that all vectors outside of the set can be deleted), with a previous result of Hermelin and Rozenberg [19, Theorem 2], which solves the special case of RAD-CLUSTER-COMPLETION for $k = |M|$, we obtain:

▶ **Observation 13.** RAD-CLUSTER-COMPLETION *parameterized by k is in* XP*.*

Together with the previously-established Theorem 12 (showing the W[1]-hardness for RAD-CLUSTER-COMPLETION w.r.t. k even for $r = 0$), this gives us an almost complete picture of the parameterized complexity of RAD-CLUSTER-COMPLETION for any combination of the parameters k, r, $\lambda(M)$. The only two questions that remain open are whether the XP result for $r + \lambda(M)$ can be improved to an FPT-result (as this has been the case for DIAM-CLUSTER-COMPLETION), and whether it is possible to obtain an FPT-algorithm either for parameter k or $k + \lambda(M)$. As a first step in this direction, we present an FPT-approximation scheme for parameter $r + \lambda(M)$ in Section 4.3. The following observation will be useful:

▶ **Observation 14.** *Given a (complete) vector $\vec{s} \in \{0, 1\}^d$, in time $\mathcal{O}(|M|d)$ we can decide if $\vec{s}$ is the center of a* RAD-CLUSTER *of k vectors in M.*

Observation 14 is straightforward since we can find all vectors $\vec{w} \in M$ that can be completed to a vector $\vec{w}^*$ at distance at most r from $\vec{s}$ by letting $\vec{w}^*[i] = \vec{s}[i]$ wherever $\vec{w}^*[i] = \square$, and then decide whether $\vec{w}^*$ is such a vector by computing $\delta(\vec{w}^*, \vec{s})$.

4.1 RAD-CLUSTER-COMPLETION Parameterized by $k + r + \lambda(M)$

We start by showing that, as in the case of DIAM-CLUSTER-COMPLETION, RAD-CLUSTER-COMPLETION parameterized by $k + r + \lambda(M)$ has a Turing kernel. The approach is similar to that in Subsection 3.1.

▶ **Theorem 15.** RAD-CLUSTER-COMPLETION *parameterized by $k + r + \lambda(M)$ has a Turing-kernel containing at most $n = k3^{\lambda(M)+2r} + \lambda(M) + 2$ vectors, each having at most* $\max\{2r(n-1) + \lambda(M), \binom{\lambda(M)}{2}(2r + 1)\}$ *coordinates.*

4.2 RAD-CLUSTER-COMPLETION Parameterized by $r + \lambda(M)$

While, it is relatively easy to see that RAD-CLUSTER-COMPLETION parameterized by $r+\lambda(M)$ can be solved in time $f(\lambda(M), r)n^{\mathcal{O}(r)}$, here we provide a more efficient algorithm by reducing the degree of the polynomial in the run-time from $\mathcal{O}(r)$ to $\log r$. Moreover, our algorithm can be applied to the CLOSEST STRING WITH OUTLIERS problem [3].

▶ **Theorem 16.** RAD-CLUSTER-COMPLETION *can be solved in time $\mathcal{O}(|M|2^{\lambda(M)}(|M|(2^{2r} + d))^{\log r+1})$ and is therefore in* XP *parameterized $r + \lambda(M)$.*

Proof Sketch. The main ideas behind the algorithm are captured by the following definition and discussions. Let $F \subseteq [d]$ and let t be an integer, where $0 \leq t \leq r$. We say that a vector $\vec{v} \in \{0, 1\}^d$ is an (F, t)-*seed* for a center $\vec{c} \in \{0, 1\}^d$ of a solution for (M, k, r) if it satisfies:

(C1) $\vec{c}$ agrees with $\vec{v}$ on all coordinates in F; and
(C2) $\vec{c}$ differs from $\vec{v}$ on at most t coordinates outside of F.
We can show the following statement. If $\vec{v}$ is an (F,t)-seed for $\vec{c}$, then either $\vec{v}$ is the center of a solution for (M,k,r), or there is a vector $\vec{m} \in M$ with $r < \delta(\vec{v},\vec{m}) \leq 2r$ and a subset $C \subset D \setminus F$, where $D = \Delta(\vec{v},\vec{m})$, such that the vector $\vec{v}'$ obtained from $\vec{v}$ by complementing all coordinates in C is an $(F \cup D, t/2)$-seed for $\vec{c}$. Note that testing the former possibility, that is, whether a vector $\vec{v} \in \{0,1\}^d$ is a center of a solution for (M,k,r), can be done in time $\mathcal{O}(|M|d)$ by Observation 14.

Since there at most $M2^{2r}$ possibilities for $\vec{m}$ and C, we can use the above statement to obtain a $(F \cup D, t/2)$-seed from a given (F,t)-seed. This can be employed within a recursive procedure that, given a $(\emptyset, r)$-seed for some center $\vec{c}$ of a solution either obtains a center of a solution or obtains a $(F', 0)$-seed (which itself is the center of a solution), in at most $\log r$ recursive steps. It only remains to find a $(\emptyset, r)$-seed for some center $\vec{c}$ of a solution, which can be achieved by guessing the completion $\vec{v}$ of any vector in $M \setminus D_M$ that will be in a solution. Note that if $M \setminus D_M$ does not contain a vector in the solution, then $k \leq \lambda(M)$ and the result follows from Theorem 15. ◀

We note that the algorithm provided by the above theorem generalizes a previous algorithm of Marx [26, Lemma 3.2] for Closest Substring to strings that may contain unknown characters. In particular, it lifts the concept of "generators" to strings with unknown characters by showing that there are $\log r$ vectors that can be computed efficiently and that define at most $r \log r$ "important" coordinates for the center of some solution.

4.3 FPT Approximation Scheme Parameterized by $r + \lambda(M)$

In this subsection we give an algorithm that, for a given instance (M,k,r) of Rad-Cluster-Completion and $\varepsilon \in \mathbb{R}$, where $0 < \varepsilon < 1$, computes in FPT-time parameterized by $r + \lambda(M) + \frac{1}{\varepsilon}$ a center of a Rad-cluster of size at least $(1-\varepsilon)k$, or it correctly concludes that no RAD-Cluster of size k exists.

▶ **Theorem 17.** *Given an instance (M,k,r) of* Rad-Cluster-Completion *and $\varepsilon \in \mathbb{R}$, where $0 < \varepsilon < 1$, there exists an* FPT *algorithm $\mathcal{A}$, parameterized by $r + \lambda(M) + \frac{1}{\varepsilon}$, such that $\mathcal{A}$ either computes a RAD-Cluster of size at least $(1-\varepsilon)k$, or correctly concludes that M does not contain a RAD-Cluster of size k.*

Proof Sketch. The algorithm starts by performing a similar branching and pre-processing to the FPT algorithm for Diam-Cluster-Completion parameterized by $r + \lambda(M)$. Fix M^* to be a completion of M that contains a maximum size RAD-Cluster, let P^* be such a maximum size RAD-Cluster in M^*, and let $\vec{s}^*$ be a center of P^*. The goal of the algorithm is to find $\vec{s}^*$, as given $\vec{s}^*$, by Observation 14, we can decide the instance in time $\mathcal{O}(|M|d)$.

If $k < \frac{2\lambda(M)}{\varepsilon} + 2$, then we use the algorithm in Theorem 15 to obtain a Turing kernel. Otherwise, we can guess two vectors $\vec{u}$ and $\vec{v}$ in $M \setminus D_M$, together with their respective completions $\vec{u}^*$ and $\vec{v}^*$, such that $\vec{u}^*$ and $\vec{v}^*$ are the farthest vectors apart in $P^* \setminus D_M$; fix $r_{\max} = \delta(\vec{v}^*, \vec{u}^*)$.We normalize all the vectors in M so that $\vec{v}^*$ becomes the all-zero vector. Finally, for each vector $\vec{w} \in M$, we can in time $r_{\max} \cdot d$, where d is the dimension of the vectors in M, check if there is a completion $\vec{w}^*$ of $\vec{w}$ such that the distance from $\vec{w}^*$ to both $\vec{v}^*$ and $\vec{u}^*$ is at most $r_{\max}$; we remove all vectors $\vec{w}$ that do not have such a completion. Note here that some vectors in $P^* \cap D_M$ could have been removed from M at this step. However, we did not remove any vector from $P^* \setminus D_M$. Hence, after the preprocessing, M contains a RAD-Cluster with center $\vec{s}^*$ and at least $k' = (1 - \frac{\varepsilon}{2})k$ vectors. Our goal is to find a center for a RAD-Cluster with at least $(1-\frac{\varepsilon}{2})k' = (1-\frac{\varepsilon}{2})^2 k \geq (1-\varepsilon)k$ vectors. For ease of exposition, we let $\varepsilon' = \frac{\varepsilon}{2}$ and we let $k' \geq (1-\varepsilon')k$ be the number of vectors of P^* still in M. Now we can show the following statement: After the above pre-processing, in time $\mathcal{O}(2^{r_{\max}} \cdot |M|)$, we can find a center for a RAD-Cluster of size $\frac{|M|}{2^{r_{\max}}}$.

Therefore, we can assume henceforth that $k' \geq \frac{|M|}{2^{r_{\max}}}$. Now the algorithm sets $\vec{s}_0^* = \vec{v}^* = \vec{0}$ and the goal is to iteratively compute $\vec{s}_1^*, \vec{s}_2^*, \vec{s}_3^*, \ldots, \vec{s}_{r'}^*$, $r' \leq r$, such that:

1. for all $i \in [r']$, we have $\Delta(\vec{s}_i^*) = \Delta(\vec{s}_{i-1}^*) \cup \{c_i\}$ for some coordinate $c_i \in \Delta(\vec{s}^*) \setminus \Delta(\vec{s}_{i-1}^*)$;
2. for all $j \in [r'-1]$, the number of vectors $\vec{w}$ with $\delta(\vec{w}, \vec{s}_j^*) \leq r$ is less than $(1-\varepsilon')k'$; and
3. the number of vectors $\vec{w}$ with $\delta(\vec{w}, \vec{s}_{r'}^*) \leq r$ is at least $(1-\varepsilon')k'$.

Let $\vec{s}_i^*$ be such that $\Delta(\vec{s}_i^*) \subseteq \Delta(\vec{s}^*)$ for some $i \in [r'-1]$. The number of vectors at distance at most r from $\vec{s}_i^*$, $i < r'$, is less than $(1-\varepsilon')k'$. This means that at least $\varepsilon' k' \geq \frac{\varepsilon'|M|}{2^{r_{\max}}}$ vectors whose completions are in P^* are at distance at least $r+1$ from $\vec{s}_i^*$. For every such vector $\vec{w}$, it is easy to see that, since $\Delta(\vec{s}_i^*) \subseteq \Delta(\vec{s}^*)$, it must be the case that $(\Delta(\vec{s}^*) \cap \Delta(\vec{w})) \setminus \Delta(\vec{s}_i^*)$ is nonempty. Note that $|\Delta(\vec{s}^*)| \leq r$, and hence there exists $c_{i+1} \in \Delta(\vec{s}^*)$ such that, for at least $\frac{\varepsilon'|M|}{2^{r_{\max}} \cdot r}$ vectors $\vec{w}$ in M at distance at least $r+1$ from $\vec{s}_i^*$, it holds that $c_{i+1} \in \Delta(\vec{w})$. Moreover, for every vector $\vec{w} \in M$, we have $|\Delta(\vec{w})| \leq r_{\max}$. It follows–by a straightforward counting argument–that there are at most $\frac{2^{r_{\max}} \cdot r}{\varepsilon} \cdot r_{\max}$ coordinates $c \in [d]$ such that, for at least $\frac{\varepsilon'|M|}{2^{r_{\max}} \cdot r}$ vectors $\vec{w}$, it holds that $c \in \Delta(\vec{w})$. Therefore, to obtain $\vec{s}_{i+1}^*$ such that $\Delta(\vec{s}_{i+1}^*) \subseteq \Delta(\vec{s}^*)$, we only need to branch on one of at most $\frac{2^{r_{\max}} \cdot r}{\varepsilon} \cdot r_{\max}$ coordinates. The statement of the theorem follows since we can exhaustively branch on the coordinates that are set to 1 in at least $\frac{\varepsilon'|M|}{2^{r_{\max}} \cdot r}$ many vectors in M, until either the number of vectors at distance at most r from $\vec{s}_i^*$ is at least $(1-\varepsilon')k'$ or $i \geq r$. ◀

5 Concluding Remarks

We studied the parameterized complexity of two fundamental problems pertaining to incomplete data that have applications in data analytics. In most cases, we were able to provide a complete landscape of the parameterized complexity of the problems w.r.t. the parameters under consideration. It is worth noting that all algorithmic upper bounds obtained in this paper can also be directly generalized to vectors (i.e., matrices) over a domain whose size is bounded by the parameter value by using the encoding described by Eiben et al. [9].

Two important open questions ensue from our work, namely determining the parameterized complexity of RAD-CLUSTER-COMPLETION w.r.t. each of the two parameterizations $k+\lambda$ and $r+\lambda$. In particular, the restrictions of these two problems to complete data (i.e., $\lambda = 0$) remain open, resulting in two important questions about the parameterized complexity of RAD-CLUSTER parameterized by the cluster size k or the cluster radius r.

References

1 Laura Balzano, Arthur Szlam, Benjamin Recht, and Robert D. Nowak. k-subspaces with missing data. *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pages 612–615, 2012.

2 Christina Boucher and Bin Ma. Closest string with outliers. *BMC Bioinformatics*, 12(S-1):S55, 2011.

3 Laurent Bulteau and Markus L. Schmid. Consensus strings with small maximum distance and small distance sum. *Algorithmica*, 82(5):1378–1409, 2020.

4 Sergio Cabello, Panos Giannopoulos, Christian Knauer, Dániel Marx, and Günter Rote. Geometric clustering: Fixed-parameter tractability and lower bounds with respect to the dimension. *ACM Trans. Algorithms*, 7(4):43:1–43:27, 2011.

5 Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *Journal of Computer and System Sciences*, 68(2):417–441, 2004.

6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

7 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`.

8 M.E Dyer and A.M Frieze. A simple heuristic for the p-centre problem. *Oper. Res. Lett.*, 3(6):285–288, 1985.

9 Eduard Eiben, Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. The parameterized complexity of clustering incomplete data. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 7296–7304. AAAI Press, 2021.

10 Ehsan Elhamifar. High-rank matrix completion and clustering under self-expressive models. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 73–81. Curran Associates, Inc., 2016.

11 Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2765–2781, 2013.

12 Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 434–444. ACM, 1988.

13 M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.

14 Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. Parameterized algorithms for the matrix completion problem. In *ICML*, volume 80 of *JMLR Workshop and Conference Proceedings*, pages 1642–1651, 2018.

15 Leszek Gąsieniec, 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*, pages 905–906, 1999.

16 Leszek Gąsieniec, Jesper Jansson, and Andrzej Lingas. Approximation algorithms for Hamming clustering problems. *Journal of Discrete Algorithms*, 2(2):289–301, 2004.

17 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

18 Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for CLOSEST STRING and related problems. *Algorithmica*, 37(1):25–42, 2003.

19 Danny Hermelin and Liat Rozenberg. Parameterized complexity analysis for the closest string with wildcards problem. *Theoretical Computer Science*, 600:11–18, 2015.

20 Tomohiro Koana, Vincent Froese, and Rolf Niedermeier. Parameterized algorithms for matrix completion with radius constraints. In Inge Li Gørtz and Oren Weimann, editors, *31st Annual Symposium on Combinatorial Pattern Matching, CPM 2020, June 17-19, 2020, Copenhagen, Denmark*, volume 161 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

21 Tomohiro Koana, Vincent Froese, and Rolf Niedermeier. Binary matrix completion under diameter constraints. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 47:1–47:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

22 Stefan Kratsch, Dániel Marx, and Magnus Wahlström. Parameterized complexity and kernelizability of max ones and exact ones problems. *TOCT*, 8(1):1:1–1:28, 2016.

23 H. W. Lenstra and Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

24 Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002.

25 Dániel Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2):153–183, 2005.

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

27 Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008.

28 J. Yi, T. Yang, R. Jin, A. K. Jain, and M. Mahdavi. Robust ensemble clustering by matrix completion. In *2012 IEEE 12th International Conference on Data Mining*, pages 1176–1181, 2012.