


Max-Distance Sparsification for Diversification and Clustering

Soh Kumabe 

CyberAgent, Tokyo, Japan

Abstract

Let \mathcal{D} be a set family that is the solution domain of some combinatorial problem. The *max-min diversification problem* on \mathcal{D} is the problem to select k sets from \mathcal{D} such that the Hamming distance between any two selected sets is at least d . FPT algorithms parameterized by $k + \ell$, where $\ell = \max_{D \in \mathcal{D}} |D|$, and $k + d$ have been actively studied recently for several specific domains.

This paper provides unified algorithmic frameworks to solve this problem. Specifically, for each parameterization $k + \ell$ and $k + d$, we provide an FPT oracle algorithm for the max-min diversification problem using oracles related to \mathcal{D} . We then demonstrate that our frameworks provide the first FPT algorithms on several new domains \mathcal{D} , including the domain of t -linear matroid intersection, almost 2-SAT, minimum edge s, t -flows, vertex sets of s, t -mincut, vertex sets of edge bipartization, and Steiner trees. We also demonstrate that our frameworks generalize most of the existing domain-specific tractability results.

Our main technical breakthrough is introducing the notion of *max-distance sparsifier* of \mathcal{D} , a domain on which the max-min diversification problem is equivalent to the same problem on the original domain \mathcal{D} . The core of our framework is to design FPT oracle algorithms that construct a constant-size max-distance sparsifier of \mathcal{D} . Using max-distance sparsifiers, we provide FPT algorithms for the max-min and max-sum diversification problems on \mathcal{D} , as well as k -center and k -sum-of-radii clustering problems on \mathcal{D} , which are also natural problems in the context of diversification and have their own interests.

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

Keywords and phrases Fixed-Parameter Tractability, Diversification, Clustering

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.46

Related Version *Full Version*: <https://arxiv.org/abs/2411.02845>

1 Introduction

1.1 Background and Motivation

The procedure for approaching real-world problems with optimization algorithms involves formulating the real-world motivations as mathematical problems and then solving them. However, real-world problems are complex, and the idea of a “good” solution cannot always be correctly formulated. The paradigm of *diversification*, introduced by Baste et al. [8] and Baste et al. [7], is a “formulation of the unformulatable problems”, which formulates *diversity measures* for a set of multiple solutions, rather than attempting to formulate the “goodness” of a single solution. By computing a set of solutions that maximize this measure, the algorithm provides effective options to evaluators who have the correct criteria for judging the “goodness” of a solution.

Let U be a finite set, $k \in \mathbb{Z}_{\geq 1}$ and $d \in \mathbb{Z}_{\geq 0}$. Let $\mathcal{D} \subseteq 2^U$ be the feasible domain of some combinatorial problem. The following problem frameworks, defined by two types of diversity measures, have been studied extensively.



© Soh Kumabe;
licensed under Creative Commons License CC-BY 4.0

33rd Annual European Symposium on Algorithms (ESA 2025).

Editors: Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman; Article No. 46; pp. 46:1–46:14

Leibniz International Proceedings in Informatics



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

Max-Min Diversification Problem on \mathcal{D} : Does there exist a k -tuple $(D_1, \dots, D_k) \in \mathcal{D}^k$ of sets in \mathcal{D} such that $\min_{1 \leq i < j \leq k} |D_i \Delta D_j| \geq d$?¹

Max-Sum Diversification Problem on \mathcal{D} : Does there exist a k -tuple $(D_1, \dots, D_k) \in \mathcal{D}^k$ of sets in \mathcal{D} such that $\sum_{1 \leq i < j \leq k} |D_i \Delta D_j| \geq d$?

These problems ensure diversity by aiming to output solutions that are as dissimilar as possible in terms of Hamming distance.

Parameterized algorithms for diversification problems have been actively studied. Particularly, FPT algorithms for the max-min diversification problems parameterized by $k + \ell$, where $\ell = \max_{D \in \mathcal{D}} |D|$ [7, 8, 22, 26, 31], as well as by $k + d$ [17, 21, 22, 23, 24, 29], have been the focus of research. Since assuming $d \leq 2\ell$ does not lose generality in the max-min diversification problem, the latter addresses a more general situation than the former. Since the max-sum diversification problem is empirically more tractable than the max-min diversification problem, for some time hereafter, we will restrict our discussion to the max-min diversification problem.

This research provides general algorithmic frameworks for FPT algorithms solving max-min (and max-sum) diversification problems for both parameterizations $k + \ell$ and $k + d$. Our frameworks are very general and can be applied to all domains [5, 6, 7, 8, 17, 21, 22, 23, 25, 26, 31] for which FPT algorithms parameterized by $k + \ell$ and $k + d$ are currently known for the case that diversity measure is defined using an unweighted Hamming distance. Moreover, our frameworks further provide the first algorithms for several domains where such algorithms were previously unknown.

Our main technical breakthrough is introducing a notion of *max-distance sparsifier* as an intermediate step, which, for the max-min diversification problem, essentially works as a *core-set* [2]. The formal definition is given in Section 1.3. The critical fact is that, when \mathcal{K} is a max-distance sparsifier of \mathcal{D} , the max-min diversification problem on \mathcal{D} is equivalent to the same problem on \mathcal{K} . Our framework constructs a constant-size max-distance sparsifier \mathcal{K} of \mathcal{D} using the oracles on \mathcal{D} , enabling us to solve the max-min diversification problems on \mathcal{D} by brute-force search on \mathcal{K} .

The power of max-distance sparsification is not limited to solving diversification problems. Specifically, the following *k-center* [27] and *k-sum-of-radii clustering problems on \mathcal{D}* [11] can also be solved via max-distance sparsification.

k-Center Clustering Problem on \mathcal{D} : Does there exist a k -tuple of subsets $(D_1, \dots, D_k) \in \mathcal{D}^k$ such that for all $D \in \mathcal{D}$, there exists an $i \in \{1, \dots, k\}$ satisfying $|D_i \Delta D| \leq d$?

k-Sum-of-Radii Clustering Problem on \mathcal{D} : Does there exist a k -tuple of subsets $(D_1, \dots, D_k) \in \mathcal{D}^k$ and a k -tuple of non-negative integers $(d_1, \dots, d_k) \in \mathbb{Z}_{\geq 0}^k$ with $\sum_{i \in \{1, \dots, k\}} d_i \leq d$ such that for all $D \in \mathcal{D}$, there exists an $i \in \{1, \dots, k\}$ satisfying $|D_i \Delta D| \leq d_i$?

When \mathcal{D} is an explicitly given set of points, parameterized algorithms for these problems have been extensively studied in the area of clustering [3, 4, 12, 15, 16, 20, 28]. Furthermore, approximation algorithms for the *relational k-means* [13, 19, 30] and *relational k-center* [1] problems are investigated, which are the *k-means* and *k-center* clustering problem defined on

¹ We define $Z_1 \Delta Z_2 := (Z_1 \setminus Z_2) \cup (Z_2 \setminus Z_1)$.

a point set represented as a *join* of given relational databases. Their setting is similar to ours as the point set are implicitly given and its size can be exponential. This research adds several new combinatorial domains to the literature on clustering problems on implicitly given domains, and also introduces a parameterized view. These problems are also natural in the context of diversification, since in real situations, the concept of diversity often means that the extracted elements cover the entire space comprehensively rather than being mutually dissimilar. This motivation is formulated by clustering problems, which extract a list of sets in \mathcal{D} such that for each set in \mathcal{D} , there is an extracted set near to it.

1.2 Our Results

This paper consists of two parts. In the first part, we design general frameworks for solving diversification and clustering problems. In the second part, we apply our frameworks to several specific domains \mathcal{D} . Table 1 shows a list of domains on which our frameworks provide the first or an improved algorithm for the max-min diversification problem.

1.2.1 The Frameworks

We define the following $(-1, 1)$ -*optimization oracle* on \mathcal{D} and the *exact extension oracle* on \mathcal{D} .

$(-1, 1)$ -Optimization Oracle on \mathcal{D} : Let U be a finite set, $\mathcal{D} \subseteq 2^U$, and $w \in \{-1, 1\}^U$ be a weight vector. Return a set $D \in \mathcal{D}$ that maximizes $\sum_{e \in D} w_e$.

Exact Extension Oracle on \mathcal{D} : Let U be a finite set, $r \in \mathbb{Z}_{\geq 0}$, $\mathcal{D} \subseteq 2^U$, and $C \in \mathcal{D}$. Let $X, Y \subseteq U$ be two disjoint subsets of U . If there exists a set $D \in \mathcal{D}$ such that $|D \triangle C| = r$, $X \subseteq D$, and $Y \cap D = \emptyset$, return one such set. If no such set exists, return \perp .

When $C = \emptyset$ and $X = \emptyset$, we specifically call the exact extension oracle on \mathcal{D} the *exact empty extension oracle* on \mathcal{D} .

Exact Empty Extension Oracle on \mathcal{D} : Let U be a finite set, $r \in \mathbb{Z}_{\geq 0}$, $\mathcal{D} \subseteq 2^U$, and $Y \subseteq U$. If there exists a set $D \in \mathcal{D}$ such that $|D| = r$ and $Y \cap D = \emptyset$, return one such set. If no such set exists, return \perp .

Let $\mathcal{P}_{\mathcal{D}}$ be the max-min/max-sum diversification problem or the k -center/ k -sum-of-radii clustering problem on \mathcal{D} . Our main result is FPT algorithms for solving $\mathcal{P}_{\mathcal{D}}$ using these oracles. We construct frameworks for both types of parameterizations, $k + \ell$ and $k + d$. The result for the parameterization by $k + \ell$ is as follows.

► **Theorem 1.** *There exists an oracle algorithm solving $\mathcal{P}_{\mathcal{D}}$ using the exact empty extension oracle on \mathcal{D} , where the number of oracle calls and time complexity are both FPT parameterized by $k + \ell$ and for each call of an oracle, r and $|Y|$ are bounded by constants that depend only on $k + \ell$.*

The result for the parameterization by $k + d$ is as follows.

► **Theorem 2.** *There exists a randomized oracle algorithm solving $\mathcal{P}_{\mathcal{D}}$ using the $(-1, 1)$ -optimization oracle on \mathcal{D} and the exact extension oracle on \mathcal{D} , where the number of oracle calls and time complexity are both FPT parameterized by $k + d$ and for each call of the exact extension oracle, $r + |X| + |Y|$ are bounded by constants that depend only on $k + d$.*

■ **Table 1** List of new results for the max-min diversification problem obtained by our frameworks. The first column represents the domain \mathcal{D} . The second column represents parameterization. For the formal definition of each domain, see the full version.

Domain	Parameter
t -Linear Matroid Intersection	$k + \ell + t$
Almost 2-SAT	$k + \ell$
Independent Set on Certain Graphs	$k + \ell$
Min Edge s, t -Flow	$k + d$
Steiner Tree	$k + d + T $
Vertex Set of Min s, t -Cut	$k + d$
Vertex Set of Edge Bipartization	$k + d + s$

1.2.2 Applications of Theorem 1

On most domains \mathcal{D} , the exact empty extension oracle can be designed by almost the same way as an algorithm to extract a single solution from \mathcal{D} . For example, consider the case where \mathcal{D} is the ℓ -path domain, i.e., a domain of sets of edges on paths of length ℓ . In this case, the exact empty extension oracle on \mathcal{D} is equivalent to the problem of finding an ℓ -path in the graph obtained by removing all edges in Y from the input. Combining Theorem 1 with this empirical fact, we can claim that, for most domains \mathcal{D} , the diversification and clustering problems parameterized by $k + \ell$ on \mathcal{D} are as easy as determining the non-emptiness of \mathcal{D} .

To demonstrate that Theorem 1 yields existing tractability results, we design the oracles for the domains of the vertex cover [7, 8], t -hitting set [7], feedback vertex set [7], and common independent set of two matroids [17, 22]. We also apply our framework on new domains, t -represented linear matroid intersection, almost 2-SAT, and independent set on *subgraph-closed IS-FPT* graph classes. Here, a graph class is *subgraph-closed IS-FPT* if it is closed under taking subgraphs and the problem of finding independent set of size ℓ is FPT parameterized by ℓ . The following theorem summarizes our results, where the precise definitions of each domain are given in the full version.

► **Theorem 3.** *Let \mathcal{D} be the domain of vertex covers, t -hitting sets, feedback vertex sets, t -represented linear matroid intersections, almost 2-SATs, or independent sets on subgraph-closed IS-FPT graph classes. Then, max-min and max-sum diversification problems and k -center and k -sum-of-radii clustering problems admit an FPT algorithm, where the parameterization is $k + \ell$ except for the t -hitting set and t -represented linear matroid intersection, which are parameterized by $k + \ell + t$.*

Theorem 1 also generalizes existing frameworks for diversification. Baste et al. [8] provided an algorithmic framework for diversification using a *loss-less kernel* [9, 10], which, roughly speaking, is a kernel that completely preserves the information of the solution space. Since loss-less kernels are known for very limited domains, their framework requires very strong assumptions. Our framework has broader applicability than theirs because it relies on a weaker oracle, as the exact empty extension oracle can be constructed using a loss-less kernel. Hanaka et al. [26] developed a color-coding-based framework for diversification. The oracle they use can be regarded as the exact empty extension oracle with an additional colorfulness constraint. Our framework again has broader applicability than theirs because our oracle can be constructed using theirs. Moreover, our framework also treats clustering problems, which these two do not.

1.2.3 Applications of Theorem 2

FPT algorithms for the max-min diversification problems on \mathcal{D} parameterized by $k + d$ are known for the cases where \mathcal{D} is the family of matroid bases [17, 22], perfect matchings [22], and shortest paths [23]. The result for the perfect matchings is later extended to the matchings of specified size, not necessarily perfect [17]. Additionally, for the cases where \mathcal{D} is the family of interval schedulings [26] and the longest common subsequences of an absolute constant number of strings [31], FPT algorithms parameterized by $k + \ell$ are known. Both of these two can be generalized to the domain of dynamic programming problems, which we define in the full version. Furthermore, the problem of finding a pair of a branching and an in-branching such that the Hamming distance between them is at least d is investigated as the name of *d-distinct branchings problem* [5, 6, 25], which admits FPT algorithm parameterized by d . This problem can naturally be extended to the case that selects k_1 branchings and k_2 in-branching, rather than one each. We give FPT algorithms parameterized by $k + d$ for all those problems, where $k = k_1 + k_2$ for the extended version of *d-distinct branchings problem*. We also give FPT algorithms on domains of minimum edge s, t -flows, Steiner trees, vertex sets of s, t -mincut, and vertex sets of edge bipartization, which are domains where no FPT algorithm for the max-min diversification problem is previously known. Remark that the domain of shortest paths [23] is the special case of the minimum edge s, t -flow domain and the minimum Steiner tree domain. The following theorem summarizes our results, where the precise definitions of each domain are given in the full version.

► **Theorem 4.** *Let \mathcal{D} be the domain of matroid bases, branchings, matchings of specified size, minimum edge s, t -flows, minimum Steiner trees, vertex sets of s, t -mincut, vertex sets of edge bipartization, and dynamic programming problems. Then, max-min and max-sum diversification problems and k -center and k -sum-of-radii clustering problems admit an FPT algorithm, where the parameterization is $k + d$ except for the Steiner tree, which is parameterized by $k + d + |T|$ for the terminal set T , and edge bipartization, which is parameterized by $k + \ell + s$, where s is the minimum number of edges to be removed to make the given graph bipartite. Furthermore, the extended version of *d-distinct branching problem* also admits FPT algorithm parameterized by $k + d$.*

Eiben et al. [17] provided a technique called *determinantal sieving*, which is a general tool to give and speed up parameterized algorithms, including that for diversification problems. Particularly, they provided a framework to solve the diversification problem by using an oracle that, roughly speaking, counts the number of solutions modulo 2. Using their framework, they improved the running times of FPT algorithms for max-min diversification problems on matchings and matroid bases, as well as the *d-distinct branchings problem*. Although not stated explicitly, their framework seems to yield FPT algorithms parameterized by $k + d$ when \mathcal{D} is the dynamic programming domain, thereby improving the parameterizations in the results of [26] and [31], respectively, as well as the extended version of *d-distinct branchings problems*. We are not sure whether our framework generalizes theirs, that is, whether our oracle can be constructed using their oracle. However, we strongly believe that our framework has broader applicability because their framework assumes counting oracles, which is often hard even modulo 2. In contrast, our framework uses optimization-type oracles, which are generally more tractable than counting. Indeed, our framework provides an FPT algorithm with the same parameterization for every domain which they explicitly considered. Moreover, we do not think their framework can give an FPT algorithm for the max-min diversification problem on the domains of minimum edge s, t -flows, vertex sets of minimum s, t -cut, and vertex sets of edge bipartization. Furthermore, our framework can be applied not only to diversification problems but also to clustering problems.

1.3 Framework Overview

In this section, we provide an overview of the entire flow of our frameworks. Our frameworks first construct max-distance sparsifiers using the corresponding oracles and then solve diversification and clustering problems using them.

1.3.1 From d -Limited k -Max-Distance Sparsifier to Diversification and Clustering

We begin by defining the max-distance sparsifiers. The key for Theorem 1 is designing the following k -max-distance sparsifier of \mathcal{D} .

► **Definition 5** (k -max-distance sparsifier). Let $k \in \mathbb{Z}_{\geq 1}$. Let U be a finite set and $\mathcal{D}, \mathcal{F} \subseteq 2^U$. We say that $\mathcal{K} \subseteq \mathcal{D}$ is a k -max-distance sparsifier of \mathcal{D} with respect to \mathcal{F} if for any $(F_1, \dots, F_k) \in \mathcal{F}^k$ and $(z_1, \dots, z_k) \in \mathbb{Z}_{\geq 0}^k$, the two conditions

- There exists $D \in \mathcal{D}$ such that for each $i \in \{1, \dots, k\}$, $|F_i \Delta D| \geq z_i$.
- There exists $K \in \mathcal{K}$ such that for each $i \in \{1, \dots, k\}$, $|F_i \Delta K| \geq z_i$.

are equivalent. Unless specifically noted, when we write k -max-distance sparsifier of \mathcal{D} , we mean the case where $\mathcal{D} = \mathcal{F}$.

Similarly, the key for Theorem 2 is designing the following d -limited k -max-distance sparsifier of \mathcal{D} .

► **Definition 6** (d -limited k -max-distance sparsifier). Let $k \in \mathbb{Z}_{\geq 1}$ and $d \in \mathbb{Z}_{\geq 0}$. Let U be a finite set and $\mathcal{D}, \mathcal{F} \subseteq 2^U$. We say that $\mathcal{K} \subseteq \mathcal{D}$ is a d -limited k -max-distance sparsifier of \mathcal{D} with respect to \mathcal{F} if for any $(F_1, \dots, F_k) \in \mathcal{F}^k$ and $(z_1, \dots, z_k) \in \{0, \dots, d\}^k$, the two conditions

- There exists $D \in \mathcal{D}$ such that for each $i \in \{1, \dots, k\}$, $|F_i \Delta D| \geq z_i$.
- There exists $K \in \mathcal{K}$ such that for each $i \in \{1, \dots, k\}$, $|F_i \Delta K| \geq z_i$.

are equivalent. Unless specifically noted, when we write d -limited k -max-distance sparsifier of \mathcal{D} , we mean the case where $\mathcal{D} = \mathcal{F}$.

The difference between the two sparsifiers is that the domain of (z_1, \dots, z_k) is $\mathbb{Z}_{\geq 0}^k$ in the former case, while it is $\{0, \dots, d\}^k$ in the latter. By definition, any k -max-distance sparsifier is also a d -limited k -max-distance sparsifier for any $d \in \mathbb{Z}_{\geq 0}$. We can prove that given a d -limited $(k-1)$ -max-distance sparsifier of \mathcal{D} with size bounded by a constant that depends only on $k+d$, we can construct FPT algorithms parameterized by $k+d$ for the max-min/max-sum diversification problems on \mathcal{D} . Similarly, we can prove that given a $(d+1)$ -limited k -max-distance sparsifier of \mathcal{D} with size bounded by a constant that depends only on $k+d$, we can construct FPT algorithms parameterized by $k+d$ for the k -center/ k -sum-of-radii clustering problems on \mathcal{D} . Therefore, to prove Theorems 1 and 2, it suffices to construct FPT oracle algorithms for designing k -max-distance sparsifiers and d -limited k -max-distance sparsifiers, respectively.

1.3.2 Computing k -Max-Distance Sparsifier

The remaining task towards Theorem 1 is to provide an FPT algorithm parameterized by $k+\ell$ that constructs a k -max-distance sparsifier of \mathcal{D} with size bounded by a constant that depends only on $k+\ell$. The key lemma toward this is that, if \mathcal{K} contains a sufficiently large *sunflower* (see Section 3 for the definition) consisting of sets of the same size, then we can safely remove one of them from \mathcal{K} while preserving the property that \mathcal{K} is a k -max-distance

sparsifier (actually, for the sake of simplifying the framework, we prove a slightly stronger statement). Starting with $\mathcal{K} = \mathcal{D}$ and exhaustively removing such sets leads to a \mathcal{K} that is still a k -max-distance sparsifier of \mathcal{D} , and by using the well-known sunflower lemma (Lemma 13), its size is bounded by a constant.

However, this observation is still not sufficient to obtain an FPT algorithm. The reason is that \mathcal{D} generally has exponential size, and removing sets one by one would require an exponential number of steps. Instead, our algorithm starts with $\mathcal{K} = \emptyset$ and exhaustively adds sets of \mathcal{D} to \mathcal{K} until \mathcal{K} becomes a k -max-distance sparsifier. In this way, the number of steps is bounded by a constant. The remaining task is to choose a set to be added at each step. For this task, we design an FPT algorithm using constant number of calls of the exact empty extension oracle.

1.3.3 Computing d -Limited k -Max-Distance Sparsifier

The algorithm in the previous section alone is insufficient to prove Theorem 2 since ℓ is unbounded and the sunflower-lemma-based bound for the number of steps cannot be used. Our algorithm divides \mathcal{D} into at most k clusters, computes a d -limited k -max-distance sparsifier for each cluster, and outputs their union. Let $p > 2d$ be a constant that depends only on $k + d$. We first find $\mathcal{C} \subseteq \mathcal{D}$ satisfying the following properties: (i) $|\mathcal{C}| \leq k$, (ii) for all distinct $C, C' \in \mathcal{C}$, $|C \Delta C'| > 2d$, and (iii) for all $D \in \mathcal{D}$, there exists $C \in \mathcal{C}$ such that $|D \Delta C| \leq p$. If such a family does not exist, a trivial d -limited k -max-distance sparsifier of \mathcal{D} will be found, and we output it and terminate.

We provide an algorithm for computing \mathcal{C} . Our algorithm starts with $\mathcal{C} = \emptyset$ and exhaustively adds sets in \mathcal{D} to \mathcal{C} until \mathcal{C} satisfies the above conditions or its size exceeds k . To choose the elements to be added, we randomly sample $w \in \{-1, 1\}^U$ and call a $(-1, 1)$ -optimization oracle. We can prove for sufficiently large constant p that if there exists $D \in \mathcal{D}$ such that $|D \Delta C| > p$ for any $C \in \mathcal{C}$, with a constant probability, the $(-1, 1)$ -optimization oracle will find a $D \in \mathcal{D}$ such that $|D \Delta C| > 2d$ for any $C \in \mathcal{C}$. Thus, if \mathcal{C} does not meet the conditions, by calling the $(-1, 1)$ -optimization oracle a sufficient number of times, we can find a set to add to \mathcal{C} with high probability.

Here, we provide an algorithm for computing a d -limited k -max-distance sparsifier of \mathcal{D} using \mathcal{C} . For each cluster $\mathcal{D}_C := \{D \in \mathcal{D} : |D \Delta C| \leq p\}$, let $\mathcal{D}_C^* := \{D \Delta C : D \in \mathcal{D}_C\}$. The algorithm computes a k -max-distance sparsifier of each \mathcal{D}_C^* and outputs their union. For technical reasons, we actually compute a slightly more general object, but we will not delve into the details here. Since each \mathcal{D}_C^* consists only of sets whose size is at most p , the k -max-distance sparsifier of \mathcal{D}_C^* can be constructed using the algorithm in Section 1.3.2. The exact empty extension oracle on \mathcal{D}_C^* corresponds to the exact extension oracle on \mathcal{D} .

Here, we note the difference between our framework and that used by Fomin et al. [22] and Funayama et al. [23] to provide FPT algorithms for the max-min diversification problem on \mathcal{D} when \mathcal{D} is the family of perfect matchings and shortest paths, respectively. Their algorithms also start by dividing \mathcal{D} into clusters. However, their algorithms perform stricter clustering than ours. Specifically, in their clustering, clusters \mathcal{D}_C corresponding to different $C \in \mathcal{C}$ must be well-separated. In contrast, we allow clusters to overlap. This simplifies the clustering step compared to their approach at the cost of a more challenging task afterward. We resolve this more challenging task by introducing and designing the d -limited k -max-distance sparsifier.

1.4 Organization

The rest of this paper is organized as follows. In Section 2, we provide FPT algorithms for solving diversification and clustering problems on \mathcal{D} using a constant-size d -limited k -max-distance sparsifier of \mathcal{D} . In Section 3, we prove Theorem 1 by providing an FPT oracle algorithm parameterized by $k + \ell$ that computes a constant-size k -max-distance sparsifier of \mathcal{D} . In Section 4, we prove Theorem 2 by providing an FPT oracle algorithm parameterized by $k + d$ that computes a constant-size d -limited k -max-distance sparsifier of \mathcal{D} . The discussion in Section 4 internally uses the results from Section 3. In the full version, we apply the results of Theorems 1 and 2 to several domains \mathcal{D} to obtain FPT algorithms for diversification and clustering problems. The full version also contains further related work and the proofs of the lemmas marked with an asterisk.

2 From Sparsifier to Diversification and Clustering

In this section, we provide FPT algorithms for diversification and clustering problems using a d -limited k -max-distance sparsifier of constant size.

2.1 Diversification

Let U be a finite set, $d \in \mathbb{Z}_{\geq 0}$, $k \in \mathbb{Z}_{\geq 1}$, and $\mathcal{D} \subseteq 2^U$. For diversification problems, we have the following.

► **Lemma 7.** *Let $\mathcal{K} \subseteq \mathcal{D}$ be a d -limited $(k - 1)$ -max-distance sparsifier of \mathcal{D} and $(D_1, \dots, D_k) \in \mathcal{D}^k$. Then, there is a k -tuple $(K_1, \dots, K_k) \in \mathcal{K}^k$ such that $\min(d, |D_i \Delta D_j|) \leq \min(d, |K_i \Delta K_j|)$ holds for all $1 \leq i < j \leq k$. Particularly, if there exists a solution to the max-min/max-sum diversification problem on \mathcal{D} , then there exists a solution consisting only of sets in \mathcal{K} .*

Proof. Assume $(D_1, \dots, D_k) \in \mathcal{D}^k \setminus \mathcal{K}^k$ and let $i \in \{1, \dots, k\}$ be an index such that $D_i \notin \mathcal{K}$. It is sufficient to prove that there is a set $K_i \in \mathcal{K}$ such that $\min(d, |D_i \Delta D_j|) \leq \min(d, |K_i \Delta D_j|)$ holds for all $j \in \{1, \dots, k\} \setminus \{i\}$. For $j \in \{1, \dots, k\} \setminus \{i\}$, let $z_j := \min(d, |D_i \Delta D_j|)$. Since \mathcal{K} is a d -limited $(k - 1)$ -max-distance sparsifier of \mathcal{D} , there exists $K_i \in \mathcal{K}$ such that $\min(d, |K_i \Delta D_j|) \geq \min(d, z_j) = \min(d, |D_i \Delta D_j|)$ holds for all $j \in \{1, \dots, k\} \setminus \{i\}$. ◀

Considering an algorithm that exhaustively searches for a subfamily of \mathcal{K} of size k , we can state the following.

► **Lemma 8.** *Assume there exists an FPT algorithm parameterized by $k + d$ to compute a d -limited $(k - 1)$ -max-distance sparsifier of \mathcal{D} with size bounded by a constant that depends only on $k + d$. Then, there exists an FPT algorithm parameterized by $k + d$ for the max-min/max-sum diversification problem on \mathcal{D} .*

We note that, under the slightly stronger assumption, the discussion in this section can directly be extended to the case where the sets D_1, \dots, D_k are taken from different domains. Specifically, let $\mathcal{D}_1, \dots, \mathcal{D}_k \subseteq 2^U$ and assume $(k - 1)$ -max-distance sparsifiers $\mathcal{K}_1, \dots, \mathcal{K}_k$ of these domains with respect to 2^U are computed. Then, we can determine whether there exists a k -tuple $(D_1, \dots, D_k) \in \mathcal{D}_1 \times \dots \times \mathcal{D}_k$ such that $\min_{1 \leq i < j \leq k} |D_i \Delta D_j| \geq d$ (or $\sum_{1 \leq i < j \leq k} |D_i \Delta D_j| \geq d$) by exhaustive search on $\mathcal{K}_1 \times \dots \times \mathcal{K}_k$.

2.2 Clustering

Let U be a finite set, $d \in \mathbb{Z}_{\geq 0}$, $k \in \mathbb{Z}_{\geq 1}$, and $\mathcal{D} \subseteq 2^U$. Here, we provide an FPT algorithm for the k -center and k -sum-of-radii clustering problems using a $(d+1)$ -limited k -max-distance sparsifier \mathcal{K} of \mathcal{D} . For $Z \subseteq U$ and $r \in \mathbb{Z}_{\geq 0}$, the *ball* of radius r centered at Z is defined as $\mathcal{B}(Z, r) := \{Z' \subseteq U : |Z \Delta Z'| \leq r\}$. The algorithm first guesses a partition of \mathcal{K} into k clusters $\mathcal{K}_1, \dots, \mathcal{K}_k$. Since $|\mathcal{K}|$ is constant, the cost of this guess is constant. Then, for each $i \in \{1, \dots, k\}$, the algorithm computes the minimum radius r_i such that there is a set $D_i \in \mathcal{D}$ satisfying $\mathcal{K}_i \subseteq \mathcal{B}(D_i, r_i)$. If $r_i > d$, the algorithm asserts it instead of computing the specific value of r_i . The k -center clustering problem and k -sum-of-radii clustering problem on \mathcal{D} are solved by checking whether the maximum and sum, respectively, of the r_i s is at most d . We show the correctness of this algorithm by proving the following.

► **Lemma 9.** *Let $(D_1, \dots, D_k) \in \mathcal{D}^k$ and $(r_1, \dots, r_k) \in \{0, \dots, d\}^k$. Assume $\mathcal{K}_i \subseteq \mathcal{B}(D_i, r_i)$ holds for all $i \in \{1, \dots, k\}$. Then, for all $D \in \mathcal{D}$, there is an index $i \in \{1, \dots, k\}$ such that $D \in \mathcal{B}(D_i, r_i)$.*

Proof. Assume the contrary. Then, there is a set $D \in \mathcal{D}$ such that for all $i \in \{1, \dots, k\}$, $|D_i \Delta D| \geq r_i + 1$. Since \mathcal{K} is a $(d+1)$ -limited k -max-distance sparsifier of \mathcal{D} , there is a set $K \in \mathcal{K}$ such that for all $i \in \{1, \dots, k\}$, $|D_i \Delta K| \geq r_i + 1$. Hence, $K \notin \mathcal{K}_i$ for all $i \in \{1, \dots, k\}$, contradicting the fact that $(\mathcal{K}_1, \dots, \mathcal{K}_k)$ is a partition of \mathcal{K} . ◀

We now provide an algorithm to decide whether there exists $D \in \mathcal{D}$ with $\mathcal{K}_i \subseteq \mathcal{B}(D, r_i)$ for each $i \in \{1, \dots, k\}$ and $r_i \in \{0, \dots, d\}$. If the domain \mathcal{D} is 2^U , this problem is equivalent to the *closest string problem* on binary strings, for which a textbook FPT algorithm parameterized by $d + |\mathcal{K}_i|$ is known [14]. Our algorithm is a modified version of this. An element $e \in U$ is *bad* if there exist both $K \in \mathcal{K}_i$ with $e \in K$ and $K \in \mathcal{K}_i$ with $e \notin K$. The following lemma is fundamental.

► **Lemma 10** ([14]). *If there are more than $d|\mathcal{K}_i|$ bad elements, no $D \in \mathcal{D}$ satisfies $\mathcal{K}_i \subseteq \mathcal{B}(D, d)$.*

Let B be the set of bad elements, and assume $|B| \leq d|\mathcal{K}_i|$. The algorithm first guesses $B' \subseteq B$. The cost of this guess is $2^{d|\mathcal{K}_i|}$. Then, it determines whether there exists $D \in \mathcal{D}$ such that $D \cap B = B'$ and $\mathcal{K}_i \subseteq \mathcal{B}(D, r_i)$. Let $K^* = \operatorname{argmax}_{K \in \mathcal{K}_i} |(K \cap B) \Delta B'|$. Then, we can claim the following.

► **Lemma 11.** *For $D \in \mathcal{D}$ such that $D \cap B = B'$, $\max_{K \in \mathcal{K}_i} |K \Delta D| = |K^* \Delta D|$.*

Proof. Let $K \in \mathcal{K}_i$. Then, $|K \Delta D| = |(K \cap B) \Delta (D \cap B)| + |(K \setminus B) \Delta (D \setminus B)|$. From the definition of B , the value of $|(K \setminus B) \Delta (D \setminus B)|$ is equal among all $K \in \mathcal{K}_i$. Thus, the maximum value of $|K \Delta D|$ for $K \in \mathcal{K}_i$ is achieved by the set K that maximizes $|(K \cap B) \Delta (D \cap B)| = |(K \cap B) \Delta B'|$. ◀

Now, it is sufficient to solve the problem of determining whether there exists $D \in \mathcal{D}$ such that $D \cap B = B'$ and $|K^* \Delta D| \leq r_i$. This corresponds to the exact extension oracle on \mathcal{D} with $r = r_i$, $X = B'$, $Y = B \setminus B'$, and $C = K^*$. Therefore, we can claim the following:

► **Lemma 12.** *Assume there exists an FPT algorithm parameterized by $k + d$ that computes a $(d+1)$ -limited k -max-distance sparsifier of \mathcal{D} with size bounded by a constant that depends only on $k + d$, and the exact extension oracle on \mathcal{D} whose time complexity is FPT parameterized by $r + |X| + |Y|$. Then, there exists an FPT algorithm parameterized by $k + d$ for the k -center/ k -sum-of-radii clustering problem on \mathcal{D} .*

■ **Algorithm 1** k -max-distance sparsification of \mathcal{D} .

```

1 Procedure KSPARSIFY( $k, r$ )
  Input:  $k \in \mathbb{Z}_{\geq 1}, r \in \mathbb{Z}_{\geq 0}$ 
2   Let  $\mathcal{K} := \emptyset$ ;
3   while true do
4     Let  $R := \bigcup_{K \in \mathcal{K}} K, f := \text{false}$ ;
5     for  $\ell' \in \{0, \dots, l\}$  do
6       Let  $\mathfrak{S}$  be the family of all sunflowers  $S \subseteq \mathcal{K}$  such that  $|S| = kr + 1$  and
7         each  $S \in \mathfrak{S}$  satisfies  $|S| = \ell'$ ;
8       for  $Y \subseteq R$  that intersects with all  $K \in \mathcal{K}$  with  $|K| = \ell'$  and the cores of all
9         sunflowers of  $\mathfrak{S}$  do
10        Let  $D = \text{EXACTEMPTYEXTENSION}(\ell', Y)$ ;
11        if  $D \neq \perp$  and  $f = \text{false}$  then
12          Add  $D$  to  $\mathcal{K}$  and  $f := \text{true}$ ;
13    if  $f = \text{false}$  then
14      break;
15  return  $\mathcal{K}$ ;

```

3 Framework for k -Max-Distance Sparification

In this section, we complete the proof of Theorem 1 by providing an FPT algorithm that uses the exact empty extension oracle on \mathcal{D} to obtain a k -max-distance sparsifier of \mathcal{D} . For further use, we show a slightly more extended result. Let $r \in \mathbb{Z}_{\geq 0}$. We construct a k -max-distance sparsifier of \mathcal{D} with respect to $\mathcal{B}(\emptyset, r)$ for $r \geq \ell$. Since $\mathcal{D} \subseteq \mathcal{B}(\emptyset, \ell) \subseteq \mathcal{B}(\emptyset, r)$ for $r \geq \ell$, this is also a k -max-distance sparsifier of \mathcal{D} (with respect to \mathcal{D}). A set family $\mathcal{S} := \{S_1, \dots, S_t\}$ is called a *sunflower* if there exists a set called *core* C such that for any $1 \leq i < j \leq t$, $S_i \cap S_j = C$. The following is well-known.

► **Lemma 13** (Sunflower Lemma [14, 18]). *Let U be a finite set, $\ell, t \in \mathbb{Z}_{\geq 0}$, and $\mathcal{K} \subseteq 2^U$ be a family consisting only of sets of size at most ℓ . If $|\mathcal{K}| > \ell!(t-1)^\ell$, then \mathcal{K} contains a sunflower of size t .*

For $t \in \mathbb{Z}_{\geq 0}$, $\mathcal{T} \subseteq 2^U$, and $Z \in 2^U \setminus \mathcal{T}$, a sunflower $\mathcal{S} \subseteq \mathcal{T}$ is a (Z, t) -sunflower of \mathcal{T} if it satisfies the following three conditions.

- $|\mathcal{S}| = t$,
- For each $S \in \mathcal{S}$, $|S| = |Z|$, and
- The core of \mathcal{S} is a subset of Z .

The following lemma is the core of our framework.

► **Lemma 14** (*). *Let U be a finite set, $\mathcal{D} \subseteq 2^U$, and $\mathcal{K} \subseteq \mathcal{D}$ be a k -max-distance sparsifier of \mathcal{D} with respect to $\mathcal{B}(\emptyset, r)$. Let $Z \in \mathcal{K}$ and assume there is a $(Z, kr + 1)$ -sunflower \mathcal{S} of $\mathcal{K} \setminus \{Z\}$. Then, $\mathcal{K} \setminus \{Z\}$ is also a k -max-distance sparsifier of \mathcal{D} with respect to $\mathcal{B}(\emptyset, r)$.*

Our algorithm is given in Algorithm 1, where $\text{EXACTEMPTYEXTENSION}(\ell', Y)$ represents the exact empty extension oracle on \mathcal{D} with arguments ℓ' and Y . The algorithm starts with $\mathcal{K} := \emptyset$ and repeatedly adds $Z \in \mathcal{D} \setminus \mathcal{K}$ such that there is no $(Z, kr + 1)$ -sunflower of \mathcal{K} to \mathcal{K} . The following lemma shows this algorithm stops after a constant number of iterations.

► **Lemma 15 (*)**. *The number of iterations of the loop starting from line 3 in Algorithm 1, as well as the size of the output family, is at most $(\ell + 1)!(kr + 1)^\ell$.*

In particular, at each step of the algorithm, since $|R| \leq |\mathcal{K}|l \leq (\ell + 1)!(kr + 1)^\ell \ell$, the size of Y chosen in line 7 is bounded by a constant. The time complexity is bounded as follows.

► **Lemma 16 (*)**. *Algorithm 1 makes at most $2^{2^{O(\ell \log(klr))}}$ calls of $\text{EXACTEMPTYEXTENSION}(\cdot)$ and has a time complexity of $2^{2^{O(\ell \log(klr))}}$.*

The correctness of the algorithm is shown as follows.

► **Lemma 17 (*)**. *Algorithm 1 outputs a k -max-distance sparsifier of \mathcal{D} with respect to $\mathcal{B}(\emptyset, r)$.*

4 Framework for d -Limited k -Max-Distance Sparsification

4.1 Overall Flow

In this section, we complete the proof of Theorem 2 by providing FPT algorithm that uses the $(-1, 1)$ -optimization oracle and the exact extension oracle on \mathcal{D} to obtain a d -limited k -max-distance sparsifier of \mathcal{D} . Actually, for further applications, we construct the slightly more general object of d -limited k -max-distance sparsifier of \mathcal{D} with respect to 2^U , not with respect to \mathcal{D} itself. Our framework consists of two steps. Let $p \in \mathbb{Z}_{\geq 0}$ be an integer with $2d < p$. The first step achieves one of the following.

- Find a set $\mathcal{C} \subseteq \mathcal{D}$ with size at most k such that $\mathcal{D} \subseteq \bigcup_{C \in \mathcal{C}} \mathcal{B}(C, p)$.
- Find a set $\mathcal{C} \subseteq \mathcal{D}$ of size $k + 1$ such that $|C \triangle C'| > 2d$ holds for any distinct $C, C' \in \mathcal{C}$.

We do this by using the following *approximate far set oracle*, which will be designed in Section 4.2.

Approximate Far Set Oracle: Let U be a finite set, $d \in \mathbb{Z}_{\geq 0}$, $\mathcal{D} \subseteq 2^U$, and $\mathcal{C} \subseteq \mathcal{D}$.

The approximate far set oracle returns one of the following.

- A set of \mathcal{D} that does not belong to $\bigcup_{C \in \mathcal{C}} \mathcal{B}(C, 2d)$.
- \perp . This option can be chosen only when $\mathcal{D} \subseteq \bigcup_{C \in \mathcal{C}} \mathcal{B}(C, p)$.

Starting with $\mathcal{C} := \emptyset$, we repeat the following steps. If the approximate far set oracle returns \perp , terminate the loop. Otherwise, add the element found by the oracle to \mathcal{C} . If the oracle returns \perp within k iterations, the first condition is achieved. If not, the set \mathcal{C} after $k + 1$ iterations satisfies the second condition. In the latter case, the following lemma shows that \mathcal{C} is a d -limited k -max-distance sparsifier.

► **Lemma 18 (*)**. *Let $r \in \mathbb{Z}_{\geq 0}$. Let \mathcal{C} be a subset of \mathcal{D} of size $k + 1$ such that for any distinct $C, C' \in \mathcal{C}$, $|C \triangle C'| \geq 2d$. Then, \mathcal{C} is a d -limited k -max-distance sparsifier of \mathcal{D} with respect to 2^U .*

Now, we assume the first condition. Let \mathcal{C} be a subset of \mathcal{D} of size at most k such that $\mathcal{D} \subseteq \bigcup_{C \in \mathcal{C}} \mathcal{B}(C, p)$. The second step involves constructing a d -limited k -max-distance sparsifier of $\mathcal{D}_C := \mathcal{D} \cap \mathcal{B}(C, p)$ with respect to $\mathcal{B}(C, p + d)$ for each $C \in \mathcal{C}$. We prove that the union of all such d -limited k -max-distance sparsifiers obtained in this manner is a d -limited k -max-distance sparsifier of \mathcal{D} with respect to 2^U .

► **Lemma 19 (*)**. *Assume $\mathcal{D} = \bigcup_{C \in \mathcal{C}} \mathcal{D}_C$. For each $C \in \mathcal{C}$, let $\mathcal{K}_C \subseteq \mathcal{D}_C$ be a d -limited k -max-distance sparsifier of \mathcal{D}_C with respect to $\mathcal{B}(C, p + d)$. Then, $\mathcal{K} := \bigcup_{C \in \mathcal{C}} \mathcal{K}_C$ is a d -limited k -max-distance sparsifier of \mathcal{D} with respect to 2^U .*

Next, we reduce the computation of d -limited k -max-distance sparsifiers to the computation of k -max-distance sparsifiers of families consisting of constant-size sets, which was discussed in Section 3. For $C \in \mathcal{C}$, let $\mathcal{D}_C^* := \{D \triangle C \mid D \in \mathcal{D}_C\}$. By the definition of \mathcal{D}_C , we have $\mathcal{D}_C^* \subseteq \mathcal{B}(\emptyset, p)$. The following holds.

► **Lemma 20 (*)**. *Let $C \in \mathcal{C}$. A subset $\mathcal{K}_C \subseteq \mathcal{D}_C$ is a d -limited k -max-distance sparsifier of \mathcal{D}_C with respect to $\mathcal{B}(C, p + d)$ if and only if $\mathcal{K}_C^* := \{K \triangle C \mid K \in \mathcal{K}_C\}$ is a d -limited k -max-distance sparsifier of \mathcal{D}_C^* with respect to $\mathcal{B}(\emptyset, p + d)$.*

If \mathcal{K}_C^* is a k -max-distance sparsifier of \mathcal{D}_C^* with respect to $\mathcal{B}(\emptyset, p + d)$, then it is also a d -limited k -max-distance sparsifier of \mathcal{D}_C^* with respect to $\mathcal{B}(\emptyset, p + d)$ for any $d \in \mathbb{Z}_{\geq 0}$. Therefore, a d -limited k -max-distance sparsifier \mathcal{K} of \mathcal{D} with respect to $\mathcal{B}(C, p + d)$ can be computed as

$$\mathcal{K} := \bigcup_{C \in \mathcal{C}} \{K^* \triangle C : K^* \in \mathcal{K}_C^*\}.$$

From the discussion in Section 3, \mathcal{K}_C^* can be obtained by calling the exact empty extension oracle on \mathcal{D}_C^* a constant number of times that depends on k , $\ell = p$, and $r = p + d$. The exact empty extension oracle for \mathcal{D}_C^* is equivalent to the exact extension oracle on \mathcal{D}_C when the inputs C, X, Y are taken to be $C, Y \cap C, Y \setminus C$, respectively. Therefore, \mathcal{K}_C^* can be obtained by calling the exact extension oracle on \mathcal{D}_C a constant number of times that depends only on k and p .

4.2 Designing the Approximate Far Set Oracle

Here, we design a randomized algorithm parameterized by $|\mathcal{C}|$ and d for the approximate far set oracle. Our algorithm repeats the following sufficient number of times: It selects a weight vector $w \in \{-1, 1\}^U$ uniformly at random and finds a set $D \in \mathcal{D}$ that maximizes $w(D) := \sum_{e \in D} w_e$. If the found D does not belong to $\bigcup_{C \in \mathcal{C}} \mathcal{B}(C, 2d)$, it outputs this D and terminates. If no such D is found after a sufficient number of iterations, it returns \perp . We now prove the correctness of the algorithm. We can claim the following.

► **Lemma 21 (*)**. *Assume $\max_{D \in \mathcal{D}} w(D) > \max_{C \in \mathcal{C}} w(C) + 2d$. Then, the D that attains the maximum on the left-hand side does not belong to $\bigcup_{C \in \mathcal{C}} \mathcal{B}(C, 2d)$.*

The following lemma is the core of the analysis.

► **Lemma 22 (*)**. *Assume $p \geq (4d + 2)^2 \cdot 2^{k-1}$ and $|\mathcal{C}| \leq k$. Let $D \in \mathcal{D}$ and assume $D \notin \bigcup_{C \in \mathcal{C}} \mathcal{B}(C, p)$. Then,*

$$\Pr \left[w(D) > \max_{C \in \mathcal{C}} w(C) + 2d \right] \geq 2^{-2^{O(k)}}.$$

By repeating the sampling of w a sufficient number of times, we can state the following.

► **Lemma 23**. *Let $\epsilon > 0$, $\mathcal{D}, \mathcal{C} \subseteq 2^U$, $k \in \mathbb{Z}_{\geq 1}$, and $d \in \mathbb{Z}_{\geq 0}$. Assume $|\mathcal{C}| \leq k$. Then, there exists a randomized algorithm that runs in time $2^{2^{O(k)}} \log \epsilon^{-1}$ and satisfies the following.*

- *If there exists $D \in \mathcal{D}$ such that $D \notin \bigcup_{C \in \mathcal{C}} \mathcal{B}(C, (4d + 2)^2 \cdot 2^k)$, the algorithm returns a set $D' \in \mathcal{D}$ satisfying $D' \notin \bigcup_{C \in \mathcal{C}} \mathcal{B}(C, 2d)$ with probability at least $1 - \epsilon$.*
- *If not, the algorithm returns either \perp or a set $D' \in \mathcal{D}$ satisfying $D' \notin \bigcup_{C \in \mathcal{C}} \mathcal{B}(C, 2d)$.*

Combining Lemma 23 with the results from Sections 3 and 4.1, we have the following.

► **Lemma 24 (*)**. Let $\epsilon > 0$, $\mathcal{D} \subseteq 2^U$, $k \in \mathbb{Z}_{\geq 1}$, and $d \in \mathbb{Z}_{\geq 0}$. Then, there exists a randomized algorithm that, with probability $1 - \epsilon$, computes a d -limited k -max-distance sparsifier of \mathcal{D} with respect to 2^U with size at most $2^{2^{O(k+\log d)}}$ in time $\left(2^{2^{O(k+\log d)}} + 2^{2^{O(k)}} \log \epsilon^{-1}\right) \text{poly}(|U|)$. It uses at most $2^{2^{O(k)}} \log \epsilon^{-1}$ calls to the $(-1, 1)$ -optimization oracle on \mathcal{D} and at most $2^{2^{O(k+\log d)}}$ calls to the exact extension oracle on \mathcal{D} such that $r \leq (4d+2)^2 2^{k-1}$ and $|X|, |Y| \leq 2^{2^{O(k+\log d)}}$.

References

- 1 Pankaj Agarwal, Aryan Esmailpour, Xiao Hu, Stavros Sintos, and Jun Yang. Computing a well-representative summary of conjunctive query results. *Proceedings of the ACM on Management of Data*, 2(5):1–27, 2024. doi:10.1145/3695835.
- 2 Pankaj Agarwal, Sarel Har-Peled, and Kasturi Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004. doi:10.1145/1008731.1008736.
- 3 Amihood Amir, Jessica Fidler, Liam Roditty, and Oren Sar Shalom. On the efficiency of the hamming c -centerstring problems. In *Combinatorial Pattern Matching (CPM)*, pages 1–10. Springer, 2014. doi:10.1007/978-3-319-07566-2_1.
- 4 Sayan Bandyapadhyay, William Lochet, and Saket Saurabh. FPT constant-approximations for capacitated clustering to minimize the sum of cluster radii. In *International Symposium on Computational Geometry (SoCG)*, volume 258, pages 12:1–12:14, 2023. doi:10.4230/LIPICS.SOCG.2023.12.
- 5 Jørgen Bang-Jensen, Kristine Vitting Klinkby, and Saket Saurabh. k -distinct branchings admits a polynomial kernel. In *European Symposium on Algorithms (ESA)*, pages 11:1–11:15, 2021. doi:10.4230/LIPICS.ESA.2021.11.
- 6 Jørgen Bang-Jensen, Saket Saurabh, and Sven Simonsen. Parameterized algorithms for non-separating trees and branchings in digraphs. *Algorithmica*, 76:279–296, 2016. doi:10.1007/S00453-015-0037-3.
- 7 Julien Baste, Michael Fellows, Lars Jaffke, Tomáš Masařík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303:103644, 2022. doi:10.1016/J.ARTINT.2021.103644.
- 8 Julien Baste, Lars Jaffke, Tomáš Masařík, Geevarghese Philip, and Günter Rote. FPT algorithms for diverse collections of hitting sets. *Algorithms*, 12(12):254, 2019. doi:10.3390/A12120254.
- 9 Clément Carbonnel and Emmanuel Hebrard. Propagation via kernelization: The vertex cover constraint. In *International Conference on Principles and Practice of Constraint Programming (CP)*, pages 147–156. Springer, 2016. doi:10.1007/978-3-319-44953-1_10.
- 10 Clément Carbonnel and Emmanuel Hebrard. On the kernelization of global constraints. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 578–584, 2017. doi:10.24963/IJCAI.2017/81.
- 11 Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. In *ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 2001. doi:10.1145/380752.380753.
- 12 Xianrun Chen, Dachuan Xu, Yicheng Xu, and Yong Zhang. Parameterized approximation algorithms for sum of radii clustering and variants. In *AAAI Conference on Artificial Intelligence*, volume 38, pages 20666–20673, 2024. doi:10.1609/AAAI.V38I18.30053.
- 13 Ryan Curtin, Benjamin Moseley, Hung Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. RK-means: Fast clustering for relational data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2742–2752. PMLR, 2020. URL: <http://proceedings.mlr.press/v108/curtin20a.html>.

- 14 Marek Cygan, Fedor Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 15 Erik Demaine, Fedor Fomin, MohammadTaghi Hajiaghayi, and Dimitrios Thilikos. Fixed-parameter algorithms for (k, r) -center in planar graphs and map graphs. *ACM Transactions on Algorithms (TALG)*, 1(1):33–47, 2005. doi:10.1145/1077464.1077468.
- 16 Eduard Eiben, Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. On the parameterized complexity of clustering problems for incomplete data. *Journal of Computer and System Sciences*, 134:1–19, 2023. doi:10.1016/J.JCSS.2022.12.001.
- 17 Eduard Eiben, Tomohiro Koana, and Magnus Wahlström. Determinantal sieving. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–423. SIAM, 2024. doi:10.1137/1.9781611977912.16.
- 18 Paul Erdős and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960.
- 19 Aryan Esmailpour and Stavros Sintos. Improved approximation algorithms for relational clustering. *Proceedings of the ACM on Management of Data*, 2(5):213:1–213:27, 2024. doi:10.1145/3695831.
- 20 Andreas Emil Feldmann and Dániel Marx. The parameterized hardness of the k -center problem in transportation networks. *Algorithmica*, 82:1989–2005, 2020. doi:10.1007/S00453-020-00683-W.
- 21 Fedor Fomin, Petr Golovach, Lars Jaffke, Geevarghese Philip, and Danil Sagunov. Diverse pairs of matchings. *Algorithmica*, 86(6):2026–2040, 2024. doi:10.1007/S00453-024-01214-7.
- 22 Fedor Fomin, Petr Golovach, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. Diverse collections in matroids and graphs. *Mathematical Programming*, 204(1):415–447, 2024. doi:10.1007/S10107-023-01959-Z.
- 23 Ryo Funayama, Yasuaki Kobayashi, and Takeaki Uno. Parameterized complexity of finding dissimilar shortest paths. *arXiv preprint arXiv:2402.14376*, 2024. doi:10.48550/arXiv.2402.14376.
- 24 Tatsuya Gima, Yuni Iwamasa, Yasuaki Kobayashi, Kazuhiro Kurita, Yota Otachi, and Rin Saito. Computing diverse pair of solutions for tractable sat. *arXiv preprint arXiv:2412.04016*, 2024. doi:10.48550/arXiv.2412.04016.
- 25 Gregory Gutin, Felix Reidl, and Magnus Wahlström. k -distinct in-and out-branchings in digraphs. *Journal of Computer and System Sciences*, 95:86–97, 2018. doi:10.1016/J.JCSS.2018.01.003.
- 26 Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, and Yota Otachi. Finding diverse trees, paths, and more. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 3778–3786, 2021. doi:10.1609/AAAI.V35I5.16495.
- 27 Wen-Lian Hsu and George Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979. doi:10.1016/0166-218X(79)90044-1.
- 28 Tanmay Inamdar and Kasturi Varadarajan. Capacitated sum-of-radii clustering: An FPT approximation. In *European Symposium on Algorithms (ESA)*, 2020.
- 29 Neeldhara Misra, Harshil Mittal, and Ashutosh Rai. On the parameterized complexity of diverse SAT. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 322, pages 50:1–50:18, 2024. doi:10.4230/LIPICS.ISAAC.2024.50.
- 30 Benjamin Moseley, Kirk Pruhs, Alireza Samadian, and Yuyan Wang. Relational algorithms for k -means clustering. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198, pages 97:1–97:21, 2021. doi:10.4230/LIPICS.ICALP.2021.97.
- 31 Yuto Shida, Giulia Punzi, Yasuaki Kobayashi, Takeaki Uno, and Hiroki Arimura. Finding diverse strings and longest common subsequences in a graph. In *Symposium on Combinatorial Pattern Matching (CPM)*, volume 296, pages 27:1–27:19, 2024. doi:10.4230/LIPICS.CPM.2024.27.