Fully dynamic clustering and diversity maximization in doubling metrics

Paolo Pellizzoni[1,2], Andrea Pietracaprina[2], and Geppino Pucci[2]

[1] Max Planck Institute of Biochemistry
Martinsried, Germany
`pellizzoni@biochem.mpg.de`
[2] Dept. of Information Engineering
University of Padova, Italy
`{andrea.pietracaprina,geppino.pucci}@unipd.it`

Abstract. We present approximation algorithms for some variants of center-based clustering and related problems in the fully dynamic setting, where the pointset evolves through an arbitrary sequence of insertions and deletions. Specifically, we target the following problems: k-center (with and without outliers), matroid-center, and diversity maximization. All algorithms employ a coreset-based strategy and rely on the use of the cover tree data structure, which we crucially augment to maintain, at any time, some additional information enabling the efficient extraction of the solution for the specific problem. For all of the aforementioned problems our algorithms yield $(\alpha+\epsilon)$-approximations, where α is the best known approximation attainable in polynomial time in the standard offline setting (except for k-center with z outliers where $\alpha = 2$ but we get a $(3+\epsilon)$-approximation) and $\epsilon > 0$ is a user-provided accuracy parameter. The analysis of the algorithms is performed in terms of the doubling dimension of the underlying metric. Remarkably, and unlike previous works, the data structure and the running times of the insertion and deletion procedures do not depend in any way on the accuracy parameter ϵ and, for the two k-center variants, on the parameter k. For spaces of bounded doubling dimension, the running times are dramatically smaller than those that would be required to compute solutions on the entire pointset from scratch. To the best of our knowledge, ours are the first solutions for the matroid-center and diversity maximization problems in the fully dynamic setting.

Keywords: dynamic algorithms · k-center · outliers · matroid center · diversity maximization · cover tree.

1 Introduction

Clustering, that is the task of partitioning a set of points according to some similarity metric, is one of the fundamental primitives in unsupervised learning and data mining, with applications in several fields, such as bioinformatics, computer vision and recommender systems [26]. A commonly used formulation

is the *k-center* problem, where, given a set of points from a metric space and a parameter k, one seeks to select k points as cluster centers so that the maximum distance from any point to its closest center is minimized. Since finding the optimal solution is known to be NP-hard [18], in practice one has to settle for approximate solutions.

In many practical applications, the set of points to be clustered is not static, but evolves with time, and the clustering algorithms have to cope with the insertion and deletion of arbitrary points efficiently. For example, in the context of social network analytics, a large number of new profiles are created every second, therefore adding new data to the pointset of interest, and the service provider has to allow for the deletion of user data at any point in time, e.g., to comply with GDPR regulations. As another example, if one wishes to track and cluster the currently watched contents from a video streaming service, users starting and stopping to watch videos result in a dynamically changing pointset, with thousands of updates per second. Because of this ever-increasing need in efficient algorithms that can cope with non-static data, in recent years there has been a surge in research efforts to develop *fully dynamic* clustering algorithms, whose main focus is on the efficient handling of arbitrary insertions and deletions [2, 10, 11, 19]. Several variants of the k-center problem have been intensely studied in recent years, tailored at addressing specific limitation of the problem or at adding additional constraints. Since k-center's objective function involves a maximum, the optimal solution may be heavily affected by points, dubbed *outliers*, which are markedly distant from the other ones. Indeed, especially when data volumes are high, the probability of observing outliers, such as from noisy or erroneous measurements, is non negligible, and being able to cope with them is of paramount importance. A robust formulation of the problem, referred to as *k-center with z outliers* has been introduced in [13], where the objective function is allowed to disregard the z most distant points from the selected centers. Another studied variant of k-center is the *matroid center* problem [14], which, given a set of points and a matroid defined on it, aims at finding a set of centers forming an independent set of the matroid which minimizes the maximum distance of any point from the closest center. This variant is extremely flexible and can be employed to model a wide range of constraints (e.g., fairness) on the solution [15]. As for the standard k-center, the matroid center problem admits a robust formulation with z outliers [14]. Finally, a problem closely related to k-center is *diversity maximization* [1], whose goal is somehow dual with respect to the one of k-center, as k points are to be selected so that a suitable notion of diversity among them is maximized. This problem has a huge impact on information retrieval applications, where it is crucial that the information provided to the user be non-redundant.

A widely-used notion of dimensionality for general metric spaces is the *doubling dimension* [20, 21], which is defined as the minimum value $D \geq 0$ such that any ball of any radius $r \geq 0$ can be covered by at most 2^D balls of radius $r/2$. In many real-world instances, the points of interest either belong to low-dimensional spaces or lie on low-dimensional manifolds of the higher-dimensional

metric space they belong to, and this property has been extensively exploited to obtain efficient clustering algorithms [8, 19, 28]. In this paper, we tackle the k-center problem, along with the aforementioned variants to it, in the fully dynamic setting, for spaces of bounded doubling dimension.

1.1 Related work

In the sequential setting, k-center admits 2-approximate algorithms, such as Gonzalez's [18], but no $(2 - \epsilon)$-approximate ones unless P=NP. For the robust formulation with z outliers, there exists a simple, combinatorial 3-approximation algorithm [13], which has been extended to weighted pointsets in [5]. More complex LP-based 2-approximate algorithms have been developed [9, 22], but they are less amenable to practical implementations. For what concerns the matroid-center problem, [14] provides a 3-approximate algorithm for the standard formulation and a 7-approximate algorithm for the formulation with outliers, which has been improved to a 3-approximation in [22]. The diversity maximization problem admits several instantiations, depending on the specific diversity function embodied in its objective, which are all NP-hard but admit polynomial-time $O(1)$-approximation algorithms. For an overview of such methods, we refer to [1, 6] and references therein.

In [10], the authors developed the first fully dynamic k-center algorithm, which is able to return a $(2 + \epsilon)$-approximate solution under arbitrary insertions and deletions of a non-adaptive adversary in general metrics. The algorithm is randomized and has an (amortized) update time of $O\left(k^2 \epsilon^{-1} \log \Delta\right)$, where Δ denotes the *aspect ratio* of the pointset, namely the ratio between the largest and the smallest distance between any two points. This approach has been recently improved in [2], where an algorithm with expected amortized $O\left((k + \log n)\epsilon^{-1} \log \Delta \log n\right)$ update time is presented, where n is the maximum number of points at any time. It has to be remarked that both these works make use of data structures for storing the dynamically changing pointset, which are statically configured to deal with *fixed* values of k and ϵ. Answering queries for different clustering granularities and/or accuracies would, in principle, require building the data structure from scratch. Also both works use data structures whose size, with respect to the number of points, is superlinear by at least a factor $O(\log \Delta/\epsilon)$. Recently, [11] presented a randomized algorithm that returns a $14 + \epsilon$ (bi-criteria) approximate solution when discarding at most $(1 + \lambda)z$ outliers. Again, their data structure works for fixed k and ϵ, although it does not depend on λ and z, and requires superlinear size by a factor $O(\log \Delta/\epsilon)$.

In [19], the authors propose a fully dynamic k-center algorithm for points belonging to a low-dimensional space based on the *navigating net* data structure [25], which affords insertions and deletions in $O\left((1/\epsilon)^{O(D)} \log \Delta \log \log \Delta \log \epsilon^{-1}\right)$ time, where D is the doubling dimension of the metric space. While their approach allows clustering queries for arbitrary values of k, the data structure is built for a specific value of the accuracy parameter ϵ, and must be rebuilt from scratch if a different accuracy ϵ' is sought for. Moreover, the data structure requires superlinear space, by a factor $O\left((1/\epsilon)^{O(D)} \log \Delta \log \log \Delta \log \epsilon^{-1}\right)$.

Finally, k-center clustering has been tackled in other simpler dynamic frameworks, such as in the insertion-only setting [5,8,12,23], or in the sliding window setting [4,16,28,29]. In these frameworks though, the focus is usually on keeping the working memory sublinear in the number of points rather than achieving fast update times.

1.2 Our contributions

In this paper, we present approximation algorithms for center-type problems in the fully dynamic setting. The core data structure at the foundation of our approach is the *cover tree* [3], which was originally designed for answering nearest neighbor queries in low-dimensional metric spaces. We augment such data structure to maintain the necessary information for solving the problems at hand, under arbitrary insertion and deletions. The augmented data structure is used to extract, at any point in time, a *coreset*, that is, a small subset of representative points that can be used to obtain solutions of any desired accuracy on the whole pointset for the problem at hand. Upon query requests, the solution is returned by running a sequential algorithm on the extracted coreset, which allows for fast execution times, independently on the current number of points. Specifically, our contributions are the following.

1. We augment the cover tree data structure to maintain efficiently, in each of its nodes, information about the subset of points stored in its subtree, such as its cardinality or a maximal independent set of the submatroid induced by such subset (in case a matroid M is defined over the whole pointset). When no matroid information is maintained, the data structure takes space linear in the number of points, and allows for insertion and deletions in $O\left(2^{O(D)} \log \Delta\right)$ time, where D is the doubling dimension of the metric space, and Δ is the aspect ratio of the current set of points. When maintaining matroid information, the space requirements and the deletion times grow by a factor at most $\text{rank}(M)$.
2. We provide an iterative formulation of the algorithms used to maintain the cover tree data structure, which, compared to the original recursive formulation of [3], affords simpler correctness proofs and more efficient implementations. In fact, it has been recently argued that the complexity analysis for the update operations presented in the original paper has some flaws [17], which we fix by making it parametric in D and Δ.
3. We devise a fully dynamic $(2 + \epsilon)$-approximate algorithm for the k-center problem. Unlike all aforementioned previous works, our data structure allows to query for solutions for *arbitrary values* of k and ϵ. Also, the query time exhibits a linear dependency on k.
4. We devise a $(3 + \epsilon)$ approximation algorithm for the fully dynamic k-center with z outliers problem. Our method allows to choose k, ϵ, and z at query time. We remark that the only previously available algorithm can only return a $(14+\epsilon)$ bi-criteria solution (i.e. with an additional slackness on the number of outliers), and requires to fix k and ϵ beforehand.

5. We present the first fully dynamic algorithm for the matroid center problem. Our algorithm return a $(3 + \epsilon)$ approximate solution.
6. We present the first fully dynamic algorithms for diversity maximization. Our algorithms return $(\alpha_{\mathrm{div}} + \epsilon)$ approximate solutions, where α_{div} is the best approximation factor achievable by a sequential algorithm on the variant of the problem at hand.

An important feature of our algorithms is that they are fully oblivious to D, in the sense that the actual value of this parameter only influences the analysis but are not needed for the algorithms to run. This is a very desirable feature, since, in practice, this value is difficult to estimate.

The rest of the paper is organized as follows. Section 2 provides the formal definition of the problems and of the theoretical tools we use in the analysis. Section 3 is dedicated to describing the structure of the novel augmented cover tree data structure, while Section 4 details how to maintain such data structure efficiently. In Section 5, we present the approximation algorithms, which rely on the cover tree data structure. Section 6 concludes the paper with some final remarks.

2 Preliminaries

This section formally defines the problems studied in this paper, and states some important technical facts.

2.1 (Robust) k-center problem

Consider a metric space (U, dist) and a set $S \subseteq U$ of n points. For any $p \in U$ and any subset $C \subseteq S$, we use the notation $\mathrm{dist}(p, C) = \min_{q \in C} \mathrm{dist}(p, q)$, and define the *radius of C with respect to S* as

$$r_C(S) = \max_{p \in S} \mathrm{dist}(p, C).$$

For a positive integer $k < n$, the *k-center* problem requires to find a subset $C \subseteq S$ of size at most k which minimizes $r_C(S)$. The points of the solution C are referred to as *centers*. Note that C induces a partition of S into $|C|$ clusters, by assigning each point to its closest center (with ties broken arbitrarily). We denote the radius of the optimal solution by $r_k^*(S)$. The popular seminal work by Gonzalez [18] presents a 2-approximation sequential algorithm for the k-center program, based on the simple, $O(nk)$-time greedy strategy that selects the first center arbitrarily and each subsequent center as the point with maximum distance from the set of previously selected ones. The author also shows that, in general metric spaces, it is impossible to achieve an approximation factor $2 - \epsilon$, for any fixed $\epsilon > 0$, unless P = NP.

The algorithms presented in this paper crucially rely on confining the computation of the solution on a succinct *coreset T* efficiently extracted from the (possibly large) input S, which contains a close enough "representative" for each point in S. The quality of a coreset T is captured by the following definition.

Definition 1. *Given a pointset S and a value $\epsilon > 0$, a subset $T \subseteq S$ is an (ϵ, k)-coreset for S (w.r.t. the k-center problem) if $r_T(S) \leq \epsilon r_k^*(S)$.*

In real world applications, large datasets often include noisy points which, if very distant from all other points, may severely distort the optimal center selection. To handle these scenarios, the following important generalization of the k-center problem has been considered. For positive $k, z < n$, the *k-center problem with z outliers* (also referred to as *robust (k, z)-center*) requires to find a subset $C \subseteq S$ of size k minimizing $r_C(S - Z_C)$, where Z_C is the set of z points in S with the largest distances from C, which are regarded as outliers to be discarded from the clustering. We denote the radius of the optimal solution of this problem by $r_{k,z}^*(S)$. Observe that the k-center problem with z outliers reduces to the k-center problem for $z = 0$. Also, it is straightforward to argue that

$$r_{k+z}^*(S) \leq r_{k,z}^*(S). \tag{1}$$

A well known 3-approximation sequential algorithm for the k-center problem with z outliers, which runs in $O\left(kn^2 \log n\right)$ time was devised in [13]. In this work, we will make use of a more general formulation of the problem, referred to as *weighted k-center with z outliers*, where each point $p \in S$ carries a positive integer weight $w(p)$, and the desired set C of k centers must minimize $r_C(S - Z_C)$, where Z_C is the set of points with the largest distances from C, of maximum cardinality and aggregate weight at most z.

2.2 Matroid center problem

Another variant of the k-center problem requires the solution C to satisfy an additional constraint, specified through a matroid. A *matroid* [27] on a pointset S is a pair $M = (S, I)$, where I is a family of subsets of S, called *independent sets*, satisfying the following properties: (i) the empty set is independent; (ii) every subset of an independent set is independent (*hereditary property*); and (iii) if $A, B \in I$ and $|A| > |B|$, then there exists $x \in A \setminus B$ such that $B \cup \{x\} \in I$ (*augmentation property*). An independent set is *maximal* if it is not properly contained in another independent set. Given a matroid $M = (S, I)$, the *matroid center problem* on M requires to determine an independent set $C \in I$ minimizing the radius $r_C(S)$. We let $r^*(M) = \min_{C \in I} r_C(S)$ to denote the radius of the optimal solution. The augmentation property ensures that all maximal independent sets of M have the same size, which is referred to as *rank* of M, denoted by $\text{rank}(M)$. It is easy to argue that $r_{\text{rank}(M)}^*(S) \leq r^*(M)$, since each solution to the matroid center problem on is also a solution to the k-center problem with $k = \text{rank}(M)$.

As customary in previous works [1, 6, 23], throughout the paper we assume that a constant-time oracle is available to check the independence of any subset of S. A combinatorial 3-approximation algorithm for the matroid center problem in general metrics is presented in [14], which requires time polynomial in $|S|$ and $\text{rank}(M)$.

Some important structural properties of matroids will be used in the derivations of our results. It is easily seen that for any subset $S' \subseteq S$, $M' = (S', I')$, where $I' \subseteq I$ is the restriction of I to subsets of S' is also a matroid. An *extended augmentation property*, stated in the following fact, was proved in [30, Lemma 2.1] (see also [23]).

Fact 1 *Let $M = (S, I)$ be a matroid. Consider an independent set $A \in I$, a subset $S' \subseteq S$, and an independent set $B \subseteq S'$ which is maximal within the submatroid $M' = (S', I')$. If there exists $y \in S' \setminus A$ such that $A \cup \{y\} \in I$, then there exists $x \in B \setminus A$ such that $A \cup \{x\} \in I$.*

The next fact can be easily proved as a consequence of the extended augmentation property.

Fact 2 *Let $M = (S, I)$ be a matroid, and let $S_1, \ldots, S_h$ be a partition of S into h disjoint subsets. If $A_1 \subseteq S_1, \ldots, A_h \subseteq S_h$ are maximal independent sets of the submatroids $M_1 = (S_1, I_1), \ldots, M_h = (S_h, I_h)$, then $\cup_{I=1}^{h} A_i$ contains a maximal independent set of M.*

Different matroids are associated to different semantics of the constraint on the centers. An important instantiation is the case of the *partition matroid $M_P = (S, I_P)$*, where each point in S is associated to one of $m \leq k$ of categories, and I_P consists of all subsets with at most k_i points of the i-th category, with $\sum_{i=0}^{m} k_i = k$. For instance, this matroid can be employed to model fairness constraints [15, 24].

2.3 Diversity maximization

Let div $: 2^S \to \mathbb{R}$ be a *diversity function* that maps any subset $X \subset S$ to some non-negative real number. For a specific diversity function div and a positive integer $k \leq n$, the goal of the *diversity maximization problem* is to find a set $C \subseteq S$ of size k that maximizes $\text{div}(C)$. We denote the optimal value of the objective function as $\text{div}_k^*(S) = \max_{C \subseteq S, |C|=k} \text{div}(C)$. In this paper, we will focus on several variants of the problem based on different diversity functions amply studied in the previous literature, which are reported in Table 1. All of these variants are known to be NP-hard, and Table 1 also lists the best known approximation ratios attainable in polynomial time (see [1, 8] and references therein).

2.4 Doubling dimension

We will relate the performance of our algorithms to the dimensionality of the data which, for a general metric space (U, dist), can be captured by the notion of doubling dimension, reviewed below. For any $p \in U$ and $r > 0$, the *ball of radius r centered at p*, denoted as $B(p, r)$, is the subset of all points of U at distance at most r from p. The *doubling dimension* of U is the minimum value D such that, for all $p \in U$, any ball $B(p, r)$ is contained in the union of at most

Problem	Diversity measure $\mathrm{div}(X)$	Sequential approx. α_{div}
remote-edge	$\min_{p,q \in X} d(p,q)$	2
remote-clique	$\sum_{p,q \in X} d(p,q)$	2
remote-star	$\min_{c \in X} \sum_{q \in X \setminus \{c\}} d(c,q)$	2
remote-bipartition	$\min_{Q \subset X, |Q|=\lfloor |X|/2 \rfloor} \sum_{q \in Q, z \in X \setminus Q} d(q,z)$	3
remote-tree	$w(\mathrm{MST}(X))$	4
remote-cycle	$w(\mathrm{TSP}(X))$	3

Table 1. Diversity functions considered in this paper. $w(\mathrm{MST}(X))$ (resp., $w(\mathrm{TSP}(X))$) denotes the minimum weight of a spanning tree (resp., Hamiltonian cycle) of the complete graph whose nodes are the points of X and whose edge weights are the pairwise distances among the points. The last column lists the best known approximation factor.

2^D balls of radius $r/2$ centered at points of U. In recent years, the notion of doubling dimension has been used extensively for a variety of applications (e.g., see [7,8,20,28] and references therein). The following fact is a simple consequence of the definition.

Fact 3 *Let X be a set of points from a metric space of doubling dimension D, and let $Y \subseteq X$ be such that any two distinct points $a, b \in Y$ have pairwise distance $\mathrm{dist}(a,b) > r$. Then for every $R \geq r$ and any point $p \in X$, we have $|B(p,R) \cap Y| \leq 2^{\lceil \log_2(2R/r) \rceil \cdot D} \leq (4R/r)^D$. If $R/r = 2^i$, then the bound can be lowered to $|B(p,R) \cap Y| \leq (2R/r)^D$.*

3 Augmented Cover Trees

Let S be a set of n points from a metric space (U, dist) of doubling dimension D. Our algorithms employ an augmented version of the cover tree data structure, defined in [3], which, for completeness, we review below. Conceptually, a *cover tree T* for S is an infinite tree, where each node corresponds to a single point of S, while each point of S is associated to one or more nodes. The levels of the tree are indexed by integers, decreasing from the root towards the leaves. For $\ell \in (-\infty, +\infty)$, we let T_ℓ to be the set of nodes of level ℓ, and let $\mathrm{pts}(T_\ell)$ be the points associated with the nodes of T_ℓ, which are required to be distinct. For each node $u \in T$ we maintain: its associated point (u.point); a pointer to its parent (u.parent); the list of pointers to its children (u.children); and the level of u in T (u.level). For brevity, in what follows, for any two nodes $u, v \in T$ (resp., any point $p \in U$ and node $u \in T$), we will use $\mathrm{dist}(u,v)$ (resp., $\mathrm{dist}(p,u)$) to denote $\mathrm{dist}(u.\mathrm{point}, v.\mathrm{point})$ (resp., $\mathrm{dist}(p, u.\mathrm{point})$). The tree must satisfy the following properties. For every level ℓ:

1. $\mathrm{pts}(T_\ell) \subseteq \mathrm{pts}(T_{\ell-1})$;

2. for each $u \in T_\ell$, $\text{dist}(u, u.\text{parent}) \leq 2^{\ell+1}$;
3. for all $u, v \in T_\ell$, $\text{dist}(u, v) > 2^\ell$.

For every $p \in S$, let $\ell(p)$ denote the largest index such that $p \in \text{pts}(T_{\ell(p)})$. The definition implies that for every $\ell \leq \ell(p)$, $p \in \text{pts}(T_\ell)$, and the node u in T_ℓ with $u.\text{point} = p$ is a *self-child* of itself, in the sense that $u.\text{parent.point} = p$, since for every other $v \in T_{\ell+1}$, with $v.\text{point} \neq p$, $\text{dist}(u, v) > 2^{\ell+1}$. Let $d_{\min}$ and $d_{\max}$ denote, respectively, the minimum and maximum distances between two points of S, and define $\Delta = d_{\max}/d_{\min}$ as the *aspect ratio* of S. It can be easily seen that for every $\ell < \log_2 d_{\min}$, $\text{pts}(T_\ell) = S$, and for every $\ell \geq \log_2 d_{\max}$, $|\text{pts}(T_\ell)| = 1$. We define $\ell_{\min}$ (resp., $\ell_{\max}$) as the largest (resp., smallest) index such that $\text{pts}(T_\ell) = S$ (resp., $|\text{pts}(T_\ell)| = 1$), and note that every node u in a level T_j with $j > \ell_{\max}$ or $j \leq \ell_{\min}$ has only the self-child in $u.\text{children}$. Therefore, we will consider only the portion of the tree constituted by the levels T_ℓ with $\ell \in [\ell_{\min}, \ell_{\max}]$ and will regard the unique node $r \in T_{\ell_{\max}}$ as the root of the tree. The above observations imply that the number of levels in this portion of the tree is $O(\log \Delta)$.

Cover trees have been initially proposed as data structures for efficiently retrieving nearest neighbors. This feature, which will also be exploited in our context, crucially relies on the notion of *cover set*. For any point p from the metric space U, and for every index $\ell \leq \ell_{\max}$ the *cover set* (*for p at level ℓ*) $Q_\ell^p \subseteq T_\ell$ is defined in an inductive way as follows:

$$
\begin{aligned}
Q_{\ell_{\max}}^p &= \{r\} \\
Q_\ell^p &= \{u \in T_\ell \;:\; u.\text{parent} \in Q_{\ell+1}^p \wedge \text{dist}(u, p) \leq 2^{\ell+1}\} \quad \text{for } \ell < \ell_{\max}
\end{aligned}
\tag{2}
$$

The next lemma shows that cover sets for a point p include, at each level of the tree, all points somewhat "close" to p, at a scale prescribed by the level.

Lemma 1. *For every point p from (U, dist), $\ell \leq \ell_{\max}$, and $u \in T_\ell - Q_\ell^p$, we have $\text{dist}(p, u) > 2^{\ell+1}$.*

Proof. The proof proceeds by (backward) induction on ℓ. For the base case $\ell = \ell_{\max}$, the statement holds vacuously. Assume now that the statement holds at level $\ell + 1$, and let $u \in T_\ell - Q_\ell^p$. The statement immediately follows when $u.\text{parent} \in Q_{\ell+1}^p$. Otherwise, since $u.\text{parent} \in Q_{\ell+1}^p - T_{\ell+1}$, by the inductive hypothesis it must be $\text{dist}(p, u.\text{parent}) > 2^{\ell+2}$, whence $\text{dist}(p, u) \geq \text{dist}(p, u.\text{parent}) - \text{dist}(u, u.\text{parent}) > 2^{\ell+1}$. $\square$

We now relate the size of the cover sets to the doubling dimension D of (U, dist).

Lemma 2. *For every point $p \in U$ and $\ell \leq \ell_{\max}$, we have that*

1. $|Q_\ell^p| \leq 4^D$.
2. $\sum_{u \in Q_\ell^p} |u.\text{children}| \leq 12^D$.

Proof. For each $u_1, u_2 \in T_\ell$, we have that $\text{dist}(u_1, u_2) > 2^\ell$. Also, $\text{pts}(Q_\ell^p) \subseteq B(p, 2^{\ell+1}) \cap T_\ell$. Then, by Fact 3, it follows that $|Q_\ell^p| \leq (2 \cdot 2^{\ell+1}/2^\ell)^D = 4^D$. Moreover, for each $u \in Q_\ell^p$ and each $u' \in u.\text{children}$, we have that $\text{dist}(u', p) \leq \text{dist}(u', u) + \text{dist}(u, p) \leq 2^\ell + 2^{\ell+1} = 3 \cdot 2^\ell$, whence $u'.\text{point} \in B(p, 3 \cdot 2^\ell) \cap \text{pts}(T_{\ell-1})$. Then, again by Fact 3, it follows that $|\{u' \in u.\text{children s.t. } u \in Q_\ell^p\}| \leq (2 \cdot 3 \cdot 2^\ell/2^{\ell-1})^D = 12^D$.

We represent the entire tree T by a pointer to the root, and recall that $r.\text{level} = \ell_{\max}$. Note that this naive representation of T, referred to as *implicit representation* in [3], requires $O(n \log \Delta)$ space. In order to save space, a more compact representation T, referred to as *explicit representation* in [3], is used, where chains of 1-child nodes, which correspond to instances of the same point, are coalesced. More precisely, this representation only maintains explicitly nodes that have children other than the self-child. Therefore, any maximal chain of nodes in $T_i, T_{i-1}, \ldots, T_{i-j}$ all associated to the same point p and such that each of them (up to Level T_{i-j+1}) is only parent to the self-child, is represented through the explicit node $u \in T_i$ with $u.\text{children}$ set to the list of children of the node of the chain in T_{i-j}. It is easy to argue that this compact representation takes only $O(n)$ space. It is important to observe that, given a point $p \in U$, the cover sets Q_ℓ^p for $\ell \leq \ell_{\max}$ can be constructed in a top-down fashion from the explicit representation of T by simply recreating the contracted chains of implicit nodes. In our algorithms, in each cover set Q_ℓ^p, an implicit node v will be represented by the explicit ancestor u associated to the head of the chain containing v. By Lemma 2, Q_ℓ^p can be constructed from $Q_{\ell+1}^p$ in $O(12^D)$ time.

3.1 Augmenting the basic structure

In this subsection, we show how to augment the cover tree data structure so to maintain at its nodes two additional data fields, which will be exploited in the target applications presented later. Suppose that a matroid $M = (U, I)$ is defined over the universe U. An *augmented cover tree* T for S (with respect to M) is a cover tree such that each node U stores the following two additional fields: a positive weight $u.\text{weight} = |S_u|$, where S_u is the subset of points of S associated with nodes in the subtree rooted at node u, and a set of points $u.\text{mis}$, which is a maximal independent set of the submatroid $M_u = (S_u, I_u)$ where I_u is the restriction of I to the subsets of S_u. The size of T becomes $O(n \cdot m)$, where m denotes the size of a maximum independent set of S. For the applications where the matroid information is not needed, the fields $u.\text{mis}$ will be always set to null (as if $M = (U, \emptyset)$), and, in this case, the size of T will be $O(n)$.

4 Dynamic maintenance of augmented cover trees

Let T be an augmented cover tree for the set S of n points. In this section, we show how to update T efficiently when a point p is added to or deleted from S.

4.1 Insertion

Let p be a new point to be inserted in T. The insertion of p is accomplished as follows. First, if p is very far from the root r, namely $\text{dist}(p, r) > 2^{\ell_{\max}}$, then both $\ell_{\max}$ and r.level are increased to $\lfloor \log_2 \text{dist}(p, r) \rfloor$. Then, an explicit node u is created with u.point $= p$, u.weight $= 1$ and u.mis $= \{p\}$. In order to determine the level $\ell(p)$ where u must be placed, all cover sets Q_ℓ^p are computed, as described above, for every $\ell \in [\bar{\ell}, \ell_{\max}]$, where $\bar{\ell}$ is the largest index in $(-\infty, \ell_{\max}]$ such that $Q_{\bar{\ell}}^p = \emptyset$. Note that such empty cover set must exist and it is easy to see that $\bar{\ell} \geq \lceil \log_2 \text{dist}(p, S) \rceil - 2$. Then, u.level is set to the smallest index $\ell(p) \geq \bar{\ell}$ such that $\text{dist}(p, \text{pts}(Q_{\ell(p)}^p)) > 2^{\ell(p)}$ and $\text{dist}(p, \text{pts}(Q_{\ell(p)+1}^p)) \leq 2^{\ell(p)+1}$. At this point, an arbitrary node $v \in Q_{\ell(p)+1}^p$ such that $\text{dist}(p, v) \leq 2^{\ell(p)+1}$ (which must exists for sure) is determined. Let $q = v$.point. If v has no explicit self-child at level $\ell(p)$, a new node w with w.point $= q$, w.level $= \ell(p)$, and w.children $= v$.children, is created, and v.children is set to $\{u, w\}$. If instead such an explicit self-child w of v exists at level $\ell(p)$, then u is simply added as a further child of v. Finally, the path from the newly added node u to r is traversed, and for every ancestor v of u, v.weight is increased by 1 and p is added to the independent set v.mis, if v.mis $\cup \{p\}$ is still an independent set. Algorithm INSERT(p, T) in the appendix, provides the pseudocode for the above procedure.

Theorem 1. *Let T be an augmented cover tree for a set S of n points, with respect to a matroid $M = (U, I)$. The insertion algorithm described above yields an augmented cover tree for $S \cup \{p\}$ in time $O\left(12^D \log \Delta\right)$ where D is the doubling dimension of the metric space and Δ is the aspect ratio of S.*

Proof. It is easy to see that the insertion algorithm enforces, for every level ℓ, Properties 1, 2, 3 of the definition of cover tree, restricted to the nodes in Q_ℓ^p plus the new node created for p (for $\ell = \ell(p)$). These properties immediately extend to the entire level ℓ by virtue of Lemma 1. For what concerns the update of the .weight and .mis fields, correctness is trivially argued for the .weight fields, while Fact 2 ensures correctness of the updates of the .mis fields. can be argued as for the insertion algorithm. The complexity bounds follow by observing that there are $O(\log \Delta)$ levels in the explicit representation of T and that, at each such level ℓ, the algorithm performs work linear in the number of children of Q_ℓ^p, which are at most 12^D, by Lemma 2.

4.2 Deletion

Let $p \in S$ be the point to be removed. We assume that p is not the only point in S, otherwise the removal is trivial. The deletion of p is accomplished as follows. In the first, top-down phase, all cover sets Q_ℓ^p are computed, for every $\ell \in [\bar{\ell}, \ell_{\max}]$, where $\bar{\ell} \leq \ell_{\max}$ is the level of the leaf node corresponding to p in the explicit tree. Also, a list $R_{\bar{\ell}}$ of explicit nodes at level $\bar{\ell}$ to be relocated is created and initialized to the empty list. In the second, bottom-up phase, the following operations are performed iteratively, for every $\ell = \bar{\ell}, \bar{\ell} + 1, \ldots, \ell_{\max} - 1$ (the case $\ell = \ell_{\max}$ will be treated separately).

– If Q_ℓ^p contains a node u with u.point $= p$ and u.level $= \ell$, the following additional operations are performed. Let v be the parent of u, and observe that all children of v must also be explicit nodes at level ℓ. If u is the self child of v (i.e., v.point $= u$.point $= p$) u is removed from T, u's siblings are detached from v and added to R_ℓ (v will be later removed at iteration v.level). If instead u is not the self child of v, but it is the only child of v besides the self-child, u is removed from T and v and its self-child are merged together in the explicit tree.

– An empty list $R_{\ell+1}$ is created. Then, R_ℓ is scanned sequentially, and, for every $w \in R_\ell$, a node $w' \in Q_{\ell+1}^p \cup R_{\ell+1}$ is searched for such that $d(w, w') \leq 2^{\ell+1}$. If no such node exists, then w is added to $R_{\ell+1}$, raising w.level to $\ell+1$. Otherwise, if w' is found, it becomes parent of w as follows. If w' is internal and its children are at level ℓ, w is simply added as a further child. Otherwise, a new explicit node z, is created with z.point $= w'$.point, z.level $= \ell$, and z.children $= w'$.children, and w'.children is set to $\{z, w\}$.

– For all nodes in $w \in Q_{\ell+1}^p \cup R_{\ell+1}$ with w.level $= \ell+1$, their .weight and .mis fields are updated based on the values of the corresponding fields of their children. The update of the .weight fields is straightforward, while, based on Fact 2, the update of the .mis field of one such node w can be accomplished by computing a maximal independent set in the union of the elements of the .mis fields of w's children.

Once the above operations are performed up to $\ell = \ell_{\max} - 1$, the following cases arise for level $\ell_{\max}$. Consider first the case $p \neq r$.point. If $R_{\ell_{\max}} = \emptyset$, we simply update $\ell_{\max}$ and r.level to 1 plus the level of the children of r; otherwise ($R_{\ell_{\max}} \neq \emptyset$), a new root r_{new} is created with r_{new}.point $= r$.point, r_{new}.level $= \ell_{\max} + 1$, and r_{new}.children $= \{r\} \cup R_{\ell_{\max}}$. Also, $\ell_{\max}$ is incremented by 1. Instead, in case $p = r$.point, it is easy to see that $R_{\ell_{\max}}$ cannot be empty, since it must contain for sure the children of r. If $R_{\ell_{\max}}$ contains only one node, say v, then v becomes the new root, and we update, $\ell_{\max}$ and v.level to 1 plus the level of the children of v; otherwise ($|R_{\ell_{\max}}| > 1$), we select an arbitrary $v \in R_{\ell_{\max}}$, a new root r_{new} is created with r_{new}.point $= v$.point, r_{new}.level $= \ell_{\max} + 1$, and r_{new}.children $= R_{\ell_{\max}}$. Also, $\ell_{\max}$ is incremented by 1. Finally, whenever level $\ell_{\max}$ is incremented by 1 and, consequently, a new root node is created, for this node the .weight and .mis fields are updated based on the values of the corresponding fields of their children, as described above. Algorithm DELETE(p, T) in the appendix, provides the pseudocode for the above procedure.

Theorem 2. *Let T be an augmented cover tree for a set S of n points, with respect to a matroid $M = (U, I)$. The deletion algorithm described above yields an augmented cover tree for $S - \{p\}$ in time $O\left((16^D + 12^D \text{rank}(M)) \log \Delta\right)$ where D is the doubling dimension of the metric space and Δ is the aspect ratio of S.*

Proof. It is easy to see that the bottom-up phase of the deletion algorithm enforces, for every level ℓ, Properties 1, 2, 3 of the definition of cover tree, restricted to the nodes in $Q_\ell^p \cup R_\ell$. These properties immediately extend to the

entire level ℓ by virtue of Lemma 1. Finally, correctness of the update of the .weight and .mis fields can be argued as for the insertion algorithm. For what concerns the complexity bound, first observe that for every level ℓ, the nodes in $Q_\ell^p \cup R_\ell$ represent the new coversets $\hat{Q}_\ell^p$ associated to p after its deletion from T, thus Lemma 2 holds. As a consequence, the work needed to process the nodes in R_ℓ is $\Theta(|R_\ell|(|Q_{\ell+1}^p \cup R_{\ell+1}|) = O(16^D)$, while, by Fact 2, recreating the .weight and .mis fields for all nodes in $Q_{\ell+1}^p \cup R_{\ell+1}$ is upper bounded by the number of their children multiplied by $(1+\mathrm{rank}(M))$. The final bound follows by observing that there are $O(\log \Delta)$ levels in the explicit representation of T.

5 Extracting solutions from the augmented cover tree

We show how to employ the augmented cover tree presented in the previous section to extract accurate solutions to the various k-center related and diversity maximization problems introduced in Section 2. For all these problems, we rely on the extraction from the cover tree of a small (ϵ, k)-coreset (see Definition 1), for suitable values of ϵ and k.

Let T be an augmented cover tree for a set S of n points from a metric space of doubling dimension D. Given ϵ and k, an (ϵ, k)-coreset for S can be constructed as follows. Let $T_{\ell(k)}$ be the level of largest index (in the implicit representation of T) such that $|T_{\ell(k)}| \leq k$ and $|T_{\ell(k)-1}| > k$. Then, define

$$\ell^*(\epsilon, k) = \max\{\ell_{\min}, \ell(k) - \lceil \log_2(8/\epsilon) \rceil\}. \tag{3}$$

(For ease of notation, in what follows we shorthand $\ell^*(\epsilon, k)$ with ℓ^* whenever the parameters are clear from the context.) We have:

Lemma 3. *The set of points* $\mathrm{pts}(T_{\ell^*})$ *is an* (ϵ, k)*-coreset for* S *of size at most* $k(64/\epsilon)^D$ *and can be constructed in time* $O\left(k((64/\epsilon)^D + \log \Delta)\right)$.

Proof. We first show that $\mathrm{pts}(T_{\ell^*})$ is an (ϵ, k)-coreset for S. If $\ell^* = \ell_{\min}$, we have $\mathrm{pts}(T_{\ell^*}) = S$, so the property is trivially true. Suppose that $\ell^* = \ell(k) - \lceil \log_2(8/\epsilon) \rceil > \ell_{\min}$ and consider an arbitrary point $p \in S$. There must exist some node $u \in T_{\ell_{\min}}$ such that $p = u.\mathrm{point}$. Let v be the ancestor of u in T_{ℓ^*}. By the properties of the cover tree we know that $\mathrm{dist}(v.\mathrm{point}, p) \leq 2^{\ell^*+1} \leq (\epsilon/4)2^{\ell(k)}$. Also, all pairwise distances among points of $\mathrm{pts}(T_{\ell(k)-1})$ are greater than $2^{\ell(k)-1}$. However, since $|T_{\ell(k)-1}| \geq k+1$, there must be two points $q, q' \in \mathrm{pts}(T_{\ell(k)-1})$ which belong to the same cluster in the optimal k-center clustering of S. Therefore, $2^{\ell(k)-1} < \mathrm{dist}(q, q') \leq 2r_k^*(S)$, which implies that $2^{\ell(k)} \leq 4r_k^*(S)$. Putting it all together, we get that for any $p \in S$, $\mathrm{dist}(p, \mathrm{pts}(T_{\ell^*})) \leq \epsilon r_k^*(S)$. Let us now bound the size of T_{ℓ^*}. By construction $|T_{\ell(k)}| \leq k$, and we observe that T_{ℓ^*} can be partitioned into $|T_{\ell(k)}|$ subsets $T_{\ell^*}^u$, for every $u \in T_{\ell(k)}$, where $T_{\ell^*}^u$ is the set of descendants of u in T_{ℓ^*}. The definition of cover tree implies that for each $u \in T_{\ell(k)}$ and $v \in T_{\ell^*}^u$, $\mathrm{dist}(u, v) \leq 2^{\ell(k)+1}$. Moreover, since the pairwise distance between points of $T_{\ell^*}^u$ is greater than 2^{ℓ^*}, by applying Fact 3 with $Y = T_{\ell^*}^u$, $R = 2^{\ell(k)+1}$ and $r = 2^{\ell^*}$, we obtain that

$$|T_{\ell^*}^u| \leq 2^{(\lceil \log_2(8/\epsilon) \rceil + 2) \cdot D} \leq (64/\epsilon)^D,$$

and the bound on $|T_{\ell^*}|$ follows. T_{ℓ^*} can be constructed on the explicit tree through a simple level-by-level visit up to level ℓ^*, which can be easily determined from $\ell(k)$ and the fact that $\ell_{\min}$ is the largest level ℓ for which all nodes in T_ℓ only have the self-child. The construction time is linear in

$$\sum_{\ell=\ell^*}^{\ell_{\max}} |T_\ell| = \sum_{\ell=\ell^*}^{\ell(k)-1} |T_\ell| + \sum_{\ell=\ell(k)}^{\ell_{\max}} |T_\ell|$$

The second summation is clearly upper bounded by $k \log \Delta$, while, using again Fact 3 it is easy to argue that $|T_\ell| \leq 2^{(\ell(k)+2-\ell)\cdot D}$, for every $\ell^* \leq \ell \leq \ell(k) - 1$, whence the first sum is $O\left((64/\epsilon)^D\right)$. The lemma follows.

Remark. Consider an arbitrary node $u \in T_{\ell^*}$ and recall that, in the augmented version of the cover tree, the fields u.weight and u.mis contain, respectively, the size and a maximal independent set of S_u, where S_u is the subset of points of S associated with the descendants of u in T. The proof of the above lemma shows that for any $p \in S_u$ (thus, for any p accounted for by u.weight and any p of the maximal independent set) $\operatorname{dist}(p, u.\text{point}) \leq \epsilon r_k^*(S)$.

5.1 Solving k-center

Suppose that an (augmented) cover tree T for S is available. We can compute an $O\left(2 + O\left(\epsilon\right)\right)$-approximate solution C to k-center on S as follows. First, we extract the coreset $Q = \operatorname{pts}(T_{\ell^*})$, where $\ell^* = \ell^*(\epsilon, k)$ is the index defined in Equation 3, and then run a sequential algorithm for k-center on Q. To do so, we could use Gonzalez's 2-approximation algorithm. However, this would contribute an $O\left(k|Q|\right)$ term to the running time, which, based on the size bound stated in Lemma 3, would yield a quadratic dependency on k. The asymptotic dependency on k can be lowered by computing the solution through an adaptation of the techniques presented in [19], as explained below. Let us define a generalization of the cover tree data structure, dubbed (α, β)-*cover tree*, where the three properties that each level ℓ must satisfy are rephrased as follows:

1. $\operatorname{pts}(T_\ell) \subseteq \operatorname{pts}(T_{\ell-1})$;
2. for each $u \in T_\ell$, $\operatorname{dist}(u, u.\text{parent}) \leq \beta \cdot \alpha^{\ell+1}$;
3. for all $u, v \in T_\ell$, $\operatorname{dist}(u, v) > \beta \cdot \alpha^\ell$.

By adapting the insertion procedure of Section 4.1 and its analysis, it is easily seen that the insertion of a new point in the data structure can be supported in $O\left(12^D \cdot \log_\alpha \Delta\right)$ time. For a given integer parameter m, we construct m generalized cover trees for Q, namely an $(\alpha, \alpha^{p/m})$-cover tree $T^{(p)}$ for every $1 \leq p \leq m$. Each cover tree is constructed by inserting one point of Q at a time. Let ℓ_p be the smallest index such that level $T_{\ell_p}^{(p)}$ in $T^{(p)}$ has at most k nodes. The returned solution C is the set $\operatorname{pts}(T_{\ell_p}^{(p)})$ such that $T_{\ell_p}^{(p)}$ minimizes $\alpha^{\ell_p+p/m}$. By selecting $\alpha = 2/\epsilon$ and $m = O\left(\epsilon^{-1} \ln \epsilon^{-1}\right)$, and by using the argument of [19], it can be shown that C is a $(2 + O\left(\epsilon\right))$-approximation for k-center on Q.

We have:

Theorem 3. *Given an augmented cover tree T for S, the above procedure returns a $(2 + O(\epsilon))$-approximation C to the k-center problem for S, and can be implemented in time $O\left((k/\epsilon)(768/\epsilon)^D \log \Delta\right)$.*

Proof. By Lemma 3, Q is an (ϵ, k)-coreset for S. Let $C^* = \{c_1, c_2, \ldots, c_k\}$ be an optimal solution for k-center on S. The coreset property of Q ensures that for each c_i there is a point $c_i' \in Q$ such that $d(c_i, c_i') \le \epsilon r_k^*(S)$. This implies that the set $C' = \{c_1', c_2', \ldots, c_k'\}$ is a solution to k-center on Q with $r_{C'}(Q) \le (1+\epsilon)r_k^*(S)$, hence $r_k^*(Q) \le (1+\epsilon)r_k^*(S)$. Suppose that the $(2+O(\epsilon))$-approximation algorithm outlined above is used in Phase 2 to compute the solution C on Q. Then, $r_C(Q) \le (2 + O(\epsilon))r_k^*(Q) \le (2 + O(\epsilon))(1 + \epsilon)r_k^*(S) = (2 + O(\epsilon))r_k^*(S)$. By the coreset property and the triangle inequality, it follows that $r_C(S) \le (2 + O(\epsilon))r_k^*$. For what concerns the running time, we have that the construction of the (ϵ, k)-coreset Q requires $O\left(k((64/\epsilon)^D + \log \Delta)\right)$ time (see Lemma 3), while the running time of Phase 2 is dominated by the construction of the $m = O\left(\epsilon^{-1} \ln \epsilon^{-1}\right)$ $(\alpha, \alpha^{p/m})$-cover trees $T^{(p)}$, for $1 \le p \le m$, by successive insertions of the elements of Q. As observed above, an insertion takes $O\left(12^D \log_\alpha \Delta\right)$ time, hence the cost for constructing each $T^{(p)}$ is $O\left(|Q|12^D \log_\alpha \Delta\right)$. Since $\alpha = 2/\epsilon$ and $|Q| \le k(64/\epsilon)^D$, the total cost is thus $O\left((k/\epsilon)(768/\epsilon)^D \log \Delta\right)$, which dominates over the cost of Phase 1.

The constants involved in the analysis are likely to provide very conservative upper bounds on the actual behavior of the data structure in practical scenarios. Moreover, for moderate values of k, the use of Gonzalez's algorithm in Phase 2 might prove a much more practical choice. We wish to remark that our algorithm improves upon the one proposed in [19] in several ways. First, the accuracy parameter ϵ can be chosen freely at query time, and does not influence the construction of the data structure. Second, our data structure requires linear space and can handle insertions and deletions at a lower asymptotic cost.

5.2 Solving k-center with z outliers

For the k-center problem with z outliers, an approach similar to the one adopted for k-center can be employed. Let T be an augmented cover tree for S. We can compute a $(3 + O(\epsilon))$-approximation to k-center with z outliers on S, by proceeding as follows. First, we extract the coreset $Q = \text{pts}(T_{\ell^*})$, where $\ell^* = \ell^*(k+z)$ is the index defined in Equation 3. Each point $q \in Q$ is associated to the weight $w_q = u.\text{weight}$, where $u \in T_{\ell^*}$ is such that $u.\text{point} = q$. Then, we extract the solution C from this weighted coreset Q using the techniques from [5], which are reviewed below.

By Lemma 3, Q is an $(\epsilon, k + z)$-coreset for S and, based on the remark made after Lemma 3, all points of S can be associated to the points of Q, such that, for every $q \in Q$, w_q points of S are associated to q (q is referred to as the *proxy* for these points) and they are all at distance at most $\epsilon r_{k+z}^*(S)$ from q. Also recall, from Equation 1, that $r_{k+z}^*(S) \le r_{k,z}^*(S)$. Suppose that algorithm OUTLIERSCLUSTER described in [5] is run on the weighted coreset

Q with parameters k, r, and ϵ, where r is a guess of the optimal radius. The analysis in [5] shows that the algorithm returns two subsets $X, Q' \subseteq Q$ such that

- $|X| \leq k$
- For every $p \in S$ whose proxy is in $Q - Q'$, $\text{dist}(p, X) \leq \epsilon r_{k,z}^*(S) + (3 + 4\epsilon)r$;
- if $r \geq r_{k,z}^*(S)$, then $\sum_{q \in Q'} w_q \leq z$.

Then, we can repeatedly run OUTLIERSCLUSTER for $r = 2^{\ell_{\max}}/(1 + \epsilon)^i$, for $i = 0, 1, \ldots$, stopping at the smallest guess r which returns a pair (X, Q') where Q' has aggregate weight at most z and returning $C = X$ as the final solution. We have:

Theorem 4. *Given an augmented cover tree T for S, the above procedure returns a $(3 + O(\epsilon))$-approximation C to the k-center problem with z outliers for S, and can be implemented in time $O\left((k + z)^2(64/\epsilon)^{2D}(1/\epsilon)\log \Delta\right)$.*

Proof. The bound on the approximation ratio immediately follows from the properties of OUTLIERSCLUSTER reviewed above. The running time is dominated by the repeated executions of OUTLIERSCLUSTER. Each execution of OUTLIERSCLUSTER can be performed in $O\left(|Q|^2 + k|Q|\right) = O\left((k + z)^2(64/\epsilon)^{2D}\right)$ time. The bound on the running time follows by observing that $2^{\ell_{\max}}/r_{k,z}^*(S) = O(\Delta)$, whence $O\left(\log_{1+\epsilon}\Delta\right) = O\left((1/\epsilon)\log \Delta\right)$ executions suffice.

5.3   Solving matroid center

Consider a matroid $M = (S, I)$ defined on a set S, and suppose that an augmented cover tree T for S w.r.t. M is available. We can compute an $O(3 + O(\epsilon))$-approximate solution C to the matroid center problem on M as follows. First we determine a coreset Q as the union of the independent sets associated with the nodes of level T_{ℓ^*} where $\ell^* = \ell^*(\epsilon, \text{rank}(M))$ is the index defined in Equation 3. Namely,

$$Q = \bigcup_{u \in T_{\ell^*}} u.\text{mis}.$$

Note that $\text{rank}(M)$ is easily obtained as the size of $r.\text{mis}$, where r is the root of T. Then, solution C is computed by running the 3-approximation algorithm by [14] on Q. We have:

Theorem 5. *Given an augmented cover tree T for S w.r.t. $M = (S, I)$, the above procedure returns a $(3 + O(\epsilon))$-approximation C to the matroid center problem on M, and can be implemented in time $O\left(\text{poly}(\text{rank}(M), (64/\epsilon)^D) + \text{rank}(M)\log \Delta\right)$.*

Proof. As remarked before, the nodes of T_{ℓ^*} induce a partition of S into subsets $\{S_u : u \in T_{\ell^*}\}$, where S_u is the subset of points of S associated with the descendants of u in T, and for each $q \in S_u$ we have $\text{dist}(p, u.\text{point}) \leq \epsilon r_{\text{rank}(M)}^*(S) \leq r^*(M)$. Consider an optimal solution $C^* = \{c_1, c_2, \ldots, c_{\text{rank}(M)}\}$ to the matroid center problem on M, and let $c_i \in S_{u_i}$, for some $u_i \in T_{\ell^*}$. We now show that

we can substitute each c_i with a distinct element of $u_i.\mathrm{mis} \subseteq Q$, so that the resulting set of substitutes is also a maximal independent. Inductively, suppose that we have substituted c_j with a point $c'_j \in u_j.\mathrm{mis}$, for every $1 \leq j < i - 1$, and that the set $C'(i-1) = \{c'_1, \ldots, c'_{i-1}, c_i, \ldots, c_{\mathrm{rank}(M)}\}$ is an independent set. By applying Fact 1 with $A = C'(i-1) - \{c_i\}$, $y = c_i$, $S' = S_{u_i}$, and $B = u_i.\mathrm{mis}$, we have that there exists a point $c'_i \in u_i.\mathrm{mis} \backslash C'(i-1)$ such that $C'(i) = (C'(i-1) \backslash \{c_i\}) \cup \{c'_i\}$ is an independent set. Let $C' = C'(\mathrm{rank}(M))$. Since c_i and c'_i belong to the same subset S_{u_i}, we have $d(c_i, c'_i) \leq 2\epsilon r^*(M)$, which immediately implies that $r_{C'}(Q) \leq (1 + 2\epsilon)r^*(M)$. Therefore, the solution C computed using the 3-approximation algorithm by [14] is such that $r_C(Q) \leq (3 + O(\epsilon))r^*(M)$. By the coreset property and the triangle inequality, it follows that $r_C(S) \leq (3 + O(\epsilon))r^*(M)$.

For what concerns the running time, we have that the determination of the level T_{ℓ^*} requires $O\left(\mathrm{rank}(M)((64/\epsilon)^D + \log \Delta)\right)$ time (see Lemma 3), and the size of the coreset Q is $O\left((\mathrm{rank}(M))^2(64/\epsilon)^D\right)$. The claimed bound follows since the algorithm by [14] runs in time polynomial in the input size.

5.4 Solving diversity maximization

In [8], the authors present a coreset-based approach to approximating the optimal solution of the diversity maximization problem on a pointset S, under all the diversity measures $\mathrm{div}(\cdot)$ listed in Table 1. Specifically, starting from an (ϵ, k)-coreset Q for k-center on S, the authors obtain the coreset $Q' = Q$ for the remote edge and the remote cycle variants of diversity maximization, while, for all the other variants, the coreset Q' is constructed by selecting, for each $p \in Q$, $\min\{k, |S_p|\}$ points from the subset S_p of a partition $\{S_p : p \in Q\}$ of S into disjoint subsets, where each S_p contains points $q \in S$ with $\mathrm{dist}(p, q) \leq \epsilon r_k^*$. It is shown in [8] that running an α approximation algorithm on Q' yields an $(\alpha + O(\epsilon))$-approximate solution for S. Observe that in all cases the coreset Q' can be easily constructed from an (augmented) cover tree T for S. Indeed, in the former, simple case, Q' is obtained as the set of points associated with the nodes of level T_{ℓ^*}, where $\ell^* = \ell^*(\epsilon, k)$ is the index defined in Equation 3. In the latter case, Q' can be obtained as follows. We need T to be an augmented cover tree w.r.t. to k-bounded cardinality matroid for S, denoted as $M_{k,S}$, whose independent sets are all subsets of S of at most k points. Then, we simply set $Q' = \cup_{u \in T_{\ell^*}} u.\mathrm{mis}$.

For each diversity variant in Table 1, let A_{div} be the polynomial-time approximation algorithm yielding the α_{div} approximation mentioned in the table, and let $t_{A_{\mathrm{div}}}(\cdot)$ denote its running time. We have:

Theorem 6. *Consider an cover tree T for S (augmented w.r.t. the k-bounded cardinality matroid $M_{k,S}$, when necessary). For each diversity variant in Table 1, running A_{div} on the coreset Q' extracted from T returns an $(\alpha_{\mathrm{div}} + O(\epsilon))$-approximate solution to the diversity maximization problem in time $O\left(t_{A_{\mathrm{div}}}(k(64/\epsilon)^D) + k \log \Delta\right)$ for the remote edge and cycle variants, and time $O\left(t_{A_{\mathrm{div}}}(k^2(64/\epsilon)^D) + k \log \Delta\right)$ for the other variants.*

Proof. The stated bounds are an immediate consequence of the above discussion and the observation that the construction of coreset Q' can be accomplished in $O\left(k((64/\epsilon)^D + \log \Delta)\right)$ time, for the remote edge and cycle variants, and in $O\left(k^2(64/\epsilon)^D + k \log \Delta)\right)$ time, for the other variants.

6   Conclusions

It is important to remark that for all problems treated in this paper, when S is large and both the spread Δ and the doubling dimension D of the metric are small, the dynamic maintenance of the augmented cover tree data structure and the extraction of solutions can be accomplished in time dramatically smaller than the time that would be required to compute solutions on the entire pointset from scratch.

Finally, the coreset-based approaches developed in [30] for robust matroid center, and in [6] for diversity maximization under matroid constraints, can be integrated with the approach presented in this paper, to yield dynamic maintenance for these more general versions of the problems, with similar accuracy-performance tradeoffs.

References

1. Abbassi, Z., Mirrokni, V.S., Thakur, M.: Diversity maximization under matroid constraints. In: Proc. ACM KDD. pp. 32–40 (2013)
2. Bateni, M., Esfandiari, H., Fichtenberger, H., Henzinger, M., Jayaram, R., Mirrokni, V., Wiese, A.: Optimal fully dynamic k-center clustering for adaptive and oblivious adversaries. In: Proc. ACM-SIAM SODA. pp. 2677–2727 (2023)
3. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proc. ICML. pp. 97–104 (2006)
4. Borassi, M., Epasto, A., Lattanzi, A., Vassilvitskii, S., Zadimoghaddam, M.: Better sliding window algorithms to maximize subadditive and diversity objectives. In: Proc. ACM PODS. pp. 254–268 (2019)
5. Ceccarello, M., Pietracaprina, A., Pucci, G.: Solving k-center clustering (with outliers) in mapreduce and stre aming, almost as accurately as sequentially. PVLDB **12**(7), 766–778 (2019)
6. Ceccarello, M., Pietracaprina, A., Pucci, G.: A general coreset-based approach to diversity maximization under matroid constraints. ACM Trans. Knowl. Discov. Data **14**(5), 60:1–60:27 (2020)
7. Ceccarello, M., Pietracaprina, A., Pucci, G., Upfal, E.: A practical parallel algorithm for diameter approximation of massive weighted graphs. In: Proc. IEEE IPDPS. pp. 12–21 (2016)
8. Ceccarello, M., Pietracaprina, A., Pucci, G., Upfal, E.: Mapreduce and streaming algorithms for diversity maximization in metric spaces of bounded doubling dimension. Proc. VLDB Endow. **10**(5), 469–480 (2017)
9. Chakrabarty, D., Goyal, P., Krishnaswamy, R.: The non-uniform k-center problem. ACM Trans. on Algorithms **16**(4), 46:1–46:19 (2020)
10. Chan, T.H., Guerqin, A., Sozio, M.: Fully dynamic k-center clustering. In: Proc. WWW. pp. 579–587 (2018)

11. Chan, T.H., Lattanzi, S., Sozio, M., Wang, B.: Fully dynamic k-center clustering with outliers. In: Proc. COCOON. pp. 150–161 (2023)
12. Charikar, M., Chekuri, C., Feder, T., Motwani, R.: Incremental clustering and dynamic information retrieval. In: Proc. ACM STOC. pp. 626–635 (1997)
13. Charikar, M., Khuller, S., Mount, D., Narasimhan, G.: Algorithms for Facility Location Problems with Outliers. In: Proc. ACM-SIAM SODA. pp. 642–651 (2001)
14. Chen, D., J.Li, H.L., Wang, H.: Matroid and knapsack center problems. Algorithmica **75**(1), 27–52 (2016)
15. Chiplunkar, A., Kale, S., Ramamoorthy, S.: How to solve fair k-center in massive data models. In: Proc. ICML. pp. 1877–1886 (2020)
16. Cohen-Addad, V., Schwiegelshohn, C., Sohler, C.: Diameter and k-center in sliding windows. In: Proc. ICALP (2016)
17. Elkin, Y., Kurlin, V.: Counterexamples expose gaps in the proof of time complexity for cover trees introduced in 2006. In: Proc. TopoInVis. pp. 9–17. IEEE (2022)
18. Gonzalez, T.: Clustering to Minimize the Maximum Intercluster Distance. Theoretical Computer Science **38**, 293–306 (1985)
19. Goranci, G., Henzinger, M., Leniowski, D., Schulz, C., Svozil, A.: Fully dynamic k-center clustering in low dimensional metrics. In: Proc. ALENEX 2021. pp. 143–153 (2021)
20. Gottlieb, L.A., Kontorovich, A., Krauthgamer, R.: Efficient classification for metric data. IEEE Trans. Information Theory **60**(9), 5750–5759 (2014)
21. Gupta, A., Krauthgamer, R., Lee, J.R.: Bounded geometries, fractals, and low-distortion embeddings. In: Proc. IEEE FOCS. pp. 534–543 (2003)
22. Harris, D., Pensyl, T., Srinivasan, A., Trinh, K.: A lottery model for center-type problems with outliers. ACM Trans. on Algorithms **15**(3), 36:1–36:25 (2019)
23. Kale, S.: Small space stream summary for matroid center. In: Proc. APPROX/RANDOM. pp. 20:1–20:22 (2019)
24. Kleindessner, M., Awasthi, P., Morgenstern, J.: Fair k-center clustering for data summarization. In: Proc. ICML. pp. 3448–3457 (2019)
25. Krauthgamer, R., Lee, J.: Navigating nets: simple algorithms for proximity search. In: Proc. ACM-SIAM SODA. pp. 798–807 (2004)
26. Leskovec, J., Rajaraman, A., Ullman, J.: Mining of Sassive Data Sets. Cambridge University Press (2014)
27. Oxley, J.: Matroid Theory. Oxford University Press (2006)
28. Pellizzoni, P., Pietracaprina, A., Pucci, G.: Dimensionality-adaptive k-center in sliding windows. In: Proc. DSAA. pp. 197–206 (2020)
29. Pellizzoni, P., Pietracaprina, A., Pucci, G.: k-center clustering with outliers in sliding windows. Algorithms **15**(2), 52 (2022)
30. Pietracaprina, A., Pucci, G., Soldà, F.: Coreset-based strategies for robust center-type problems. arXiv **2002.07463** (2020)

Appendix

Pseudocode for the insert procedure

Algorithm 1 details the pseudocode of the insertion procedure. It takes in input the point p to be inserted and the cover tree, represented by its root r.

Algorithm 1: INSERT(Point p, Root r)

---

1 **if** $\text{dist}(p, r) > 2^{\ell_{\max}}$ **then**
2 $\quad$ $\ell_{\max} = \lfloor \log_2 \text{dist}(p, r) \rfloor$
3 $\quad$ $r.\text{level} = \ell_{\max}$
4 let $\ell = \ell_{\max}$
5 $Q_\ell = \{r\}$
6 **while** $Q_\ell \neq \emptyset$ **do**
7 $\quad$ $Q_{\ell-1} = \emptyset$
8 $\quad$ **for** $q \in Q_\ell$ **do**
9 $\quad\quad$ **if** $q.\text{children} == \emptyset$ OR $q.\text{children}[0].\text{level} \; != \ell - 1$ **then**
10 $\quad\quad\quad$ **if** $\text{dist}(q, p) \leq 2^\ell$ **then**
11 $\quad\quad\quad\quad$ $Q_{\ell-1} = Q_{\ell-1} \cup \{q\}$
12 $\quad\quad$ **else**
13 $\quad\quad\quad$ $Q_{\ell-1} = Q_{\ell-1} \cup \{q' \in q.\text{children s.t dist}(p, q') \leq 2^\ell\}$
14 $\quad$ $\ell = \ell - 1$
15 **while** $\text{dist}(p, Q_{\ell+1}) > 2^{\ell+1}$ **do**
16 $\quad$ $\ell = \ell + 1$
17 let $v \in Q_{\ell+1}$ be s.t. $\text{dist}(p, v) \leq 2^{\ell+1}$
18 $u = $ new node with $u.\text{point} = p$, $u.\text{level} = \ell$
19 **if** $v.\text{children} == \emptyset$ OR $v.\text{children}[0].\text{level} \; != \ell$ **then**
20 $\quad$ $w = $ new node with $w.\text{point} = v.\text{point}$, $w.\text{level} = \ell$, $w.\text{children} = v.\text{children}$
21 $\quad$ $v.\text{children} = \{u, w\}$
22 **else**
23 $\quad$ $v.\text{children} = v.\text{children} \cup \{u\}$
24 $t = u$
25 **while** $t \neq \text{null}$ **do**
26 $\quad$ $t.\text{weight} = t.\text{weight} + 1$
27 $\quad$ add p to $t.\text{mis}$ if it remains an independent set
28 $\quad$ $t = r.\text{parent}$

Pseudocode for the delete procedure

Algorithm 2 details the pseudocode of the deletion procedure. It takes in input the point p to be deleted and the cover tree, represented by its root r.

Algorithm 2: DELETE(Point p, Root r)

1   let $\ell = \ell_{\max}$
2   $Q_\ell = \{r\}$
3   **while** true **do**
4     $Q_{\ell-1} = \emptyset$
5   **for** $q \in Q_\ell$ **do**
6   **if** q.point $== p$ AND q.children $== \emptyset$ **then**
7   **break while**
8   **if** q.children $== \emptyset$ OR q.children[0].level $!= \ell - 1$ **then**
9   **if** $\mathrm{dist}(q, p) \leq 2^\ell$ **then**
10           $Q_{\ell-1} = Q_{\ell-1} \cup \{q\}$
11   **else**
12         $Q_{\ell-1} = Q_{\ell-1} \cup \{q' \in q.\text{children s.t } \mathrm{dist}(p, q') \leq 2^\ell\}$
13     $\ell = \ell - 1$
14   $R_\ell = \emptyset$
15   **while** $\ell \leq \ell_{\max} - 1$ **do**
16   **if** $\exists u \in Q_\ell$ s.t. u.point $== p$ and u.level $== \ell$ **then**
17       $v = u$.parent
18   delete u from v.children
19   **if** v.point $== p$ **then**
20         $R_\ell = R_\ell \cup v$.children
21   **else if** $|v$.children$| == 1$ **then**
22   v.children $= v$.children[0].children
23   delete v.children[0]
24   let $R_{\ell+1} = \emptyset$
25   **for** $w \in R_\ell$ **do**
26   **if** $\exists w' \in Q_{\ell+1} \cup R_{\ell+1}$ s.t. $\mathrm{dist}(w, w') \leq 2^{\ell+1}$ **then**
27   **if** w'.children $\neq \emptyset$ AND w'.children[0].level $== \ell$ **then**
28   w'.children $= w'$.children $\cup \{w\}$
29   **else**
30           $z = $ new node with z.point $= w'$.point, z.level $= \ell$,
31   z.children $= w'$.children
32   w'.children $= \{z, w\}$
33   **else**
34   w.level $= \ell + 1$
35         $R_{\ell+1} = R_{\ell+1} \cup \{w\}$
36   **for** $w \in Q_{\ell+1} \cup R_{\ell+1}$ **do**
37   update w.weight and w.mis
38     $\ell = \ell + 1$
39   **if** $p \neq r$.point **then**
40   **if** $R_\ell == \emptyset$ **then**
41       $\ell_{\max} = r$.children[0].level $+ 1$
42   r.level $= \ell_{\max}$
43   **else**
44       $r_{\text{new}} = $ new root with r_{new}.point $= r$.point, r_{new}.level $= \ell + 1$,
45   r_{new}.children $= \{r\} \cup R_\ell$
46   **else**
47   **if** $|R_\ell| == 1$ **then**
48   let $v = R_\ell[0]$ be the new root
49       $\ell_{\max} = r$.children[0].level $+1$
50   v.level $= \ell_{\max}$
51   update v.weight and v.mis
52   **else**
53   let $v \in R_\ell$
54       $r_{\text{new}} = $ new root with r_{new}.point $= v$.point, r_{new}.level $= \ell + 1$, r_{new}.children $= R_\ell$
55       $\ell_{\max} = \ell + 1$
56   update r_{new}.weight and r_{new}.mis