

A General Coreset-Based Approach to Diversity Maximization under Matroid Constraints

MATTEO CECCARELLO, Free University of Bozen

ANDREA PIETRACAPRINA and GEPPINO PUCCI, University of Padova

Diversity maximization is a fundamental problem in web search and data mining. For a given dataset S of n elements, the problem requires to determine a subset of S containing $k \ll n$ "representatives" which minimize some diversity function expressed in terms of pairwise distances, where distance models dissimilarity. An important variant of the problem prescribes that the solution satisfy an additional orthogonal requirement, which can be specified as a matroid constraint (i.e., a feasible solution must be an independent set of size k of a given matroid). While unconstrained diversity maximization admits efficient coreset-based strategies for several diversity functions, known approaches dealing with the additional matroid constraint apply only to one diversity function (sum of distances), and are based on an expensive, inherently sequential, local search over the entire input dataset. We devise the first coreset-based algorithms for diversity maximization under matroid constraints for various diversity functions, together with efficient sequential, MapReduce and Streaming implementations. Technically, our algorithms rely on the construction of a small coreset, that is, a subset of S containing a feasible solution which is no more than a factor $1 - \epsilon$ away from the optimal solution for S . While our algorithms are fully general, for the partition and transversal matroids, if ϵ is a constant in $(0, 1)$ and S has bounded doubling dimension, the coreset size is independent of n and it is small enough to afford the execution of a slow sequential algorithm to extract a final, accurate, solution in reasonable time. Extensive experiments show that our algorithms are accurate, fast and scalable, and therefore they are capable of dealing with the large input instances typical of the big data scenario.

CCS Concepts: • **Theory of computation** → **Facility location and clustering; Streaming, sublinear and near linear time algorithms; MapReduce algorithms.**

Additional Key Words and Phrases: Diversity Maximization, Matroids, Coresets, MapReduce, Streaming, Doubling Spaces, Approximation Algorithms

1 INTRODUCTION

In many application domains, data analysis often requires the extraction of a succinct and significant summary of a large dataset which may take the form of a small subset of elements as diverse as possible from one another. The summary can be either presented to the user or employed as input for further processing [1, 25, 29, 30]. More specifically, given a dataset S of points in a metric space and a constant k , *diversity maximization* requires to determine a subset of k points of S maximizing some diversity objective function defined in terms of the distances between the points.

There are several ways of characterizing the diversity function. In general, the diversity of a set of k points can be captured by a specific graph-theoretic measure defined on the points, which are seen as the nodes of a k -clique where each edge is weighted with the distance between its endpoints [13]. The diversity functions considered in this paper are defined in Table 1. The maximization problems under these functions are all known to be NP-hard [13] and the development of efficient approximation algorithms has attracted a lot of interest in the recent literature (see [10] and references therein).

An important variant of diversity maximization requires that the k points to be returned satisfy some additional orthogonal constraint such as, for example, covering a variety of pre-specified categories attached to the data. This

Authors' addresses: Matteo Ceccarello, mceccarello@unibz.it, Free University of Bozen, Bolzano, Italy; Andrea Pietracaprina, andrea.pietracaprina@unipd.it; Geppino Pucci, geppino.pucci@unipd.it, University of Padova, Department of Information Engineering, Padova, Italy.

variant has been recently investigated under the name *Diversity Maximization under Matroid Constraint (DMMC)* where the subset of k points to be returned is required to be an independent set of a given matroid [1, 8, 12]. (A more formal definition of the problem is given in Section 2.) As a concrete example, suppose that S is a set of Wikipedia pages, each associated with one or more topics. A solution of the DMMC problem identifies a subset of pages that are most diverse in terms of some pre-specified distance metric (e.g., cosine distance [24]) but also “well spread” among the topics. This latter property can be suitably controlled by imposing a partition or transversal matroid constraint, depending on whether topics overlap or not.

In this paper, we contribute to this line of work and present novel algorithms for diversity maximization under matroid constraint for the diversity functions in Table 1, both for the traditional sequential setting and for big-data oriented computation frameworks such as MapReduce [16] and Streaming [20].

1.1 Related work

Unconstrained diversity maximization has been studied for over two decades within the realm of facility location (see [13] for an account of early results). In recent years, several works have devised efficient algorithms for various diversity maximization problems in different computational frameworks. Specifically, in [11] PTAS’s are devised in the sequential setting for metric spaces of constant doubling dimension (a notion that will be formalized in Subsection 2.3). In [4, 10, 17, 21], MapReduce and/or Streaming algorithms are proposed for several diversity measures, which are based on confining the expensive computations required by standard sequential algorithms to small subsets of the input (*coresets*), cleverly extracted so to contain high-quality global solutions. For general metric spaces, the algorithms in [4, 17, 21], require sublinear working memory per processor at the expense of a constant worsening in the approximation ratio with respect to the best ratio attained by sequential algorithms, while, for metric spaces of constant doubling dimension, the algorithms in [10] retain sublinear space but feature approximation ratios that can be made arbitrarily close to the best sequential ones. Finally, unconstrained diversity maximization is also studied in [7] in the sliding-window framework.

The literature on diversity maximization under matroid constraints is much thinner. Unlike the unrestricted case, existing approaches only target the sum-DMMC variant (see Table 1) in the sequential setting, and are based on expensive local search strategies over the entire input S . Specifically, both [8] and [1] present $(1/2)$ -approximation algorithms for sum-DMMC whose running times are at least quadratic in the input size, hence impractical for large inputs. In fact, these algorithms guarantee polynomial time only at the expense of a slightly worse approximation ratio $1/2 - \gamma$, for any fixed $\gamma > 0$. In the preliminary conference version of our work [9], sequential, MapReduce and Streaming algorithms for the sum-DMMC variant have been presented, which achieve the same approximation quality as [1, 8]. For metric spaces of constant doubling dimension, the sequential algorithm is much faster than its competitors, while the MapReduce/Streaming algorithms enable the processing of massive datasets in 2 rounds/1 pass. The improvements of this present work over [9] are discussed in more detail at the end of the next subsection.

Recently, an extension of the above local search approaches for sum-DMMC to non-metric spaces with negative-type distances has been proposed in [12]. To the best of our knowledge, no polynomial-time algorithms featuring nontrivial approximation guarantees are known for the other DMMC variants listed in Table 1.

Finally, it has been proved that, under the widely accepted *planted clique hypothesis*, it is hard to attain an approximation ratio larger than $1/2$ for sum-DMMC [5, 8] (in fact, this inapproximability result holds also for the unconstrained case).

1.2 Our contribution

In this paper, we present coreset-based strategies which can be employed to provide good approximations to all DMMC variants listed in Table 1, and which are amenable to efficient implementations in the sequential, MapReduce and Streaming settings. For all variants, the coreset constructions revolve around the same key idea of clustering the input dataset into subsets of close-by points, and then selecting suitable representatives from each subset, depending on the matroid type. The essence of this idea was pioneered in previous work on unconstrained diversity maximization [4, 10, 21]. More specifically, for any DMMC instance and every chosen $\varepsilon \in (0, 1)$, our approach builds a $(1 - \varepsilon)$ -coreset, that is, a coreset which contains a feasible solution to the instance whose diversity is within a factor $(1 - \varepsilon)$ from the optimal diversity. The coreset size is analyzed in terms of the matroid type, the size k of the solution, and the doubling dimension D of the input dataset S (see Subsection 2.3 for a formal definition of doubling dimension). For constant D , our coreset constructions can be implemented using work linear in n and polynomial in k and $1/\varepsilon$ in all three computational settings. It is important to remark that while k and ε are given in input together with the dataset S , the value D (hard to estimate in practice) is used in only in the analysis and needs not be explicitly provided to the algorithms. The constructions can be accomplished in MapReduce in one round with sublinear memory, and in Streaming in a single pass with working memory proportional to the (small) coreset size. While our coreset constructions are fully general, for the important cases of the partition and transversal matroids and of datasets with constant D , the resulting coreset size becomes independent of $n = |S|$. In the MapReduce setting, the coreset size also depends on the degree of available parallelism ℓ , which can in turn be a function of n . However, a coreset size independent of n can always be achieved regardless of the value of ℓ in an extra round, by performing a second (sequential) coreset construction on the first coreset.

In all three settings, once a $(1 - \varepsilon)$ -coreset T is computed for S , the final solution can be obtained by running on T a sequential algorithm for the DMMC variant under consideration. For sum-DMMC, running the algorithm of [1] on T yields a $(1/2 - O(\varepsilon))$ -approximation, with total work (including the coreset construction phase) which is linear in n and, for constant D , polynomial in k and $1/\varepsilon$. For the other DMMC variants of Table 1, since no nontrivial polynomial-time approximations are known, we resort to running an (exact) exhaustive search for the best solution on T . This approach yields a $(1 - \varepsilon)$ -approximation, with total work which is linear in n and, for constant D , polynomial in $1/\varepsilon$ but exponential in k . For small values of k , a range of definite interest for real applications, ours are the first feasible algorithms providing provably accurate solutions for these DMMC variants. Finally, we remark that our MapReduce and Streaming algorithms provide the first practically viable approaches to the solutions of all DMMC variants in the big data scenario.

Our theoretical results are complemented with extensive experiments on real-world datasets. For concreteness, we focus on the sum-DMMC variant, the only variant for which there is a known sequential competitor [1]. In the sequential and Streaming settings, the experiments provide clear evidence that the accuracy scales with the coreset size, in the sense that larger coreset sizes afford solutions of higher quality. Moreover, for a given target accuracy, both our sequential and Streaming algorithms run up to two orders of magnitude faster than the pure local-search of [1]. In the MapReduce setting, the experiments show scalability of performance with respect to the available parallelism. In essence, the experiments confirm that, on sufficiently large instances, the time for the extraction of the final solution,

now confined to a small coreset rather than the entire input, becomes negligible with respect to overall running time, while the running time is dominated by the highly-scalable, linear-work coreset construction.

Novelty with respect to conference version. The novel results of this work over those present in the preliminary conference version [9] are the following. (a) The results have been generalized to several diversity functions, through the introduction of the notion of *average fairness* and the derivation of its relation to the diameter of the dataset (Lemma 1). As a consequence of this generalization, the current paper presents the first feasible algorithms in the literature providing provably accurate solutions for the DMMC variants associated to these functions. (b) The streaming implementation of the coreset construction is novel and simpler. More importantly, unlike to the one presented in [9], it is oblivious to the doubling dimension D of the dataset. (c) An experimental analysis of the streaming algorithm and an extensive comparison between all of our algorithms have been added.

1.3 Organization of the paper

The rest of the paper is structured as follows. A formal definition of the problem and some key concepts and notations are given in Section 2. The coresets constructions are described in Section 3, while their implementations in the various settings and the resulting DMMC algorithms are presented in Section 4. The experimental results are reported in Section 5. Section 6 closes the paper with some final remarks and open problems.

2 PRELIMINARIES

2.1 Matroids

Let $S = \{s_1, s_2, \dots, s_n\}$ be a set from a metric space with distance function $d(\cdot, \cdot)$. Recall that d is nonnegative, symmetric, equal to 0 only on pairs of identical elements, and obeys the triangle inequality. A *matroid* [26] based on S is a pair $\mathcal{M} = (S, \mathcal{I}(S))$, where $\mathcal{I}(S)$ 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 $A \in \mathcal{I}(S)$ and $B \in \mathcal{I}(S)$, and $|A| > |B|$, then there exist $x \in A \setminus B$ such that $B \cup \{x\} \in \mathcal{I}(S)$ (*augmentation property*). An independent set is maximal if it is not properly contained in another independent set. A basic property of a matroid \mathcal{M} is that all of its maximal independent sets have the same size, which is called the *rank* of the matroid and is denoted by $\text{rank}(\mathcal{M})$. In this paper we concentrate on two well-known types of matroid, namely, *partition matroids* and *transversal matroids*, which are defined as follows.

DEFINITION 1 (PARTITION MATROID). Consider a partition of S into h disjoint subsets A_1, A_2, \dots, A_h , and let $k_i \leq |A_i|$ be a nonnegative integer, for $1 \leq i \leq h$. Define $\mathcal{I}(S)$ as the family of subsets $X \subseteq S$ with $|X \cap A_i| \leq k_i$, for $1 \leq i \leq h$. Then, $\mathcal{M} = (S, \mathcal{I}(S))$ is a partition matroid based on S .

DEFINITION 2 (TRANSVERSAL MATROID). Consider a covering family $\mathcal{A} = \{A_1, \dots, A_h\}$ of (possibly non-disjoint) subsets of S , that is, $S = \bigcup_{i=1}^h A_i$, and consider the bipartite graph $(S, \mathcal{A}; E)$ where E consists of all edges $\{s_i, A_j\}$ with $s_i \in A_j$, for $1 \leq i \leq n$ and $1 \leq j \leq h$. Define $\mathcal{I}(S)$ as the family of subsets $X \subseteq S$ corresponding to the left endpoints of some matching in the above graph. Then, $\mathcal{M} = (S, \mathcal{I}(S))$ is a transversal matroid based on S .

In the following, the sets A_1, A_2, \dots, A_h in the definition of partition and transversal matroids will be referred to as *categories*. We make the reasonable assumption that for the transversal matroids each element of the input belongs to a constant number of categories. Also, without loss of generality, we assume that each individual element of S makes

Problem	Diversity function $\text{div}(X)$
sum-DMMC	$\sum_{u,v \in X} d(u,v)$
star-DMMC	$\min_{c \in X} \sum_{u \in X \setminus \{c\}} d(c,u)$
tree-DMMC	$w(\text{MST}(X))$
cycle-DMMC	$w(\text{TSP}(X))$
bipartition-DMMC	$\min_{Q \subset X, Q =\lfloor X /2 \rfloor} \sum_{u \in Q, v \in X \setminus Q} d(u,v)$

Table 1. Instantiations of the DMMC problem considered in this paper, with related diversity measures. $w(\text{MST}(X))$ (resp., $w(\text{TSP}(X))$) 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.

a singleton independent set. In fact, elements for which this is not the case can be eliminated since, by the hereditary property of matroids, they cannot belong to larger independent sets.

2.2 Problem definition

Let $\text{div} : 2^S \rightarrow \mathbb{R}$ be a *diversity function* that maps any subset $X \subset S$ to some nonnegative real number. For a specific diversity function div , a matroid $\mathcal{M} = (S, \mathcal{I}(S))$, and a positive integer $k \leq \text{rank}(\mathcal{M})$, the goal of the *Diversity Maximization problem under Matroid Constraint* (DMMC problem, for brevity) is to find an independent set $X \in \mathcal{I}(S)$ of size k that maximizes $\text{div}(X)$. We denote the optimal value of the objective function as

$$\text{div}_{k, \mathcal{M}}(S) = \max_{X \in \mathcal{I}(S), |X|=k} \text{div}(X)$$

In this paper, we will focus on several instantiations of the DMMC problem presented in Table 1, characterized by different diversity functions amply studied in the previous literature¹ [4, 13, 21]. Throughout the paper, the generic term “DMMC problem” will be used whenever a statement applies to all instantiations of Table 1.

Returning to the example mentioned in the introduction, concerning a set S of Wikipedia pages, the covering of various topics, viewed as categories, can be enforced by a partition matroid constraint when each page is labeled by a single topic, or by a transversal matroid constraint when pages may refer to multiple topics.

The algorithms presented in this paper use clustering as a subroutine. For a given positive integer τ , a τ -*clustering* of S is a pair (C, Z) , where $C = \{C_1, \dots, C_\tau\}$ is a partition of S , and $Z = \{z_1, \dots, z_\tau\} \subset S$ is such that $z_i \in C_i$, for $1 \leq i \leq \tau$. Each z_i is said to be the *center* of its respective *cluster* C_i . We define the *radius* of the clustering as

$$r(C, Z) = \max_{1 \leq i \leq \tau} \max_{s \in C_i} d(s, z_i).$$

The problem of finding a τ -clustering of minimum radius is NP-hard, as it is also NP-hard to achieve an approximation factor of $2 - \varepsilon$ in general metric spaces, for any $\varepsilon > 0$ [18]. In the paper, we will make use of the well-known sequential 2-approximation clustering algorithm of [18] (known as GMM in the literature) as a key tool in both our sequential and MapReduce algorithms for the DMMC problem. Instead, in the streaming setting we rely on a strategy reminiscent of the seminal streaming clustering algorithm of [14]. In fact, both [18] and [14] limit themselves to identifying a suitable set $Z = \{z_1, \dots, z_\tau\}$ of centers which implicitly induce a clustering $(C = \{C_1, \dots, C_\tau\}, Z)$ with the desired approximation quality, where each C_i is the set of elements which are closer to z_i than any other center.

¹Observe that function $\text{div}(X) = \min_{u \neq v \in X} d(u, v)$, also well studied in the literature for the unconstrained variant of the problem is missing in the table since we were not able to obtain meaningful results for it. We will discuss this issue in the conclusions.

2.3 Doubling dimension

Our algorithms will be analyzed in terms of the dimensionality of the dataset S , as captured by the well-established notion of doubling dimension. Formally, for a given point $x \in S$, let the *ball of radius r centered at x* be the subset of points of S at distance at most r from x . The *doubling dimension* of S is defined as the smallest value D such that any ball of radius r centered at an element x is covered by at most 2^D suitably centered balls of radius $r/2$. Observe that the doubling dimension of a dataset S of size n is upper bounded by $\log_2 n$. The algorithms that will be presented in this paper adapt automatically to the doubling dimension D of the input dataset and attain their best performance when D is small, possibly constant. This is the case, for instance, of datasets S whose points belong to low-dimensional Euclidean spaces, or represent nodes of mildly-expanding network topologies under shortest-path distances. The characterization of datasets (or metric spaces) through their doubling dimension has been used in the literature in several contexts, including routing [23], clustering [2], nearest neighbour search [15], and machine learning [19].

3 CORESETS

The notion of *coreset* has been introduced in [3] as a tool for the development of efficient algorithms for optimization problems on large datasets. In broad terms, for a given computational objective, a coreset is a small subset of the input which embodies a feasible solution whose cost is a good approximation to the cost of an optimal solution over the entire input. Coreset constructions have been successfully developed for the unconstrained diversity maximization problem [4, 10, 21]. In fact, these constructions feature an additional *composability* property, meaning that the construction can be applied independently to the subsets of an arbitrary input partition so that the union of the coresets extracted from each subset is itself a coreset for the entire input. This additional property enables the development of scalable distributed (e.g., MapReduce) algorithms. Indeed, the coreset constructions devised in this section for the DMMC problem are also composable.

Throughout the section, we refer to an arbitrary input to a DMMC problem, which is specified by a set S of size n , a matroid $\mathcal{M} = (S, \mathcal{I}(S))$, and an integer $k \leq \text{rank}(\mathcal{M})$. The formal definition of coreset for the problem is the following.

DEFINITION 3. *For a positive real-valued $\beta \leq 1$, a subset $T \subseteq S$ is a β -coreset for the DMMC problem if $\text{div}_{k, \mathcal{M}}(T) \geq \beta \text{div}_{k, \mathcal{M}}(S)$.*

We aim at β -coresets with β close to 1. Before describing how to construct such coresets, we need to establish some technical results. Let $\Delta_S = \max_{a, b \in S} (d(a, b))$ be the *diameter* of S . We have:

FACT 1. *For $k > 1$, there exists an independent set $X \in \mathcal{I}(S)$ of size k containing two points a, b such that*

- $d(a, b) \geq \Delta_S/2$
- $\forall c \in X \setminus \{a, b\}: d(a, c) \geq \Delta_S/4$ or $d(b, c) \geq \Delta_S/4$.

PROOF. We first show that there exists an independent set $\{a, b\}$ of two elements at distance at least $\Delta_S/2$ from one another. Let $p, q \in S$ be such that $\Delta_S = d(p, q)$. If $\{p, q\} \in \mathcal{I}(S)$ then the statement clearly holds with $a = p$ and $b = q$. Otherwise, by the augmentation property, there must exist a point $r \notin \{p, q\}$ such that both $\{p, r\}$ and $\{q, r\}$ are independent sets. Clearly, by the triangle inequality we have that $\max\{d(p, r), d(q, r)\} \geq \Delta_S/2$. If $d(p, r) \geq \Delta_S/2$ (resp., $d(q, r) \geq \Delta_S/2$) the statement holds with $a = p$ (resp., $a = q$) and $b = r$. The set $\{a, b\}$ can be augmented to a set X of size k using the augmentation property $k - 2$ times. Then, the first property stated by the fact immediately established, and the second property follows since, by the triangle inequality, any point in $X \setminus \{a, b\}$ must be at distance at least $\Delta_S/4$ from either a or b . \square

Observe that each diversity function div listed in Table 1 is a sum of $f(k)$ distances between points, where $f(k) = \binom{k}{2}$, for max-DMMC, $f(k) = k - 1$ for star-DMMC and tree-DMMC, $f(k) = k$ for cycle-DMMC, and $f(k) = \lfloor k/2 \rfloor \lceil k/2 \rceil$ for bipartition-DMMC. A crucial parameter for the analysis of our algorithms is the *average fairness*

$$\rho_{S,k} = \frac{\text{div}_{k, \mathcal{M}(S)}}{f(k)}.$$

The following lemma provides a lower bound to the average fairness as a function of the diameter of the input set, for each of the diversity functions considered in this paper.

LEMMA 1. *For $k > 1$, we have that*

$$\rho_{S,k} \geq \begin{cases} \Delta_S/(2k) & \text{for sum-DMMC} \\ \Delta_S/(4(k-1)) & \text{for star-DMMC} \\ \Delta_S/(2(k-1)) & \text{for tree-DMMC} \\ \Delta_S/k & \text{for cycle-DMMC} \\ \Delta_S/(2(k+1)) & \text{for bipartition-DMMC} \end{cases}$$

PROOF. Consider the independent set X of size k , whose existence is proved in Fact 1, which contains two points a, b with $d(a, b) \geq \Delta_S/2$ and such that the remaining $k - 2$ points are at distance at least $\Delta_S/4$ from a or b , and observe that

$$\rho_{S,k} \geq \frac{\text{div}(X)}{f(k)}.$$

For the sum-DMMC problem, $\text{div}(X) \geq (k-1)\Delta_S/4$ since $d(a, b) \geq \Delta_S/2$ and for each $c \in X \setminus \{a, b\}$, $d(a, c) + d(b, c) \geq \Delta_S/4$. The bound follows since, for this diversity function, $f(k) = \binom{k}{2}$. For the star-DMMC problem, $\text{div}(X) \geq \Delta_S/4$ since for any $c \in X$ there exists at least one point $u \in X \setminus \{c\}$ such that $d(c, u) \geq \Delta_S/4$. The bound follows since, for this diversity function, $f(k) = k - 1$. For the tree-DMMC problem, $\text{div}(X) \geq \Delta_S/2$, since any spanning tree connecting the points of X includes a path between a and b which has length at least $d(a, b) \geq \Delta_S/2$. The bound follows since, for this diversity function, $f(k) = k - 1$. Similarly, for the cycle-DMMC problem, $\text{div}(X) \geq \Delta_S$ since any Hamiltonian cycle connecting the points of X is made of two edge-disjoint paths between a and b whose aggregate length is at least $2d(a, b) \geq \Delta_S$. The bound follows since, for this diversity function, $f(k) = k$. For the bipartition-DMMC problem, let $(Q, X \setminus Q)$, with $|Q| = \lfloor k/2 \rfloor$, be the bipartition of X minimizing the sum $\sum_{u \in Q, v \in X \setminus Q} d(u, v)$. We distinguish two cases. In case a and b belong to the same subset of the bipartition, then each point c in the other subset will contribute at least $d(a, c) + d(b, c) \geq \Delta_S/4$ to the sum, whence $\text{div}(X) \geq \lfloor k/2 \rfloor \Delta_S/4$. Otherwise, assume w.l.o.g. that $a \in Q$ and $b \in X \setminus Q$ and observe that these two points contribute at least $\Delta_S/2$ to the sum. Out of the remaining $k - 2$ points in X , we can create $\lfloor k/2 \rfloor - 1$ mutually disjoint pairs $\{u_i, v_i\}$, with $u_i \in Q$ and $v_i \in X \setminus Q$, for $1 \leq i < \lfloor k/2 \rfloor$. We have that either $d(a, v_i) + d(b, u_i) \geq \Delta_S/4$ or, by the triangle inequality, $d(u_i, v_i) \geq d(a, b) - d(a, v_i) - d(b, u_i) > \Delta_S/2 - \Delta_S/4 = \Delta_S/4$. Thus, pair $\{u_i, v_i\}$ contributes at least $\Delta_S/4$ to the sum, hence $\text{div}(X) \geq (\lfloor k/2 \rfloor - 1)\Delta_S/4 + d(a, b) \geq \lfloor k/2 \rfloor \Delta_S/4$. The bound follows since, for this diversity function, $f(k) = \lfloor k/2 \rfloor \lceil k/2 \rceil$. \square

It is easy to argue that there are instances of the problem for which the lower bound to $\rho_{S,k}$ is tight, up to constant factors.

Intuitively, a good coreset in our setting is a set of points that contains, for each independent set in $\mathcal{I}(S)$, an independent set of comparable diversity. The following lemma (which holds for every instantiation of the DMMC problem) formalizes this intuition.

LEMMA 2. Let $\varepsilon < 1$ be a positive value. Consider a subset $T \subseteq S$ such that for each $X \in \mathcal{I}(S)$ of size k there is an injective proxy function $p : X \rightarrow T$ satisfying (i) $\{p(x) : x \in X\} \in \mathcal{I}(S)$; and (ii) $d(x, p(x)) \leq (\varepsilon/2)\rho_{S,k}$, for every $x \in X$. Then, T is a $(1 - \varepsilon)$ -coreset.

PROOF. Let $O \subseteq S$ be an optimal solution to the DMMC instance and consider the set of proxies $p(O) = \{p(o) : o \in O\} \subseteq T$, which is an independent set of size k by hypothesis and is thus a feasible solution. By the triangle inequality and the properties of the proxy function, for each pair $o_1, o_2 \in O$, we have that

$$\begin{aligned} d(p(o_1), p(o_2)) &\geq d(o_1, o_2) - d(o_1, p(o_1)) - d(o_2, p(o_2)) \\ &\geq d(o_1, o_2) - \varepsilon\rho_{S,k}. \end{aligned}$$

It follows that

$$\text{div}(p(O)) \geq \text{div}(O) - f(k)(\varepsilon\rho_{S,k}),$$

where $f(k)$ denotes, as stated before, the number of distances which contribute to div . Since $\text{div}(O) = \text{div}_{k, \mathcal{M}}(S) = f(k)\rho_{S,k}$, it follows that

$$\text{div}(p(O)) \geq (1 - \varepsilon) \text{div}_{k, \mathcal{M}}(S).$$

The lemma follows, since $\text{div}_{k, \mathcal{M}}(T) \geq \text{div}(p(O))$. \square

In the next subsections, we will develop clustering-based constructions of small coresets meeting the requirements of Lemma 2 for partition and transversal matroids. We will also point out how to extend these constructions to the case of general matroids, at the expense of a possible blow-up in the coreset size.

3.1 Coreset construction

Fix an arbitrary positive constant $\varepsilon < 1$, and consider a τ -clustering (C, Z) of the input set S , where $C = \{C_1, \dots, C_\tau\}$ and $Z = \{z_1, \dots, z_\tau\}$, with radius

$$r(C, Z) \leq \frac{\varepsilon}{4}\rho_{S,k}. \quad (1)$$

Observe that such a clustering surely exists, as long as τ is large enough, since the trivial n -clustering where each element of S is a singleton cluster has radius 0. Our coresets are obtained by selecting a suitable subset from each cluster of C so that the properties (i) and (ii) specified in Lemma 2 are satisfied. In particular, the bound on the clustering radius is functional to establish property (ii).

The effectiveness of this approach relies on the existence of a clustering with suitably small τ , so that the resulting coreset size is significantly smaller than n . Although it is not easy to determine a meaningful upper bound to τ in the general case, in the next subsection we show that for the important case of metric spaces of bounded doubling dimension, τ is upper bounded by a constant w.r.t. n .

Technically, throughout this section, we work under the hypothesis that a τ -clustering (C, Z) whose radius satisfies Equation 1 is available, and postpone the description of its explicit construction to Section 4, since different constructions will be employed for the different computational settings considered in this paper.

Below, we describe, separately for each matroid type, how a coresets can be derived from the τ -clustering (C, Z) of S of radius $r(C, Z) \leq (\varepsilon/4)\rho_{S,k}$.

3.1.1 Partition matroid. Consider a partition matroid $\mathcal{M} = (S, \mathcal{I}(S))$ with categories A_1, \dots, A_h and cardinality bounds k_1, \dots, k_h . We build the coreset T for S as follows. From each cluster C_i of C we select a largest independent set $T_i \subseteq C_i$ of size at most k , and let $T = \bigcup_{i=1}^r T_i$. The effectiveness of this simple strategy is stated by the following theorem.

THEOREM 1. *The set T computed by the above procedure from a τ -clustering (C, Z) of S of radius $r(C, Z) \leq (\varepsilon/4)\rho_{S,k}$ is a $(1 - \varepsilon)$ -coreset of size $O(k\tau)$ for the DMMC problem.*

PROOF. First observe that the bound on the size of T is immediate by construction. As for the approximation, we now show that for any independent set $X \in \mathcal{I}(S)$ with $|X| = k$, there is an injective function $p : X \rightarrow T$ such that $\{p(x) : x \in X\} \in \mathcal{I}(S)$ and $d(x, p(x)) \leq (\varepsilon/2)\rho_{S,k}$. The result will then follow from Lemma 2. Consider the set of clusters as partitioned in two families: C^ℓ includes those clusters that contain an independent set of size k , whereas $C^s = C - C^\ell$ includes the remaining clusters that contain only independent sets of size strictly less than k . For each cluster $C_i \in C^s$, consider the independent set $T_i \subseteq C_i$ included in T by the algorithm. It is easy to see that for each category A_j , we have $|X \cap C_i \cap A_j| \leq |T_i \cap A_j|$, since $|T_i| < k$ and that T_i is a largest independent set in C_i . Therefore, each point $x \in X \cap C_i$ can be associated with a distinct point $p(x) \in T_i$ belonging to the same category. Let $P = \{p(x) : x \in X \cap (\bigcup_{C \in C^s} C)\}$ and note that P is an independent set, since $X \cap (\bigcup_{C \in C^s} C)$ is an independent set (by the hereditary property of matroids) and the number of elements per category is the same in X and in P . If P has size k , then X has no points in clusters of C^ℓ and the lemma is proved. If instead P has size strictly less than k , we consider the clusters in C^ℓ containing the remaining $k - |P|$ points of X . For each such C_i , we expand P to a larger independent set by adding $n_i = |X \cap C_i|$ elements from T_i using the augmentation property n_i times, exploiting the fact that T_i is an independent set of size k . These n_i elements of T_i can act as distinct proxies of the elements in $X \cap C_i$ under function p . After these additions, we obtain an independent set P of k distinct proxies for the elements of X . Since, for each $x \in X$, $p(x) \in P$ is taken from the same cluster of C , we have that, by the triangle inequality and by Equation (1), $d(x, p(x)) \leq (\varepsilon/2)\rho_{S,k}$. \square

3.1.2 Transversal matroid. The construction of the coreset T is more involved in the case of transversal matroids. Consider a transversal matroid $\mathcal{M} = (S, \mathcal{I}(S))$ defined over a family $\mathcal{A} = \{A_1, \dots, A_h\}$ of h (not necessarily disjoint) categories. The fact that the categories can now overlap complicates the coreset construction, resulting in slightly larger coresets. For each cluster C_i of C we begin by selecting, as before, a largest independent set U_i of size at most k . If $|U_i| = k$, then we set $T_i = U_i$. Instead, if $|U_i| < k$, let $\mathcal{A}' \subseteq \mathcal{A}$ be a subfamily of categories of the points of U_i . We construct T_i by augmenting U_i in such a way that for each category $A \in \mathcal{A}'$ there are $\min\{k, |A \cap C_i|\} \leq k$ points of category A in T_i . (Observe that a point contributes to the count for all of its categories in \mathcal{A}') Finally, we let $T = \bigcup_{i=1}^r T_i$.

THEOREM 2. *The set T computed by the above procedure from a τ -clustering (C, Z) of S of radius $r(C, Z) \leq (\varepsilon/4)\rho_{S,k}$ is a $(1 - \varepsilon)$ -coreset of size $O(k^2\tau)$ for the DMMC problem.*

PROOF. The proof follows the lines of the one of Theorem 1. The bound on the size of T follows from the assumption that each point belongs to a constant number of categories, while for the approximation guarantee of the coreset it is sufficient to show that for any independent set $X \in \mathcal{I}(S)$ with $|X| = k$, there is an injective function $p : X \rightarrow T$ such that $\{p(x) : x \in X\} \in \mathcal{I}(S)$ and $d(x, p(x)) \leq (\varepsilon/2)\rho_{S,k}$. Again, we split C into C^ℓ (clusters containing an independent set of size k) and C^s (remaining clusters). We first consider clusters in C^s . Fix a cluster $C_i \in C^s$ and let

$\mathcal{A}' \subseteq \mathcal{A}$ be the subfamily of categories of the points of U_i , and recall that, by construction, for each $A \in \mathcal{A}'$ there are $\min\{k, |A \cap C_i|\}$ points of category A in T_i . Let $C_i \cap X = \{x_1, x_2, \dots, x_m\}$, and let $A_1^X, A_2^X, \dots, A_m^X \in \mathcal{A}$ be distinct categories that can be matched to x_1, x_2, \dots, x_m . Note that the maximality of U_i implies that $|U_i| \geq m$. Without loss of generality, assume that $x_1, \dots, x_j \in T_i$ while $x_{j+1}, \dots, x_m \notin T_i$, for some $0 \leq j \leq m$, that is, assume that exactly those j points of X are included in T_i . We initially set the proxies $p(x_1) = x_1, \dots, p(x_j) = x_j$. Then, consider category A_{j+1}^X and observe that $A_{j+1}^X \in \mathcal{A}'$, otherwise x_{j+1} (matched to A_{j+1}^X) could be added to U_i contradicting its maximality within C_i . Since x_{j+1} was not included in T_i , there must be at least $k \geq m$ elements of A_{j+1}^X in T_i . We can thus select one such element (distinct from $p(x_1), p(x_2), \dots, p(x_j)$) as proxy $p(x_{j+1})$, matched to A_{j+1}^X . By repeating this step for x_{j+2}, \dots, x_m , we obtain an independent set of proxies for the elements of $X \cap C_i$, each matched to the same category as its corresponding element. Then, by iterating this construction over all clusters of C^S we get an independent set P of proxies for the elements of X belonging to such clusters, each matched to the same category as its corresponding element. If P has size strictly less than k , the remaining proxies for the elements of X residing in clusters of C^ℓ can be chosen by reasoning as in the proof of Theorem 1, using the augmentation property. Also, by the bound on the radius of (C, Z) , we have $d(x, p(x)) \leq (\varepsilon/2)\rho_{S,k}$, for each $x \in X$. \square

We remark that the $O(k^2\tau)$ bound on the coreset size is a rather conservative worst-case estimate. In fact, as reported in the experimental section, it is conceivable that, in practice, much smaller sizes can be expected. Another important observation is that the constructions for both the partition and the transversal matroid yield coresets whose size is independent of $n = |S|$.

3.1.3 General Matroids. Consider now a constraint specified as a general matroid $\mathcal{M}(S, \mathcal{I}(S))$. We can still build a coreset T in this general scenario from the τ -clustering (C, Z) of S . Specifically, for each cluster C_i of C we compute a largest independent set U_i of size at most k . If $|U_i| = k$, then we set $T_i = U_i$, otherwise we set $T_i = C_i$. Finally, we let $T = \bigcup_{i=1}^{\tau} T_i$. Note that, unlike the case of partition and transversal matroids, this coreset may potentially grow very large when clusters do not contain large enough independent sets. However, for small enough values of τ and k (hence, large cluster sizes) we expect that each cluster may reasonably contain an independent set of size k , hence an actual coreset size of $O(k\tau)$ is conceivable. We have:

THEOREM 3. *The set T set computed by the above procedure from a τ -clustering (C, Z) of S of radius $r(C, Z) \leq (\varepsilon/4)\rho_{S,k}$ is a $(1 - \varepsilon)$ -coreset for the DMMC problem.*

PROOF. As in the proofs of Theorems 1 and 2, it is sufficient to determine a suitable proxy function p for every independent set $X \in \mathcal{I}(S)$ of size k . We set function p to be the identity function for those points belonging to clusters C_i such that $T_i = C_i$. The proxies for the elements of X residing in the other clusters can then be chosen through repeated applications of the augmentation property. Specifically, consider a cluster C_i such that $T_i \subset C_i$ (hence, T_i is an independent set of size k by construction) and let $m_i = |X \cap C_i|$. Since T_i is an independent set of size k , we can apply the augmentation property m_i times to select the m_i proxies $X \cap C_i$ from T_i so that the union of all proxies is an independent set. Also, by the bound on the radius of (C, Z) , we have $d(x, p(x)) \leq (\varepsilon/2)\rho_{S,k}$, for each $x \in X$, hence the thesis follows from Lemma 2. \square

3.2 Relating cluster granularity to doubling dimension

In this subsection, we show that for datasets of bounded doubling dimension, very coarse clusterings of small radius exist, and discuss how this feature can be exploited to obtain coresets of small size for the DMMC problem.

THEOREM 4. *Let S be an n -point dataset of doubling dimension D . For any integer τ , with $1 \leq \tau \leq n$, the minimum radius of any τ -clustering of S , denoted with $r_\tau^*(S)$, is*

$$r_\tau^*(S) \leq \frac{2\Delta_S}{\tau^{1/D}}.$$

PROOF. Observe that the whole set S is contained in the ball of radius Δ_S centered at any element of S . By applying the definition of doubling dimension i times, starting from such a ball, we can cover S with 2^{iD} balls of radius at most $\Delta_S/2^i$. Let j be such $2^{jD} \leq \tau < 2^{(j+1)D}$. The theorem follows since the minimum radius of any τ -clustering of S is upper bounded by the minimum radius of any 2^{jD} -clustering of S , which is in turn upper bounded by $\Delta_S/2^j \leq 2\Delta_S/\tau^{1/D}$. \square

In order to appreciate the relevance of the above theorem, consider an algorithm \mathcal{A} that, given a target number of clusters τ , returns a τ -clustering for S whose radius $r_\tau^{\mathcal{A}}(S)$ is a factor at most $\sigma > 1$ larger than the minimum radius attainable by any τ -clustering for S , that is $r_\tau^{\mathcal{A}}(S) \leq \sigma \cdot r_\tau^*(S)$. The theorem implies that, by setting $\tau = (32\sigma k/\epsilon)^D$, Algorithm \mathcal{A} returns a τ -clustering for S with radius at most

$$r_\tau^{\mathcal{A}}(S) \leq \sigma r_\tau^*(S) \leq \sigma \frac{2\Delta_S}{\tau^{1/D}} \leq \frac{\epsilon \Delta_S}{16k} \leq \epsilon \frac{\rho_{S,k}}{4},$$

for any of the DMMC instantiations in Table 1, where the last inequality follows from Lemma 1. Observe that such a small-radius clustering can be used as the base for the coreset constructions illustrated in the previous subsection, and its limited granularity ensures that the coresets have size independent of n for the partition and transversal matroids. This feature will be crucial for obtaining efficient sequential, distributed, and streaming algorithms for all instantiations of the DMMC problem considered in this paper. Note that this approach requires the knowledge of the doubling dimension D in order to set τ properly. However, in the next section, we will devise implementations of the constructions that return coresets of comparable quality and size without knowledge of D . This is a very desirable feature for practical purposes since the doubling dimension of a dataset is hard to estimate.

4 DMMC ALGORITHMS

In this section, we will devise efficient sequential, MapReduce and Streaming algorithms for the instantiations of the DMMC problem defined in Table 1. At the core of these algorithms are efficient implementations of the coreset constructions presented in the previous section. Consider any of the instantiations of the DMMC problem and an arbitrary input specified by a set S of size n , a matroid $\mathcal{M} = (S, \mathcal{I}(S))$, and an integer $k \leq \text{rank}(\mathcal{M})$. Our general approach is to extract a $(1 - \epsilon)$ -coreset T from S and then run the best available sequential algorithm on T . In Subsections 4.1, 4.2, and 4.3, we present the implementations of the $(1 - \epsilon)$ -coreset construction in the sequential, MapReduce and Streaming setting, respectively. Finally, in Subsection 4.4 we will show how to employ these constructions to yield the final algorithms in the various setting and analyze their performance.

As in previous works, we assume that constant-time oracles are available to compute the distance between two elements of S and to check whether a subset of S is an independent set [1].

4.1 Sequential coreset construction

Our sequential implementation of the $(1 - \epsilon)$ -coreset construction presented in the previous section, dubbed SEQ-CORESET (see Algorithm 1 for the pseudocode) leverages the well-known 2-approximate clustering algorithm GMM mentioned before [18]. For a given input S , GMM determines the set Z of cluster centers in $|Z|$ iterations, by initializing Z with an arbitrary element of $z_1 \in S$, and then iteratively adding to Z the element in S of maximum distance from

the current Z . The algorithm can be instrumented to maintain the set of clusters $C = \{C_z : z \in Z\}$ centered at Z , with each element of S assigned to the cluster of its closest center, and the radius of such a clustering. Each iteration of GMM can be easily implemented in time linear in n (see Procedure GMM-ITERATION in Algorithm 1) and, as proved in the original paper [18], the clustering resulting at the end of the i -th iteration has radius which is no more than twice as large as the minimum radius of any i -clustering.

Since the second center z_2 selected by GMM is the farthest point from the first (arbitrarily selected) center z_1 , it is easy to see that the distance $\delta = d(z_1, z_2)$ between these first two centers is such that $\Delta_S/2 \leq \delta \leq \Delta_S$. In order to build the $(1 - \epsilon)$ -coreset T , we run GMM for a number τ of iterations sufficient to reduce the radius of the clustering to a value at most $\epsilon\delta/(16k)$. Once the clustering is computed, for each cluster C_z a largest independent set $U_z \subseteq C_z$ of size at most k is determined. For the partition matroid, the coreset T is obtained as the union of the U_z 's. For the transversal matroid, T is obtained by first augmenting each U_z in such a way that for each category A of a point of U_i , the augmented set contains $\min\{k, |A \cap C_i|\}$ points of $A \cap C_z$, and then taking the union of these augmented sets. Finally for all other matroids, T is obtained by first augmenting each independent set U_z of size $|U_z| < k$ to the entire cluster C_z , and then again taking the union of the U_z 's (see Procedure EXTRACT in Algorithm 1).

Algorithm 1: SEQCORESET(S, k, ϵ)

```

Let  $S = \{x_1, x_2, \dots, x_n\}$ 
 $z_1 \leftarrow x_1$ 
 $z_2 \leftarrow \arg \max_{x \in S} \{d(z_1, x)\}$ 
 $\delta \leftarrow d(z_1, z_2)$ 
 $Z \leftarrow \{z_1, z_2\}$ 
for  $i \in \{1, 2\}$  do  $C_{z_i} \leftarrow \{x \in S : z_i = \arg \min_{z' \in Z} \{d(x, z')\}\}$ 
 $C \leftarrow \{C_{z_1}, C_{z_2}\}$ 
while  $(r(C, Z) > \epsilon\delta/(16k))$  do  $(C, Z) \leftarrow \text{GMM-ITERATION}(S, Z)$ 
for  $z \in Z$  do  $U_z \leftarrow \text{EXTRACT}(C_z, k)$ 
return  $T = \cup_{z \in Z} U_z$ 

```

```

procedure GMM-ITERATION( $S, Z$ )
 $y \leftarrow \arg \max_{x \in S} \{\min_{z \in Z} \{d(x, z)\}\}$ 
 $Z \leftarrow Z \cup \{y\}$ 
for  $z \in Z$  do  $C_z \leftarrow \{x \in S : z = \arg \min_{z' \in Z} \{d(x, z')\}\}$ 
 $C \leftarrow \{C_z : z \in Z\}$ 
return  $(C, Z)$ 

```

```

procedure EXTRACT( $C, k$ )
 $U \leftarrow$  maximal independent set in  $C$  of size  $\leq k$ 
if  $(|U| = k) \vee (\text{matroid type} = \text{partition})$  then return  $U$ 
else
  switch ( $\text{matroid type}$ ) do
    case transversal do
      if  $(\exists \text{ Category } A \text{ of } x \in U : |A \cap U| < k)$  then
        add to  $U$  extra points from  $C$  so to have  $\min\{k, |A \cap C|\}$  points of Category  $A$  in  $U$ 
      case other do  $U \leftarrow C$ 
return  $U$ 

```

We have:

THEOREM 5. *Let $\varepsilon < 1$ be an arbitrary positive constant. The above algorithm computes a $(1 - \varepsilon)$ -coreset T for the DMMC problem in time $O(n\tau)$. For the partition (resp., transversal) matroid, T has size $O(k\tau)$ (resp., $O(k^2\tau)$). If the set S has constant doubling dimension D , then $\tau = O\left((k/\varepsilon)^D\right)$.*

PROOF. It is easy to see that the distance δ between the first two centers selected by GMM is such that $\Delta_S/2 \leq \delta \leq \Delta_S$, thus, by Lemma 1, the radius of the τ -clustering is $\varepsilon\delta/(16k) \leq \varepsilon\Delta_S/(16k) \leq \varepsilon\rho_{S,k}/4$. The fact that T is $(1 - \varepsilon)$ -coreset for the DMMC problem with the stated sizes then follows by Theorems 1 and 2. As for the running time, the cost of the GMM algorithm is $O(n\tau)$, while, the subsequent extraction of the coreset can be accomplished using a total of $O(n)$ invocations of the independent-set oracle to determine a largest independent set in each cluster, exploiting the augmentation property, and additional $O(nk) = O(n\tau)$ operations on suitable dictionary structures to determine the extra elements for each category (in the sole case of the transversal matroid). Finally, the bound on the cluster granularity can be established by noticing that for $\tau = (128k/\varepsilon)^D$, Theorem 4 together with the fact that GMM is a 2-approximate clustering algorithm, implies that after this many iterations of GMM the radius of the clustering is at most

$$\frac{4\Delta_S}{(\tau)^{1/D}} \leq \frac{\varepsilon\delta}{16k}.$$

Note that for constant D , $\tau = O\left((k/\varepsilon)^D\right)$. □

It is conceivable that the large constants involved in the coreset sizes are an artifact of the analysis since the experiments reported in Section 5, show that much smaller coresets yield very accurate solutions. We also wish to stress that, thanks to the incremental nature of GMM, the coreset construction needs not know the doubling dimension D of the metric space in order to attain the desired bound on τ .

4.2 MapReduce coreset construction

A MapReduce (MR) algorithm executes as a sequence of *rounds* where, in a round, a multiset of key-value pairs is transformed into a new multiset of pairs through two user-specified map and reduce functions, as follows: first the map function is applied to each individual pair returning a set of new pairs; then the reduce function is applied independently to each subset of pairs having the same key, again producing a set of new pairs. Each application of the reduce function to a subset of same-key pairs is referred to as *reducer*. The model is parameterized by the total memory available to the computation, denoted with M_T , and by the maximum amount of memory locally available to run each map and reduce function, denoted with M_L . The typical goal for a MR algorithm is to run in as few rounds as possible while keeping M_T (resp., M_L) linear (resp., substantially sublinear) in the input size [16, 22, 28].

To obtain MR implementations of our $(1 - \varepsilon)$ -coreset constructions we will crucially exploit an additional property of these constructions, known as *composability* [21]. Formally, composability ensures that a $(1 - \varepsilon)$ -coreset for a set S can be obtained as the union of $(1 - \varepsilon)$ -coresets extracted from each subset of any given partition of S . Therefore, a coreset T can be built in one MR round as follows. First, the set S is partitioned evenly but arbitrarily into $\ell > 0$ disjoint subsets S_1, \dots, S_ℓ , through a map function, where ℓ is a design parameter (corresponding to the degree of parallelism) to be set in the analysis. Then, each S_i is assigned to a distinct reducer, which builds a $(1 - \varepsilon)$ -coreset T_i for S_i based on a τ_i -clustering of radius at most $\varepsilon\delta_i/(16k)$, by running $\text{SEQCORESET}(S_i, k, \varepsilon)$ (see Algorithm 1), where $\delta_i \in [\Delta_{S_i}/2, \Delta_{S_i}]$ now represents the distance between the first two centers selected by the algorithm. Coreset T is simply the union of the T_i 's.

THEOREM 6. *Let $\varepsilon < 1$ be an arbitrary positive constant and let $\tau = \sum_{1 \leq i \leq \ell} \tau_i$. The above 1-round MR algorithm computes a $(1 - \varepsilon)$ -coreset T for the DMMC problem with memory requirements $M_T = O(n)$ and $M_L = O(n/\ell)$. For the partition (resp., transversal) matroid, T has size $O(k\tau)$ (resp., $O(k^2\tau)$). If the set S has constant doubling dimension D , then $\tau = O(\ell(k/\varepsilon)^D)$.*

PROOF. The coreset $T = \bigcup_{i=1, \ell} T_i$ computed by the algorithm can be regarded as being derived from a τ -clustering of S of radius at most

$$\varepsilon \frac{\max_{i=1, \ell} \Delta_{S_i}}{16k} \leq \varepsilon \frac{\Delta_S}{16k} \leq \varepsilon \frac{\rho_{S,k}}{4},$$

where the last inequality follows by Lemma 1. Hence, by Theorems 1 and 2, T is $(1 - \varepsilon)$ -coreset for S . The bound on the total and local memory is immediate, since the sequential algorithm used to extract each T_i runs in linear space. For what concerns the bound on τ , consider a generic subset S_i . It is easy to adapt the proof of Theorem 4 to show that for $\tau_i = (256k/\varepsilon)^D$, there is a τ_i -clustering of S_i of radius $\varepsilon \Delta_{S_i}/(128k)$, whose centers are points of S which may not belong to S_i . By recentering each cluster on a point of S_i , the radius at most doubles. Therefore, after τ_i iterations GMM returns a clustering of radius at most

$$\frac{4\varepsilon \Delta_{S_i}}{128k} \leq \frac{\varepsilon \delta_i}{16k}.$$

Hence, $\tau = \sum_{1 \leq i \leq \ell} \tau_i \leq \ell(256k/\varepsilon)^D$, thus for constant D , $\tau = O(\ell(k/\varepsilon)^D)$. \square

As will be illustrated in Subsection 4.4, in a second round the coreset T can be gathered in a single reducer which will then extract the final solution by running on T the best available sequential approximation algorithm. The degree of parallelism ℓ can then be fixed in such a way to balance the local memory requirements of both rounds. However, if this balancing results in a large value of ℓ (possibly a function of n), since the approximation algorithm used to extract the final solution is computationally intensive, the work in the second round could easily grow too large. To circumvent this problem, the slow approximation algorithm can be run on a smaller coreset T' of size independent of n , which can be computed from T using our sequential coreset construction, at the expense of an extra $1 - \varepsilon$ factor in the final approximation ratio.

4.3 Streaming coreset construction

In the streaming setting [20] the computation is performed by a single processor with a small-size working memory, and the input is provided as a continuous stream of items which is usually too large to fit in the working memory. Typically, streaming strategies aim at a single pass on the input but in some cases few additional passes may be needed. Key performance indicators are the size of the working memory and the number of passes.

In this subsection, we describe a 1-pass streaming algorithm, dubbed `STREAMCORESET` which implements the $(1 - \varepsilon)$ -coreset construction for the DMMC problem described in Section 3.1. At the core of the algorithm is the computation of a set Z of centers implicitly defining a $|Z|$ -clustering of radius at most $\varepsilon \Delta_S/(16k)$, which is in turn upper bounded by $\varepsilon \rho_{S,k}/4$ by Lemma 1. Specifically, Z is obtained by combining a center selection strategy akin to the one presented in [14], with a progressive estimation of Δ_S , needed to obtain the desired bound on the cluster radius. As Z is computed, a set of extra points, dubbed *delegates*, is selected from each of the clusters induced by Z so that, at the end of the stream, all selected points provide the desired coreset for each matroid type.

Algorithm `STREAMCORESET` works as follows (see Algorithm 2 for the complete pseudocode). The algorithm maintains the following variables: an estimate R for the diameter of the first i points of the stream, the current set of centers Z , and a set of delegates D_z for each center $z \in Z$. For $1 \leq i \leq n$, let x_i denote the i -th point of S in the stream. Initially,

we set $R = d(x_1, x_2)$, $Z = \{x_1, x_2\}$, and $D_{x_i} = \{x_i\}$, for $i = 1, 2$. Let $c > 0$ be a suitable constant which will be set in the analysis. For $i \geq 3$, the processing of point x_i is performed as follows. Let z be the center in Z closest to x_i . If $d(x_i, z) > 2\epsilon R/(ck)$, then x_i is added to Z and a new set of delegates $D_{x_i} = \{x_i\}$ is created. Otherwise, x_i triggers an update of the delegate set D_z , carried out by calling procedure $\text{HANDLE}(x_i, z, D_z)$, whose actions vary with the matroid type and will be described later. Also, if $d(x_i, x_1) > 2R$, the diameter estimate is updated by setting $R = d(x_i, x_1)$. Finally, if this latter update occurs, a restructuring of Z takes place, where Z is shrunk to a maximal subset Z' of centers at distance greater than $\epsilon R/(ck)$ from one another, and the delegate set of each discarded center $z \in Z - Z'$ is “merged” into the delegate set of its closest center $z' \in Z'$ by invoking $\text{HANDLE}(x, z', D_{z'})$ on each $x \in D_z$.

Given a point x , a center $z \in Z$, and its associated delegate set D_z , procedure $\text{HANDLE}(x, z, D_z)$ first checks whether D_z is an independent set of size k and discards x if this is the case. Otherwise, D_z is updated differently according to the matroid type. For the partition matroid, x is added to D_z only if $D_z \cup \{x\}$ is an independent set (clearly of size at most k). For the transversal matroid, x is added to D_z only if one of the categories of x is still short of k delegates in D_z , while for any other matroid type, x is always added to D_z . In all cases, after x is added to D_z , if D_z contains an independent set D' of size k , D' becomes the new D_z and all other points in $D_z - D'$ are discarded.

In order to analyze the algorithm, we need to make some preliminary observations and introduce some notation. Let Z_i be the set of centers after processing the i -th point of the stream. For each point x_j we define the sequence of centers $z_{x_j}^{(j)}, z_{x_j}^{(j+1)}, \dots, z_{x_j}^{(n)}$, with $z_{x_j}^{(i)} \in Z_i$ for each $i \geq j$, as follows: $z_{x_j}^{(j)}$ is the center in Z_j closest to x_j (possibly x_j itself); for $i > j$, if $z_{x_j}^{(i-1)} \in Z_i$ then $z_{x_j}^{(i)} = z_{x_j}^{(i-1)}$, otherwise $z_{x_j}^{(i)}$ is the center of Z_i closest to $z_{x_j}^{(i-1)}$. For each x_j we say that $z_{x_j}^{(n)}$ is its *reference center*. Let $Z = Z_n$ be the set of centers at the end of the algorithm. For every $z \in Z$, let C_z denote the set all points $x \in S$ for which z is the reference center (note that z is the reference center for itself). Clearly, $(C = \{C_z : z \in Z\}, Z)$ is a $|Z|$ -clustering of S . Also, by observing that every time a delegate x is transferred from a set D_z to a set $D_{z'}$ its reference center becomes z' , it is immediate to conclude that at the end of the algorithm, $D_z \subseteq C_z$ for each $z \in Z$.

LEMMA 3. *Consider the execution of $\text{STREAMCORESET}(S, k, \epsilon, c)$ for an arbitrary positive constant ϵ and $c = 32$. The $|Z|$ -clustering (C, Z) defined above has radius $r(C, Z) < (\epsilon/4)\rho_{S,k}$. Also, if the set S has constant doubling dimension D , then $|Z| = O\left((k/\epsilon)^D\right)$.*

PROOF. Let $S_i = \{x_1, x_2, \dots, x_i\}$, and let R_i be the value of variable R after processing x_i . Analogously, as defined above, Z_i is the value of Z after processing x_i . To prove the first part of the lemma, we show that the following invariants are inductively maintained for every $i \geq 2$:

- (1) $\Delta_{S_i}/4 \leq R_i \leq \Delta_{S_i}$;
- (2) For any two distinct centers $u, v \in Z_i$, $d(u, v) > \epsilon R_i/(ck)$;
- (3) For every $h \leq i$, it holds $d(x_h, z_{x_h}^{(i)}) < 2\epsilon R_i/(ck)$.

Initially ($i = 2$) the invariants trivially hold for any $\epsilon < 1$ and $c \geq 1$. Assume now that the invariants hold after processing x_{i-1} and consider the processing of x_i . Consider first Invariant 1, and note that

$$\begin{aligned} \Delta_{S_i} &= \max\{\Delta_{S_{i-1}}, \max_{j < i}\{d(x_i, x_j)\}\} \\ &\leq \max\{4R_{i-1}, \max_{j < i}\{d(x_i, x_1) + d(x_1, x_j)\}\}, \end{aligned}$$

where the last passage follows from Invariant 1 at step $i - 1$ and from the triangle inequality. We distinguish two cases. In case $R_i = R_{i-1}$, we have that $d(x_i, x_1) \leq 2R_i$ and, moreover, for any $j < i$, $d(x_1, x_j) \leq 2R_i$ as well, since the

Algorithm 2: STREAMCORESET(S, k, ε, c)

```

Let  $S = x_1, x_2, \dots$ 
 $R \leftarrow d(x_1, x_2)$ 
 $Z \leftarrow \{x_1, x_2\}$ 
 $D_{x_1} \leftarrow \{x_1\}; D_{x_2} \leftarrow \{x_2\}$ 
for ( $i \geq 3$ ) do
   $z \leftarrow \arg \min_{w \in Z} \{d(x_i, w)\}$ 
  if ( $d(x_i, z) > 2\varepsilon R/(ck)$ ) then
     $Z \leftarrow Z \cup \{x_i\}; D_{x_i} \leftarrow \{x_i\}$ 
  else HANDLE( $x_i, z, D_z$ )
  if ( $d(x_i, x_1) > 2R$ ) then
     $R \leftarrow d(x_i, x_1)$ 
     $Z' \leftarrow$  maximal subset of  $Z$  such that
       $\forall u \neq v \in Z' : d(u, v) > \varepsilon R/(ck)$ 
    for ( $z \in Z - Z'$ ) do
       $z' \leftarrow \arg \min_{w \in Z'} \{d(z, w)\}$ 
      for ( $x \in D_z$ ) do HANDLE( $x, z', D_{z'}$ )
  return  $T = \cup_{z \in Z} D_z$ 

procedure HANDLE( $x, z, D_z$ )
if ( $(|D_z| = k) \wedge (D_z \in \mathcal{I}(S))$ ) then discard  $x$ 
else
  switch (matroid type) do
    case partition do
      if ( $D_z \cup \{x\} \in \mathcal{I}(S)$ ) then  $D_z \leftarrow D_z \cup \{x\}$ 
      else discard  $x$ 
    case transversal do
      if ( $\exists$  Category  $A$  of  $x : |A \cap D_z| < k$ ) then
         $D_z \leftarrow D_z \cup \{x\}$ 
        if ( $\exists D' \subseteq D_z : D' \in \mathcal{I}(S) \wedge |D'| = k$ ) then
           $D_z \leftarrow D'$ 
          discard all other points
        else discard  $x$ 
    case other do
       $D_z \leftarrow D_z \cup \{x\}$ 
      if ( $\exists D' \subseteq D_z : D' \in \mathcal{I}(S) \wedge |D'| = k$ ) then
         $D_z \leftarrow D'$ 
        discard all other points

```

algorithm enforces that $d(x_1, x_j) \leq 2R_j$, and $R_j \leq R_i$. Therefore, by the above relation it follows that $\Delta_{S_i}/4 \leq R_i$. Also, $R_i = R_{i-1} \leq \Delta_{S_{i-1}} \leq \Delta_{S_i}$. If instead $R_i \neq R_{i-1}$, then $R_i = d(x_i, x_1)$, and an easy induction shows that $d(x_i, x_1) = \max_{j \leq i} \{d(x_j, x_1)\}$. Let $\Delta_{S_i} = d(x_r, x_t)$ for some $1 \leq r, t \leq i$. Then, $\Delta_{S_i} \leq d(x_r, x_1) + d(x_1, x_t) \leq 2d(x_i, x_1) = 2R_i$. Since, trivially, $d(x_i, x_1) \leq \Delta_{S_i}$, it follows that

$$\Delta_{S_i}/4 \leq \Delta_{S_i}/2 \leq R_i \leq \Delta_{S_i},$$

which ensures that Invariant 1 holds. Invariant 2 is explicitly enforced by the algorithm. As for Invariant 3, we distinguish again two cases. In case $R_i = R_{i-1}$, then it is immediate to see that the invariant still holds. Otherwise, we have that $R_i = d(x_i, x_1) > 2R_{i-1}$. Now, for any point x_h with $h \leq i$, if $z_{x_h}^{(i)} = z_{x_h}^{(i-1)}$ then $d(x_h, z_{x_h}^{(i)}) < 2\epsilon R_i / (ck)$ by the fact that the invariant holds at step $i-1$ and that $R_{i-1} < R_i$. If instead, $z_{x_h}^{(i)} \neq z_{x_h}^{(i-1)}$ then, by the triangle inequality,

$$\begin{aligned} d(x_h, z_{x_h}^{(i)}) &\leq d(x_h, z_{x_h}^{(i-1)}) + d(z_{x_h}^{(i-1)}, z_{x_h}^{(i)}) \\ &\leq \epsilon(2R_{i-1} + R_i) / (ck) \\ &< 2\epsilon R_i / (ck). \end{aligned}$$

By fixing $c = 32$ we have that the invariants imply that at the end of the algorithm, the distance between any $x_j \in S$ and its reference center in $Z = Z_n$ is

$$d(x_j, z_{x_j}^{(n)}) < 2\epsilon R_n / (ck) = \epsilon \Delta_S / (16k).$$

The stated bound on the radius of the clustering (C, Z) follows since $\Delta_S / (4k) \leq \rho_{S,k}$.

For what concerns the bound on $|Z|$, let τ be the smallest integer such that the radius of an optimal τ -clustering of S is at most $\epsilon \Delta_S / (256k)$. By Theorem 4 it follows that

$$\tau \leq \left\lceil \left(\frac{512k}{\epsilon} \right)^D \right\rceil.$$

We now prove that $|Z| \leq \tau$. If this were not the case, by the pigeonhole principle there would be two distinct points $z_1, z_2 \in Z$ belonging to the same cluster of an optimal τ -clustering. Then, by the triangle inequality and Invariant 1, we would have that

$$d(z_1, z_2) \leq \frac{2\epsilon \Delta_S}{256k} \leq \frac{\epsilon R_n}{32k},$$

thus contradicting Invariant 2. \square

THEOREM 7. *Consider the execution of $\text{STREAMCORESET}(S, k, \epsilon, c)$ for an arbitrary positive constant ϵ and $c = 32$. Then the returned set $T = \bigcup_{z \in Z} D_z$ is a $(1 - \epsilon)$ -coreset for the DMMC problem. The algorithm performs a single pass on the stream and uses a working memory of size $O(|T|)$. If the set S has constant doubling dimension D , then we have that $|T| = O(k(k/\epsilon)^D)$, for the partition matroid, and $|T| = O(k^2(k/\epsilon)^D)$, for the transversal matroid.*

PROOF. Let Z be the final set of centers computed by the algorithm and let $(C = \{C_z : z \in Z\}, Z)$ be the $|Z|$ -clustering of S . Lemma 3 shows that the radius of this clustering is at most $(\epsilon/4)\rho_{S,k}$. Recall that in Subsections 3.1.1, 3.1.2, and 3.1.3 we showed how to construct $(1 - \epsilon)$ -coresets for the various matroid types starting from a any clustering of radius at most $(\epsilon/4)\rho_{S,k}$. We now show that at the end of Algorithm STREAMCORESET each set $D_z \subseteq C_z$ complies with the requirements of those constructions. Then, the fact that $T = \bigcup_{z \in Z} D_z$ is a $(1 - \epsilon)$ -coreset will follow immediately from Theorems 1, 2, and 3.

For the partition matroid, consider an arbitrary delegate set D_z and observe that the algorithm ensures that it is an independent set. Hence, if $|D_z| = k$, then D_z clearly complies with the construction requirements. If instead $|D_z| < k$, consider a point x_j whose associated sequence of centers is $\{z_{x_j}^{(i)} : j \leq i \leq n\}$, and whose reference center is $z_{x_j}^{(n)} = z$. Let A be the category of x_j and let k_A be the cardinality constraint on A . It is easy to see that if x_j is discarded at some

time $i \geq j$ then every $D_{z_{x_j}^{(\ell)}}$, with $\ell \geq i$, contains k_A elements of A . Thus, x_j cannot contribute to an independent set of C_z larger than D_z .

For the transversal matroid, consider as before an arbitrary D_z . Again, if D_z is an independent set of size k , then we are done. Otherwise, let Q be a largest independent set in D_z , (thus $|Q| < k$). We first show that Q is also a largest independent set in C_z . If this were not the case, by the augmentation property there would exist a point $x_j \in C_z - D_z$ such that $Q \cup \{x_j\}$ is an independent set. Let A be one of the categories that can be associated with x_j to provide a matching witnessing the independence of $Q \cup \{x_j\}$. Let also $\{z_{x_j}^{(i)} : j \leq i \leq n\}$ be the sequence of centers associated with x_j (hence, $z_{x_j}^{(n)} = z$). It is easy to see that if x_j is discarded at some time $i \geq j$, then every $D_{z_{x_j}^{(\ell)}}$, with $\ell \geq i$, must contain k elements of A . Therefore, D_z contains k points of A and one such point can be added to Q yielding a larger independent set, which contradicts the maximality of Q . To conclude the proof, we need to show that D_z contains at least $\min\{k, |A \cap C_z|\}$ points for each category A of a point in Q . By contradiction, if $|A \cap D_z| < \min\{k, |A \cap C_z|\}$ for one such category A , there would exist a point $x_j \in A \cap C_z$ which has been discarded by the algorithm at some time $i \geq j$ when A had less than k points in $D_{z_{x_j}^{(i)}}$. This is in contrast with the workings of the algorithm.

The case of the general matroid follows from an easy inductive argument, which shows that at the end of the algorithm each $D_z \subseteq C_z$ is either C_z or an independent set of C_z of size k .

Finally, for what concerns the size of T , Lemma 3 ensures that $|Z| = O\left((k/\varepsilon)^D\right)$. For the case of the partition matroid the claimed bound follows the fact that at any time a delegate set D_z contains at most k points. Instead, for the transversal matroid, the assumption that each point belongs to at most a constant number of categories (say $\gamma \in O(1)$) and the fact that a point x is added to a delegate set D_z only if one of the categories of x has less than k representatives in D_z , imply that at any time $|D_z| < \gamma k^2$ since otherwise D_z would contain an independent set of size k and the algorithm would retain only such an independent set. \square

4.4 Final algorithms

Let S be the input set of n points. In the previous subsections we presented efficient sequential, MR and streaming algorithms to construct $(1 - \varepsilon)$ -coresets $T \subseteq S$ for all variants of the DMMC problem considered in this paper. For all settings and all variants, the final approximation algorithm can be obtained by running a sequential α -approximation algorithm \mathcal{A} on T , which will yield an $(\alpha - \eta)$ -approximate solution, where $\eta = \varepsilon/\alpha$. We remark that while \mathcal{A} may exhibit very high running time, the advantage of the coreset-based approach is that the use of \mathcal{A} is confined on a much smaller subset of the input, retaining a comparable approximation quality while enabling the solution of very large instances. In what follows, we concentrate on the partition and transversal matroids, since for the general matroids no meaningful worst-case time/space bounds can be claimed, even if we believe that our approach can be of practical use even in the general case.

4.4.1 Sequential and streaming algorithms. Theorem 5 shows that a $(1 - \varepsilon)$ -coreset $T \subseteq S$ can be computed in time $O(n|T|)$, where $|T| = O\left(k(k/\varepsilon)^D\right)$ for the partition matroid and $|T| = O\left(k^2(k/\varepsilon)^D\right)$ for the transversal matroid. For the sum-DMMC variant, the local-search based, polynomial-time $(1/2 - \gamma)$ -approximation algorithm of [1] can serve as algorithm \mathcal{A} , yielding (for $\gamma = \varepsilon$) a final $(1/2 - 2\varepsilon)$ -approximation in polynomial time. For all other variants, for which no polynomial-time constant-approximation algorithms are known, we can run an exhaustive search for the best solution on the coreset T , yielding a $(1 - \varepsilon)$ -approximation in time $O\left(n|T| + |T|^k\right)$. We observe that, in both cases, the dependence on the input size n is merely linear and that for small values of k , which is typical for many real-world applications, and for constant ε and D , the overall running time is within feasible bounds even for very large instances.

Table 2. Datasets used in the experimental evaluation, n is the number of elements.

	n	Matroid rank	Matroid type
Wikipedia	5,886,692	100	transversal
Songs	237,698	89	partition

For the streaming setting, Theorem 7 states that a $(1 - \epsilon)$ -coreset $T \subseteq S$ can be computed in one pass with working memory $O(|T|)$, where the sizes of T for the partition and transversal matroids are the same as those claimed before for the sequential setting. Therefore, by running the algorithm of [1] or an exhaustive search on T at the end of the pass, we obtain the same approximation guarantees stated above.

4.4.2 MapReduce algorithms. Theorem 6 states that a $(1 - \epsilon)$ -coreset $T \subseteq S$ can be computed in one MR round with linear total memory and $O(n/\ell)$ local memory, where ℓ is the number of subsets in the partition of S . T has size $O(\ell k(k/\epsilon)^D)$ (resp., $O(\ell k^2(k/\epsilon)^D)$) for the partition (resp., transversal) matroid. By gathering T in one reducer in a second round, we may apply our novel sequential algorithms to extract the final solution. Clearly, this second round requires local memory $O(|T|)$. In order to balance the local-memory requirements between the two rounds we can fix $\ell = \sqrt{n/k}$ (resp., $\ell = \sqrt{n/k^2}$) for the partition (resp., transversal) matroid yielding overall local-memory requirements of $O(\sqrt{nk}(k/\epsilon)^D)$ (resp., $O(\sqrt{nk}(k/\epsilon)^D)$). (Observe that a better choice of ℓ yielding improved bounds on the local memory could be made if D were known.) For what concerns the quality of the solutions, it is easy to see that in this fashion we can obtain the same spectrum of approximations as in the sequential and streaming settings.

5 EXPERIMENTS

In this section, we report on three sets of experiments run on a cluster of 16 machines, each equipped with a 18GB RAM and a 4-core Intel I7 processor, connected by a 10Gbit Ethernet network. The first set (Subsection 5.1) compares the performance of our coreset-based approach with the state of the art in the sequential setting. The other two explore its applicability to very large inputs, focusing on the Streaming (Subsection 5.2) and MapReduce models (Subsection 5.3), respectively. The source code of our implementation is publicly available².

As testbeds, we use two real-world datasets, whose characteristics are summarized in Table 2. One dataset is derived from a recent dump of the English Wikipedia³, comprising 5,886,692 pages. Each Wikipedia page is associated to a number of *categories*, out of 1,102,435 overall categories defined by the Wikipedia users, which naturally induce a transversal matroid. Observe however that due to the sheer number of categories, for any reasonable value of k , any subset of k pages would very likely be an independent set, thus making the matroid constraint immaterial. As a workaround, we applied the *Latent Dirichlet Allocation* model [6] to derive a much smaller set of 100 *topics*, which we use as new categories, together with a probability distribution over these topics for each page. We then assign each page to the most likely topics (probability ≥ 0.1), thus obtaining a transversal matroid of rank 100. Finally, each page is mapped to a 25-dimensional real-valued vector using the *Global Vectors for Word Representation* (GloVe) model [27]. The other dataset is a set of 237,698 songs⁴, each represented by the bag of words of its lyrics and associated to a unique *genre*, out of a total of 16 genres. Since genres define a partition of the dataset, they induce a partition matroid. For each genre g , we fixed the associated cardinality threshold k_g in the matroid as the minimal nonzero value proportional to

²<https://github.com/Cecca/diversity-maximization>

³<https://dumps.wikimedia.org/backup-index.html>, accessed on 2019-07-20

⁴<http://millionsongdataset.com/musixmatch/>

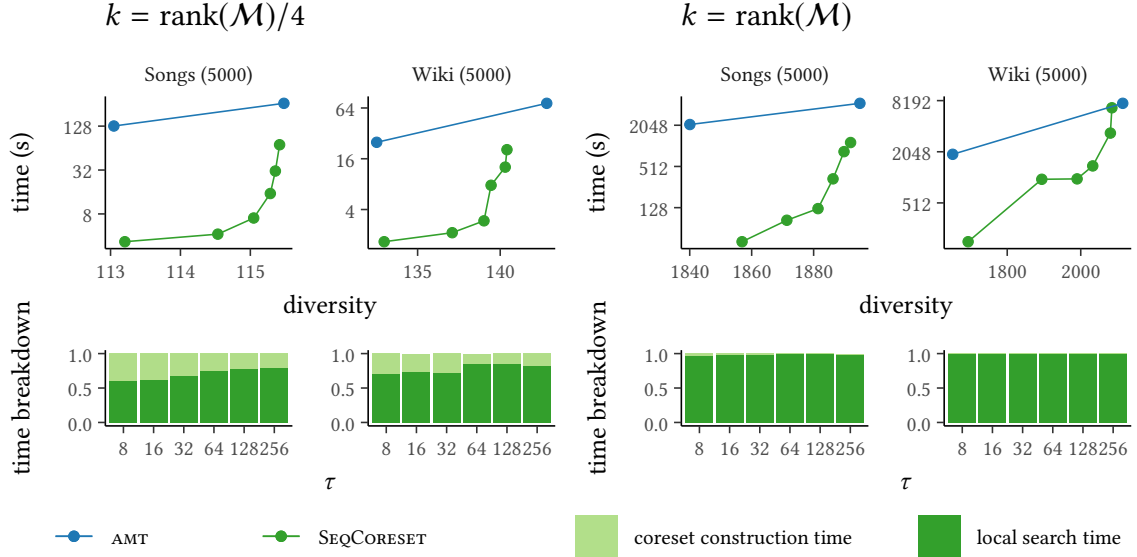


Fig. 1. Time vs. diversity for AMT and SEQCORESET (top), and running time breakdown for SEQCORESET (bottom). In the top plots the y scale is logarithmic, and the τ parameter increases from left to right for the SEQCORESET algorithm. The best performance is towards the bottom-right corner: low running time and high diversity of the solution.

the number of songs of the genre in the dataset, thus obtaining a partition matroid of rank 89. Each page is mapped to a sparse vector, with a coordinate for each of the 5000 words of the dataset’s vocabulary, each set to the number of occurrences of the corresponding word in the lyrics of the song. (Along with the source code we also provide the scripts that we used to preprocess the datasets.)

For both datasets we use as distance the metric version of the *cosine distance* [24]. All results reported are obtained as averages over at least 10 runs. To evaluate the stability of the solution quality with respect to the initial data layout, the dataset is randomly permuted before each run. All figures in the pdf version of the paper can be clicked upon, so to open an online interactive version which provides additional information about the experiments pictured in the figure.

As discussed in Section 4.4, for the sum-DMMC problem we can use the local search algorithm of [1] to compute the final solution⁵, whereas for other diversity measures no approximation algorithm is known but we can run an exhaustive search on a small enough coreset returned by our strategy. For concreteness, we restrict the attention to the sum-DMMC problem. In fact, the benefits of our coreset-based approach are evident for the other problems, where the only alternative (with provable approximation guarantees) to our strategy is an unfeasible exhaustive search on the entire input.

5.1 Sequential setting

We compare our sequential algorithm SEQCORESET, described in Subsection 4.1, against the algorithm in [1], which we refer to as AMT in the following. We recall that AMT runs a local search over the entire input, and features a parameter γ to limit swaps to those providing an improvement of a factor at least $(1 + \gamma)$ to the current solution quality, thus

⁵For any coreset-based algorithm studied in this section, the final output is computed using the local search algorithm with $\gamma = 0$.

exercising a tradeoff between approximation guarantee ($\frac{1}{2} - \gamma$) and running time. Albeit polynomial for constant $\gamma > 0$, AMT is quite expensive, since it may require to check a large number of candidate swaps (possibly quadratic in the input size) where each check entails a call to the independent set oracle, which can be a costly operation, depending on the matroid type. For this reason, in order to keep the running times of AMT within reasonable limits, we tested both algorithms on scaled-down versions of the datasets obtained as samples of 5,000 elements drawn at random from each dataset. However, we want to stress that SEQCORESET is able to process the entire datasets within reasonable time bounds, as will be shown by the MapReduce experiments, where SEQCORESET represents the case of parallelism 1 (see Subsection 5.3).

For what concerns SEQCORESET, rather than using parameter ε , we control the radius of the clustering underlying the coreset construction indirectly through the number of clusters τ to be found by GMM, where larger values of τ yield smaller radii, and thus correspond to smaller values of ε . Specifically, we set τ to powers of two from 8 to 256. For comparison, we ran several instances of AMT with values of gamma in the range $[0, 0.9]$, by increments of 0.001. To avoid overcrowding the plot with too many points and to gauge the performance of our algorithm versus its competitor in terms of time and accuracy, we report the results of two specific runs of AMT: the one with the value γ returning the largest diversity (ties broken in favor of fastest running time); and the one with the value γ returning a solution with a quality just below the lowest one found by SEQCORESET. All other tested runs of AMT featured running times and diversities between those of the two extreme runs. Also, for each dataset we used two values of k , namely $k = \text{rank}(\mathcal{M})$ and $k = \text{rank}(\mathcal{M})/4$, where \mathcal{M} is the associated matroid.

In the top row of Figure 1, we plot the running times of the two algorithms (y -axis) against the diversity yielded by each parameter configuration (x -axis), so to compare the time taken by each algorithm to compute a solution of similar quality. The bottom row of plots reports, for each parameter configuration of SEQCORESET, the breakdown of the running time between the two components of the algorithm: coreset construction (light green) and local search on the coreset (dark green). We observe that our algorithm returns solutions of quality comparable to the ones returned by AMT and, as emphasized by the logarithmic scale, in most cases it runs one or two orders of magnitude faster.

In all experiments, the coreset construction performed by SEQCORESET never dominates the overall running time. For $k = \text{rank}(\mathcal{M})/4$, the coreset construction takes between 50% of the time (for $\tau = 8$) and 20% of the total time (for $\tau = 256$). For $k = \text{rank}(\mathcal{M})$, building the coreset takes negligible time compared to the total time (below 2%). This is a consequence of both the limited input size and the expensive nature of the local search task. In fact, in the context of the MapReduce experiments reported in Subsection 5.3, we also ran SEQCORESET on both the full Songs and Wikipedia datasets, with $\tau = 64$. The results, reported in Figure 3, show that with a much larger dataset the coreset construction task dominates the running time.

Finally, the shape of the curve in Figure 1 representing the performance of our algorithm shows that the parameter τ can be effectively used to control a tradeoff between accuracy and running time, while parameter γ of AMT seems less effective in that respect.

5.2 Streaming setting

We evaluate the performance of our streaming strategy by analyzing the relationship between coreset size, quality of approximation, and running time. As done for the sequential setting, we implemented a variant of STREAMCORESET (Algorithm 2) so to control the number τ of clusters directly, rather than having the value of τ to be determined implicitly as a function of the approximation parameter ε . Controlling the size of the coreset directly also enables an easier comparison with the MapReduce results presented in the next section.

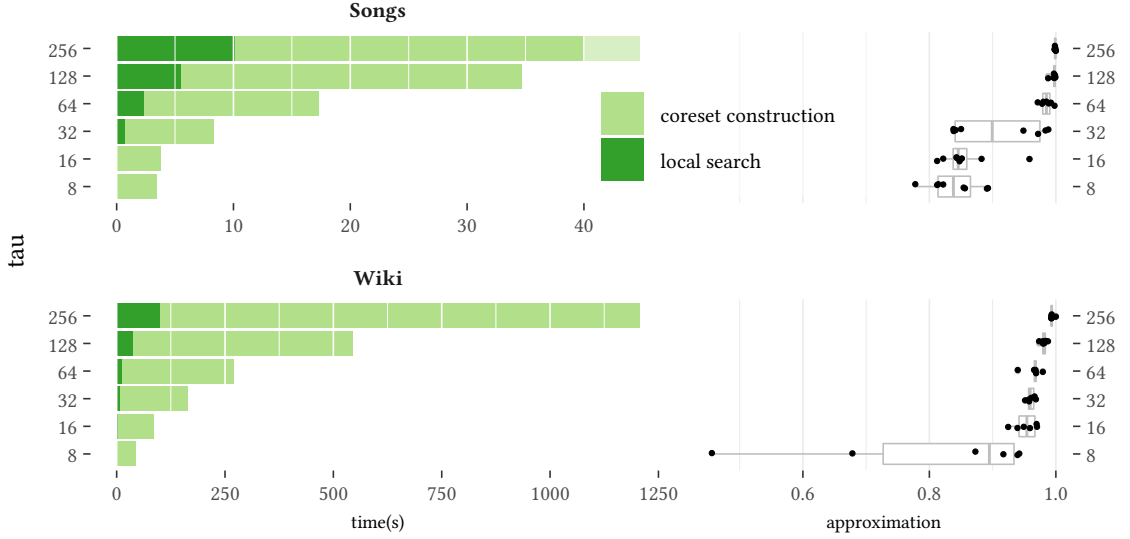


Fig. 2. Performance of the streaming algorithm. For different coreset sizes, the box plots on the right show the diversity attained by different configurations of the algorithm. The bar charts on the left report the breakdown of the overall running time.

To control τ directly, the implemented variant maintains in variable R an estimate of the *radius* of the τ -clustering built so far, rather than an estimate of the diameter of the dataset. For each point of the stream, if it falls within distance $2 \cdot R$ from any of the centers, it is handled using procedure `HANDLE` of `STREAMCORESET`. Otherwise it is added as a new cluster center. As soon as there are more than τ clusters, the algorithm restructures the set of centers as in `STREAMCORESET` and doubles R . This variant is reminiscent of the k -center streaming algorithm by [14], and, using an analysis similar to the one in Subsection 4.3, it can be shown that by setting $\tau = \Theta\left(\frac{k}{\varepsilon}^D\right)$ it returns a $(1 - \varepsilon)$ -coreset.

We run the algorithm on the full Wikipedia and Songs datasets, fixing $k = \text{rank}(\mathcal{M})/4$. As for the coreset size, we run the algorithm so to build $\tau \in \{8, 16, 32, 64, 128, 256\}$ clusters, each containing the appropriate number of delegate points, depending on the matroid type.

We report the results of these experiments in Figure 2. While the bars on the left side of the picture plot the running time for each value of τ , the box-plots on the right report, for each dataset, the distribution of the approximation ratios in the various runs. Such ratios are computed with respect to the best solution ever found by *any run of any algorithm in any setting* (on the same dataset and for the same value of k), with results close to 1 denoting better solutions. We observe that despite the high dimensionality of the dataset, increasing the coreset size has a beneficial effect on the solution quality, at the expense of a roughly linear increase in the running time. Note also that as the size of the coreset increases, all runs tend to give solutions with diversity values which are more concentrated.

We report the results obtained by `STREAMCORESET` with $\tau = 64$ also in Figure 3 (red bars), to compare with the performance of the algorithms in the other settings for a fixed coreset size. From the figure, we observe that, compared to `SEQCORESET`, the streaming algorithm is around 7 times faster on the Songs dataset, and 4 times faster on Wikipedia. This might be surprising, given that `STREAMCORESET` may perform more distance computations than `SEQCORESET` to build the coreset, namely $\Theta(\tau^2 n)$ instead of $\Theta(\tau n)$ in the worst case. However, this worst case rarely happens in

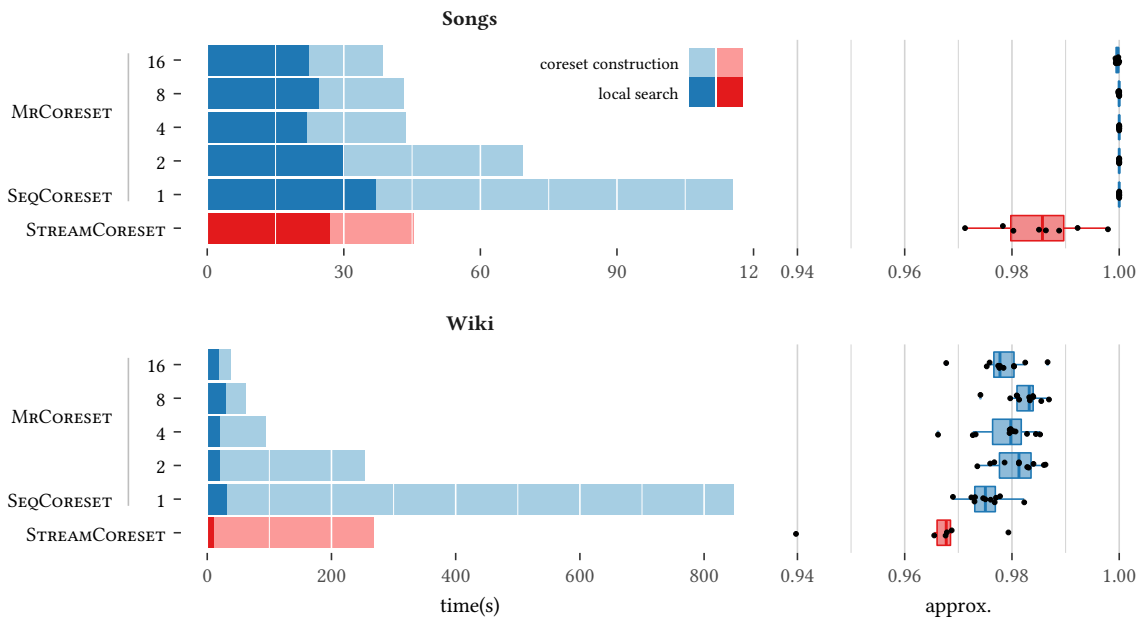


Fig. 3. Comparison between all the algorithms, with $\tau = 64$, on the full datasets. The bars on the left report the running time. For MRCoreSet we report on the performance with 1, 2, 4, 8, and 16 machines. The results with 1 machine correspond to the performance of SeqCoreSet. The portion of the bars with saturated colors reports the time employed for the local search task, the part with dimmed colors reports the coreset construction times. Note that the scale is linear and that, for readability, each dataset has its own time scale. The box plots on the left report the quality of the solution found by each algorithm.

practice. In fact, STREAMCORESET is considerably more cache-efficient, since for each point of the stream it computes the distance with $O(\tau)$ cluster centers, which easily fit into cache. Conversely, SEQCORESET benefits less from data locality, since it iterates τ times over the the entire input, not reusing information already in the cache. However, despite having a worse running time, SEQCORESET yields solutions of better quality than STREAMCORESET, as can be observed in the box plots of Figure 3. This is not surprising: SEQCORESET derives its coreset starting from the 2-approximation GMM clustering algorithm; STREAMCORESET, instead, uses a clustering strategy similar to [14], which is an 8-approximation.

5.3 MapReduce setting

We implemented the MapReduce version of our coreset-based strategy described in Section 4.2 (dubbed MRCoreSet), using the Spark framework [31], and ran it on the full Wikipedia and Songs datasets. As observed at the end of Subsection 5.1 and confirmed by the streaming experiments, with large inputs the main bottleneck of our approach becomes the coreset construction task, where the whole input is involved. However, this task is fully parallelized in MRCoreSet, thus yielding scalable performance. To emphasize this aspect, we fixed a cluster granularity, namely $\tau = 64$, which provides a coreset whose size is small enough to limit the impact of the local search task, but sufficient to embody a good quality solution. Then, we ran MRCoreSet, on $\ell = 1, 2, 4, 8$ and 16 machines, setting the number of clusters

computed by each machine to τ/ℓ , so to extract the final coreset always from a τ -clustering. For both datasets, we fixed $k = \text{rank}(\mathcal{M})/4$.

By the choice of parameters, the final coreset is small enough that it does not make sense to apply SEQCORESET in the second round, as described in Section 4.4.2, since SEQCORESET would not reduce the size of the coreset significantly, while it might obfuscate the evaluation of the impact of parallelism on the solution quality. Thus, we ran AMT with $\gamma = 0$ directly on the coreset computed in the first MapReduce round, hence making the case $\ell = 1$ coincide with SEQCORESET (in fact, in Subsection 5.1 we used MRCORESET with $\ell = 1$ as implementation of SEQCORESET).

The results of our experiments are reported in the four plots of Figure 3. The left plots show the running times broken down into coreset construction and local search time, under the different levels of parallelism, while the right plots are box-plots for the approximation ratios attained by MRCORESET, computed as described for the streaming setting. The figure reports also the performance of SEQCORESET and STREAMCORESET with $\tau = 64$, so to allow a full comparison of all our algorithms. Indeed, the bars of these two algorithms show clearly that, on large inputs, the majority of the work goes into the coreset construction, and the bars corresponding to larger levels of parallelism show that such construction scales well. This scalability effect is more evident on the Wikipedia dataset which, due to its larger size, is able to take better advantage of the available parallelism. Not surprisingly, on this larger dataset the coreset construction scales more than linearly. This is due to the fact that the complexity of the clustering required to compute the local coresets is roughly inversely proportional to ℓ^2 , since τ/ℓ clusters must be computed in subsets of the dataset S of size $|S|/\ell$. Importantly, we have that parallelizing the coreset construction, which may in theory yield coresets of worse quality, does not seem to affect the quality of the final solution significantly.

As for a comparison with the streaming algorithm, consider the red bars of the figure. For the Songs dataset, which is not very large, the performance of the streaming algorithm (which employs a single processor) is comparable with the performance of the MapReduce algorithm with 16 processor. This is due to the overhead of communication and synchronization between machines inherent in the Spark platform. On the Wikipedia dataset, on the other hand, the benefits of parallelization emerge more evidently, and the running time of the streaming algorithm is already matched with parallelism 2 but with a better solution quality.

Finally, we remark that the high complexity of AMT makes it impractical for such large inputs, thus ruling out a direct comparison with MRCORESET. However, by pairing the results of the comparison between AMT and SEQCORESET from subsection 5.1 and between SEQCORESET and MRCORESET from this subsection, we can infer that the latter holds the promise to provide solutions of similar quality as AMT but in a scalable fashion.

6 CONCLUSIONS

Coreset-based strategies provide an effective way of processing massive datasets by building a succinct summary of the input dataset S , which can then be analyzed with a (possibly computationally-intensive) sequential algorithm of choice. In this paper, we have seen how to leverage coresets to build fast sequential, MapReduce and Streaming algorithms for diversity maximization under matroid constraints for a wide family of DMMC variants. For the sum-DMMC variant, our algorithms feature an accuracy which can be made arbitrarily close to that of the state-of-the-art (computationally expensive) sequential algorithms, while for all other variants they provide the the first viable $(1 - \epsilon)$ -approximate solutions in all of the aforementioned computational frameworks, by confining exhaustive search to the coreset, whose size is independent of $|S|$. For the important cases of partition and transversal matroids, and under reasonable assumptions on the dimensionality of the dataset, the algorithms require work linear in the input size, and, as demonstrated by experiments conducted on real world datasets, their performance can be orders of magnitude

faster than that of existing algorithms. Moreover, the Streaming and MapReduce versions of the algorithms can be effectively employed in big data scenarios where solutions of very large instances are sought.

In our algorithms the coreset size, which cannot grow too large to avoid incurring large overheads in the extraction of the final solution, exhibits an exponential dependency on the doubling dimension D of the input dataset. A challenging, yet important open problem is to provide a tighter analysis of our algorithms, if at all possible, or to develop improved strategies that retain efficiency even for large values of D .

Recall that for the transversal matroid we made the assumption that each point belongs to a constant number of categories. In fact, our sequential and MapReduce algorithms can be easily modified to work without this assumption, while maintaining the same quality. Another interesting open problem is to modify our streaming algorithm so that the assumption can be lifted.

Finally, we wish to remark that our coreset-based approach for diversity maximization under matroid constraints does not yield efficient algorithms for another well-studied variant (dubbed *min-DMMC*), where the diversity function of a subset of points is defined as the minimum distance between these points. Devising solutions for this important variant remains an interesting open problem.

REFERENCES

- [1] Zeinab Abbassi, Vahab S. Mirrokni, and Mayur Thakur. 2013. Diversity maximization under matroid constraints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 32–40. <https://doi.org/10.1145/2487575.2487636>
- [2] Marcel R. Ackermann, Johannes Blömer, and Christian Sohler. 2010. Clustering for metric and nonmetric distance measures. *ACM Trans. Algorithms* 6, 4 (2010), 59:1–59:26. <https://doi.org/10.1145/1824777.1824779>
- [3] Pankaj K. Agarwal, Sarel Har-Peled, and Kasturi R. Varadarajan. 2005. Geometric approximation via coresets. *J. of Combinatorial and Computational Geometry* 52 (2005), 1–30.
- [4] Sepideh Aghamolaei, Majid Farhadi, and Hamid Zarrabi-Zadeh. 2015. Diversity Maximization via Composable Coresets. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG)*.
- [5] Aditya Bhaskara, Mehrdad Ghadiri, Vahab S. Mirrokni, and Ola Svensson. 2016. Linear Relaxations for Finding Diverse Elements in Metric Spaces. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS)*. 4098–4106. <http://papers.nips.cc/paper/6500-linear-relaxations-for-finding-diverse-elements-in-metric-spaces>
- [6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. of Machine Learning Research* 3 (2003), 993–1022. <http://jmlr.org/papers/v3/blei03a.html>
- [7] Michele Borassi, Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. 2019. Better Sliding Window Algorithms to Maximize Subadditive and Diversity Objectives. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. ACM, 254–268. <https://doi.org/10.1145/3294052.3319701>
- [8] Allan Borodin, Hyun Chul Lee, and Yuli Ye. 2012. Max-Sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. 155–166. <https://doi.org/10.1145/2213556.2213580>
- [9] Matteo Ceccarello, Andrea Pietracaprina, and Geppino Pucci. 2018. Fast Coreset-based Diversity Maximization under Matroid Constraints. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 81–89. <https://doi.org/10.1145/3159652.3159719>
- [10] Matteo Ceccarello, Andrea Pietracaprina, Geppino Pucci, and Eli Upfal. 2017. MapReduce and Streaming Algorithms for Diversity Maximization in Metric Spaces of Bounded Doubling Dimension. *Proc. of the VLDB Endowment* 10, 5 (2017), 469–480. <http://www.vldb.org/pvldb/vol10/p469-ceccarello.pdf>
- [11] Alfonso Cevallos, Friedrich Eisenbrand, and Sarah Morell. 2018. Diversity Maximization in Doubling Metrics. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC) (LIPIcs)*, Vol. 123. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 33:1–33:12. <https://doi.org/10.4230/LIPIcs.ISAAC.2018.33>
- [12] Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. 2017. Local Search for Max-Sum Diversification. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 130–142. <https://doi.org/10.1137/1.9781611974782.9>
- [13] Barun Chandra and Magnús M. Halldórsson. 2001. Approximation Algorithms for Dispersion Problems. *J. Algorithms* 38, 2 (2001), 438–465. <https://doi.org/10.1006/jagm.2000.1145>
- [14] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. 2004. Incremental Clustering and Dynamic Information Retrieval. *SIAM J. on Computing* 33, 6 (2004), 1417–1440. <https://doi.org/10.1137/S0097539702418498>
- [15] Richard Cole and Lee-Ad Gottlieb. 2006. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*. 574–583. <https://doi.org/10.1145/1132516.1132599>

- [16] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI)*. 137–150. <http://www.usenix.org/events/osdi04/tech/dean.html>
- [17] Alessandro Epasto, Vahab S. Mirrokni, and Morteza Zadimoghaddam. 2019. Scalable Diversity Maximization via Small-size Composable Core-sets (Brief Announcement). In *Proceedings of the 31st ACM on Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, 41–42. <https://doi.org/10.1145/3323165.3323172>
- [18] Teofilo F. Gonzalez. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science* 38 (1985), 293–306. [https://doi.org/10.1016/0304-3975\(85\)90224-5](https://doi.org/10.1016/0304-3975(85)90224-5)
- [19] Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. 2014. Efficient Classification for Metric Data. *IEEE Trans. Information Theory* 60, 9 (2014), 5750–5759. <https://doi.org/10.1109/TIT.2014.2339840>
- [20] Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. 1998. Computing on data streams. In *Proceedings of a DIMACS Workshop on External Memory Algorithms*. 107–118.
- [21] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. 2014. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. 100–108. <https://doi.org/10.1145/2594538.2594560>
- [22] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. 2010. A Model of Computation for MapReduce. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 938–948. <https://doi.org/10.1137/1.9781611973075.76>
- [23] Goran Konjevod, Andréa W. Richa, and Donglin Xia. 2008. Dynamic Routing and Location Services in Metrics of Low Doubling Dimension. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC)*. 379–393. https://doi.org/10.1007/978-3-540-87779-0_26
- [24] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. 2014. *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press.
- [25] Michael Masin and Yossi Bukchin. 2008. Diversity Maximization Approach for Multiobjective Optimization. *Operations Research* 56, 2 (2008), 411–424. <https://doi.org/10.1287/opre.1070.0413>
- [26] James G. Oxley. 2006. *Matroid Theory*. Oxford University Press.
- [27] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 18th Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, 1532–1543. <https://www.aclweb.org/anthology/D14-1162/>
- [28] Andrea Pietracaprina, Geppino Pucci, Matteo Riondato, Francesco Silvestri, and Eli Upfal. 2012. Space-round tradeoffs for MapReduce computations. In *Proceedings of the 25th International Conference on Supercomputing (ICS)*. ACM, 235–244. <https://doi.org/10.1145/2304576.2304607>
- [29] Y.C. Wu. 2013. Active Learning Based on Diversity Maximization. *Applied Mechanics and Materials* 347, 10 (2013), 2548–2552. <https://doi.org/10.4028/www.scientific.net/AMM.347-350.2548>
- [30] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G. Hauptmann. 2015. Multi-Class Active Learning by Uncertainty Sampling with Diversity Maximization. *International J. of Computer Vision* 113, 2 (2015), 113–127. <https://doi.org/10.1007/s11263-014-0781-x>
- [31] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*. USENIX Association. <https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets>