6-2021

Minimum coresets for maxima representation of multidimensional data

Yanhao WANG

Michael MATHIOUDAKIS

Yuchen LI
Singapore Management University, yuchenli@smu.edu.sg

Kian-Lee TAN

Citation

1

Minimum Coresets for Maxima Representation of Multidimensional Data

Yanhao Wang
University of Helsinki
Helsinki, Finland
yanhao.wang@helsinki.fi

Michael Mathioudakis
University of Helsinki
Helsinki, Finland
michael.mathioudakis@helsinki.fi

Yuchen Li
Singapore Management University
Singapore
yuchenli@smu.edu.sg

Kian-Lee Tan
National University of Singapore
Singapore
tankl@comp.nus.edu.sg

ABSTRACT

Coresets are succinct summaries of large datasets such that, for a given problem, the solution obtained from a coreset is provably competitive with the solution obtained from the full dataset. As such, coreset-based data summarization techniques have been successfully applied to various problems, e.g., geometric optimization, clustering, and approximate query processing, for scaling them up to massive data. In this paper, we study coresets for the maxima representation of multidimensional data: Given a set P of points in $\mathbb{R}^d$, where d is a small constant, and an error parameter $\varepsilon \in (0, 1)$, a subset $Q \subseteq P$ is an ε-coreset for the maxima representation of P iff the maximum of Q is an ε-approximation of the maximum of P for any vector $u \in \mathbb{R}^d$, where the maximum is taken over the inner products between the set of points (P or Q) and u. We define a novel minimum ε-coreset problem that asks for an ε-coreset of the smallest size for the maxima representation of a point set. For the two-dimensional case, we develop an optimal polynomial-time algorithm for the minimum ε-coreset problem by transforming it into the shortest-cycle problem in a directed graph. Then, we prove that this problem is NP-hard in three or higher dimensions and present polynomial-time approximation algorithms in an arbitrary fixed dimension. Finally, we provide extensive experimental results on both real and synthetic datasets to demonstrate the superior performance of our proposed algorithms.

CCS CONCEPTS

• **Theory of computation → Data structures and algorithms for data management**; **Computational geometry**.

KEYWORDS

Coreset; maxima representation; ε-kernel; convex hull; regret minimizing set

ACM Reference Format:
Yanhao Wang, Michael Mathioudakis, Yuchen Li, and Kian-Lee Tan. 2021. Minimum Coresets for Maxima Representation of Multidimensional Data. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3452021.3458322

1 INTRODUCTION

Scaling data analysis tasks to large-scale datasets is a key challenge in big data processing. A standard approach to addressing the scalability issue is *data summarization*, which aims to reduce massive datasets to concise summaries of much smaller sizes. As such, computationally expensive algorithms can be scaled to massive datasets by restricting the computation to summaries only. One established paradigm for data summarization that has attracted much attention recently is *coresets*. Specifically, a coreset is a small subset of a dataset such that the solution of a given problem computed from the coreset can approximate the solution of the same problem computed from the whole dataset with a provable guarantee. Coreset-based data summarization techniques have been considered in many real-world problems, ranging from shape fitting [1, 41, 45], to clustering [11, 27], to regression [16], and even to neural networks [33].

Motivated by numerous applications in computational geometry, machine learning, databases, and data mining, e.g., [1, 3, 14, 35, 43–45], there has been considerable work on computing coresets for *extent measures* of a set P of n points in $\mathbb{R}^d$. Here the term "extent measure" typically refers to certain statistics (e.g., diameter and width) that capture the range covered by either the point set P itself or some geometric shape (e.g., sphere, convex hull, bounding box, cylinder, etc.) enclosing P. In this paper, we consider the problem of computing coresets for a specific extent measure of a point set P, namely maxima representation. Given a vector $u \in \mathbb{R}^d$, we define the extreme point $\varphi(P, u)$ of P w.r.t. u as the point having the largest inner product with u among all points in P – i.e., $\varphi(P, u) = \arg\max_{p \in P} \langle p, u \rangle$. The maximum $\omega(P, u)$ of P w.r.t. u is defined by the inner product of the extreme point and u accordingly – i.e., $\omega(P, u) = \max_{p \in P} \langle p, u \rangle$. Then, we formally define the notion of *coreset* for the maxima representation of P as follows: a subset $Q \subseteq P$ is an ε-coreset of P iff the maximum of Q is within an ε-approximation of the maximum of P for every vector $u \in \mathbb{R}^d$ – i.e., $\omega(Q, u) \geq (1 - \varepsilon) \cdot \omega(P, u)$ for all $u \in \mathbb{R}^d$.

PODS '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8381-3/21/06…$15.00
https://doi.org/10.1145/3452021.3458322

We establish the theoretical connections between ε-coresets for maxima representation and several well-known notions, e.g., ε-kernels [1], convex hulls [13, 38], and regret minimizing sets [35, 42] (see Sections 2 and 3 for more details). In light of these connections, the ε-coresets for maxima representation can find applications in various real-world problems in which previous methods based on similar notions are used. For example, geometric optimization problems that can be approximated by ε-kernels, including diameter, minimum enclosing cylinder, minimum bounding box, convex hull volume, to just name a few, can also be approximated by ε-coresets for maxima representation. As another example, ε-coresets for maxima representation can find applications in several database problems, including approximate top-k queries with linear ranking functions [32, 44] and representative skyline queries [9, 35].

In all the above applications, smaller coresets are preferred to larger ones, as smaller coresets lead to higher efficiency in optimization, lower overhead in query processing, or more compact representation in data reduction. Therefore, how to find coresets of smaller sizes is a problem of great interest. Although there have been many methods [1, 7, 8, 18, 21, 45] to compute ε-coresets of size $O\big(\frac{1}{\varepsilon^{(d-1)/2}}\big)$, which is optimal in the worst case, much smaller coresets may exist for a specific point set. However, the problem of finding the smallest possible coreset of a given point set has not been explored yet. To address this problem, we formulate a novel *Minimum ε-Coreset* (MC) problem in this paper: Given a point set $P \subset \mathbb{R}^d$ and an error parameter $\varepsilon \in (0, 1)$, find the smallest ε-coreset Q^*_ε for the maxima representation of P.

1.1 Prior Work

There has been a line of work on *ε-kernel computation* [1, 4, 7, 8, 18–21, 30, 45, 46]. The notion of ε-kernels was first introduced by Agarwal et al. [1]. They proved the existence of ε-kernels of size $O\big(\frac{1}{\varepsilon^{(d-1)/2}}\big)$ for an arbitrary point set in $\mathbb{R}^d$, which is optimal in the worst case. Then, they proposed an algorithm to compute such an ε-kernel in $O\big(n + \frac{1}{\varepsilon^{3(d-1)/2}}\big)$ time. Chan [18] proposed two improved algorithms for ε-kernel construction running in $O\big(n + \frac{1}{\varepsilon^{d-3/2}}\big)$ and $O\big((n+\frac{1}{\varepsilon^{d-2}})\cdot\log\frac{1}{\varepsilon}\big)$ time, respectively. Arya and Chan [7] developed an algorithm for computing ε-kernels in $O\big(n+\sqrt{n}\cdot\frac{1}{\varepsilon^{d/2}}\big)$ time using discrete Voronoi diagrams. Arya et al. [8] improved the time complexity of ε-kernel construction to $O\big(n\cdot\log\frac{1}{\varepsilon}+\frac{1}{\varepsilon^{(d-1)/2+c}}\big)$ for some $c = O(1)$. Chan [21] proposed an $O\big((\frac{n}{\varepsilon^{1/2}} + \frac{1}{\varepsilon^{d/2+1}})\cdot\log^c\frac{1}{\varepsilon}\big)$-time algorithm for ε-kernel computation using Chebyshev polynomials. The problem of computing ε-kernels was also considered in different settings. Streaming algorithms for ε-kernel construction that processed all points in only one pass and used sublinear update time and space were studied in [1, 5, 7, 18, 46]. Maintaining ε-kernels in dynamic settings where points could be inserted, deleted, and updated was considered in [1, 4, 19, 20]. Huang et al. [30] investigated the problem of computing ε-kernels on noisy and uncertain data. Nevertheless, the aforementioned works were limited to theoretical analysis and did not consider how to compute ε-kernels efficiently in practice. Yu et al. [45] implemented an approximate nearest neighbor (ANN) based algorithm for ε-kernel computation based on the algorithms in [1, 18]. This ANN-based implementation was regarded as the standard approach to computing ε-kernels and widely used in many applications [43, 44] (and also compared to as a baseline in our experiments). Note that the above methods for *ε-kernel computation* provide ε-kernels of size $O\big(\frac{1}{\varepsilon^{(d-1)/2}}\big)$ and, as will be shown in Section 2, they can also be used to provide ε-coresets of the same size. However, they do not provide any guarantee on the minimality of ε-coresets.

Another notion related to our MC problem is ε-hulls [13–15]. Given a point set $P \subset \mathbb{R}^d$, a subset $Q \subseteq P$ is called an ε-hull of P iff any point in P is either in the convex hull $\mathcal{CH}(Q)$ of Q or within distance ε from $\mathcal{CH}(Q)$. The difference between ε-hulls and ε-coresets in terms of convex hull approximation is that the former is defined by additive errors but the latter is defined by multiplicative errors. In a seminal work, Bentley et al. [13] proposed an algorithm for computing ε-hulls of size $O\big(\frac{1}{\varepsilon^{d-1}}\big)$. Blum et al. [15] first studied the "minimization" of ε-hulls and proposed an approximation algorithm to compute an ε-hull of size $O(d\cdot\texttt{OPT}\cdot\log\texttt{OPT})$ for any point set P, where $\texttt{OPT}$ is the size of the smallest ε-hull of P. Blum et al. [14] further investigated the minimum ε-hull problem in data streams. However, due to the differences in definitions, algorithms for the minimum ε-hull problem cannot be used for our MC problem and vice versa. In fact, the reduction between both minimization problems is still open [14].

Finally, our MC problem is relevant to the regret minimizing set (RMS) problem [35], which is a restricted version of MC where all points and vectors are nonnegative. The RMS problem was first introduced by Nanongkai et al. [35] for finding the k-representative skyline. Since the seminal work by Nanongkai et al., different approximation and heuristic algorithms were proposed for RMS, e.g., [3, 9, 17, 31, 35, 37, 39, 43]. Interested readers can refer to [42] for a survey of algorithmic techniques for RMS. However, most existing algorithms for RMS cannot be used for MC because they rely on the non-negativity of points and vectors for solution computation. To the best of our knowledge, the algorithms in [3, 9, 31] that reduce RMS to the set cover problem are the only ones that can be adapted to MC. Our SCMC algorithm in this paper is an adaptation of the algorithms in [3, 9] that takes into account all points and vectors other than the nonnegative ones only.

1.2 Our Contributions

The main contributions of this paper are summarized as follows:

- In Section 2, we introduce the notion of ε-coreset for maxima representation of multidimensional data and indicate its connections with the ε-kernel and the convex hull. Then, we define the minimum ε-coreset (MC) problem.
- In Section 3, we prove that finding the minimum ε-coreset of a point set $P \subset \mathbb{R}^d$ is NP-hard for any constant $d \geq 3$ by the reduction from the *regret minimizing set* (RMS) problem [17], a known NP-hard problem. Then, in Section 4, we introduce the background on *inner-product Voronoi diagrams* [10] and provide a geometric interpretation of the MC problem based on Voronoi diagrams, on which our proposed algorithms will be further built.
- In Section 5, we propose an $O(n^3)$-time optimal algorithm OptMC for MC on a point set $P \subset \mathbb{R}^2$, where $n = |P|$. OptMC utilizes the Voronoi diagram-based geometric interpretation of MC to transform the problem of computing the optimal

solution of MC into finding the shortest cycle of a directed graph. We also note that OptMC runs much faster than the worst-case time of $O(n^3)$ for small values of ε.

- In Section 6, we develop two polynomial-time approximation algorithms for MC in an arbitrary fixed dimension. The high-level ideas of both algorithms are transforming MC into simpler covering problems, followed by computing solutions of MC using the greedy algorithm. We first propose the DSMC algorithm, which simplifies MC as a dominating set problem on a weighted and directed graph that encapsulates the information about the loss of representing one point with another in terms of maxima representation. We prove that DSMC provides valid ε-coresets in polynomial time while achieving an approximation ratio of at most $O(\frac{\xi}{d})$, where ξ is the number of extreme points in P. Furthermore, we find that the transformation from RMS to the set cover problem based on the notion of δ-nets [28] proposed in [3, 9] can also work for MC. Thus, we propose the SCMC algorithm by slightly adapting the results in [3, 9]. Theoretically, SCMC runs in $O(\frac{n}{\varepsilon^{d-1}})$ time and provides an ε-coreset of size $O(d \cdot \mathtt{OPT}_{\varepsilon/2} \cdot \log \frac{1}{\varepsilon})$, where $\mathtt{OPT}_{\varepsilon/2}$ is the size of the smallest $\frac{\varepsilon}{2}$-coreset.
- In Section 7, we conduct extensive experiments on real and synthetic datasets to evaluate the performance of our proposed algorithms. The results show that OptMC provides optimal solutions for MC on two-dimensional data in reasonable time. In terms of effectiveness, both DSMC and SCMC provide solutions of significantly higher quality (i.e., smaller coresets) than the ANN-based algorithm on all datasets with dimensions ranging from 2 to 10. In terms of efficiency, SCMC is close to or slightly worse than ANN but DSMC runs much faster than both ANN and SCMC, particularly so for smaller values of ε.

2 PRELIMINARIES

Let P be a set of n points in $\mathbb{R}^d$ and $p = (p_1, \dots, p_d)$ be a point in P. We assume that the dimension d is a small constant throughout this paper. In addition, we consider that P is in general linear position. For any vector[1] $u \in \mathbb{S}^{d-1}$, $\varphi(P, u) = \arg\max_{p \in P} \langle p, u \rangle$ is the extreme point of P for u and $\omega(P, u) = \max_{p \in P} \langle p, u \rangle$ is the maximum of P for u, where $\langle \cdot, \cdot \rangle$ denotes the inner product function. Moreover, we define the α-fatness of a point set as follows:

Definition 2.1 (α-fatness). A point set P is α-fat iff $\omega(P, u) > 0$ for any $u \in \mathbb{S}^{d-1}$ and

$$\min_{u, v \in \mathbb{S}^{d-1}} \frac{\omega(P, u)}{\omega(P, v)} \geq \alpha$$

According to the analysis in [1], there always exists an affine transformation T from an arbitrary point set P in $\mathbb{R}^d$, which is in general linear position, to an α_d-fat point set $\widetilde{P} = T(P)$ in $[-1, 1]^d$, where α_d is a constant depending only on d. In what follows, we will assume that the point set P has been transformed to be α-fat in $[-1, 1]^d$ for some constant $\alpha \in (0, 1)$.

Given a point set P and an error parameter $\varepsilon \in (0, 1)$, an ε-coreset for the maxima representation of P is defined as a subset $Q \subseteq P$ that approximates the maximum of P within a relative error ε for every vector $u \in \mathbb{S}^{d-1}$. Formally,

Definition 2.2 (ε-coreset). For a point set $P \subset \mathbb{R}^d$ and an error parameter $\varepsilon \in (0, 1)$, a subset $Q \subseteq P$ is an ε-coreset for the maxima representation of P iff $\omega(Q, u) \geq (1 - \varepsilon) \cdot \omega(P, u)$ for any $u \in \mathbb{S}^{d-1}$.

For ease of analysis, we denote the largest loss in the maxima of Q w.r.t. P by $l(Q, P) = \max_{u \in \mathbb{S}^{d-1}} 1 - \frac{\omega(Q,u)}{\omega(P,u)}$. Obviously, the condition in the definition of ε-coreset can be equivalently expressed by $l(Q, P) \leq \varepsilon$. Note that P will be dropped from the loss function l when the context is clear.

One important notion closely related to the ε-coreset for maxima representation is the ε-coreset for directional width, commonly known as *ε-kernel* [1]. Here the directional width of a point set P for a vector $u \in \mathbb{S}^{d-1}$ is defined by

$$\overline{\omega}(P, u) = \max_{p \in P} \langle p, u \rangle - \min_{q \in P} \langle q, u \rangle = \omega(P, u) + \omega(P, -u)$$

A subset of points is an ε-kernel of P iff its directional width is within an ε-approximation of the directional width of P for every vector $u \in \mathbb{S}^{d-1}$. The ε-coreset for maxima representation can be seen as a stronger version of the ε-kernel, since the maxima of two opposing directions u and $-u$, instead of their sum only, are both constrained. The connection between both notions is made formal and explicit in the following theorem.

Theorem 2.3. *Let P be an α-fat point set in $[-1, 1]^d$. If Q is an ε-coreset for the maxima representation of P, then Q is an ε-kernel of P. Conversely, if Q is an $\frac{\alpha\varepsilon}{1+\alpha}$-kernel of P, then Q is an ε-coreset for the maxima representation of P.*

Proof. If Q is an ε-coreset of P, then $\omega(Q, u) \geq (1 - \varepsilon) \cdot \omega(P, u)$ and $\omega(Q, -u) \geq (1 - \varepsilon) \cdot \omega(P, -u)$ for each $u \in \mathbb{S}^{d-1}$. So we have:

$$\begin{aligned} \overline{\omega}(Q, u) &= \omega(Q, u) + \omega(Q, -u) \\ &\geq (1 - \varepsilon) \cdot \big(\omega(P, u) + \omega(P, -u)\big) = (1 - \varepsilon) \cdot \overline{\omega}(P, u) \end{aligned}$$

and thus Q is an ε-kernel of P.

Conversely, if Q is an $\frac{\alpha\varepsilon}{1+\alpha}$-kernel of P, then $\overline{\omega}(Q, u) \geq (1 - \frac{\alpha\varepsilon}{1+\alpha}) \cdot \overline{\omega}(P, u)$ for each $u \in \mathbb{S}^{d-1}$. So we have:

$$\big(\omega(P, u) + \omega(P, -u)\big) - \big(\omega(Q, u) + \omega(Q, -u)\big) \leq \frac{\alpha\varepsilon}{1 + \alpha} \cdot \overline{\omega}(P, u) \quad (1)$$

Furthermore, because $\omega(P, -u) - \omega(Q, -u) \geq 0$ for $Q \subseteq P$ and $\frac{\omega(P,u)}{\omega(P,-u)} \geq \alpha \Leftrightarrow \overline{\omega}(P, u) \leq \frac{1+\alpha}{\alpha} \cdot \omega(P, u)$ from the α-fatness of P, Eq. 1 is reduced to:

$$\omega(P, u) - \omega(Q, u) \leq \frac{\alpha\varepsilon}{1 + \alpha} \cdot \frac{1 + \alpha}{\alpha} \cdot \omega(P, u) = \varepsilon \cdot \omega(P, u)$$

and we conclude the proof accordingly. □

Another notion related to the ε-coreset for maxima representation is the *convex hull* [38]. The convex hull $\mathcal{CH}(P)$ of a point set P is the smallest convex set that contains all points in P. The ε-coreset for maxima representation can be regarded as an approximate convex hull since it approximately contains P with bounded multiplicative errors, as shown in the following theorem.

[1] We normalize all (nonzero) vectors in $\mathbb{R}^d$ to the set of unit vectors on the $(d-1)$-dimensional unit sphere $\mathbb{S}^{d-1}$ since the relative inner products of points in P with each vector are norm-invariant. In addition, the case of the zero vector is trivial as $\langle p, 0 \rangle = 0$ for any $p \in \mathbb{R}^d$ and thus ignored in our problem.

Theorem 2.4. *Let Q be an ε-coreset for the maxima representation of P. For any point $p \in P$, it holds that p is either contained in the convex hull $\mathcal{CH}(Q)$ of Q or within distance $\varepsilon \cdot \|p\|^2$ from $\mathcal{CH}(Q)$, where $\|p\|$ is the Euclidean norm of p.*

Theorem 2.4 is a slight variant of the results for approximating convex hulls with ε-kernels in [2] and ε-hulls in [14].

As discussed in Section 1.1, existing algorithms provide ε-kernels of size $O\big(\frac{1}{\varepsilon^{(d-1)/2}}\big)$. Since $\alpha = O(1)$, Theorem 2.3 implies that any of these algorithms can also provide an ε-coreset of size $O\big(\frac{1}{\varepsilon^{(d-1)/2}}\big)$. Note however that, for a specific point set, there may exist ε-coresets of much smaller sizes, but none of the existing algorithms for ε-kernel computation can provide any guarantee on the minimality of ε-kernels (as well as ε-coresets). In this paper, we study a novel minimization version of coreset computation, which aims to find the smallest possible ε-coreset among all valid ε-coresets for the compactness of data representation. We formally define this problem as Minimum ε-Coreset (MC) in the following:

Definition 2.5 (Minimum ε-Coreset). Given a point set $P \subset \mathbb{R}^d$ and an error parameter $\varepsilon \in (0,1)$, find the smallest ε-coreset Q^*_ε for the maxima representation of P. Formally,

$$Q^*_\varepsilon = \operatorname*{arg\,min}_{Q \subseteq P : l(Q) \le \varepsilon} |Q|$$

Note that there is a dual formulation of MC– i.e., given a size constraint $r \in \mathbb{Z}^+$, find a subset Q of size at most r with the smallest $l(Q)$. One can trivially adapt an algorithm $\mathcal{A}$ for MC to solve the dual problem: By performing a binary search on ε and computing a solution of MC using $\mathcal{A}$ for each value of ε, one can find the minimum value of ε such that the size of the ε-coreset is at most r. If $\mathcal{A}$ is optimal for MC, the adapted algorithm is also optimal for the dual problem at the expense of an additional logarithmic factor (for binary search) in running time. Considering the equivalence of both formulations, we will focus on MC in Definition 2.5.

3 HARDNESS

In this section, we show that the minimum ε-coreset (MC) problem is NP-hard in $\mathbb{R}^d$ for any constant $d \ge 3$. Obviously, MC in $\mathbb{R}^1$ is trivial since the two extreme points with the minimum and maximum values are always an optimal solution, which can be computed in $O(n)$ time. In case of $d = 2$, we will show that MC in $\mathbb{R}^2$ is in P by presenting an optimal polynomial-time algorithm in Section 5. Next, we prove that MC in $\mathbb{R}^3$ is NP-hard by reducing from the regret-minimizing set (RMS) problem in $\mathbb{R}^3$, which is known to be NP-hard [17], to MC in $\mathbb{R}^3$.

Theorem 3.1. *The minimum ε-coreset problem is NP-hard in $\mathbb{R}^3$.*

Proof. The decision version of MC is formulated as follows: given a set P of points in $\mathbb{R}^3$, an error parameter $\varepsilon \in (0,1)$, and a positive integer $r \in \mathbb{Z}^+$, determine whether there exists a subset $Q \subseteq P$ of size r such that $l(Q) \le \varepsilon$. For a set P^+ of points in the positive orthant $\mathbb{R}^3_+$, an error parameter $\varepsilon \in (0,1)$, and a positive integer $r \in \mathbb{Z}^+$, the regret-minimizing set (RMS) problem asks whether there exists a subset $Q^+ \subseteq P^+$ of size r such that $\omega(Q^+, u) \ge (1-\varepsilon) \cdot \omega(P^+, u)$ for any positive vector $u \in \mathbb{S}^2_+$. Intuitively, RMS is a restricted version of MC where both points and vectors are positive. We restrict $P^+ \subset [0,1]^3$ because of the scale-invariance of RMS [35]. Similar to the definition of $l(Q)$, we use $l'(Q^+) = \max_{u \in \mathbb{S}^2_+} 1 - \frac{\omega(Q^+,u)}{\omega(P^+,u)}$ to denote the loss of Q^+ w.r.t. P^+ in RMS.

To prove the theorem, we construct a three-dimensional MC instance $\mathrm{MC}(P_1, r_1)$ from any three-dimensional RMS instance RMS (P_0, r_0) satisfying that there is a subset $Q_0 \subseteq P_0$ of size r_0 such that $l'(Q_0) \le \varepsilon$ if and only if there is a subset $Q_1 \subseteq P_1$ of size r_1 such that $l(Q_1) \le \varepsilon$ for an arbitrary parameter $\varepsilon \in (0,1)$. Given an RMS instance $\mathrm{RMS}(P_0, r_0)$ and $\varepsilon \in (0,1)$, we add three new points $B = \{b_x, b_y, b_z\}$ to P_0. Let $b_x = (1-\eta, 1, 1)$, $b_y = (1, 1-\eta, 1)$, and $b_z = (1, 1, 1-\eta)$, where the value of η is greater than 3 and determined by P_0 and ε as discussed later. We will show that there is a subset $Q_0 \subseteq P_0$ of size r_0 with $l'(Q_0) \le \varepsilon$ if and only if there is a subset $Q_1 = Q_0 \cup B$ of size $r_1 = r_0 + 3$ for $P_1 = P_0 \cup B$ with $l(Q_1) \le \varepsilon$. To prove this, we need to show *(i)* if $l'(Q_0) \le \varepsilon$ and $Q_1 = Q_0 \cup B$, then $l(Q_1) \le \varepsilon$ and *(ii)* if $l(Q_1) \le \varepsilon$, then $B \subset Q_1$ and $l'(Q_0) \le \varepsilon$ for $Q_0 = Q_1 \setminus B$ and $P_0 = P_1 \setminus B$.

To prove *(i)*, we first consider the case of $u \in \mathbb{S}^2_+$. Let $p^* = \arg\max_{p \in P_1} \langle p, u \rangle$ for some $u \in \mathbb{S}^2_+$. If $p^* \in B$, then $\omega(Q_1, u) = \omega(P_1, u)$ because $B \subset Q_1$. Otherwise, since $p^* \in P_0$ and $l'(Q_0) \le \varepsilon$, there must exist some $p \in Q_0 \subset Q_1$ such that $\langle p^*, u \rangle \le (1-\varepsilon) \cdot \langle p, u \rangle$. Next, we consider the case of $u \in \mathbb{S}^2 \setminus \mathbb{S}^2_+$. We show that the extreme point for any $u \in \mathbb{S}^2 \setminus \mathbb{S}^2_+$ is always from B and thus $\omega(Q_1, u) = \omega(P_1, u)$ in this case. There are three different cases based on the orthant of $u = (u_1, u_2, u_3)$ as follows:

- **Case 1 ($u_1 \ge 0, u_2 \ge 0, u_3 \le 0$):** For any $p \in P_0$, $\langle p, u \rangle \le p_1 u_1 + p_2 u_2 < u_1 + u_2 \le \sqrt{2}$. Meanwhile, $\langle b_z, u \rangle = u_1 + u_2 + (1-\eta) \cdot u_3 \ge \sqrt{2}$ for $\eta \ge 3$. Thus, $\langle b_z, u \rangle \ge \langle p, u \rangle$ for any $p \in P_0$. This result also holds for b_x or b_y when $u_1 \le 0$ or $u_2 \le 0$ and the remaining dimensions are nonnegative.
- **Case 2 ($u_1 \ge 0, u_2 \le 0, u_3 \le 0$):** For any $p \in P_0$, $\langle p, u \rangle < u_1 \le 1$. Moreover, $\langle b_y, u \rangle = u_1 + u_2 + u_3 - \eta u_2$ and $\langle b_z, u \rangle = u_1 + u_2 + u_3 - \eta u_3$. If $u_2 \le u_3$, then $\langle b_y, u \rangle \ge \langle b_z, u \rangle$, and vice versa. So the minimum of the maximum between $\langle b_y, u \rangle$ and $\langle b_z, u \rangle$ is always reached when $u_2 = u_3$. In this case, $\langle b_y, u \rangle = \langle b_z, u \rangle = u_1 + (2-\eta) u_2$. By setting $u_2 = u_3 = \beta$ and $u_1 = \sqrt{1 - 2\beta^2}$ accordingly, we define the following function of β where $\beta \in [-1, 0]$:

  $$f(\beta) = \langle b_y, u \rangle = \sqrt{1 - 2\beta^2} + (2-\eta)\beta$$

  As $f(\beta)$ first increases and then decreases in $[-1, 0]$, its minimum is reached when $\beta = -1$ or 0. Thus, we have $\langle b_y, u \rangle \ge f(0) = 1$ and $\langle b_y, u \rangle \ge f(-1) = \eta - 2 > 1$ for $\eta \ge 3$. So the larger one between $\langle b_y, u \rangle$ and $\langle b_z, u \rangle$ is always greater than $\langle p, u \rangle$ for any $p \in P_0$ in this case. A similar result can also be implied when $u_2 \ge 0$ or $u_3 \ge 0$ and the remaining dimensions are not positive.
- **Case 3 ($u_1 \le 0, u_2 \le 0, u_3 \le 0$):** For any $p \in P_0$, $\langle p, u \rangle \le 0$. And for the minimum u_{min} among $\{u_1, u_2, u_3\}$, we have $u_{min} \le -\frac{\sqrt{3}}{3}$. Taking $u_{min} = u_1 \le -\frac{\sqrt{3}}{3}$ as an example, we have $\langle b_x, u \rangle \ge \frac{\sqrt{3}}{3}\eta - \sqrt{3} > 0$ due to $\eta > 3$. A similar result can also be acquired for u_2 or u_3.

We prove *(i)* for any $u \in \mathbb{S}^2 \setminus \mathbb{S}^2_+$ from the above three cases.

To verify *(ii)*, we need to show *(a)* if $B \not\subset Q_1$, then $l(Q_1) > \varepsilon$ and *(b)* if $l'(Q_0) > \varepsilon$, then $l(Q_0 \cup B) > \varepsilon$ w.r.t. $P_1 = P_0 \cup B$. The correctness of *(a)* is obvious: Taking $u = (-1, 0, 0) \in \mathbb{S}^2$, $\langle b_x, u \rangle > 0$

and $\langle p, u\rangle < 0$ for any $p \in P_1 \setminus \{b_x\}$. So if $B \not\subset Q_1$, then $l(Q_1) > 1$. The proof of *(b)* involves determining the value of η. If $l'(Q_0) > \varepsilon$, then there is some point $p' \in P_0 \setminus Q_0$ and a vector $u' \in \mathbb{S}^2_+$ such that $(1-\varepsilon)\cdot\langle p', u'\rangle > \omega(P_0, u')$. Since the maximum loss $l'(Q_0)$ and the vector where $l'(Q_0)$ is reached can be computed from a linear program [35], such a point p' and a vector u' can be found in polynomial time. When $\eta > \frac{3-(1-\varepsilon)\cdot\langle p',u'\rangle}{u_i}$, we have $\langle b_i, u'\rangle < u_1 + u_2 + u_3 - 3 + (1-\varepsilon)\cdot\langle p', u'\rangle < (1-\varepsilon)\cdot\langle p', u'\rangle$ for all $i = 1, 2, 3$. Therefore, if $l'(Q_0) > \varepsilon$, then $l(Q_0 \cup B) > \varepsilon$, once η is a large enough constant. We prove *(ii)* from *(a)* and *(b)*.

We have thus completed the reduction from an RMS instance in $\mathbb{R}^3_+$ to an MC instance in $\mathbb{R}^3$ in polynomial time and proved the NP-hardness of MC in $\mathbb{R}^3$. □

Since the reduction procedure can be generalized to any constant $d > 3$, the minimum ε-coreset (MC) problem is still NP-hard in $\mathbb{R}^d$ for any constant $d > 3$.

4 INNER-PRODUCT VORONOI DIAGRAM

In this section, we introduce the *Voronoi diagrams* [10], from which we can draw geometric insights for the MC problem and build the theoretical foundation of our proposed algorithms. A Voronoi diagram for a point set P is defined in terms of a similarity measure, which is the *inner product* function $\langle\cdot,\cdot\rangle$ here. Prior work has investigated the relationships between the inner-product Voronoi diagram and graph-based maximum inner product search [34, 40, 47]. We will first consider using the inner-product Voronoi diagram for coreset construction in this work.

Formally, for a point set P, its Voronoi diagram is a collection of *Voronoi cells*, one defined for each point $p \in P$. The Voronoi cell $R(p)$ of point $p \in P$ is defined as the set of vectors

$$R(p) := \{u \in \mathbb{S}^{d-1} \,:\, \langle p, u\rangle \geq \omega(P, u)\}$$

i.e., the set of unit vectors for which p is the maximum. By definition, a point $p \in P$ is an extreme point if and only if its Voronoi cell $R(p)$ is non-empty. It is also not difficult to see that the set X of extreme points consists exactly of the set of vertices of the convex hull $\mathcal{CH}(P)$ of P.

To illustrate MC geometrically, we extend the notion of Voronoi cells to ε-approximate Voronoi cells. Specifically, the ε-approximate Voronoi cell $R_\varepsilon(p)$ of point $p \in P$ is defined as the vector set

$$R_\varepsilon(p) := \{u \in \mathbb{S}^{d-1} \,:\, \langle p, u\rangle \geq (1-\varepsilon)\cdot\omega(P, u)\}$$

i.e., the set of unit vectors for which the inner product of p is within an ε-approximation to the maximum of P. Equivalently, the loss in the maxima of Q w.r.t. P is at most ε for any vector in $R_\varepsilon(p)$ if $p \in Q$. So, from a geometric perspective, MC can be regarded as the set cover problem of *finding the minimum subset of points from P such that the union of their ε-approximate Voronoi cells is $\mathbb{S}^{d-1}$*.

A notion closely related to the inner-product Voronoi diagram is the Inner-Product Delaunay Graph (IPDG), which will be used for our DSMC algorithm in Section 6.1. The IPDG of point set P is a graph that records the adjacency information of the Voronoi cells of extreme points in P. Formally, it is an undirected graph $\mathcal{G}(P) = (V, E)$ where $V = X$, and there exists an edge $\{p, q\} \in E$ if and only if $R(p) \cap R(q) \neq \emptyset$. The number of edges in $\mathcal{G}(P)$ grows exponentially with d and building an exact IPDG is often not computationally feasible when $d > 3$ [40]. Nevertheless, the IPDG $\mathcal{G}(P)$ of point set P can be built from the convex hull $\mathcal{CH}(P)$ efficiently in $\mathbb{R}^2$ or $\mathbb{R}^3$ since the edges in $\mathcal{CH}(P)$ exactly correspond to the edges in $\mathcal{G}(P)$.

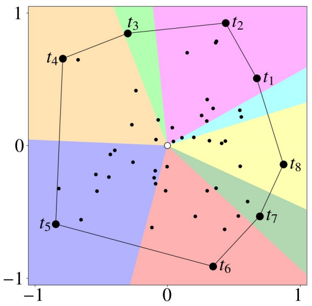


Figure 1: An example of Voronoi diagram on a point set in $\mathbb{R}^2$. The extreme points are indexed as $\{t_1, \ldots, t_8\}$ with their corresponding Voronoi cells in different colors. We also draw the IPDG of the point set, where each pair of extreme points with adjacent Voronoi cells is connected.

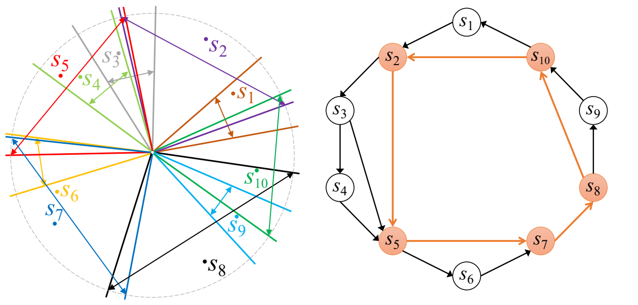
5 ALGORITHM IN 2D

In this section, we present OptMC, our optimal polynomial-time algorithm for MC in $\mathbb{R}^2$. Before delving into the details, we discuss geometric properties of MC in $\mathbb{R}^2$ that are useful for the design of the OptMC algorithm. As we have seen in Section 4, in $\mathbb{R}^2$, exact and ε-approximate Voronoi cells of any point can be represented as arcs of the one-dimensional unit sphere $\mathbb{S}^1$. Therefore, MC in $\mathbb{R}^2$ is equivalent to *finding the minimum number of arcs* (resp. *ε-approximate Voronoi cells of points*) *that fully cover* $\mathbb{S}^1$. One straightforward solution for this problem is to compute the ε-approximate Voronoi cells of all points for an input parameter ε and solve the aforementioned arc-covering problem. Furthermore, the (exact) Voronoi cell of each extreme point can be computed via simple comparisons with its two neighbors in the IPDG (note that the degree of each vertex of $\mathcal{G}(P)$ is 2 for $P \subset \mathbb{R}^2$). Computing the ε-approximate Voronoi cell of each point also requires comparisons with extreme points only. Once all Voronoi cells are obtained, the optimal solution of the arc-covering problem can be found in polynomial time because it is one-dimensional. In the OptMC algorithm we propose below, we further manage to avoid explicit representations of ε-approximate Voronoi cells via a graph-based transformation for reducing the computational cost.

5.1 Algorithmic Description

The main idea behind the OptMC algorithm is to build a directed graph G where each (directed) cycle corresponds to a feasible solution for MC. Then, the shortest cycle in G provides an optimal solution for MC. In general, the OptMC algorithm proceeds in three main steps, namely *candidate selection*, *graph construction*, and *solution computation*. The point set P, the parameter ε, and the set X of extreme points are provided as inputs to OptMC. We can obtain X from P in $O(n \log \xi)$ time, where $\xi = |X|$, by running any convex hull algorithm such as Qhull [12]. For ease of illustration, we arrange all extreme points in a counterclockwise direction as

```
Algorithm 1: OptMC
  Input  : A point set $P$; the set $X$ of extreme points in $P$; the error
           parameter $\varepsilon \in (0,1)$
  Output : The optimal solution $Q^*_\varepsilon$ of MC on $P$
1 Let $u^*_i$ be the vector $u \in \mathbb{S}^1$ where $\langle t_i, u\rangle = \langle t_{i+1}, u\rangle$ and
    $\langle t_i, u\rangle > 0$, and $U^* = \{u^*_i : i \in [\xi]\}$ where $\xi = |X|$;
2 Initialize a candidate set $S \leftarrow X$;
3 foreach point $p \in P \setminus X$ do
4     if $\exists u \in U^* : \langle p, u\rangle \geq (1-\varepsilon) \cdot \langle t_i, u\rangle$ then
5         $S \leftarrow S \cup \{p\}$;
6 Arrange the points of $S$ in a counterclockwise direction and index
    them as $[s_1, \ldots, s_\varsigma]$ where $\varsigma = |S|$;
7 Let $G = (V, E)$ be a directed graph with $V = S$ and $E = \emptyset$;
8 foreach $i, j \in [\varsigma]$ and $i \neq j$ do
9     if $\angle s_i O s_j \geq \pi$ then skip Lines 10–12 directly;
10    Let $u^*$ be $u \in \mathbb{S}^1$ where $\langle s_i, u\rangle = \langle s_j, u\rangle$ and $\langle s_i, u\rangle \geq 0$;
11    if $\exists u \in U^* \cup \{u^*\} : \max_{t \in X} 1 - \min\left(\frac{\langle s_i, u\rangle}{\langle t, u\rangle}, \frac{\langle s_j, u\rangle}{\langle t, u\rangle}\right) \leq \varepsilon$ then
12        Add a directed edge $(s_i \rightarrow s_j)$ to $E$;
13 $C^* \leftarrow$ ShortestCycle($G$);
14 return $Q^*_\varepsilon \leftarrow \{q : q \in C^*\}$;
```



(a) Candidates and Voronoi cells (b) Graph for solution computation

Figure 2: An illustration of OptMC. In Figure 2(a), we index the candidates picked from the point set in Figure 1 for $\varepsilon = 0.1$ as $\{s_1, \ldots, s_{10}\}$ and draw the boundaries of their 0.1-approximate Voronoi cells in different colors. The graph G for $\varepsilon = 0.1$ is shown in Figure 2(b), where each pair of candidates with overlapping Voronoi cells is connected by a directed edge. The shortest cycle C^* of G is highlighted in orange and the points in C^* are the optimal solution of MC with $\varepsilon = 0.1$.

$X = \{t_1, \ldots, t_\xi\}$ based on their corresponding angles from 0 to 2π in the polar coordinate system. For any point p or vector u in $\mathbb{R}^2$, we use $\theta(p)$ or $\theta(u) \in [0, 2\pi)$ to denote the angle of p or u. Next, we will present the three steps of OptMC in detail.

Candidate Selection: The purpose of this step is to identify a subset $S \subseteq P$ of points that may be included in the solution, while pruning from consideration points that are certainly not. To find the candidate points, OptMC essentially ignores all points that are never within an ε-approximation from the maxima. In other words, $p \in S$ if and only if the ε-approximate Voronoi cell of p is non-empty – i.e., $R_\varepsilon(p) \neq \emptyset$. This obviously holds for the extreme points in X. To determine whether it holds for a non-extreme point $p \in P \setminus X$, OptMC compares p with each extreme point $t \in X$ and computes the minimum loss of p w.r.t. t across all vectors in $R(t)$. If the minimum loss is at most ε for any extreme point t, then $R_\varepsilon(p)$ is not empty; otherwise, it is. Moreover, the minimum loss is always reached at a boundary vector where the inner product of t is equal to the inner product of either the previous or next extreme point. Putting everything together, OptMC first computes the boundary vector u^*_i of $R(t_i)$ and $R(t_{i+1})$ for each pair of neighboring extreme points $t_i, t_{i+1} \in X$ (for $i = \xi$, u^*_ξ is computed from t_ξ and t_1). Then, it computes the relative loss of p for each u^*_i and adds p to S if its loss is at most ε for some u^*_i. Finally, all points in S are arranged in a counterclockwise direction from 0 to 2π and indexed by $[1, \ldots, \varsigma]$ accordingly, where $\varsigma = |S|$.

Graph Construction: The purpose of this step is to build a directed graph $G = (V, E)$ where $V = S$ with the following property: there is an edge between a pair of vertices $s_i, s_j \in S$ if and only if their ε-approximate Voronoi cells are overlapping. Specifically, if the losses of s_i and s_j are both at most ε for either the vector u^* where s_i and s_j have the same inner product or any of the boundary vectors between extreme points, then there will exist a directed edge $(s_i \rightarrow s_j)$. OptMC will decide whether an edge should be added between each pair of distinct candidates. If the vector angle between s_i and s_j is greater than π, the check for $(s_i \rightarrow s_j)$ will be skipped directly to avoid duplicated computation.

Solution Computation: Given a directed graph $G = (V, E)$ constructed according to the above procedure, the final step is to find the shortest cycle C^* of G and return the vertices of C^* as the optimal solution Q^*_ε of MC on P. Here, any shortest-path or shortest-cycle algorithm, e.g., the algorithms in [23, 26, 36], can be used for finding the shortest cycle C^* from G.

We summarize the detailed procedure of OptMC in Algorithm 1, where the *candidate selection* step is presented in Lines 1–6, the *graph construction* step is described in Lines 7–12, and the *solution computation* step is shown in Lines 13–14.

5.2 Theoretical Analysis

Next, we analyze the optimality and time complexity of the OptMC algorithm theoretically. Firstly, Lemma 5.1 shows the validity of *candidate selection*. Secondly, Lemma 5.2 proves the correctness of *graph construction*. Thirdly, Lemma 5.3 verifies the equivalence between computing the optimal solution of MC on P and finding the shortest cycle from G. Finally, considering the results of Lemmas 5.1–5.3 collectively, we prove the optimality of OptMC and analyze its time complexity in Theorem 5.4.

Lemma 5.1. *For the candidate set S computed in Algorithm 1, it holds that a point $p \in S$ if and only if $R_\varepsilon(p) \neq \emptyset$.*

Proof. On the one hand, it is obvious that if $p \in S$ then $R_\varepsilon(p) \neq \emptyset$, since $u^*_i \in R_\varepsilon(p)$. On the other hand, we will prove that if $p \notin S$ then $R_\varepsilon(p) = \emptyset$, by showing that the minimum of $l_u(p) = 1 - \frac{\langle p, u\rangle}{\omega(P,u)}$ of p for any $u \in \mathbb{S}^1$ is greater than ε. By considering each extreme point $t_i \in X$ for $l_u(p)$ separately, we need to show that $1 - \frac{\langle p, u\rangle}{\langle t_i, u\rangle} > \varepsilon$ for each $u \in R(t_i)$. We define a function f to denote the loss of p

w.r.t. t_i as follows:

$$f(u) = 1 - \frac{\langle p, u\rangle}{\langle t_i, u\rangle} = 1 - \frac{\|p\|}{\|t_i\|} \cdot \frac{\cos(\theta(p) - \theta(u))}{\cos(\theta(t_i) - \theta(u))}$$

where $\theta(p)$ is the angle of the point-vector p and $u \in R(t_i) = [u^*_{i-1}, u^*_i]$. According to Line 4 of Algorithm 1, we have $1 - \frac{\langle p,u^*_i\rangle}{\langle t_i,u^*_i\rangle} > \varepsilon$ for all $i \in [\xi]$ if $p \notin S$. Furthermore, if $\theta(p) < \theta(t_i)$, then $f(u)$ monotonically increases with $\theta(u)$; if $\theta(p) > \theta(t_i)$, then $f(u)$ monotonically decreases with $\theta(u)$; and if $\theta(p) = \theta(t_i)$, then $f(u)$ is a constant $1 - \frac{\|p\|}{\|t_i\|}$. Therefore, the minimum of $f(u)$ is reached when either $u = u^*_{i-1}$ or u^*_i. In addition, $f(u^*_{i-1}) = 1 - \frac{\langle p,u^*_{i-1}\rangle}{\langle t_{i-1},u^*_{i-1}\rangle} > \varepsilon$ and $f(u^*_i) = 1 - \frac{\langle p,u^*_i\rangle}{\langle t_i,u^*_i\rangle} > \varepsilon$. Combining all above results, we prove that $l_u(p) > \varepsilon$ for any $u \in \mathbb{S}^1$ if $\frac{\langle p,u\rangle}{\langle t_i,u\rangle} < 1 - \varepsilon$ for any $t_i \in X$ and conclude the proof. □

Lemma 5.2. *Let s_i, s_j $(i < j)$ be two points in S. For the graph G constructed by Algorithm 1, there exists an edge $(s_i \longrightarrow s_j)$ if and only if $R_\varepsilon(s_i) \cap R_\varepsilon(s_j) \neq \varnothing$.*

Proof. First of all, it is easy to see that if there is an edge $(s_i \longrightarrow s_j)$ in G then $R_\varepsilon(s_i) \cap R_\varepsilon(s_j) \neq \varnothing$ — because the vector u in $U^* \cup \{u^*\}$ satisfying the condition in Line 11 must be in $R_\varepsilon(s_i) \cap R_\varepsilon(s_j)$. Next, we need to show that $R_\varepsilon(s_i) \cap R_\varepsilon(s_j) = \varnothing$ if $(s_i \longrightarrow s_j)$ does not exist in G. For $s_i, s_j \in S (i < j)$ and $t \in X$, we define function f_{ij} to denote the maximum loss of either point s_i and s_j w.r.t. t as follows:

$$f_{ij}(u) = 1 - \min\left(\frac{\langle s_i, u\rangle}{\langle t, u\rangle}, \frac{\langle s_j, u\rangle}{\langle t, u\rangle}\right)$$

where $u \in R(t)$. There is a vector $u \in R(t)$ such that $u \in R_\varepsilon(s_i) \cap R_\varepsilon(s_j)$ if and only if the minimum of $f_{ij}(u)$ is at most ε. By extending the result in the proof of Lemma 5.1, we can find three different cases for the minimum of $f_{ij}(u)$ in $R(t)$: (1) if $\theta(t) \geq \theta(s_j)$ or $\theta(t) \leq \theta(s_i)$, then the minimum of $f_{ij}(u)$ is reached at the boundary of $R(t)$ – i.e., some vector in U^*; (2) if $\theta(s_i) < \theta(t) < \theta(s_j)$ and $u^* \in R(t)$, then the minimum of $f_{ij}(u)$ is reached when $\langle s_i, u\rangle = \langle s_j, u\rangle$ – i.e., just at u^*; (3) if $\theta(s_i) < \theta(t) < \theta(s_j)$ and $u^* \notin R(t)$, then the minimum of $f_{ij}(u)$ is also reached at the boundary of $R(t)$. Thus, to find the minimum of f_{ij}, it is enough to check all vectors in $U^* \cup \{u^*\}$ for each extreme point. If the condition in Line 11 is not satisfied for any of them, it is safe to say that $f_{ij}(u) > \varepsilon$ for any $u \in \mathbb{S}^1$ and $R_\varepsilon(s_i) \cap R_\varepsilon(s_j) = \varnothing$. □

In Lemma 5.2, we only consider the case of $i < j$. It is easy to verify that Lemma 5.2 also holds when $i > j$.

Before presenting Lemma 5.3, we define the notion of the "locally minimal solution" as follows: A solution Q of MC on P is locally minimal if Q is a feasible solution and $Q \backslash \{q\}$ is not feasible anymore for each point $q \in Q$.

Lemma 5.3. *The graph G constructed by Algorithm 1 must satisfy that: (i) if C is a cycle in G, then $Q = \{q : q \in C\}$ is a feasible solution of MC on P; and (ii) if Q is a locally minimal solution of MC on P, then there exists a cycle C in G corresponding to Q.*

Proof. Let $\{q_1, \ldots, q_{|Q|}\}$ be the points in Q arranged in a counterclockwise direction. Because C is a cycle of G, there is an edge $(q_i \longrightarrow q_{i+1})$ for each $i \in [|Q|]$. According to Lemma 5.2, $R_\varepsilon(q_i) \cap R_\varepsilon(q_{i+1}) \neq \varnothing$ for each $i \in [|Q|]$. Therefore, $\bigcup_{q \in Q} R_\varepsilon(q) = \mathbb{S}^1$ and Q is a feasible solution of MC.

Then, we show any locally minimal solution must correspond to a cycle of G. First, if Q is a locally minimal solution for MC, then $Q \subseteq S$, where S is the set of all candidates with nonempty ε-approximate Voronoi cells. For any $p \notin S$, if a subset Q' where $p \in Q'$ is feasible, it will hold that $Q' \setminus \{p\}$ is still feasible since $R_\varepsilon(p) = \varnothing$, which implies that Q' is not locally minimal. Next, we prove by contradiction that if Q is locally minimal then there is an edge $(q_i \longrightarrow q_{i+1})$ in G for every $i \in [|Q|]$. If $(q_i \longrightarrow q_{i+1}) \notin E$, then either (1) $\theta(q_{i+1}) - \theta(q_i) > \pi$ or (2) $R_\varepsilon(q_i) \cap R_\varepsilon(q_{i+1}) = \varnothing$. In the former case, we have $l(Q) > 1$ and Q is not feasible. In the latter case, if there does not exist any $i' < i$ or $i'' > i + 1$ such that $R_\varepsilon(q_{i'}) \cap R_\varepsilon(q_{i+1}) \neq \varnothing$ or $R_\varepsilon(q_i) \cap R_\varepsilon(q_{i''}) \neq \varnothing$, there will exist some vector u between $R_\varepsilon(q_i)$ and $R_\varepsilon(q_{i+1})$ where the loss of Q is greater than ε and thus Q is not feasible; otherwise, if there exists such $i' < i$ or $i'' > i + 1$, there will be an edge $(q_{i'} \longrightarrow q_{i+1})$ or $(q_i \longrightarrow q_{i''})$ in G, which implies that either q_i or q_{i+1} is redundant in Q and Q is not locally minimal. Based on all above results, we conclude that there exists a cycle C of G corresponding to Q as long as Q is a locally minimal solution for MC on P. □

Theorem 5.4. *OptMC returns the optimal solution for MC with a parameter $\varepsilon \in (0, 1)$ on a point set $P \subset \mathbb{R}^2$ in $O(n^3)$ time.*

Proof. Based on Lemma 5.1, OptMC excludes all redundant points from computation. According to Lemmas 5.2 and 5.3, it is guaranteed that any locally minimal solution for MC on P forms a cycle in G. Therefore, the optimal solution Q^*_ε of MC on P – i.e., the smallest among all feasible solutions for MC on P, must correspond to the shortest cycle of G, as the globally minimal solution must also be locally minimal. Hence, OptMC is optimal for MC in $\mathbb{R}^2$.

The time complexity of *candidate selection* is $O(\xi + n \cdot \log \xi)$. Here, a binary search is used to find the index i and thus it takes $O(\log \xi)$ time to decide whether to include p into S or not. The time complexity of *graph construction* is $O(\varsigma^2 \cdot \xi)$. The time complexity of finding the shortest cycle in a directed graph is $O(|V| \cdot |E| + |V|^2 \cdot \log |E|)$ when the Dijkstra's algorithm in [26] is used for computing the shortest path from each vertex. Recently, an $O(|V| \cdot |E|)$ time algorithm [36] was also proposed for finding the shortest directed cycle. In the worst case (i.e., ε is close to 1), since $\varsigma = |V| = O(n)$ and $|E| = O(\varsigma^2)$, the time complexity of OptMC is $O(n^3)$. When ε is much smaller than 1, it is reasonable to assume that $\varsigma = |V| \ll O(n)$ and $|E| = O(|V|)$ and the time complexity of OptMC is reduced to $O(n \cdot \log \xi + \varsigma^2 \cdot \xi)$. □

6 ALGORITHMS IN MD

In the case of $d \geq 3$, MC becomes much more challenging due to NP-hardness. As discussed in Section 4, MC can be viewed as a set cover problem defined on Voronoi cells. Solving it directly as such would require the invocation of set operations between Voronoi cells to compute whether the coverage condition is satisfied. However, the geometric shapes of Voronoi cells in $\mathbb{R}^d$ when $d \geq 3$ are convex polyhedra that are defined by intersections of half-spaces embedded on $\mathbb{S}^{d-1}$, which makes set operations [6, 22] very inefficient. Even if the Voronoi cells of all points are given as inputs and the set operations are assumed to be performed by an oracle, finding the optimal solution of MC is still infeasible unless P=NP due to the combinatorial complexity of geometric set cover [25].

In this section, we propose two different algorithms for MC. The high-level idea of both algorithms is to approximate the set operations on Voronoi cells so as to transform MC into simpler cover problems. First, in Section 6.1, we propose the DSMC algorithm that transforms MC into a dominating-set problem on a directed graph denoting the inclusion relationships among the Voronoi cells of points. Furthermore, we find that MC can also be transformed into a set cover problem via the discretization of Voronoi cells by adapting the results in [3, 9] for the transformation from the regret minimizing set (RMS) problem to a set cover problem based on the notion of δ-nets [28]. We propose the SCMC algorithm for MC accordingly. Since the analyses for SCMC are similar to those for the hitting-set based algorithm in [3], we here omit the details of SCMC and leave them to Appendix A.

6.1 Dominating Set Based Algorithm

In this subsection, we present the DSMC algorithm for the MC problem. DSMC is based on two simplifications in relations to MC. Firstly, we only use the set X of extreme points, instead of the whole point set, to compute the solutions for MC. Secondly, when considering an extreme point $t \in X$ for the solution set, we restrict ourselves to merely two possibilities: either t is in the solution; or there exists another extreme point t' in the solution whose ε-approximate Voronoi cell fully incorporates the exact Voronoi cell of t, in which case we say that t' *dominates* t. In other words, we do not consider the case that $R(t)$ is covered by a *union* of the ε-approximate Voronoi cells of two or more points for the inefficiency of set operations. The above simplifications allow us to develop an approach that is expressed in terms of a graph structure, to which we refer as the *dominance graph*, because it encapsulates information about the dominance relationships among vertices. The resulting approach can be seen as targeting a simplified formulation of the original MC problem – i.e., finding a dominating set of the dominance graph as the solution for MC.

Dominance Graph: We first consider how to construct the dominance graph for a point set P. Let the extreme points in $X \subseteq P$ be indexed by $[\xi]$ as $\{t_1, t_2, \ldots, t_\xi\}$, where $\xi = |X|$. The dominance graph $\mathcal{H} = (V, E)$ is a weighted and directed graph where V is equal to X. A directed edge $(t_i \longrightarrow t_j)$ with an associated weight $\varepsilon_{ij} \in (0, 1)$ exists if and only if the ε_{ij}-approximate Voronoi cell of t_i fully incorporates the Voronoi cell of t_j. Therefore, the presence of edge $(t_i \longrightarrow t_j)$ signifies that, in any solution of MC with $\varepsilon \geq \varepsilon_{ij}$, t_i can replace t_j without a violation of the solution validity.

The edge weight ε_{ij} for each pair of points t_i, t_j can be computed from the linear program (LP) in Eq. 2.

$$\begin{aligned} \max \quad & 1 - t_i \cdot u \\ s.t. \quad & (t_j - t) \cdot u \geq 0, \forall t \in N(t_j) \\ & t_j \cdot u = 1 \end{aligned} \tag{2}$$

In Eq. 2, $N(t)$ is the set of neighbors of point t in the IPDG $\mathcal{G}(P)$ of P, i.e., the set of extreme points whose Voronoi cells are adjacent to $R(t)$. The first set of inequality constraints represents the feasible region of the linear program as the Voronoi cell $R(t_j)$ of t_j, which is defined by the intersections of $|N(t_j)|$ closed half-spaces. Each half-space corresponds to the region where the inner product of t_j is greater than or equal to that of $t \in N(t_j)$. Hence, $\langle t_j, u\rangle \geq \langle t, u\rangle \Leftrightarrow (t_j - t) \cdot u \geq 0$. The second inequality constraint scales the vector u so that the inner product of t_j is 1. With this constraint, for a given vector u, the loss of t_i w.r.t. t_j is equal to $1 - t_i \cdot u$. The LP in Eq. 2 finds the largest loss ε_{ij} of t_i w.r.t. t_j over the feasible region $R(t_j)$.

Algorithm 2: DominanceGraph

```
Input  : The set $X$ of extreme points in $P$; the IPDG $\mathcal{G}(P)$
Output : The dominance graph $\mathcal{H}$
Initialize $\mathcal{H} = (V, E)$ where $V = X$ and $E = \varnothing$;
foreach $i, j \in [\xi]$ and $i \neq j$ do
    Solve the LP in Eq. 2 for $t_i, t_j$ to compute $\varepsilon_{ij}$;
    if $\varepsilon_{ij} \in (0, 1)$ then
        Add a directed edge $(t_i \longrightarrow t_j)$ to $E$;
return $\mathcal{H}$;
```

Algorithm 3: DSMC

```
Input  : The dominance graph $\mathcal{H}$; a parameter $\varepsilon \in (0, 1)$
Output : The solution $Q_\varepsilon$ of MC
Let $\mathcal{H}_\varepsilon = (V, E_\varepsilon)$ be the subgraph of $\mathcal{H}$ with
    $E_\varepsilon = \{(t_i \longrightarrow t_j) \in E : \varepsilon_{ij} \leq \varepsilon\}$;
Initialize $Q_\varepsilon \leftarrow \varnothing$ and $U' \leftarrow X$;
for $i \leftarrow 1, \ldots, \xi$ do
    $Dom(t_i) \leftarrow \{t_i\} \cup \{t_j \in X : (t_i \longrightarrow t_j) \in E_\varepsilon\}$;
while $U' \neq \varnothing$ do
    $t^* \leftarrow \arg\max_{t_i \in X \setminus Q_\varepsilon} |Dom(t_i) \cap U'|$;
    $Q_\varepsilon \leftarrow Q_\varepsilon \cup \{t^*\}$ and $U' \leftarrow U' \setminus Dom(t^*)$;
return $Q_\varepsilon$;
```

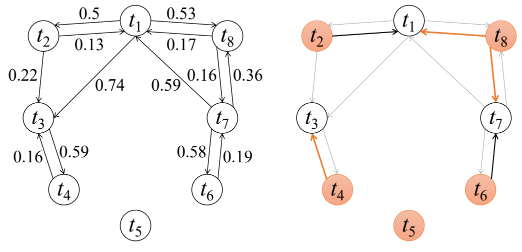
The procedure of building the dominance graph $\mathcal{H}$ is shown in Algorithm 2. For a point set P, the set X of extreme points and the IPDG $\mathcal{G}(P)$ are provided as inputs. Since computing $\mathcal{G}(P)$ exactly is costly when $d > 3$, we first assume the exact IPDG $\mathcal{G}(P)$ is available here and will discuss how to replace it with an approximate IPDG later in this subsection.

Solution Computation: Given the dominance graph $\mathcal{H}$, one can use it to compute a solution for MC. In particular, for a parameter $\varepsilon \in (0, 1)$, a solution for MC can be obtained as any subset $S \subseteq X$ of points satisfying the following condition: for each $t_j \in X$, either $t_j \in S$ or there exists an edge $(t_i \longrightarrow t_j)$ with $\varepsilon_{ij} \leq \varepsilon$ for some $t_i \in S$. Let $\mathcal{H}_\varepsilon = (V, E_\varepsilon)$ be the subgraph of $\mathcal{H}$ where $V = X$ and $E_\varepsilon \subseteq E$ is the subset of edges with weights at most ε. Then, a solution of MC can be obtained by finding the minimum dominating set of $\mathcal{H}_\varepsilon$. In practice, DSMC runs the greedy algorithm for the minimum dominating set problem on $\mathcal{H}_\varepsilon$. The detailed procedure of DSMC is summarized in Algorithm 3.

Theoretical Analysis: We first show that the solution Q_ε returned by DSMC is a valid ε-coreset for MC in Theorem 6.1.

Theorem 6.1. *The solution Q_ε returned by DSMC must satisfy that $l(Q_\varepsilon) \leq \varepsilon$.*

Proof. For any vector $u \in \mathbb{S}^{d-1}$, there exists a point t_j such that $u \in R(t_j)$. Since Q_ε is a dominating set of $\mathcal{H}_\varepsilon$, we have either



(a) Dominance graph $\mathcal{H}$ (b) Dominating set on $\mathcal{H}_{0.2}$

Figure 3: Illustrations of the dominance graph and DSMC. In Figure 3(a), we show the dominance graph $\mathcal{H}$ built on the dataset in Figure 1, where all edge weights are computed by Eq. 2. Then, in Figure 3(b), we show how $\mathcal{H}$ is used to compute the solution of MC for $\varepsilon = 0.2$. Specifically, $\mathcal{H}_{0.2}$ is a subgraph of $\mathcal{H}$ only containing the edges with weights at most 0.2. DSMC runs the greedy algorithm on $\mathcal{H}_{0.2}$ to find the dominating set (in orange) as a solution of MC.

$t_j \in Q_\varepsilon$ or there exists an edge $(t_i \longrightarrow t_j) \in E_\varepsilon$ for some $t_i \in Q_\varepsilon$. In the previous case, we have $\omega(Q_\varepsilon, u) = \omega(P, u)$; In the latter case, we have $\omega(Q_\varepsilon, u) \geq (1 - \varepsilon_{ij}) \cdot \omega(P, u) \geq (1 - \varepsilon) \cdot \omega(P, u)$. Considering both cases collectively, we have $l(Q_\varepsilon) \leq \varepsilon$. □

How large is the size of the solution of DSMC compared to the optimal one for MC? While the approximation factor of DSMC on the solution size is still open, we do have an upper bound for it, as given in Theorem 6.2.

Theorem 6.2. *If* $\mathtt{OPT}_\varepsilon$ *is the size of the smallest* ε*-coreset of* P *and* Q_ε *is the solution of MC returned by DSMC, it holds that* $|Q_\varepsilon| = O(\frac{\xi}{d}) \cdot \mathtt{OPT}_\varepsilon$*, where* $\xi = |X|$.

Proof. First, the size of Q_ε is at most ξ since $Q_\varepsilon \subseteq X$. Second, the smallest ε-coreset Q^*_ε of P must contain at least $d + 1$ points to guarantee $l(Q^*_\varepsilon) < 1$. This is because, for any size-d point set Q in $\mathbb{R}^d$, one can find a vector u perpendicular to a hyperplane containing all points in Q such that $\langle p, u \rangle = 0$ for each $p \in Q$ and $l_X(Q) = 1$. Thus, $|Q_\varepsilon| \leq \frac{\xi}{d+1} \cdot |Q^*_\varepsilon| = O(\frac{\xi}{d}) \cdot \mathtt{OPT}_\varepsilon$. □

Finally, we prove that both procedures of DSMC in Algorithms 2 and 3 run in polynomial time in Theorem 6.3.

Theorem 6.3. *The time complexities of Algorithm 2 and Algorithm 3 are* $O(\xi^2 \cdot \Delta \cdot d^{3.5})$ *and* $O(\xi^2 \cdot D)$*, respectively, where* $\xi = |X|$, $\Delta = \max_{t \in X} |N(t)|$*, and* $D = \max_{t \in X} Dom(t)$.

Proof. First of all, the number of LPs solved for dominance graph construction is $O(\xi^2)$, as it computes the weight for each pair of extreme points. Each LP in Eq. 2 has $|N(t_j)| + 1$ constraints and d variables. When the interior point method is used as the LP solver, the worst-case time complexity of $O(\Delta \cdot d^{3.5})$ is always guaranteed. Therefore, the time complexity of Algorithm 2 is $O(\xi^2 \cdot \Delta \cdot d^{3.5})$. Then, the time to extract the subgraph and build a set system for the subgraph is $O(\xi \cdot D)$. The greedy algorithm evaluates the intersection of $Dom(t)$ and U' for each $t \in X$ at each iteration in $O(\xi \cdot D)$ time. The greedy algorithm runs $|Q_\varepsilon| = O(\xi)$ iterations. Hence, the time complexity of Algorithm 3 is $O(\xi^2 \cdot D)$. □

Remark: Since constructing the exact IPDG of a point set is computationally intensive when $d > 3$, we consider using an approximate IPDG instead for dominance graph construction. A practical approach to building an approximate IPDG proposed in [40] is used in our implementation. We show that using an approximate IPDG $\widehat{\mathcal{G}}(P)$ instead of the exact IPDG $\mathcal{G}(P)$ does not affect the correctness of DSMC– i.e., the solution of DSMC computed from a dominance graph built based on $\widehat{\mathcal{G}}(P)$ is still a valid ε-coreset. Compared with $\mathcal{G}(P)$, $\widehat{\mathcal{G}}(P)$ may both contain additional edges and miss existing edges. On the one hand, an additional edge has no effect on the solution of the LP in Eq. 2. This is because its feasible region is exactly the Voronoi cell $R(t_j)$ of t_j. Hence, an additional edge leads to a redundant constraint that does not reduce the feasible region at all. On the other hand, a missing edge may cause the solution value of the LP in Eq. 2 to be greater than the maximum loss, because the feasible region becomes larger than $R(t_j)$. As a result, if we use $\widehat{\mathcal{G}}(P)$ instead of $\mathcal{G}(P)$, $\mathcal{H}_\varepsilon$ for any $\varepsilon \in (0, 1)$ will contain strictly equal or fewer edges, and thus the solution Q_ε may include more points. Intuitively, if $\widehat{\mathcal{G}}(P)$ is closer to $\mathcal{G}(P)$, the size of the solution Q_ε of DSMC will tend to be smaller. Nevertheless, Q_ε is still guaranteed to be valid for MC when $\widehat{\mathcal{G}}(P)$ is used.

In addition, since DSMC may not provide ε-coresets of the smallest sizes, we can invoke Algorithm 3 with some $\varepsilon' > \varepsilon$ for obtaining smaller valid ε-coresets. In practice, we try different values of ε' picked from $[\varepsilon, 3\varepsilon]$ and invoke Algorithm 3 for each value of ε' to find the largest value of ε' such that $l(Q_{\varepsilon'}) \leq \varepsilon$ and return $Q_{\varepsilon'}$ as the solution of DSMC.

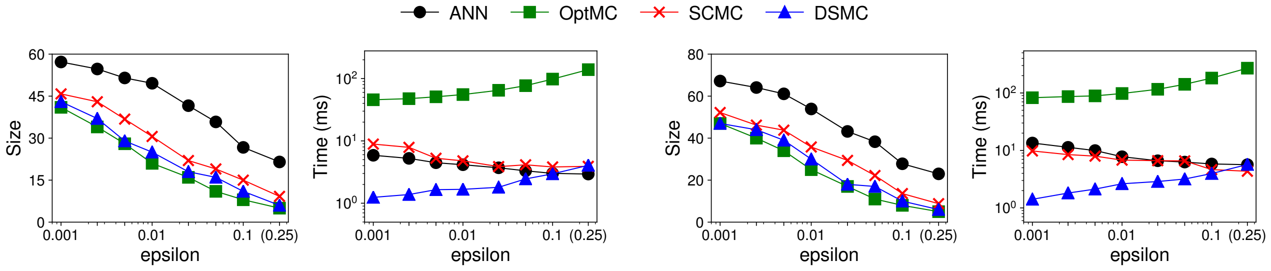
7 EXPERIMENTAL EVALUATION

In this section, we do extensive experiments on real and synthetic datasets to evaluate the performance of our proposed algorithms. We implemented our proposed algorithms – i.e., OptMC, SCMC, and DSMC, in C++11. The ANN library[2] was used for maximum inner product search in SCMC to speed up the set system construction. The GLPK library[3] was used as the LP solver for constructing the dominance graph in DSMC. The dominance graph of each dataset is precomputed and provided for solution computation in DSMC. Our implementation is publicly available on https://github.com/yhwang1990/minimum-coresets. In addition, we used the implementation of the standard ANN-based algorithm (ANN for short) for ε-kernel computation in [3, 45], which is published on https://users.cs.duke.edu/~ssintos/kRMS_SEA, as a baseline. We followed the parameter settings for ANN as described in [3]. All experiments were conducted on a server running Ubuntu 18.04.1 with a 2.3 GHz processor and 256 GB memory.

Datasets: We use the following public real-world datasets in our experiments. The statistics of these datasets are reported in Table 1.

- FourSquare[4]: This dataset contains check-ins in NYC and Tokyo collected on FourSquare from 12 April 2012 to 16 February 2013. We extracted the coordinates of 37,000 distinct locations in NYC and 59,955 distinct locations in Tokyo for the experiments on two-dimensional data.

[2] http://www.cs.umd.edu/~mount/ANN
[3] https://www.gnu.org/software/glpk
[4] https://www.kaggle.com/chetanism/foursquare-nyc-and-tokyo-checkin-dataset



(a) FourSquare-NYC

(b) FourSquare-TKY

Figure 4: Performance on two-dimensional datasets with varying ε.

Table 1: Statistics of real-world datasets, where n is the dataset size, d is the dimensionality, ξ is the number of extreme points, and *DG Time (s)* is the CPU time in seconds for dominance graph construction.

Dataset	n	d	ξ	DG Time (s)
FourSquare-NYC	37,000	2	50	0.021
FourSquare-TKY	59,955	2	60	0.028
RoadNetwork	434,874	3	182	0.333
Climate	566,262	4	888	12.81
AirQuality	383,980	6	532	7.39
Colors	68,040	9	1,961	343.6

- RoadNetwork[5]: This dataset is a collection of 434,874 spatial objects in North Jutland, Denmark. Each object has three attributes: *longitude*, *latitude*, and *elevation*.
- Climate [29]: This dataset contains the average temperatures aggregated by seasons in 566,262 weather stations.
- AirQuality[6]: This dataset includes 383,980 records for the concentrations of six air pollutants collected from 12 air-quality monitoring sites in Beijing, China.
- Colors[7]: This is a collection of color moments collected from 68,040 images.

We normalized all dimensions of each dataset to the range $[-1, 1]$ in a preprocessing step.

Furthermore, to evaluate the performance of algorithms in controlled settings, we use two synthetic datasets, namely Normal and Uniform, in our experiments. In Normal, each attribute is independently drawn from the standard normal distribution and rescaled to $[-1, 1]$. In Uniform, each attribute is independently drawn from a uniform distribution in the range $[-1, 1]$. For both datasets, we vary the dataset size n from 10^3 to 10^7 and the dimensionality d from 2 to 10 to test the effect of n and d. By default, we use the datasets with $n = 10^5$ and $d = 6$.

7.1 Results on Two-Dimensional Data

In this subsection, we describe the experimental results on two-dimensional datasets – i.e., two real-world datasets FourSquare-NYC and FourSquare-TKY and one synthetic dataset Normal with $d = 2$. The results on Uniform with $d = 2$ are similar to those on Normal and thus omitted due to space limitations.

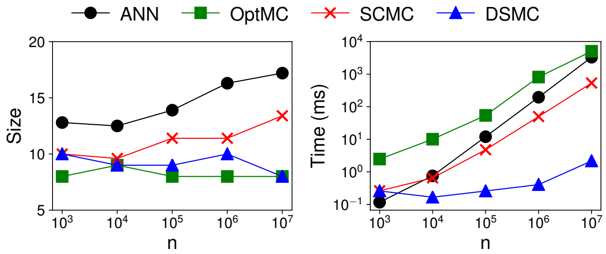


Figure 5: Performance on Normal (2D) with varying n.

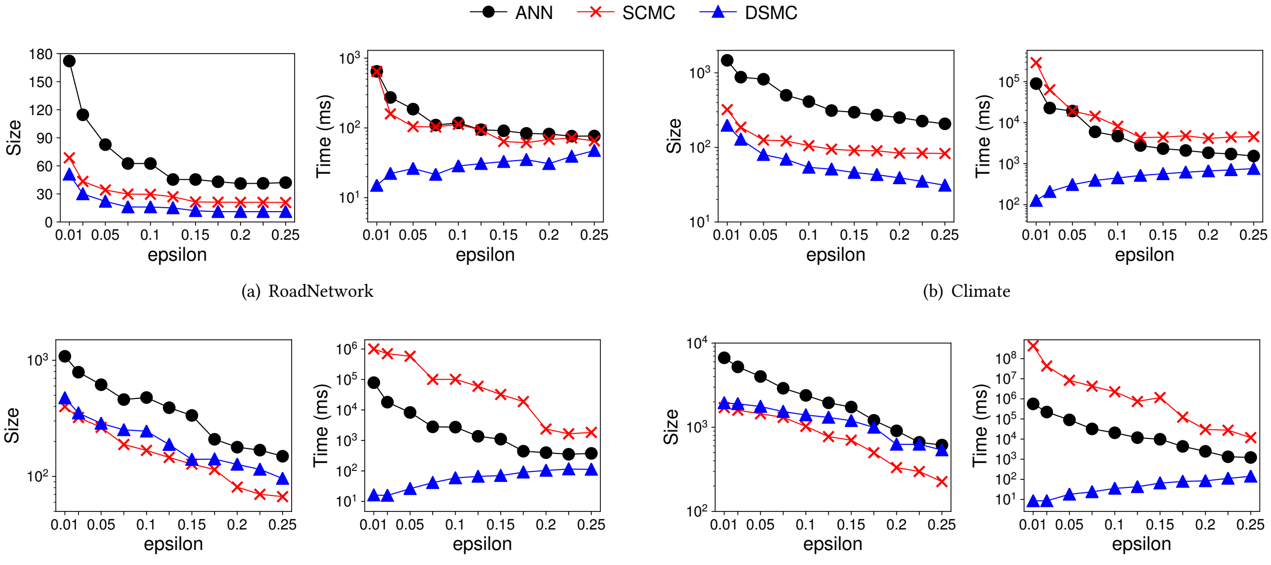
We present the performance of each algorithm in terms of solution quality (i.e., sizes of coresets found) and efficiency (i.e., running time) with varying the value of ε from 0.001 to 0.25 in Figure 4. First of all, all algorithms return smaller coresets when ε is larger. Such results are intuitive since larger ε means larger losses in maxima representation and thus smaller coresets. Among all algorithms, OptMC always provides the smallest (optimal) coresets in all cases as expected. Moreover, DSMC and SCMC provide near-optimal (or even optimal) solutions for MC in case of $d = 2$, both of which achieve significantly better solution quality than ANN. These results show that the schemes of approximating Voronoi diagrams in DSMC and SCMC are effective on two-dimensional data. In terms of efficiency, OptMC and DSMC run slower while ANN and SCMC run faster when ε increases, as the graphs used by OptMC and DSMC for solution computation have more edges for larger ε but the sample sizes of DSMC and SCMC are smaller for larger ε. Although OptMC has the slowest efficiency among four algorithms, its running time is still within one second when $\varepsilon = 0.25$. The running time of ANN and SCMC is determined by sample sizes and mostly close to each other on different datasets. The running time for the solution computation of DSMC is significantly shorter than all the other algorithms, especially when ε is small. In addition, as shown in Table 1, the running time for the dominance graph construction of DSMC is only 20-30 ms when $d = 2$ since the numbers of extreme points and edges in IPDG are very small.

Next, we fix $\varepsilon = 0.1$ and vary the size n of Normal with $d = 2$ from 10^3 to 10^7 to evaluate the scalability of each algorithm w.r.t. the size of dataset. The results are shown in Figure 5. The ranking for solution quality is the same as that of varying ε: (1) OptMC (optimal), (2) DSMC, (3) SCMC, and (4) ANN. In terms of efficiency,

[5] https://archive.ics.uci.edu/ml/datasets/3D+Road+Network+%28North+Jutland%2C+Denmark%29

[6] https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data

[7] https://archive.ics.uci.edu/ml/datasets/corel+image+features



(a) RoadNetwork

(b) Climate

(c) AirQuality

(d) Colors

Figure 6: Performance on multidimensional datasets with varying ε.

the running time of OptMC, ANN, and SCMC is linear with n, as indicated by their time complexities (note that the time complexity of OptMC is near linear when ε is small). The running time of DSMC grows very slowly with n since it computes the solution directly from the dominance graph that only contains extreme points.

For two-dimensional data, OptMC always provides an optimal solution of MC in reasonable time. In addition, both DSMC and SCMC exhibit better solution quality than ANN at all times as well as higher efficiency than ANN in most cases.

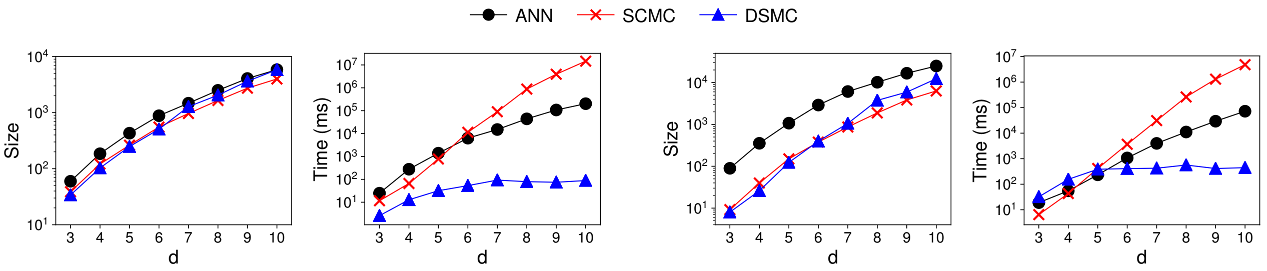
7.2 Results on Multidimensional Data

In this subsection, we describe the experimental results on datasets with $d > 2$, including four real datasets RoadNetwork, Climate, AirQuality, and Colors, and two synthetic datasets Normal and Uniform where d is ranged from 3 to 10.

We first vary ε from 0.01 to 0.25 for evaluating the performance of each algorithm. The results are presented in Figure 6. Here we do not present the results when $\varepsilon < 0.01$ because ANN and SCMC run too slow for extremely large sample sizes. Similar to the two-dimensional case, the sizes of coresets decrease with increasing ε. Both DSMC and SCMC provide solutions of significantly higher quality than ANN, particularly so for smaller values of ε. The coresets of DSMC and SCMC are up to 5 times smaller than that of ANN when $\varepsilon = 0.01$. Furthermore, DSMC shows higher solution quality than SCMC when $d \leq 5$ but performs worse than SCMC when $d > 5$. This is because DSMC uses approximate IPDGs for dominance graph construction, as building exact IPDGs is computationally expensive when $d > 3$. Therefore, in higher dimensions, a large number of edges in exact IPDGs are missing from approximate IPDGs, which leads to inaccurate edge weights in dominance graphs – and thus DSMC becomes less effective. The overall trend of running time is also similar to that of the case when $d = 2$: ANN and SCMC run faster for larger ε while DSMC runs slower. We observe that SCMC shows lower efficiency than ANN on several datasets. Although the sample sizes of both algorithms are close, ANN only finds one approximate nearest neighbor of each sampled vector for solution computation but SCMC performs one exact maximum inner product search followed by a range search to find all points that are within ε-approximation to the maximum of each sampled vector for set system construction. This causes the higher computational overhead for SCMC compared with ANN. Nevertheless, DSMC runs significantly faster than ANN and SCMC in almost all cases. In particular, when $\varepsilon = 0.01$, DSMC achieves speedups of two to four orders of magnitude than ANN and SCMC.

We also report the dominance graph construction time of DSMC on multidimensional datasets in Table 1. Since the time is determined by the number of extreme points and the dimensionality as shown in Theorem 6.3, it takes longer time for dominance graph construction on multidimensional datasets but this is still within reasonable time (from 300 ms when $d = 3$ to 6 min when $d = 9$).

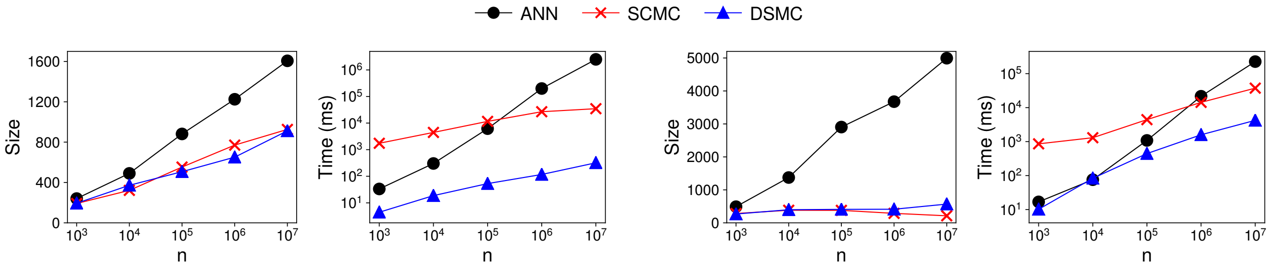
We further evaluate the scalability of each algorithm w.r.t. dimensionality d and dataset size n on two synthetic datasets. In these experiments, we fix ε to 0.1. The results for varying d and n are presented in Figures 7 and 8, respectively. Moreover, the dominance graph construction time of DSMC w.r.t. d and n is shown in Figure 9. First of all, the coreset sizes of all algorithms grow rapidly with d due to "the curse of dimensionality". But DSMC and SCMC still provide smaller coresets than ANN in different dimensions. Meanwhile, the running time of ANN and SCMC as well as the dominance graph construction time of DSMC increase rapidly with d because of the increases in sample sizes and numbers of extreme points. But the solution computation time of DSMC grows much slower since it only depends on the dominance graphs. Then, we observe that the scalabilities of DSMC and SCMC are obviously better than that of ANN. In terms of solution quality, the gaps in coreset sizes become greater when n is larger. In particular, the



(a) Normal ($n = 10^5$)

(b) Uniform ($n = 10^5$)

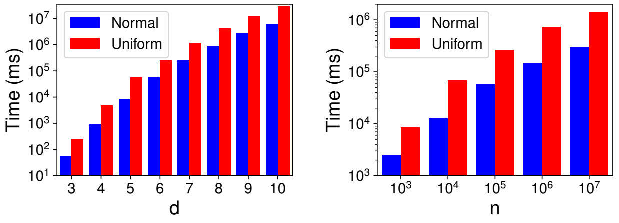
Figure 7: Performance on synthetic datasets with varying d.



(a) Normal (6D)

(b) Uniform (6D)

Figure 8: Performance on synthetic datasets with varying n.



(a) Varying d ($n = 10^5$)

(b) Varying n ($d = 6$)

Figure 9: CPU time for the dominance graph construction on synthetic datasets with varying d and n.

coreset sizes of DSMC and SCMC are 23.4x and 8.8x smaller than the coreset size of ANN on Uniform when $n = 10^7$. In terms of time efficiency, the relative performance of DSMC and SCMC also becomes better than ANN for larger n. DSMC and SCMC achieve one to four orders of magnitude speedups over ANN when $n = 10^7$.

For multidimensional data, DSMC and SCMC achieves significantly higher solution quality than ANN. Although SCMC runs slower than ANN sometimes, DSMC has much higher efficiency than ANN and SCMC, especially when the values of ε is smaller.

8 CONCLUSION

In this paper, we proposed the *minimum ε-coreset* (MC) problem to find the smallest ε-coreset for the maxima representation of a multidimensional point set. We proved the NP-hardness of MC in any constant dimension $d \geq 3$. Following a geometric interpretation of MC based on Voronoi diagrams, we designed an optimal $O(n^3)$-time algorithm for MC in two dimension and two polynomial-time approximation algorithms for MC in an arbitrary constant dimension. Finally, we conducted extensive experiments on real and synthetic datasets to evaluate the performance of our proposed algorithms. The results confirmed the effectiveness, efficiency, and scalability of our proposed algorithms for MC compared with existing algorithms in a variety of settings.

There are still many interesting open questions to answer in future work. First, although the worst-case complexities of the sizes of ε-coresets for maxima representation, ε-kernels, and ε-hulls are the same, it is still not known whether their corresponding minimization problems can be reduced to each other. Second, does DSMC have a better approximation ratio than $O(\frac{\xi}{d})$? Third, is it possible to remove the exponential dependency of SCMC on d?

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. We thank Dr. Sintos for his valuable comments on this paper. Yanhao Wang and Michael Mathioudakis have been supported by the MLDB project of the Academy of Finland (decision number: 322046).

REFERENCES

[1] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. 2004. Approximating extent measures of points. *J. ACM* 51, 4 (2004), 606–635.
[2] Pankaj K Agarwal, Sariel Har-Peled, Kasturi R Varadarajan, et al. 2005. Geometric approximation via coresets. In *Combinatorial and computational geometry*. Cambridge University Press, 1–30.
[3] Pankaj K. Agarwal, Nirman Kumar, Stavros Sintos, and Subhash Suri. 2017. Efficient Algorithms for k-Regret Minimizing Sets. In *SEA*. 7:1–7:23.
[4] Pankaj K. Agarwal, Jeff M. Phillips, and Hai Yu. 2010. Stability of *epsilon*-Kernels. In *ESA (1)*. 487–499.
[5] Pankaj K. Agarwal and Hai Yu. 2007. A space-optimal data-stream algorithm for coresets in the plane. In *SoCG*. 1–10.

[6] Boris Aronov, Micha Sharir, and Boaz Tagansky. 1997. The Union of Convex Polyhedra in Three Dimensions. *SIAM J. Comput.* 26, 6 (1997), 1670–1688.

[7] Sunil Arya and Timothy M. Chan. 2014. Better ϵ-Dependencies for Offline Approximate Nearest Neighbor Search, Euclidean Minimum Spanning Trees, and ϵ-Kernels. In *SoCG*. 416–425.

[8] Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. 2017. Near-Optimal epsilon-Kernel Construction and Related Problems. In *SoCG*. 10:1–10:15.

[9] Abolfazl Asudeh, Azade Nazi, Nan Zhang, and Gautam Das. 2017. Efficient Computation of Regret-ratio Minimizing Set: A Compact Maxima Representative. In *SIGMOD*. 821–834.

[10] Franz Aurenhammer. 1991. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM Comput. Surv.* 23, 3 (1991), 345–405.

[11] Olivier Bachem, Mario Lucic, and Andreas Krause. 2018. Scalable k-Means Clustering via Lightweight Coresets. In *KDD*. 1119–1127.

[12] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. 1996. The Quickhull Algorithm for Convex Hulls. *ACM Trans. Math. Softw.* 22, 4 (1996), 469–483.

[13] Jon Louis Bentley, Mark G. Faust, and Franco P. Preparata. 1982. Approximation Algorithms for Convex Hulls. *Commun. ACM* 25, 1 (1982), 64–68.

[14] Avrim Blum, Vladimir Braverman, Ananya Kumar, Harry Lang, and Lin F. Yang. 2018. Approximate Convex Hull of Data Streams. In *ICALP*. 21:1–21:13.

[15] Avrim Blum, Sariel Har-Peled, and Benjamin Raichel. 2016. Sparse Approximation via Generating Point Sets. In *SODA*. 548–557.

[16] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. 2013. Near-Optimal Coresets for Least-Squares Regression. *IEEE Trans. Inf. Theory* 59, 10 (2013), 6880–6892.

[17] Wei Cao, Jian Li, Haitao Wang, Kangning Wang, Ruosong Wang, Raymond Chi-Wing Wong, and Wei Zhan. 2017. k-Regret Minimizing Set: Efficient Algorithms and Hardness. In *ICDT*. 11:1–11:19.

[18] Timothy M. Chan. 2006. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.* 35, 1-2 (2006), 20–35.

[19] Timothy M. Chan. 2009. Dynamic Coresets. *Discret. Comput. Geom.* 42, 3 (2009), 469–488.

[20] Timothy M. Chan. 2016. Dynamic Streaming Algorithms for Epsilon-Kernels. In *SoCG*. 27:1–27:11.

[21] Timothy M. Chan. 2017. Applications of Chebyshev Polynomials to Low-Dimensional Computational Geometry. In *SoCG*. 26:1–26:15.

[22] Gautam Das and Michael T. Goodrich. 1997. On the Complexity of Optimization Problems for 3-dimensional Convex Polyhedra and Decision Trees. *Comput. Geom.* 8 (1997), 123–137.

[23] Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numer. Math.* 1 (1959), 269–271.

[24] Uriel Feige. 1998. A Threshold of ln n for Approximating Set Cover. *J. ACM* 45, 4 (1998), 634–652.

[25] Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. 1981. Optimal Packing and Covering in the Plane are NP-Complete. *Inf. Process. Lett.* 12, 3 (1981), 133–137.

[26] Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* 34, 3 (1987), 596–615.

[27] Sariel Har-Peled and Soham Mazumdar. 2004. On coresets for k-means and k-median clustering. In *STOC*. 291–300.

[28] Sariel Har-Peled and Manor Mendel. 2006. Fast Construction of Nets in Low-Dimensional Metrics and Their Applications. *SIAM J. Comput.* 35, 5 (2006), 1148–1184.

[29] Ian Harris, Timothy J Osborn, Phil Jones, and David Lister. 2020. Version 4 of the CRU TS monthly high-resolution gridded multivariate climate dataset. *Sci. Data* 7, 1 (2020), 1–18.

[30] Lingxiao Huang, Jian Li, Jeff M. Phillips, and Haitao Wang. 2016. epsilon-Kernel Coresets for Stochastic Points. In *ESA*. 50:1–50:18.

[31] Nirman Kumar and Stavros Sintos. 2018. Faster Approximation Algorithm for the *k*-Regret Minimizing Set and Related Problems. In *ALENEX*. 62–74.

[32] Gang Luo, Kun-Lung Wu, and Philip S. Yu. 2009. Answering linear optimization queries with an approximate stream index. *Knowl. Inf. Syst.* 20, 1 (2009), 95–121.

[33] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2020. Coresets for Data-efficient Training of Machine Learning Models. In *ICML*. 6950–6960.

[34] Stanislav Morozov and Artem Babenko. 2018. Non-metric Similarity Graphs for Maximum Inner Product Search. In *NeurIPS*. 4726–4735.

[35] Danupon Nanongkai, Atish Das Sarma, Ashwin Lall, Richard J. Lipton, and Jun (Jim) Xu. 2010. Regret-Minimizing Representative Databases. *PVLDB* 3, 1 (2010), 1114–1124.

[36] James B. Orlin and Antonio Sedeño-Noda. 2017. An $O(nm)$ time algorithm for finding the min length directed cycle in a graph. In *SODA*. 1866–1879.

[37] Peng Peng and Raymond Chi-Wing Wong. 2014. Geometry approach for k-regret query. In *ICDE*. 772–783.

[38] Franco P. Preparata and Michael Ian Shamos. 1985. *Computational Geometry - An Introduction*. Springer.

[39] Suraj Shetiya, Abolfazl Asudeh, Sadia Ahmed, and Gautam Das. 2019. A Unified Optimization Algorithm For Solving "Regret-Minimizing Representative" Problems. *Proc. VLDB Endow.* 13, 3 (2019), 239–251.

[40] Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. 2019. On Efficient Retrieval of Top Similarity Vectors. In *EMNLP/IJCNLP (1)*. 5235–5245.

[41] Yanhao Wang, Yuchen Li, and Kian-Lee Tan. 2019. Coresets for Minimum Enclosing Balls over Sliding Windows. In *KDD*. 314–323.

[42] Min Xie, Raymond Chi-Wing Wong, and Ashwin Lall. 2020. An experimental survey of regret minimization query and variants: bridging the best worlds between top-k query and skyline query. *VLDB J.* 29, 1 (2020), 147–175.

[43] Min Xie, Raymond Chi-Wing Wong, Jian Li, Cheng Long, and Ashwin Lall. 2018. Efficient k-Regret Query Algorithm with Restriction-free Bound for any Dimensionality. In *SIGMOD*. 959–974.

[44] Albert Yu, Pankaj K. Agarwal, and Jun Yang. 2012. Processing a large number of continuous preference top-*k* queries. In *SIGMOD*. 397–408.

[45] Hai Yu, Pankaj K. Agarwal, Raghunath Poreddy, and Kasturi R. Varadarajan. 2008. Practical Methods for Shape Fitting and Kinetic Data Structures using Coresets. *Algorithmica* 52, 3 (2008), 378–402.

[46] Hamid Zarrabi-Zadeh. 2011. An Almost Space-Optimal Streaming Algorithm for Coresets in Fixed Dimensions. *Algorithmica* 60, 1 (2011), 46–59.

[47] Zhixin Zhou, Shulong Tan, Zhaozhuo Xu, and Ping Li. 2019. Möbius Transformation for Fast Inner Product Search on Graph. In *NeurIPS*. 8216–8227.

A SET COVER BASED ALGORITHM

In this section, we describe the SCMC algorithm for MC by formulating it as an instance of the set cover problem. Our transformation scheme is adapted from the ones in [3, 9] for transforming RMS to set cover based on the notion of δ-net [28]. We show that such a transformation still works for MC under an assumption that P is α-fat in $[-1,1]^d$ and propose the SCMC algorithm accordingly.

Let us first introduce the background on the set cover problem. Given a set system $\Sigma = (\mathcal{U}, \mathcal{S})$ where $\mathcal{U}$ is the set of elements in the universe and $\mathcal{S}$ is a collection of subsets of $\mathcal{U}$, a subset $C \subseteq \mathcal{S}$ is called a set-cover solution of Σ if $\bigcup_{S \in C} S = \mathcal{U}$. The set cover problem asks to compute a set-cover solution of the minimum size on Σ. It is a classical NP-complete problem [24], and an $O(\log m)$-approximation greedy algorithm, where $m = |\mathcal{U}|$, is a well-known method for this problem.

The key idea of SCMC is to construct a set system on which a set-cover solution is a feasible solution for MC. The construction of such a set system is based on the following observation: Although representing the ε-approximate Voronoi cells and performing set operations on Voronoi cells are hard in $\mathbb{R}^d$ when $d \geq 3$, determining the membership of a vector u in $R_\varepsilon(p)$ for a point p is much easier. Specifically, given a point set $P \subset \mathbb{R}^d$, a vector $u \in \mathbb{S}^{d-1}$, a point $p \in P$, and a parameter $\varepsilon \in (0,1)$, we have $u \in R_\varepsilon(p)$ if and only if $\langle p, u \rangle \geq (1-\varepsilon) \cdot \omega(P, u)$. So it is possible to discretize Voronoi cells by sampling a set of vectors on $\mathbb{S}^{d-1}$ and performing the membership tests for all vectors w.r.t. each point. In this way, we can approximately represent $R_\varepsilon(p)$ by the subset of vectors in it. Intuitively, the more vectors we sample, the smaller the resulting error. This intuition is formalized by the notion of δ-net: For a given parameter $\delta \in (0,1)$, a set $\mathcal{N}$ of vectors is called a δ-net if, for any vector v in $\mathbb{S}^{d-1}$, there is a vector $u \in \mathcal{N}$ with angular distance at most δ to v. A δ-net of size $O(\frac{1}{\delta^{d-1}})$ can be computed by drawing vectors from a uniform grid on $\mathbb{S}^{d-1}$. Let $\mathcal{N}$ be a $\frac{\alpha\varepsilon}{4d}$-net of $\mathbb{S}^{d-1}$. If we construct a set system $\Sigma_{\mathcal{N}} = (\mathcal{N}, \mathcal{S}_{\mathcal{N}})$ where $S_p \subseteq \mathcal{N}$ is the subset of vectors in $R_{\varepsilon/2}(P)$ and $\mathcal{S}_{\mathcal{N}} = \{S_p : p \in P\}$, it is guaranteed that a set-cover solution of $\Sigma_{\mathcal{N}}$ will correspond to a feasible solution of MC on P for a given ε, as will be proven in Theorem A.2.

Putting everything together, we summarize the procedure of the SCMC algorithm in Algorithm 4, where the construction of the set system Σ is presented in Lines 1–6 and the greedy algorithm for solution computation on Σ is described in Lines 7–11.

Algorithm 4: SCMC

```
Input  : A point set $P$; the error parameter $\varepsilon \in (0,1)$
Output : The solution $Q_\varepsilon$ of MC on $P$
Let $\mathcal{N}$ be an $\frac{\alpha\varepsilon}{4d}$-net of $\mathbb{S}^{d-1}$;
Set $S_p = \varnothing$ for each $p \in P$;
foreach $u \in \mathcal{N}$ do
    foreach $p \in P$ do
        $S_p \leftarrow S_p \cup \{u\}$ if $\langle p, u\rangle \geq (1-\frac{\varepsilon}{2}) \cdot \omega(P,u)$;
Construct $\Sigma_{\mathcal{N}} = (\mathcal{N}, \mathcal{S}_{\mathcal{N}})$ where $\mathcal{S}_{\mathcal{N}} = \{S_p : p \in P\}$;
Initialize $Q_\varepsilon = \varnothing$ and $U' = \mathcal{N}$;
while $U' \neq \varnothing$ do
    $p^* \leftarrow \arg\max_{p \in P \setminus Q_\varepsilon} |S_{p^*} \cap U'|$;
    $Q_\varepsilon \leftarrow Q_\varepsilon \cup \{p^*\}$ and $U' \leftarrow U' \setminus S_{p^*}$;
return $Q_\varepsilon$;
```

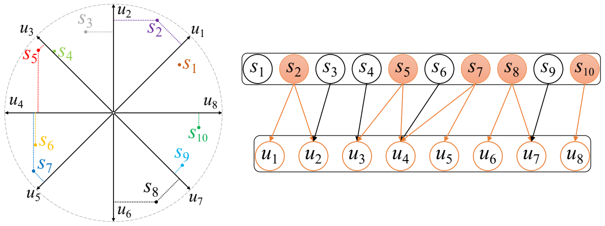


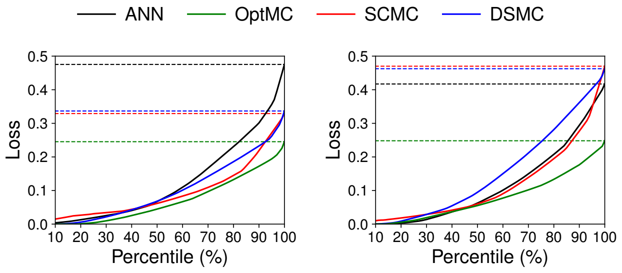
Figure 10: An illustration of SCMC. We first draw 8 vectors $\{u_1, \ldots, u_8\}$ uniformly from $\mathbb{S}^1$ for computation. Then, for each vector, we show the points whose inner products are within a 0.1-approximation to the maximum. Next, we construct the set system and represent it as a bipartite graph. Finally, we run the greedy algorithm to compute the solution of MC (highlighted in orange).

Theoretical Analysis: We first prove the correctness of the transformation from MC to the set cover problem in Lemma A.1. Note that Lemma A.1 is a slight variant of Lemma 8 for the RMS_HS algorithm in [3]. There are two differences between them: (1) some parameter values are different since SCMC is based on the α-fatness of P but RMS_HS is based on the fact that P is at least $\frac{1}{\sqrt{d}}$-fat when scaled to $[0,1]^d$; (2) the analyses of "basis points" in RMS_HS is not necessary anymore for SCMC and thus removed.

Lemma A.1. *Let $\mathcal{N}$ be a $\frac{\alpha\delta}{d}$-net of $\mathbb{S}^{d-1}$ and $\mathcal{S}_{\mathcal{N}}$ be a collection of subsets of $\mathcal{N}$ that represent the γ-approximate Voronoi cells of all points in P for some $\gamma \in (0,1)$. If C is a set-cover solution on $\Sigma_{\mathcal{N}} = (\mathcal{N}, \mathcal{S}_{\mathcal{N}})$, then $Q = \{p \in P : S_p \in C\}$ satisfies that $l(Q) \leq 2\delta + \gamma$.*

Proof. As discussed in Section 2, we assume that P is an α-fat point set in $[-1,1]^d$. Without loss of generality, we further consider that there exists at least one point in P with value -1 or 1 on each dimension. Then, we have $\max_{u \in \mathbb{S}^{d-1}} \omega(P,u) \geq 1$ and $\min_{u \in \mathbb{S}^{d-1}} \omega(P,u) \geq \alpha$ for the α-fatness of P, i.e., $\omega(P,u) \geq \alpha$ for every $u \in \mathbb{S}^{d-1}$. For any $u \in \mathbb{S}^{d-1}$, there exists a vector $u' \in \mathcal{N}$ such that $\angle uOu' \leq \frac{\alpha\delta}{d}$. According to the cosine rule, we have:

$$\|u - u'\| = \sqrt{2 - 2\cos\angle uOu'} = 2\sin\frac{\angle uOu'}{2} \leq \frac{\alpha\delta}{d} \qquad (3)$$



(a) FourSquare-NYC

(b) FourSquare-TKY

Figure 11: Loss distributions ($r = 5$).

Therefore, for any point $p \in P$, we have:

$$|\langle p, u\rangle - \langle p, u'\rangle| = |\langle p, u - u'\rangle| \leq \|p\| \cdot \|u - u'\| \leq \sqrt{d} \cdot \frac{\alpha\delta}{d} < \alpha\delta$$

where the first inequality is based on the Cauchy-Schwarz inequality, the second inequality is acquired from Eq. 3 and $\|p\| \leq \sqrt{d}$ because $p \in [-1,1]^d$, and the third inequality naturally holds for any $d > 1$. Let p^* be the extreme point in direction u – i.e., $p^* = \arg\max_{p \in P} \langle p, u\rangle$. Then, we have:

$$\langle p^*, u'\rangle \geq \langle p^*, u\rangle - \alpha\delta \geq (1-\delta) \cdot \langle p^*, u\rangle = (1-\delta) \cdot \omega(P,u) \qquad (4)$$

Let q^* be the point in Q such that $u' \in S_{q^*}$. Such a point q^* must exist because C is a set-cover solution. We have:

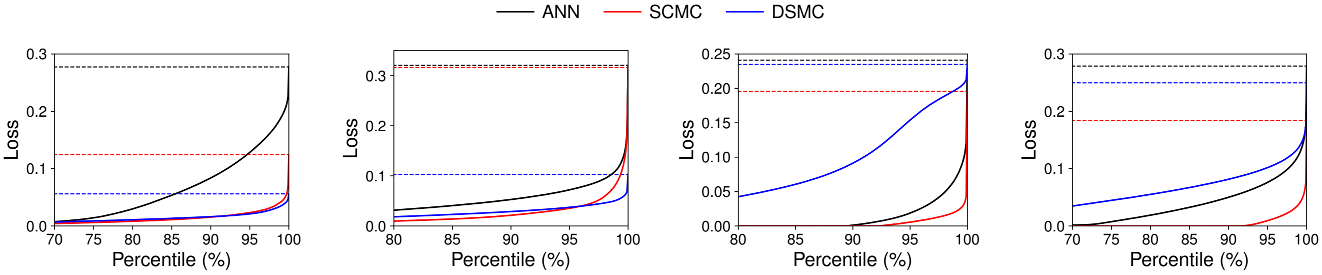
$$\begin{aligned}
\langle q^*, u\rangle &\geq \langle q^*, u'\rangle - \alpha\delta \\
&\geq (1-\gamma) \cdot \omega(P,u') - \alpha\delta \\
&\geq (1-\gamma) \cdot \langle p^*, u'\rangle - \alpha\delta \\
&\geq (1-\gamma)(1-\delta) \cdot \omega(P,u) - \delta \cdot \omega(P,u) \\
&> (1 - 2\delta - \gamma) \cdot \omega(P,u)
\end{aligned}$$

where the first inequality is the same as Eq. 4, the second inequality holds from $u' \in S_{q^*}$, the third inequality is based on $\omega(P,u') = \max_{p \in P} \langle p, u'\rangle$, the fourth inequality is the result of Eq. 4 as well as $\omega(P,u) \geq \alpha$, and the fifth inequality is due to $\delta, \gamma \in (0,1)$. According to the above results, we conclude that there is a point $q \in Q$ such that $\langle q, u\rangle \geq (1 - 2\delta - \gamma) \cdot \omega(P,u)$ for any $u \in \mathbb{S}^{d-1}$ and thus $l(Q) \leq 2\delta + \gamma$. □

The SCMC algorithm is a special case of Lemma A.1 when $\delta = \frac{\varepsilon}{4}$ and $\gamma = \frac{\varepsilon}{2}$. We provide the approximation factor and time complexity of SCMC in Theorem A.2. Note that Theorem A.2 is a variation of Theorem 9 in [3], which considers all points and vectors in $\mathbb{R}^d$ for MC instead of only nonnegative ones for RMS.

Theorem A.2. *For a point set $P \subset \mathbb{R}^d$ and a parameter $\varepsilon \in (0,1)$, SCMC returns a solution Q_ε such that (1) $l(Q_\varepsilon) \leq \varepsilon$ and (2) $|Q_\varepsilon| = O(d \cdot \mathtt{OPT}_{\varepsilon/2} \cdot \log\frac{1}{\varepsilon})$ in $O(\frac{n}{\varepsilon^{d-1}})$ time, where $\mathtt{OPT}_{\varepsilon/2}$ is the size of the smallest $\frac{\varepsilon}{2}$-coreset of P.*

Proof. First of all, since we use $\delta = \frac{\varepsilon}{4}$ and $\gamma = \frac{\varepsilon}{2}$ in the SCMC algorithm, it is obvious that $l(Q_\varepsilon) \leq \varepsilon$ according to Lemma A.1. In addition, since $\gamma = \frac{\varepsilon}{2}$, any $\frac{\varepsilon}{2}$-coreset $Q_{\varepsilon/2}$ of P must correspond to a set-cover solution C on $\Sigma_{\mathcal{N}}$. Otherwise, once C is not a set-cover solution, we will find an uncovered vector u such that $\langle q, u\rangle < (1-\frac{\varepsilon}{2}) \cdot \omega(P,u)$ for all $q \in Q_{\varepsilon/2}$ and thus $Q_{\varepsilon/2}$ is not an $\frac{\varepsilon}{2}$-coreset.



(a) RoadNetwork ($r = 20$)

(b) Climate ($r = 50$)

(c) AirQuality ($r = 100$)

(d) Colors ($r = 500$)

Figure 12: Loss distributions on multidimensional datasets with fixed coreset sizes.

Therefore, the optimal set-cover solution on $\Sigma_{\mathcal{N}}$ has at most the same size of the smallest $\frac{\varepsilon}{2}$-coreset of P. Since we use an $O(\log m)$-approximation greedy algorithm to compute the set-cover solution, where $m = O(\frac{1}{\varepsilon^{d-1}})$, the solution Q_{ε} of SCMC satisfies that $|Q_{\varepsilon}| = O(d \cdot \mathsf{OPT}_{\varepsilon/2} \cdot \log \frac{1}{\varepsilon})$, where $\mathsf{OPT}_{\varepsilon/2}$ is the size of the smallest $\frac{\varepsilon}{2}$-coreset of P. Finally, the construction of $\Sigma_{\mathcal{N}}$ runs in $O(\frac{n}{\varepsilon^{d-1}})$ time while the greedy algorithm takes $O(\frac{|Q_{\varepsilon}|}{\varepsilon^{d-1}})$ time. So the time complexity of SCMC is $O(\frac{n}{\varepsilon^{d-1}})$ in total. □

Remark: The values of δ and γ in SCMC are not limited to $\frac{\varepsilon}{4}$ and $\frac{\varepsilon}{2}$. In fact, they can take any positive real numbers such that $2\delta + \gamma \leq \varepsilon$. When δ is larger, SCMC will run faster because of smaller sample sizes. Conversely, when γ is larger, SCMC will provide better solutions since Q^*_{γ} will be closer to Q^*_{ε}. Additionally, since the sampling complexity of SCMC increases exponentially with d and thus becomes impractical when d is large, an alternative sampling strategy is often adopted in the implementation. Rather than sampling all the $O(\frac{1}{\varepsilon^{d-1}})$ vectors at once, we perform the sampling iteratively in multiple stages. In the first stage, we sample m vectors from $\mathbb{S}^{d-1}$ and compute a solution Q on the sampled vectors using Algorithm 4. Then, if $l(Q) \leq \varepsilon$, we return Q as the solution for MC. Otherwise, we double the sample size m, sample new vectors from $\mathbb{S}^{d-1}$, and compute a new solution Q in the next stage until $l(Q) \leq \varepsilon$.

Finally, after the acceptance of this paper, we notice that Algorithm 1 in [31] can also be adapted for MC. According to the analyses in [31], the adapted algorithm would have an improved time complexity of $O\big(n + \frac{1}{\varepsilon^{d-1}} + \frac{\log^2 \frac{1}{\varepsilon}}{\varepsilon^{3(d-1)/2}}\big)$ over SCMC.

B ADDITIONAL EXPERIMENTS ON LOSS DISTRIBUTIONS

According to the definition of ε-coreset, each algorithm only restricts the largest loss among all vectors. To provide more detailed information about the losses of coresets, we draw one million vectors and compute the entire distributions of losses of the coresets returned by different algorithms for all vectors. For fair comparison, we limit the coresets returned by different algorithms to the same size r. We run each algorithm with different values of ε to find the smallest one such that the coreset size is at most r.

The results for loss distributions for $r = 5$ on two-dimensional datasets are illustrated in Figure 11, where solid lines denote the losses of different algorithms by percentiles and dash horizontal lines present the largest losses among all vectors. We observe that OptMC has the smallest losses in almost all percentiles due to its optimality. In addition, SCMC exhibits smaller losses than ANN and DSMC in most percentiles because it tends to select the points that are among the maximum in more directions into the solution and naturally leads to smaller average losses.

We illustrate the loss distributions of the fixed-size coresets returned by different algorithms on multidimensional datasets in Figure 12. We also use solid lines to denote the losses of different algorithms by percentiles and dash horizontal lines to present the largest losses among all vectors. The coresets of SCMC exhibit smaller losses than ANN in different quantiles. The coresets of DSMC exhibit the smallest losses on RoadNetwork and Climate but the largest (average) losses on AirQuality and Colors. The results indicate that the dominance graph built on an approximate IPDG with a large number of missing edges becomes less effective in higher dimensions. In addition, the results on loss distributions also imply possible improvements for ANN and SCMC. Both methods sample vectors uniformly from spheres for solution computation. But in practice, we observe that the losses of most vectors have been dropped to 0 while the losses for a few vectors (resp. very small "corner" regions) are still larger than the given ε. Therefore, if we could identify these "corner" regions and do the sampling from them first, ANN and SCMC would provide valid solutions more efficiently. Although the sampling complexities of ANN and SCMC are known to be optimal in the worst case without considering the data distribution, lower running time might be achieved by adopting an alternative data-dependent sampling strategy.