Robust Wasserstein k-center Clustering: Algorithms and Acceleration

Anonymous Authors[1]

Abstract

The k-center problem is widely used in data representation tasks. However, real-world datasets often contain noise and exhibit complex structures, making the traditional k-center problem insufficient for such scenarios. To address these issues, we introduce the **R**obust **W**asserstein **C**enter clustering (RWC-clustering) problem, and provide both initialization and post-processing algorithms to solve it. Nevertheless, when dealing with large-scale datasets, the storage and computation become highly resource-intensive. To mitigate this, we design a *coreset* method to improve the computational and storage efficiency by compressing the dataset. Roughly speaking, this coreset method allows us to calculate the objective value on a small-size coreset, while ensuring a close approximation to the value on the original dataset in theory; thus, it substantially saves the storage and computation resources. Finally, experimental results show the effectiveness and efficiency of our approaches.

1. Introduction

The k-center problem (Hakimi, 1964) is widely used in data compression (Łacki et al., 2024) and representation learning (Bateni et al., 2023). Its objective is to select k centers, forming a k-center set C, such that the maximum distance from any data point to its closest center is minimized. More formally, for a given dataset Q in metric space $(\mathcal{X}, \mathrm{dist})$, the k-center problem can be formulated as

$$\min_{C\subseteq\mathcal{X},|C|=k} \max_{\mu\in Q} \min_{\nu\in C} \mathrm{dist}(\mu,\nu). \qquad (k\text{-center problem})$$

This problem is equivalent to covering the dataset Q with k balls of equal radius, while minimizing the radius. The centers of these k balls form the k-center set.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Data in combinatorial optimization (Luo et al., 2023; Grinsztajn et al., 2023; Drakulic et al., 2023) and biomedical fields (Thual et al., 2022; Bazeille et al., 2019) often exhibit complex structures and are typically represented as probability distributions. Nevertheless, the traditional Euclidean distance falls short in describing the geometric structure of such data. In contrast, the Wasserstein distance (Peyré et al., 2017) skills at capturing the geometric structure, making it a powerful tool for quantifying the difference between these complex data items.

However, the real-world datasets are often contaminated by noise, and the Wasserstein distance is sensitive to outliers (Nietert et al., 2022) due to its stringent marginal constraints. Specifically, even a single outlier with negligible mass can substantially distort the final result by adjusting its position, thereby limiting its utility in practical scenarios. To address this issue, we adopt the Robust Wasserstein Distance (RWD) (Nietert et al., 2022) to measure the similarity between the data items. Based on this, we introduce the **R**obust **W**asserstein **C**enter clustering (RWC-clustering) problem (in Definition 2.1) to effectively represent these complex datasets.

Solving the RWC-clustering is a typical non-convex optimization problem. Initialization is crucial for non-convex optimization, as it directly affects whether the optimization algorithm can escape the local minima and effectively find the global optimum. In the k-center problem, Gonzalez's algorithm (Gonzalez, 1985) is often used as a seeding algorithm to provide a good initialization; a local search algorithm (Lattanzi & Sohler, 2019; Choo et al., 2020) is then used as a post-processing step to further refine the solution. Inspired by these methods, we design a corresponding initialization and post-processing algorithm for our RWC-clustering problem.

Except for designing algorithms, scalability is also a key consideration. When handling large datasets, solving the RWC-clustering problem becomes extremely time-consuming and requires significant storage space. To address this issue, we introduce *coreset* (Ros & Guillaume, 2020), a widely used data compression technique. A coreset can be regarded as a summary of the original dataset with respect to certain objective; it enhances computational and storage efficiency by reducing the dataset size. Roughly

speaking, it enables us to approximate the value computed on the original dataset by the value on a small-size coreset. Thus, it helps save computational and storage resources substantially while maintaining accuracy closely.

Although many coreset techniques(Huang et al., 2024; Huang et al.; 2023) have been developed for the classical clustering problems, they are primarily designed for metric spaces. However, RWD is not a metric; thus, although existing techniques may provide useful insights, new theoretical analysis is still necessary for the design of our coreset.

Our contributions:

- For effectively representing datasets with complex structures and outliers, we introduce the RWC-clustering problem. To solve this problem, we propose a seeding algorithm based on Gonzalez's algorithm (Gonzalez, 1985) to obtain a proper initialization. Following this, we design a local search-based post-processing algorithm (Lattanzi & Sohler, 2019; Choo et al., 2020) to further refine the solution.
- Then, to enhance scalability, we design a coreset method. More specifically, we design an algorithm to compute the lower bound of the RWC-clustering problem. Based on this lower bound, a coreset construction scheme was developed to accelerate the computation by compressing the dataset. Additionally, we theoretically demonstrate that the coreset is a good proxy of the original dataset.
- We experimentally demonstrated the effectiveness of our RWC-clustering problem and seeding algorithm, as well as the efficiency of the coreset method.

1.1. Other related works

Optimal transport (OT) is a popular tool for quantifying the difference between probability measures. Several algorithms have been developed for solving the OT problem. Peyré et al. (2017) introduced an ϵ_+-approximation algorithm by using the interior point method within $\widetilde{\mathcal{O}}(n^3)$ time, where ϵ_+ denotes the additive error. Subsequently, Dvurechensky et al. (2018) proposed the Sinkhorn's algorithm, which reduces the time complexity to $\widetilde{\mathcal{O}}(n^2/\epsilon_+^2)$ by solving the entropic regularization version (Cuturi, 2013). Especially, Jambulapati et al. (2019) further improved this result by leveraging the area-convexity and dual extrapolation techniques, achieving $\widetilde{\mathcal{O}}(n^2/\epsilon_+)$ time complexity.

Gonzalez's algorithm (Gonzalez, 1985), a 2-approximation algorithm for the k-center problem, is often used as an initialization method in clustering tasks. It iteratively selects the point farthest from the currently chosen centers as the new center. The sequential nature of center selection leads to dependencies between steps, which poses challenges for achieving parallel computation. It takes $\mathcal{O}(mk)$ time, where m is the size of the dataset, and k represents the number of centers. When k or m is large, the computational complexity becomes a bottleneck.

Hierarchical Gonzalez's algorithm (Murtagh & Contreras, 2012) is a variation of the Gonzalez's algorithm tailored to address hierarchical clustering problems. This algorithm constructs a tree structure by recursively splitting data at different levels of granularity. It selects cluster centers sequentially within localized regions using the Gonzalez's algorithm while incorporating a globally parallelizable design, resulting in high efficiency.

We can utilize the Hierarchical Gonzalez's algorithm to compress datasets before applying the k-center algorithm. By combining both methods, the scalability of the existing k-center algorithms can be significantly improved.

2. Preliminaries

Notations: We adopt some notation conventions from (Nietert et al., 2022; Wang et al., 2024). We define $[n] := \{1, \dots, n\}$. Let $(\mathcal{X}, \mathrm{dist})$ be a metric space and $\mathbb{R}_+$ be the set of non-negative real numbers. We use $\mathcal{M}_+(\mathcal{X})$ to denote the positive measure space on $\mathcal{X}$, and $\mathcal{P}(\mathcal{X})$ the corresponding probability measure space.

Matrices are denoted by capital boldface letters, such as $\mathbf{P}$; P_{ij} denotes its element in the i-th row and j-th column. Similarly, vectors are represented by lowercase boldface letters, such as $\mathbf{a} := (a_1, \dots, a_d)^T \in \mathbb{R}^d$; a_i is its i-th element. Let $|Q|$ be the cardinality of the set Q. For measures $\mu', \mu \in \mathcal{M}_+(\mathcal{X})$, the notation $\mu' \le \mu$ means that $\mu'(A) \le \mu(A)$ for any set $A \subseteq \mathcal{X}$.

Wasserstein distance: Let $\mu = \sum_{i=1}^n a_i \delta_{x_i}, \nu = \sum_{j=1}^n b_j \delta_{y_j}$ be two discrete probability measures [1] in $\mathcal{P}(\mathcal{X})$, where $\mathbf{a} = (a_1, \dots, a_n)^T, \mathbf{b} = (b_1, \dots, b_n)^T \in \mathbb{R}_+^n$ are their weight vectors and δ is the Dirac delta function. Given any real number $z \ge 1$ and a cost matrix $\mathbf{D} \in \mathbb{R}_+^{n \times n}$ with $D_{ij} = \mathrm{dist}^z(x_i, y_j)$, the z^{th}-Wasserstein distance between μ and ν is defined as

$$W(\mu, \nu) := \left(\min_{\mathbf{P} \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{P}, \mathbf{D} \rangle \right)^{1/z}, \tag{1}$$

where $\Pi(\mathbf{a}, \mathbf{b}) := \left\{ \mathbf{P} \in \mathbb{R}_+^{n \times n} \mid \mathbf{P}\mathbf{1} = \mathbf{a}, \mathbf{P}^T\mathbf{1} = \mathbf{b} \right\}$ is the set of all feasible couplings and $\mathbf{1}$ is the vector of all ones.

Optimal Transport (OT) shares a similar formulation with Wasserstein distance, but their cost matrices differ. The

[1] To simplify the expression, the support size of all measures in this paper is set to n.

cost matrix in OT is derived from a positive function. In contrast, the cost matrix for Wasserstein distance has stricter requirements—it must be induced by a distance function. Thus, the Wasserstein distance is a metric, while OT is not necessarily one. Despite these differences, OT algorithms can still be effectively used to compute Wasserstein distance.

Robust Wasserstein distance: Although the Wasserstein distance (Villani et al., 2009; Peyré et al., 2017) is widely used for measuring the difference between two probability measures, its sensitivity to outliers limits its applicability in noisy scenarios. To overcome this limitation, several robust variants (Nietert et al., 2022; Le et al., 2021; Chapel et al., 2020) have been proposed. This paper focuses on the following robust version.

Definition 2.1 (Robust Wasserstein distance (Nietert et al., 2022; Wang et al., 2024))**.** Let μ and ν be the same as in Equation (1). Given two pre-specified parameters $0 \leq \zeta_\mu, \zeta_\nu < 1$, the robust Wasserstein distance $\widetilde{\mathcal{W}}(\mu, \nu)$ between μ and ν is formulated as

$$\widetilde{\mathcal{W}}(\mu,\nu) := \min_{\substack{\mu',\nu' \in \mathcal{M}_+(\mathcal{X}) \\ \mu' \leq \mu, \|\mu-\mu'\|_{\mathrm{TV}}=\zeta_\mu \\ \nu' \leq \nu, \|\nu-\nu'\|_{\mathrm{TV}}=\zeta_\nu}} W\left(\frac{\mu'}{1-\zeta_\mu}, \frac{\nu'}{1-\zeta_\nu}\right), \tag{2}$$

where $\|\cdot\|_{\mathrm{TV}}$ denotes the total variation (TV) norm.

Nietert et al. (2022) claimed that $\widetilde{\mathcal{W}}(\cdot,\cdot)$ almost achieves the minimax optimal robust estimation under the Huber contamination model (HUBER, 1964). The total mass of $\frac{\mu'}{1-\zeta_\mu}, \frac{\nu'}{1-\zeta_\nu}$ is exactly 1, making them valid probability measures. Thus, $\frac{\mu'}{1-\zeta_\mu}, \frac{\nu'}{1-\zeta_\nu}$ can be regarded as robust proxies for the probability measures μ, ν, respectively.

Moreover, Equation (2) can be reformulated as an (augmented) OT problem (Wang et al., 2024) by introducing a dummy point, allowing it to be solved efficiently by using the existing OT solvers.

Note: Henceforth, we denote the Wasserstein distance between μ and ν by $W(\mu, \nu)$. The notation $\widetilde{\mathcal{W}}(\mu, \nu)$ represents the robust Wasserstein distance when both μ and ν contain ζ mass of outliers. Specially, $\mathcal{W}(\mu, \nu)$ refers to the case where μ contains ζ mass of outliers while ν has no outliers.

Robust clustering: We propose a robust version of the Wasserstein k-center clustering problem. Its goal is to cover all data points using k balls of equal radius under robust Wasserstein distance $\mathcal{W}(\cdot,\cdot)$, while minimizing the radius of these balls.

Definition 2.2 (RWC-clustering)**.** Given a set of probability measures $Q = \{\mu^i\}_{i\in[m]} \subseteq \mathcal{P}(\mathcal{X})$, the k-RWC-clustering problem is to find a k-center set $C \subseteq \mathcal{P}(\mathcal{X})$ with $|C| = k$ such that the following objective is minimized.

$$\mathsf{cost}(Q, C) := \max_{\mu \in Q} \mathcal{W}(\mu, C),$$

where $\mathcal{W}(\mu, C) := \min_{\nu \in C} \mathcal{W}(\mu, \nu)$.

The input data points in Q contain outliers. However, our goal is to obtain clean cluster centers. Thus, we define the RWC-clustering problem using $\mathcal{W}(\cdot,\cdot)$ instead of $\widetilde{\mathcal{W}}(\cdot,\cdot)$. Further illustrations are provided in Section 3.

Coreset: When the dataset is large, both computation and storage become resource-intensive. To address this issue, we introduce, coreset, a popular data compression technique.

Definition 2.3 (Coreset)**.** Given a set of probability measures $Q = \{\mu^i\}_{i\in[m]} \subseteq \mathcal{P}(\mathcal{X})$ and a real number $\epsilon > 0$, a set S is an ϵ-coreset for the k-RWC-clustering problem on Q, if the following inequality holds for all k-center set $C \subseteq \mathcal{P}(\mathcal{X})$.

$$|\mathsf{cost}(Q, C) - \mathsf{cost}(S, C)| \leq \epsilon \cdot \mathsf{cost}(Q, C)$$

Essentially, a coreset is a small proxy of the original dataset. To approximate the objective value, we can execute algorithms on this small-size coreset instead of the full dataset. Overall, this approach significantly reduces computational and storage requirements while preserving the objective value.

Organization: This paper is organized as follows. In Section 3, we design a seeding algorithm to obtain a proper initialization, and further propose a post-processing algorithm to refine the solution. In Section 4, we present a method for calculating a lower bound of the RWC-clustering problem; based on this, we design a coreset algorithm to accelerate the computation. Finally, in Section 5, we validate the effectiveness of the proposed methods through experiments.

3. Our Algorithms

This section introduces our algorithms for solving the RWC-clustering problem. Section 3.1 presents an initialization method inspired by Gonzalez's algorithm. Section 3.2 provides a post-processing algorithm based on local search to further refine the solution.

Intuition of RWC-clustering problem: In our RWC-clustering problem, we essentially replace the metric $\mathrm{dist}(\cdot,\cdot)$ in k-center problem with $\mathcal{W}(\cdot,\cdot)$. To illustrate why $\mathcal{W}(\mu, \nu)$ is chosen to measure the distance between a data point μ and its center ν, rather than using $\widetilde{\mathcal{W}}(\mu, \nu)$, we consider the following example.

Example 3.1 (Intuition). Let $x_0 = (0, 0)$ and $x_1 = (0, 1000)$ be two points in $\mathbb{R}^2$. Let $\mu^0 = \delta_{x_0}$ and $\mu^1 = \delta_{x_1}$

be two data points, and let $\nu = 0.5 \cdot \delta_{x_0} + 0.5 \cdot \delta_{x_1}$ be a center. Here, we set $\zeta = 0.5$.

Case1: When employing $\widetilde{\mathcal{W}}(\cdot,\cdot)$ to measure the differences, we have $\widetilde{\mathcal{W}}(\mu^0,\nu) = 0$, $\widetilde{\mathcal{W}}(\mu^1,\nu) = 0$ and $\widetilde{\mathcal{W}}(\mu^0,\mu^1) = 1000$. In this case, both μ^0 and μ^1 are contained within a ball of arbitrarily small radius centered at ν under $\widetilde{\mathcal{W}}(\cdot,\cdot)$. However, the difference between μ and ν can be large. In other words, two points within a small ball could exhibit significant differences. Nevertheless, the goal of clustering is to group similar points together. This situation is obviously unreasonable and contradicts the goal of clustering.

Case2: In contrast, when using $\mathcal{W}(\cdot,\cdot)$, if both μ^0 and μ^1 lie within a small-radius ball centered at ν, their difference under $\widetilde{\mathcal{W}}(\cdot,\cdot)$ remains small. This implies that points within the same small ball exhibit high similarity, which is consistent with the goal of the traditional clustering. (The detailed proofs supporting this claim are provided in Lemma B.1.)

Based on this analysis, it is more reasonable to define the RWC-clustering problem using $\mathcal{W}(\cdot,\cdot)$. Naturally, the centers in RWC-clustering problem should be clean.

3.1. Seeding algorithm

Our Algorithm 1 is inspired by Gonzalez's algorithm (Gonzalez, 1985). It takes as input a set Q of probability measures and a parameter k, and outputs a k-center set C consisting of k probability measures. This provides a good initialization for the subsequent optimization in the post-processing stage.

The original Gonzalez's algorithm (Gonzalez, 1985) selects centers directly from the input set Q. In our RWC-clustering scenarios, we desire clean centers. However, the points in set Q contains outliers. Therefore, after selecting a candidate center, a purification step is required to remove the outliers.

Specifically, we select the first candidate center[2] ν from the set Q uniformly at random, perform a purification step to obtain a clean center $\tilde{\nu}$, and add it to the center set C. For the subsequent $k-1$ epochs, during each epoch, we select a point $\nu \in Q$ that is the farthest from the center set C under $\mathcal{W}(\cdot,\cdot)$; that is, ν satisfies that

$$\nu \in \arg\max_{\nu' \in Q} \mathcal{W}(\nu', C). \tag{3}$$

Then, we perform the purification step on ν to obtain a clean center $\tilde{\nu}$, and add $\tilde{\nu}$ to C.

Purification step: Select the τ closest points to the candidate center ν from the set Q under $\widetilde{\mathcal{W}}(\cdot,\cdot)$; that is,

[2]In our paper, the candidate center contains outliers, while the center is clean.

Algorithm 1 Seeding

Input: a set $Q = \{\mu^i\}_{i\in[m]}$ of probability measures, and a parameter k
Initialize the center set as $C = \emptyset$.
for $i = 1$ **to** k **do**
  ▷Select candidate center ν
  if $i = 1$ **then**
    Sample a measure ν from Q uniformly at random.
  else
    Select the point ν that is farthest from the center set C under $\mathcal{W}(\cdot,\cdot)$ according to Equation (3).
  end if
  ▷Purification step: purify ν to obtain $\tilde{\nu}$
  Perform the purification step on candidate center ν, and obtain its corresponding clean center $\tilde{\nu}$ according to Equations (4) to (6).
  Add $\tilde{\nu}$ to center set C.
end for
Output: a k-center set C

$$D \in \arg\min_{D' \subseteq Q, |D'|=\tau} \sum_{\mu \in D'} \widetilde{\mathcal{W}}(\mu,\nu). \tag{4}$$

These τ points in D can induce [3] τ clean centers $\tilde{\nu}'$.

$$\widetilde{C} = \left\{ \tilde{\nu}' \mid \widetilde{\mathcal{W}}(\mu,\nu) = \mathcal{W}(\mu,\tilde{\nu}'), \mu \in D \right\} \tag{5}$$

Then, choose the point $\tilde{\nu} \in \widetilde{C}$ that covers all the points in D with the smallest radius under $\mathcal{W}(\cdot,\cdot)$; that is,

$$\tilde{\nu} \in \arg\min_{\tilde{\nu}' \in \widetilde{C}} \max_{\mu \in D} \mathcal{W}(\mu,\tilde{\nu}'). \tag{6}$$

Then, $\tilde{\nu}$ in Equation (6) is the corresponding purified clean center of candidate center ν. After the purification step, the locations of the candidate center ν and clean center $\tilde{\nu}$ remain unchanged; only the weights are adjusted.

Remark 3.2 (Intuition of the purification step). **i)** Both the candidate center ν and the data points $\mu \in Q$ contain outliers. Thus, we use $\widetilde{\mathcal{W}}(\cdot,\cdot)$ to measure the similarity between μ and ν. **ii)** Since the candidate center ν induces different clean centers for different $\mu \in Q$, we retain the smallest τ values of $\widetilde{\mathcal{W}}(\cdot,\nu)$ in Equation (5), rather than selecting only one. **iii)** Note that ν is a candidate center of certain cluster. Points from other clusters mixed into D can damage the purification of the candidate center. Therefore, to minimize the impact of points from other clusters, the parameter τ should not be too large.

[3]In Equation (5), for each μ, there may exist infinitely many $\tilde{\nu}'$ that satisfy the condition, but we select only one of them.

3.2. Post-processing algorithm

Algorithm 2 Post-processing

Input: a set $Q = \{\mu^i\}_{i\in[m]}$ of probability measures, and a k-center set C
for $i = 1$ **to** Z **do**
  ▷Sampling
  Sample $\nu \in Q$ with probability $\frac{\mathcal{W}(\nu,C)}{\sum_{\nu'\in Q}\mathcal{W}(\nu',C)}$.
  ▷Purification
  Purify ν into $\tilde{\nu}$ according to Equations (4) to (6).
  ▷Swapping
  if $\exists \nu' \in C$, s.t., $\mathsf{cost}(Q, C\setminus\{\nu'\}\cup\{\tilde{\nu}\}) < \mathsf{cost}(Q, C)$ **then**
    $C = C\setminus\{\nu'\}\cup\{\tilde{\nu}\}$.
  end if
end for
Output: a k-center set C

Post-processing algorithm: Algorithm 2 is inspired by the local search algorithm (Lattanzi & Sohler, 2019; Choo et al., 2020), and serves as a post-processing procedure for Algorithm 1 to further refine the solution. The input is a set Q of probability measures and an initialized solution (i.e., k-center set), while the output is the refined solution.

The solution C is refined over Z epochs, with each epoch consisting of three steps: sampling, purification, and swapping. Specifically, in each epoch, we sample a candidate center $\nu \in Q$ according to a probability proportional to its cost, i.e., $\frac{\mathcal{W}(\nu,C)}{\sum_{\nu'\in Q}\mathcal{W}(\nu',C)}$. Then, we apply the purification step in Algorithm 1 to purify the candidate center ν into a clean center $\tilde{\nu}$. Next, if there exists a center $\nu' \in C$ such that replacing ν' with $\tilde{\nu}$ results in a reduction of the cost, we replace ν' with $\tilde{\nu}$.

Time Complexity [4]**:** Let $\mathcal{O}(\mathcal{T})$ denote the time complexity of computing RWD, and τ be a constant. In Algorithm 1, selecting candidate centers during each epoch requires $\mathcal{O}(m\cdot\mathcal{T})$ time, and the purification step also takes $\mathcal{O}(m\cdot\mathcal{T})$ time. With k epochs in total, the overall time complexity is $\mathcal{O}(km\cdot\mathcal{T})$.

For Algorithm 2, initializing the distance matrix between C and Q takes $\mathcal{O}(km\cdot\mathcal{T})$ time. During each epoch, the operations require $\mathcal{O}(km+m\cdot\mathcal{T})$ time. Assuming Z epochs in total, the total time complexity is $\mathcal{O}(km\cdot\mathcal{T} + Z\cdot(km + m\cdot\mathcal{T}))$.

[4]We assume that the distance between any two points in $\mathcal{X}$ can be computed within $\mathcal{O}(1)$ time.

4. Acceleration

This section presents an acceleration method for the RWC-clustering problem. Specifically, Section 4.1 provides a lower bound; based on this, Section 4.2 constructs a coreset to accelerate computation by reducing the data set size.

4.1. Lower bound

Algorithm 3 essentially replaces the metric in the classic Gonzalez's algorithm with $\widetilde{\mathcal{W}}(\cdot,\cdot)$. Specifically, let C_i be the center set containing i centers. We initially select a point μ randomly from the input dataset Q, and initialize the center set as $C_1 = \{\mu\}$. Then, for the i-th epoch with $2 \le i \le k$, we choose the point $\mu \in Q$ that is farthest from the previous center set C_{i-1}, and set $C_i = C_{i-1}\cup\{\mu\}$; formally, the center μ selected, except in the first epoch, satisfies that

$$\mu \in \arg\max_{\mu'\in Q} \widetilde{\mathcal{W}}(\mu', C_{i-1}), \tag{7}$$

where $\widetilde{\mathcal{W}}(\mu, C_{i-1}) := \min_{\nu\in C_{i-1}} \widetilde{\mathcal{W}}(\mu,\nu)$. Notably, no purification step is applied during this process, thus the centers in C_i for $i \in [k]$ contains outliers.

As described in Theorem 4.1, Algorithm 3 computes a lower bound for RWC-clustering problem, which provides a theoretical guidance for subsequent coreset construction.

Algorithm 3 Lower bound

Input: a set $Q = \{\mu^i\}_{i\in[m]}$ of probability measures, and a parameter k
Sample $\mu \in Q$ uniformly at random, and set $C_1 = \{\mu\}$.
for $i = 2$ **to** k **do**
  Select $\mu \in Q$ that is farthest from the center set C_{i-1} under $\widetilde{\mathcal{W}}(\cdot,\cdot)$ according to Equation (7).
end for
Output: a k-center set C_k

Theorem 4.1 (Lower bound)**.** *Let Δ be the optimal value of the k-RWC-clustering problem, i.e., $\Delta = \min_{C\subseteq\mathcal{P}(\mathcal{X}),|C|=k} cost(Q,C)$. Algorithm 3 takes set Q as input and outputs a k-center set C_k within $\mathcal{O}(km\cdot\mathcal{T})$ time. We define $\Gamma := \max_{\mu\in Q}\widetilde{\mathcal{W}}(\mu, C_k)$. Then, we have $\Gamma \le 2\Delta$.*

4.2. Coreset

Algorithm 4 describes a coreset construction method, which is inspired by (Ding et al., 2021; Krauthgamer & Lee, 2004; Har-Peled & Mendel, 2005). The algorithm takes as input a set Q of probability measures, its doubling dimension[5]

[5]Given a metric space (Q, W), its doubling dimension (Huang et al., 2018; Wang et al., 2024) is defined as the smallest integer

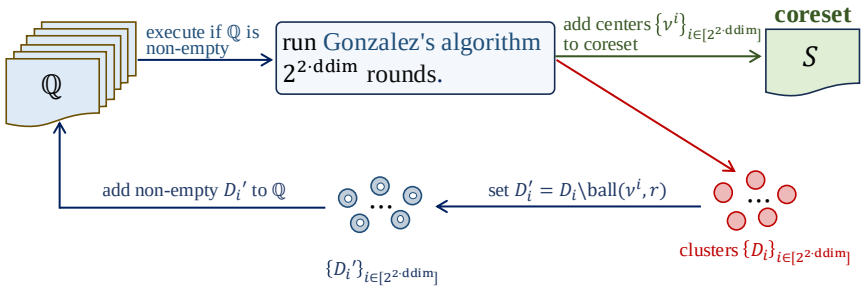


Figure 1: Coreset construction.

ddim, and a parameter r, and outputs a coreset S. The coreset construction relies on the Wasserstein distance, which serves as the key metric throughout the process.

Figure 1 provides an intuitive and comprehensible understanding of this method. Specifically, we begin by initializing the family $\mathbb{Q}$ of sets as $\mathbb{Q} = \{Q\}$. The following *local procedure* is then executed on every set $D \in \mathbb{Q}$ until $\mathbb{Q}$ becomes empty:

- Execute the Gonzalez's algorithm $2^{2\cdot \mathsf{ddim}}$ rounds on $D \in \mathbb{Q}$, yielding a set of centers $\{\nu^i\}_{i\in[2^{2\cdot \mathsf{ddim}}]}$ and their corresponding clusters $\{D_i\}_{i\in[2^{2\cdot \mathsf{ddim}}]}$. The centers are added to the coreset S.

- For each cluster D_i, we construct D_i' by removing all points within a ball of radius r centered at ν^i, formally defined as

$$D_i' = D_i \backslash \mathsf{ball}(\nu^i, r), \tag{8}$$

  where $\mathsf{ball}(\nu^i, r) := \{\mu \mid W(\mu, \nu^i) \le r, \mu \in D_i\}$.

- If D_i' is non-empty, we add it to $\mathbb{Q}$. Remove the set D from $\mathbb{Q}$.

Theorem 4.2 (Coreset property)**.** *Let ddim be the doubling dimension of Q and R be the radius of Q under Wasserstein distance, i.e., $W(\mu,\nu) \le 2R$ for any $\mu, \nu \in Q$. We set $r = \mathcal{O}(\epsilon\Gamma)$, then Algorithm 4 outputs an ϵ-coreset S with $|S| = \mathcal{O}((\frac{R}{r})^{2\cdot ddim})$ for k-RWC-clustering problem on Q within $\mathcal{O}(2^{2\cdot ddim} \cdot |Q| \cdot \mathcal{T} \cdot \log \frac{R}{r})$ time.*

Corollary 4.3. *Algorithm 4 takes Q as its input and outputs the coreset S. The output S satisfies the following property*

$$\Big|\min_{C\in\mathcal{P}(\mathcal{X}),|C|=k} \mathit{cost}(Q,C) - \min_{C'\in\mathcal{P}(\mathcal{X}),|C'|=k} \mathit{cost}(S,C')\Big| \le \min_{C\in\mathcal{P}(\mathcal{X}),|C|=k} \epsilon\cdot \mathit{cost}(Q,C).$$

ddim such that any ball with radius $2r$ can be covered by at most 2^{ddim} balls of radius r.

Algorithm 4 Coreset

Input: a set $Q = \{\mu^i\}_{i\in[m]}$ of probability measures, doubling dimension ddim and a parameter r
Initialize $\mathbb{Q} = \{Q\}$ and $S = \emptyset$.
for set D **in** $\mathbb{Q}$ **do**
  ▷local procedure
  Run Gonzalez's algorithm $2^{2\cdot \mathsf{ddim}}$ rounds on D, yielding centers $\{\nu^i\}_{i\in[2^{2\cdot \mathsf{ddim}}]}$ and clusters $\{D_i\}_{i\in[2^{2\cdot \mathsf{ddim}}]}$.
  Set $S = S \cup \{\nu^i\}_{i\in[2^{2\cdot \mathsf{ddim}}]}$.
  Construct D_i' by removing points within a ball of radius r centered at ν^i according to Equation (8).
  Add all non-empty D_i' to $\mathbb{Q}$; i.e., $\mathbb{Q} = \mathbb{Q} \cup \{D_i'\}$.
  Set $\mathbb{Q} = \mathbb{Q} \backslash \{D\}$.
end for
Input: coreset S

A good proxy: As stated in Theorem 4.2, for any subset $C \subseteq \mathcal{P}(\mathcal{X})$ with $|C| = k$, the values computed on the coreset can closely approximate the values on the original dataset within an ϵ-relative error. That is,

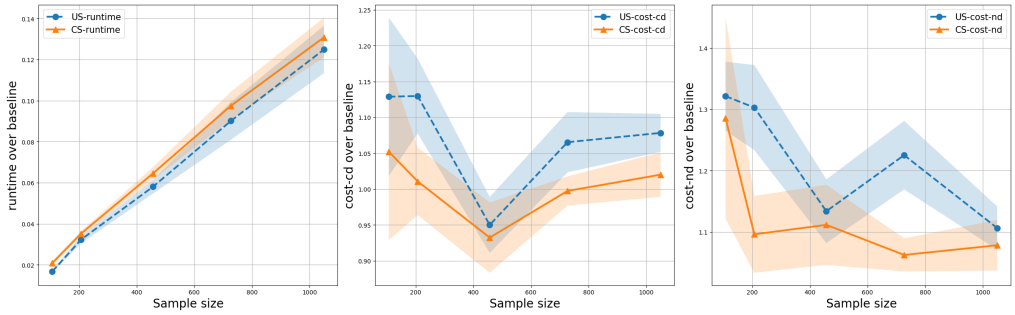
$$\mathsf{cost}(S,C) \approx \mathsf{cost}(Q,C). \tag{9}$$

Furthermore, according to Corollary 4.3, the optimal value computed on the coreset is approximately the same as the optimal value computed on the original dataset. That is,

$$\min_{C\subseteq Q,|C|=k} \mathsf{cost}(S,C) \approx \min_{C\subseteq Q,|C|=k} \mathsf{cost}(Q,C). \tag{10}$$

According to Equations (9) and (10), we have that the coreset S serves as a good proxy of the original dataset Q for the RWC-clustering problem.

Remark 4.4 (Enhancing Scalability for Coreset Construction). We can accelerate coreset construction by leveraging the merge-and-reduce framework (Bentley & Saxe, 1980; Har-Peled & Mazumdar, 2004), which is efficient in both computation and communication. Specifically, a large dataset can be partitioned into smaller subsets and distributed across multiple machines for parallel computation,



(a) Comparison of runtime (b) Comparison of cost-cd (c) Comparison of cost-nd

Figure 2: Comparison of the US method and our CS method across varying sample sizes on MNIST dataset.

which significantly improve time efficiency. Furthermore, since the coreset is a subset of the original dataset, only the indices of the data points, rather than the data items themselves, need to be transmitted during machine synchronization. This makes the communication overhead negligible, ensuring excellent scalability of our coreset approach. Additionally, the merge-and-reduce framework enables our approach to adapt seamlessly to streaming data, making it highly effective and efficient in dynamic data processing scenarios.

Remark 4.5. In the process of constructing the coreset, Wasserstein distance is employed primarily to ensure theoretical rigor. In practice, the construction of the coreset can also utilize $\mathcal{W}(\cdot,\cdot)$.

5. Experiments

This section demonstrated the effectiveness of our RWC-clustering problem and seeding algorithm, as well as the efficiency of the coreset method. All the experiments were performed on a server with 2.40GHz Intel CPU, 60GB RAM, and Python 3.12. We utilized the POT library (Flamary et al., 2021) to compute the Wasserstein distance (WD) (Bonneel et al., 2011) and unbalanced optimal transport (UOT) (Chizat et al., 2018; Frogner et al., 2015).

Due to space constraints, we validated our methods on the following two datasets. Additional experiments on ModelNet10 (Wu et al., 2015) dataset are in the Appendix.

i) *Geometric shapes* is a toy dataset designed by us, consisting of five geometric shapes. It is used to verify the advantages of our seeding algorithm and the RWC-clustering problem. Each shape is represented by a point set.

ii) *MNIST* (LeCun et al., 1998) is a well-known handwritten digit dataset. For each image, we extract the pixels with higher grayscale values to form the corresponding point set. The reported results are averaged over five runs.

For the point set $\{x_i\}_{i\in[n]}$ corresponding to a specific data item in the above datasets, we represent it as a probability measure $\mu = \sum_{i=1}^{n} \frac{1}{n}\delta_{x_i}$ to construct the clean dataset Q^0. The noisy dataset Q is generated by adding clustered noise with ζ mass following a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ to each clean probability measure.

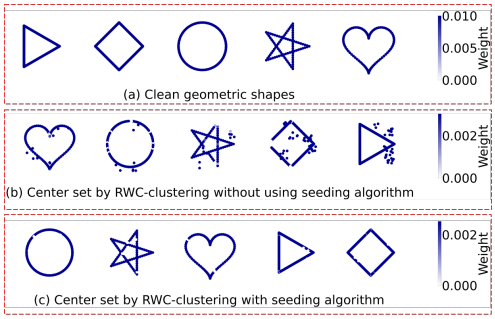


Figure 3: Effectiveness of our seeding algorithm.

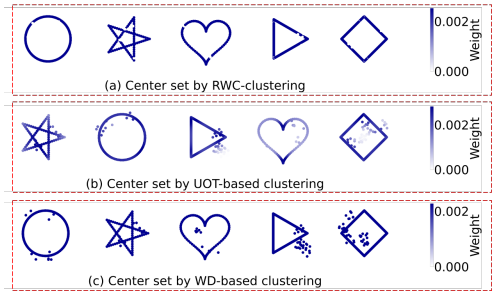


Figure 4: Comparing Our RWC-clustering with WD-Based clustering and UOT-Based clustering.

To evaluate the performance of our methods, we consider the following three criteria: i) Runtime: This includes the sampling time and the clustering time required to compute the k-center set C. ii) cost-cd: Defined as $\max_{\mu\in Q^0} \min_{\nu\in C} W(\mu,\nu)$, it quantifies the distance between the center set C and the original clean dataset Q^0. iii) cost-nd: Defined as $\max_{\mu\in Q} \min_{\nu\in C} \mathcal{W}(\mu,\nu)$, it evaluates the distance from center set C to the noisy dataset Q. The baselines for these three criteria are established

using the results computed on the original full dataset for comparison.

Effectiveness of our method: To demonstrate the effectiveness of our seeding algorithm and the RWC-clustering problem, we conducted a series of experiments on the Geometric shapes dataset. We visualized the clean geometric shapes in Figure 3(a), and the noisy geometric shapes are in the appendix (in Figure 5).

Figure 3(b) shows poor denoising performance without using the seeding algorithm for initialization. In contrast, by using the seeding algorithm, the resulting centers in Figure 3(c) are more closely with the original five clean shapes. This implies the importance of a good initialization, and demonstrates the advantage of our seeding algorithm in providing a good starting point.

Figure 4 compares the center sets computed by UOT-based clustering, WD-based clustering, and RWC-clustering. Among these, the WD-based clustering exhibits the worst performance, showing almost no denoising capability. The UOT-based clustering outperforms the WD-based clustering; however, the inclusion of an entropy regularization term causes a diffusion effect that hinders noise removal, leaving some residual noise inevitably. In contrast, our RWC-clustering approach achieves the best denoising performance, outperforming the other two methods.

Table 1: Comparison of the US method and our CS method with varying values of k. We fix the sample size as 483, $\sigma = 1$, and $\zeta = 0.1$.

k	SM	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
10	US	$18.0_{\pm 1.8}$	$14.7_{\pm 0.9}$	$775.1_{\pm 49.6}$
	CS	$\mathbf{15.7}_{\pm 0.6}$	$\mathbf{13.8}_{\pm 0.8}$	$822.6_{\pm 81.4}$
20	US	$15.7_{\pm 0.6}$	$12.0_{\pm 0.8}$	$830.1_{\pm 16.2}$
	CS	$\mathbf{14.2}_{\pm 0.3}$	$\mathbf{11.9}_{\pm 0.2}$	$858.5_{\pm 15.4}$
30	US	$14.0_{\pm 1.4}$	$10.8_{\pm 0.5}$	$901.6_{\pm 62.1}$
	CS	$\mathbf{12.9}_{\pm 0.9}$	$\mathbf{10.7}_{\pm 0.7}$	$996.1_{\pm 75.0}$

Table 2: Comparison of the US method and our CS method using varying noisy intensity σ. We fix $k = 10$ and $\zeta = 0.1$.

σ	SM	SS	cost-nd(↓)	cost-cd(↓)	Runtime (↓)
1	US	474	$22.1_{\pm 0.7}$	$14.5_{\pm 1.3}$	$776.0_{\pm 40.6}$
	CS	474	$\mathbf{19.8}_{\pm 2.6}$	$\mathbf{13.6}_{\pm 0.4}$	$828.5_{\pm 44.9}$
2	US	479	$17.3_{\pm 1.1}$	$14.5_{\pm 0.9}$	$804.9_{\pm 77.6}$
	CS	479	$\mathbf{15.2}_{\pm 0.4}$	$\mathbf{13.7}_{\pm 0.7}$	$853.6_{\pm 38.7}$
3	US	457	$15.6_{\pm 1.5}$	$14.1_{\pm 0.6}$	$748.3_{\pm 42.7}$
	CS	457	$\mathbf{15.4}_{\pm 0.8}$	$\mathbf{13.8}_{\pm 0.8}$	$838.0_{\pm 39.1}$

Efficiency of our coreset method: We show the efficiency of our coreset (CS) method by comparing it against the uniform sampling (US) method. To ensure fairness, both sampling methods (SM) employed the same sampling size (SS). Figure 2 illustrates the performance of our CS method on MNIST dataset. Although our CS method is slightly more time-consuming compared to the US method, it remains significantly more efficient than processing the original dataset. Moreover, on both cost-cd and cost-nd criteria, our CS method consistently outperforms the US method.

Table 3: Comparison of the US method and our CS method using varying mass ζ of noise. We fixed $k = 10$ and $\sigma = 1$.

ζ	SM	SS	cost-nd(↓)	cost-cd(↓)	Runtime (↓)
0.1	US	479	$17.3_{\pm 1.1}$	$14.4_{\pm 0.8}$	$695.9_{\pm 77.6}$
	CS	479	$\mathbf{15.2}_{\pm 0.4}$	$\mathbf{13.7}_{\pm 0.7}$	$741.6_{\pm 38.6}$
0.2	US	474	$22.1_{\pm 0.7}$	$14.5_{\pm 1.3}$	$726.1_{\pm 40.6}$
	CS	474	$\mathbf{19.8}_{\pm 2.6}$	$\mathbf{13.6}_{\pm 0.4}$	$778.5_{\pm 44.8}$
0.3	US	452	$33.4_{\pm 3.7}$	$\mathbf{15.9}_{\pm 3.7}$	$721.8_{\pm 22.9}$
	CS	452	$\mathbf{32.2}_{\pm 1.7}$	$16.2_{\pm 2.5}$	$757.5_{\pm 78.6}$

Due to the randomness inherent of our algorithms and approximate nature of the derived k-center set, some fluctuations are inevitable. Overall, as the sample size increases, the performance improves. Particularly, for the primary metric cost-nd in this study, our CS method reflects this trend more clearly.

Ablation experiments: Taking the MNIST dataset as an example, we explore the effects of the parameter k, noise mass ζ, and noise intensity σ on the performance of our coreset method.

Table 1 demonstrates that our CS method consistently outperforms the US method across different values of k under both the cost-cd and cost-nd criteria. Table 2 shows that under varying noise intensity levels σ, the CS method consistently achieves better performance compared to the US method. Table 3 illustrates that the CS method consistently outperforms the US method under the cost-nd criterion. Under the cost-cd criterion, the CS method generally performs better, except for a slight degradation when $\zeta = 0.3$. This minor degradation can be attributed to the stochastic nature of our algorithm, which is inevitable and reasonable.

6. Conclusion and Future Work

In this paper, we introduce the **R**obust **W**asserstein **C**enter clustering (RWC-clustering) problem, and propose an efficient algorithm to solve it. Additionally, we develop a coreset method to accelerate computations by compressing the dataset. For future work, we will explore the corresponding *robust Wasserstein k-means clustering* problem, along with its approximation algorithms and coreset techniques.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., and Savarese, S. 3d semantic parsing of large-scale indoor spaces. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1534–1543, 2016.

Bateni, M., Esfandiari, H., Fichtenberger, H., Henzinger, M., Jayaram, R., Mirrokni, V., and Wiese, A. Optimal fully dynamic k-center clustering for adaptive and oblivious adversaries. In Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2677–2727. SIAM, 2023.

Bazeille, T., Richard, H., Janati, H., and Thirion, B. Local optimal transport for functional brain template estimation. In Information Processing in Medical Imaging: 26th International Conference, IPMI 2019, Hong Kong, China, June 2–7, 2019, Proceedings 26, pp. 237–248. Springer, 2019.

Bentley, J. L. and Saxe, J. B. Decomposable searching problems i. static-to-dynamic transformation. Journal of Algorithms, 1(4):301–358, 1980.

Bonneel, N., Van De Panne, M., Paris, S., and Heidrich, W. Displacement interpolation using lagrangian mass transport. In Proceedings of the 2011 SIGGRAPH Asia conference, pp. 1–12, 2011.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

Chapel, L., Alaya, M. Z., and Gasso, G. Partial optimal tranport with applications on positive-unlabeled learning. Advances in Neural Information Processing Systems, 33: 2903–2913, 2020.

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F. Scaling algorithms for unbalanced optimal transport problems. Math. Comput., 87(314):2563–2609, 2018. doi: 10.1090/MCOM/3303. URL https://doi.org/10.1090/mcom/3303.

Choo, D., Grunau, C., Portmann, J., and Rozhon, V. k-means++: few more steps yield constant approximation. In International Conference on Machine Learning, pp. 1909–1917. PMLR, 2020.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. Advances in neural information processing systems, 26, 2013.

Ding, H., Chen, T., Yang, F., and Wang, M. A data-dependent algorithm for querying earth mover's distance with low doubling dimensions. In Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), pp. 630–638. SIAM, 2021.

Drakulic, D., Michel, S., Mai, F., Sors, A., and Andreoli, J.-M. BQ-NCO: Bisimulation quotienting for efficient neural combinatorial optimization. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=BRqlkTDvvm.

Dvurechensky, P., Gasnikov, A., and Kroshnin, A. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In International conference on machine learning, pp. 1367–1376. PMLR, 2018.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., et al. Pot: Python optimal transport. Journal of Machine Learning Research, 22(78):1–8, 2021.

Frogner, C., Zhang, C., Mobahi, H., Araya, M., and Poggio, T. A. Learning with a wasserstein loss. Advances in neural information processing systems, 28, 2015.

Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In 2012 IEEE conference on computer vision and pattern recognition, pp. 3354–3361. IEEE, 2012.

Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. Theoretical computer science, 38: 293–306, 1985.

Grinsztajn, N., Furelos-Blanco, D., Surana, S., Bonnet, C., and Barrett, T. D. Winner takes it all: Training performant rl populations for combinatorial optimization. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

Hakimi, S. L. Optimum locations of switching centers and the absolute centers and medians of a graph. Operations research, 12(3):450–459, 1964.

Har-Peled, S. and Mazumdar, S. On coresets for k-means and k-median clustering. In Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pp. 291–300, 2004.

Har-Peled, S. and Mendel, M. Fast construction of nets in low dimensional metrics, and their applications. In Proceedings of the twenty-first annual symposium on Computational geometry, pp. 150–158, 2005.

Huang, L., Jiang, S. H.-C., Lou, J., and Wu, X. Near-optimal coresets for robust clustering. In The Eleventh International Conference on Learning Representations.

Huang, L., Jiang, S. H.-C., Li, J., and Wu, X. Epsilon-coresets for clustering (with outliers) in doubling metrics. In 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 814–825. IEEE, 2018.

Huang, L., Huang, R., Huang, Z., and Wu, X. On coresets for clustering in small dimensional euclidean spaces. In International Conference on Machine Learning, pp. 13891–13915. PMLR, 2023.

Huang, L., Li, J., and Wu, X. On optimal coreset construction for euclidean (k, z)-clustering. In Proceedings of the 56th Annual ACM Symposium on Theory of Computing, pp. 1594–1604, 2024.

HUBER, P. Robust estimation of a location parameter. Ann. Math. Statist., 35:73–101, 1964.

Jambulapati, A., Sidford, A., and Tian, K. A direct tilde {O}(1/epsilon) iteration parallel algorithm for optimal transport. Advances in Neural Information Processing Systems, 32, 2019.

Krauthgamer, R. and Lee, J. R. Navigating nets: simple algorithms for proximity search. In SODA, volume 4, pp. 798–807, 2004.

Łacki, J., Haeupler, B., Grunau, C., Jayaram, R., and Rozhoň, V. Fully dynamic consistent k-center clustering. In Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 3463–3484. SIAM, 2024.

Lattanzi, S. and Sohler, C. A better k-means++ algorithm via local search. In International Conference on Machine Learning, pp. 3662–3671. PMLR, 2019.

Le, K., Nguyen, H., Nguyen, Q. M., Pham, T., Bui, H., and Ho, N. On robust optimal transport: Computational complexity and barycenter computation. Advances in Neural Information Processing Systems, 34:21947–21959, 2021.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

Luo, F., Lin, X., Liu, F., Zhang, Q., and Wang, Z. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

Mohri, M. Foundations of machine learning, 2018.

Murtagh, F. and Contreras, P. Algorithms for hierarchical clustering: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(1): 86–97, 2012.

Nietert, S., Goldfeld, Z., and Cummings, R. Outlier-robust optimal transport: Duality, structure, and statistical analysis. In International Conference on Artificial Intelligence and Statistics, pp. 11691–11719. PMLR, 2022.

Peyré, G., Cuturi, M., et al. Computational optimal transport. Center for Research in Economics and Statistics Working Papers, (2017-86), 2017.

Ros, F. and Guillaume, S. Sampling techniques for supervised or unsupervised tasks. Springer, 2020.

Shalev-Shwartz, S. and Ben-David, S. Understanding machine learning: From theory to algorithms. Cambridge university press, 2014.

Thual, A., TRAN, Q. H., Zemskova, T., Courty, N., Flamary, R., Dehaene, S., and Thirion, B. Aligning individual brains with fused unbalanced gromov wasserstein. Advances in Neural Information Processing Systems, 35: 21792–21804, 2022.

Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, D. T., and Yeung, S.-K. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In International Conference on Computer Vision (ICCV), 2019.

Villani, C. et al. Optimal transport: old and new, volume 338. Springer, 2009.

Wang, X., Huang, J., Yang, Q., and Zhang, J. On robust wasserstein barycenter: The model and algorithm. In Proceedings of the 2024 SIAM International Conference on Data Mining (SDM), pp. 235–243. SIAM, 2024.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1912–1920, 2015.

A. Other Preliminaries

Definition A.1 (Optimal transport (OT) (Peyré et al., 2017))**.** Let $\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$ and $\nu = \sum_{j=1}^{n} b_j \delta_{y_j}$ be two probability measures with weights $\mathbf{a}, \mathbf{b} \in \mathbb{R}_+^n$, respectively. Given a cost matrix $\mathbf{D} \in \mathbb{R}_+^{n \times n}$, the OT distance between μ and ν is

$$\mathcal{OT}(\mu, \nu) := \min_{\mathbf{P} \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{P}, \mathbf{D} \rangle,$$

where $\Pi(\mathbf{a}, \mathbf{b}) := \left\{ \mathbf{P} \in \mathbb{R}_+^{n \times n} \mid \mathbf{P}\mathbf{1} = \mathbf{a}, \mathbf{P}^T \mathbf{1} = \mathbf{b} \right\}$ is the set of all feasible couplings and $\mathbf{1}$ is the vector of all ones.

The Wasserstein distance is a special case of OT. The cost matrix $\mathbf{D}$ of Wasserstein distance must be induced by a distance function, while the cost matrix of OT only needs to be induced by a positive function. Thus, the Wasserstein distance is a metric on $\mathcal{P}(\mathcal{X})$, whereas OT is not a metric.

The doubling dimension provides a means for describing the growth rate of the data set with respect to certain metric.

Definition A.2 (Doubling dimension (Huang et al., 2018; Wang et al., 2024))**.** Let (Q, W) be a metric space, where $W(\cdot, \cdot)$ is a metric on Q. The doubling dimension of (Q, W) is the smallest integer ddim such that every ball of radius $2r$ can be covered by at most 2^{ddim} balls of radius r.

In real-world datasets, data often exhibit inherent regularities, leading to a relatively low intrinsic dimension. Therefore, assuming a low doubling dimension is usually reasonable.

Definition A.3 (r-cover(Shalev-Shwartz & Ben-David, 2014; Mohri, 2018))**.** Given a metric space $(\mathcal{X}, \mathrm{dist})$, a set $A \subseteq \mathcal{X}$ is an r-cover of $Q \subseteq \mathcal{X}$, if for any $x \in Q$, there exists $x' \in A$ satisfying $\mathrm{dist}(x, x') \leq r$.

Note that, a cover of a set does not need to be a subset of it.

Gonzalez's algorithm (Gonzalez, 1985), a 2-approximation algorithm for the k-center problem, is often used as an initialization method in clustering tasks.

Let Q be a data set and $\mathrm{dist}(\cdot, \cdot)$ be the metric on Q. We iteratively select the point farthest from the currently chosen centers as the new center. The sequential nature of center selection leads to dependencies between steps, which poses challenges for parallelization. It takes $\mathcal{O}(mk)$ time, where m is the size of the dataset, and k represents the number of centers. When k or m is large, the computational complexity becomes a bottleneck. The details are in Algorithm 5.

Algorithm 5 Gonzalez's algorithm

Input: data set Q, and a parameter k
Sample $c \in Q$ uniformly at random, and set $C_1 = \{c\}$.
for $i = 2$ **to** k **do**
  Select $c \in Q$ that is farthest from the center set C_{i-1}.
end for
Set $C = C_k$.
Output: a k-center set C

B. Omitted Proof

B.1. Intuition of RWC-clustering problem

Let $Q = \left\{ \mu^i \right\}_{i \in [m]}$ be a set of probability measures. We define the ball center at ν with radius r under metric $\mathcal{W}(\cdot, \cdot)$ as

$$\mathsf{ball}_{\mathcal{W}}(\nu, r) := \left\{ \mu \mid \mathcal{W}(\mu, \nu) \leq r, \mu \in Q \right\}. \tag{11}$$

Lemma B.1. *If $\mu^0, \mu^1 \in \mathcal{P}(\mathcal{X})$ are in $\mathsf{ball}_{\mathcal{W}}(\nu, r)$, we have $\widetilde{\mathcal{W}}(\mu^0, \mu^1) \leq 2r$.*

Proof. Since μ^0, μ^1 are in $\mathsf{ball}_{\mathcal{W}}(\nu, r)$, we have $\mathcal{W}(\mu^0, \nu) \leq r, \quad \mathcal{W}(\mu^1, \nu) \leq r$. Assume that $\hat{\mu}^0, \hat{\mu}^1$ are the clean probability measures induced by μ^0, μ^1, respectively. That is,

$$\mathcal{W}(\mu^0, \nu) = W(\hat{\mu}^0, \nu), \quad \mathcal{W}(\mu^1, \nu) = W(\hat{\mu}^1, \nu). \tag{12}$$

According to Definition 2.1, we have $\widetilde{\mathcal{W}}(\mu^0, \mu^1) \leq \mathcal{W}(\mu^0, \hat{\mu}^1) \leq W(\hat{\mu}^0, \hat{\mu}^1)$. Then, by combining triangle inequality property of Wasserstein distance with Equation (12), we obtain

$$\widetilde{\mathcal{W}}(\mu^0, \mu^1) \leq W(\hat{\mu}^0, \hat{\mu}^1) \leq W(\hat{\mu}^0, \nu) + W(\hat{\mu}^1, \nu) = \mathcal{W}(\mu^0, \nu) + \mathcal{W}(\mu^1, \nu) \leq 2r.$$

□

Lemma B.1 implies that, under $\mathcal{W}(\cdot, \cdot)$, the difference between two data points located within a small ball is relatively small.

B.2. Analysis of lower bound (Theorem 4.1):

Theorem 4.1 (Lower bound)**.** *Let Δ be the optimal value of the k-RWC-clustering problem, i.e., $\Delta = \min_{C \subseteq \mathcal{P}(\mathcal{X}), |C|=k} \mathit{cost}(Q, C)$. Algorithm 3 takes set Q as input and outputs a k-center set C_k within $\mathcal{O}(km \cdot \mathcal{T})$ time. We define $\Gamma := \max_{\mu \in Q} \widetilde{\mathcal{W}}(\mu, C_k)$. Then, we have $\Gamma \leq 2\Delta$.*

Let $C^* = \{\nu_*^i\}_{i \in [k]}$ be the optimal solution to the RWC-clustering problem, and Δ be its corresponding optimal value; that is,

$$\min_{C \subseteq \mathcal{P}(\mathcal{X}), |C|=k} \mathsf{cost}(Q, C) = \mathsf{cost}(Q, C^*) = \Delta. \tag{13}$$

Let $Q_i^*, i \in [k]$ be the clusters induced by $\nu_*^i, i \in [k]$, where each point is assigned to the nearest cluster center. That is,

$$\mathcal{W}(\mu, \nu_*^i) = \mathcal{W}(\mu, C^*) \leq \Delta \quad \text{for } \mu \in Q_i^*. \tag{14}$$

The set C_k is the output of Algorithm 3. The centers in C_k contain outliers, whereas the centers in C^* are clean.

Lemma B.2. *For any $\mu \in Q_i^*, i \in [k]$, we have $\widetilde{\mathcal{W}}(\mu, C_k) \leq 2\Delta$ if $|Q_i^* \cap C_k| \geq 1$.*

Proof of Lemma B.2. Let $\nu \in Q_i^* \cap C_k$. Since ν_*^i is the nearest center of $\mu \in Q_i^*$, by using Equation (13), we have

$$\mathcal{W}(\mu, \nu_*^i) \leq \Delta \text{ for all } \mu \in Q_i^*. \tag{15}$$

We know that both μ, ν are in cluster Q_i^*, thus $\mu, \nu \in \mathsf{ball}_{\mathcal{W}}(\nu_*^i, \Delta)$. Then, by using Lemma B.1, we obtain $\widetilde{\mathcal{W}}(\mu, \nu) \leq 2 \cdot \Delta$. According to the definition $\widetilde{\mathcal{W}}(\mu, C_k) := \min_{\nu' \in C_k} \widetilde{\mathcal{W}}(\mu, \nu')$, we have $\widetilde{\mathcal{W}}(\mu, C_k) \leq \widetilde{\mathcal{W}}(\mu, \nu)$. Till now, we obtain $\widetilde{\mathcal{W}}(\mu, C_k) \leq 2\Delta$.

□

Proof of Theorem 4.1. In order to prove the conclusion, we will discuss the proof in two cases, which is inspired by (Gonzalez, 1985).

Case 1: Each Q_i^* contains exactly one center $\nu \in C_k$.

According to Lemma B.2, we have $\widetilde{\mathcal{W}}(\mu, C_k) \leq 2\Delta$ for all $\mu \in Q_i^*, i \in [k]$. Since the collection $\{Q_i^*\}_{i \in [k]}$ forms a partition of Q, we have

$$Q = \sqcup_{i=1}^k Q_i^*.$$

Thus, it follows that $\widetilde{\mathcal{W}}(\mu, C_k) \leq 2\Delta$ for all $\mu \in Q$. That is, $\Gamma \leq 2\Delta$ holds.

Case 2: Some Q_i^* contains multiple centers, i.e., $|C_k \cap Q_i^*| \geq 2$.

Without loss of generality, suppose that C_{i_0} is the first center set such that $|C_{i_0} \cap Q_i^*| = 2$ for some $i \in [k]$, with $C_{i_0} \cap Q_i^* = \{\nu, \nu^{i_0}\}$, where ν^{i_0} is the last center added to C_{i_0}.

From the Line 3 of Algorithm 3, we know that

$$\widetilde{\mathcal{W}}(\mu, C_{i_0-1}) \leq \widetilde{\mathcal{W}}(\nu^{i_0}, C_{i_0-1}) \quad \text{for all } \mu \in Q.$$

Since $\nu \in C_{i_0-1}$, we have

$$\widetilde{\mathcal{W}}(\nu^{i_0}, C_{i_0-1}) \leq \widetilde{\mathcal{W}}(\nu^{i_0}, \nu) \quad \text{for } i \in [k-1].$$

We know that both ν, ν^{i_0} are in cluster Q_i^*, thus $\nu, \nu^{i_0} \in \mathsf{ball}_{\mathcal{W}}(\nu_*^i, \Delta)$. Then, by using Lemma B.1, we obtain $\widetilde{\mathcal{W}}(\nu, \nu^{i_0}) \leq 2 \cdot \Delta$.

Till now, we achieve that

$$\widetilde{\mathcal{W}}(\mu, C_{i_0-1}) \leq 2\Delta \quad \text{for all } \mu \in Q.$$

From Line 3 of Algorithm 3, it follows that

$$\widetilde{\mathcal{W}}(\mu, C_k) \leq \widetilde{\mathcal{W}}(\mu, C_{i_0-1}) \quad \text{for all } \mu \in Q.$$

Therefore, we obtain

$$\Gamma \leq 2\Delta \quad \text{for Case 2.}$$

In conclusion, for both Case 1 and Case 2, the inequality $\Gamma \leq 2\Delta$ holds.

Time complexity: The time complexity for selecting each center is $\mathcal{O}(m \cdot \mathcal{T})$. Since we need to select k centers in total, the overall time complexity is $\mathcal{O}(km\mathcal{T})$.

□

B.3. Analysis of coreset (Theorem 4.2)

Theorem 4.2 (Coreset property)**.** *Let ddim be the doubling dimension of Q and R be the radius of Q under Wasserstein distance, i.e., $W(\mu, \nu) \leq 2R$ for any $\mu, \nu \in Q$. We set $r = \mathcal{O}(\epsilon\Gamma)$, then Algorithm 4 outputs an ϵ-coreset S with $|S| = \mathcal{O}((\frac{R}{r})^{2\cdot \mathsf{ddim}})$ for k-RWC-clustering problem on Q within $\mathcal{O}(2^{2\cdot \mathsf{ddim}} \cdot |Q| \cdot \mathcal{T} \cdot \log \frac{R}{r})$ time.*

We introduce two sets, $\mathsf{Set}(\mu)$ and $\mathsf{Set}(\xi)$, defined as follows

$$\mathsf{Set}(\mu) = \left\{ \mu'' = \frac{\mu'}{1-\zeta} \in \mathcal{P}(\mathcal{X}) \mid \mu' \leq \mu, \|\mu' - \mu\|_{\mathrm{TV}} = \zeta \right\}$$

and

$$\mathsf{Set}(\xi) = \left\{ \xi'' = \frac{\xi'}{1-\zeta} \in \mathcal{P}(\mathcal{X}) \mid \xi' \leq \xi, \|\xi' - \xi\|_{\mathrm{TV}} = \zeta \right\},$$

where $\mathsf{Set}(\mu)$ and $\mathsf{Set}(\xi)$ represent the sets of feasible clean probability measures with ζ mass of outliers removed from μ, ν according to $\mathcal{W}(\mu, \cdot)$ and $\mathcal{W}(\xi, \cdot)$, respectively.

Roughly speaking, the following theorem illustrates that if two probability measures μ and ξ are similar, then the sets of feasible clean probability measures they induce, denoted by $\mathsf{Set}(\mu)$ and $\mathsf{Set}(\xi)$, respectively, are also similar, i.e., $\mathsf{Set}(\mu) \approx \mathsf{Set}(\xi)$. Specifically, for any μ'' in $\mathsf{Set}(\mu)$, there exists a corresponding ξ'' in $\mathsf{Set}(\xi)$ that is close to μ'' under Wasserstein distance. Conversely, for every ξ'' in $\mathsf{Set}(\xi)$, there exists a $\mu'' \in \mathsf{Set}(\mu)$ that is close to ζ''. Consequently, the sets $\mathsf{Set}(\mu)$ and $\mathsf{Set}(\xi)$ are r-cover of each other.

Lemma B.3. *Given any $W(\mu, \xi) \leq r$, the sets $\mathsf{Set}(\mu)$ and $\mathsf{Set}(\xi)$ are $\frac{r}{1-\zeta}$-cover of each other.*

Proof of Lemma B.3. Without loss of generality, let $\mu = \sum_{i=1}^{n} a_i \delta_{x_i}$ and $\xi = \sum_{j=1}^{n} b_j \delta_{y_j}$. Let $\mathbf{P}^*$ denote the optimal coupling induced by $W(\mu, \xi)$; that is,

$$W(\mu, \xi) = \langle \mathbf{P}^*, \mathbf{D} \rangle,$$

where $\mathbf{D}$ is the cost matrix between μ and ξ.

For any $\mu'' \in \mathsf{Set}(\mu)$, we have $\mu' = (1-\zeta) \cdot \mu'' = \sum_{i=1}^{n} a'_i \delta_{x_i}$ with $\mathbf{a}'$ being its weights. We can construct a $\mathbf{P}'$ satisfying

$$\mathbf{P}'\mathbf{1} = \mathbf{a}', \quad \mathbf{P}'\mathbf{1}^T \leq \mathbf{b}, \quad \mathbf{P}' \leq \mathbf{P}^*, \quad \mathbf{P}' \in \mathbb{R}^{n \times n}.$$

Let $\mathbf{b}' = \mathbf{P}'\mathbf{1}^T$, and construct $\xi' = \sum_{j=1}^{n} b'_j \delta_{y_j}$. Transferring μ' to ξ' according to the flow matrix $\mathbf{P}'$, we achieve $\langle \mathbf{P}', \mathbf{D} \rangle \leq \langle \mathbf{P}^*, \mathbf{D} \rangle = W(\mu, \xi) \leq r$. Clearly, $\frac{\mathbf{P}'}{1-\zeta}$ is a feasible flow for $\mu'' = \frac{\mu'}{1-\zeta}$ and $\xi'' = \frac{\xi'}{1-\zeta}$ under the Wasserstein distance. Therefore, $W(\mu'', \xi'') \leq \langle \frac{\mathbf{P}'}{1-\zeta}, \mathbf{D} \rangle \leq \frac{r}{1-\zeta}$.

From the above, we can find a measure $\mu'' = \frac{\mu'}{1-\zeta} \in \mathsf{Set}(\mu)$ such that $W\left(\frac{\mu'}{1-\zeta}, \frac{\xi'}{1-\zeta}\right) \leq \frac{r}{1-\zeta}$ holds for any $\xi'' = \frac{\xi'}{1-\zeta} \in \mathsf{Set}(\xi)$. Similarly, we can find a measure $\xi'' \in \mathsf{Set}(\xi)$ such that $W\left(\frac{\mu'}{1-\zeta}, \frac{\xi'}{1-\zeta}\right) \leq \frac{r}{1-\zeta}$ holds for any $\mu'' \in \mathsf{Set}(\mu)$.

Thus, we have demonstrated that the sets $\mathsf{Set}(\mu)$ and $\mathsf{Set}(\xi)$ are $\frac{r}{1-\zeta}$-covers of each other.

□

Let $g : \mathcal{P}(\mathcal{X}) \to \mathbb{R}, a \mapsto g(a)$ be a function. The following lemma illustrates that if for any $a \in A$, there exists $b \in B$ such that $g(a) \approx g(b)$, and vice vise; then, the minimum value of $g(\cdot)$ over the two sets A and B is close.

Lemma B.4. *If for every $a \in A$, there exists $b \in B$ such that $g(a) \in g(b) \pm r$, and vice versa, then the minimum values of g over sets A and B are approximately equal; that is,*

$$\left| \min_{a' \in A} g(a') - \min_{b' \in B} g(b') \right| \leq r.$$

[Proof of Lemma B.4.] From the given conditions, for any $a \in A$, there exists $b \in B$ such that

$$g(b) - r \leq g(a) \leq g(b) + r. \tag{16}$$

Similarly, for any $b \in B$, there exists $a \in A$ such that

$$g(a) - r \leq g(b) \leq g(a) + r. \tag{17}$$

Let b^* be the point where $g(\cdot)$ achieves its minimum on B, i.e.,

$$g(b^*) = \min_{b' \in B} g(b').$$

According to Equation (17), there exists $a' \in A$ such that

$$g(a') - r \leq g(b^*) \leq g(a') + r.$$

This implies that

$$g(a') - g(b^*) \leq r.$$

Then, we have

$$\min_{a' \in A} g(a') - \min_{b' \in B} g(b') = \min_{a' \in A} g(a') - g(b^*) \leq r. \tag{18}$$

Similarly, by using Equation (16), we also obtain

$$\min_{b' \in B} g(b') - \min_{a' \in A} g(a') \leq r. \tag{19}$$

By combining Equations (18) and (19), it follows that

$$\left| \min_{a' \in A} g(a') - \min_{b' \in B} g(b') \right| \leq r.$$

□

The following theorem illustrates that if two functions are approximately equal pointwise, then their minimum values are also approximately the same.

Lemma B.5. *Given two functions*

$$g, f : \mathcal{P}(\mathcal{X}) \to \mathbb{R}, \tag{20}$$

if $|g(\mu) - f(\mu)| \leq r$ *for all* $\mu \in Q$*, then we have* $|\min_{\mu \in Q} g(\mu) - \min_{\mu' \in Q} f(\mu')| \leq r$.

Proof of Lemma B.5. Assume that $\min_{\mu \in Q} g(\mu) > \min_{\mu' \in Q} f(\mu')$. Let f achieve its minimum at μ^*, that is, $\min_{\mu' \in Q} f(\mu') = f(\mu^*)$. Then, we have

$$\left| \min_{\mu \in Q} g(\mu) - \min_{\mu' \in Q} f(\mu') \right| = \min_{\mu \in Q} g(\mu) - \min_{\mu' \in Q} f(\mu') = \min_{\mu \in Q} g(\mu) - f(\mu^*).$$

According to Equation (20), there must exist $g(\mu^*) - f(\mu^*) \leq r$, implying $\min_{\mu \in Q} g(\mu) - f(\mu^*) \leq r$. Consequently, $\min_{\mu \in Q} g(\mu) - \min_{\mu' \in Q} f(\mu') \leq r$ holds. Similarly, for the case $\min_{\mu \in Q} g(\mu) \leq \min_{\mu' \in Q} f(\mu')$, we can also derive $\min_{\mu \in Q} f(\mu) - \min_{\mu' \in Q} g(\mu') \leq r$.

Till now, we obtain the final conclusion. □

Proof of Theorem 4.2. For any center $\nu \in C \subseteq \mathcal{P}(\mathcal{X})$ and $\mu'' \in \mathsf{Set}(\mu)$, there exists $\xi'' \in \mathsf{Set}(\xi)$ such that $W(\mu'', \xi'') \leq \frac{r}{1-\zeta}$ according to Lemma B.3.

By using the triangle inequality, we have

$$W(\xi'', \nu) - W(\mu'', \xi'') \leq W(\mu'', \nu) \leq W(\xi'', \nu) + W(\mu'', \xi''). \tag{21}$$

This leads to the following inequality

$$|W(\mu'', \nu) - W(\xi'', \nu)| \leq W(\mu'', \xi'') \leq \frac{r}{1-\zeta}. \tag{22}$$

The above result shows that for any $\mu'' \in \mathsf{Set}(\mu)$, there exists $\xi'' \in \mathsf{Set}(\xi)$ such that $W(\mu'', \nu) \in W(\xi'', \nu) \pm \frac{r}{1-\zeta}$.

Similarly, for any $\xi'' \in \mathsf{Set}(\xi)$, we also have $W(\xi'', \nu) \in W(\mu'', \nu) \pm \frac{r}{1-\zeta}$.

Let $g(\cdot) := W(\cdot, \nu)$. By applying Lemma B.4, we can deduce

$$\left| \min_{\mu'' \in \mathsf{Set}(\mu)} W(\mu'', \nu) - \min_{\xi'' \in \mathsf{Set}(\xi)} W(\xi'', \nu) \right| \leq \frac{r}{1-\zeta}, \tag{23}$$

which exactly implies $|\mathcal{W}(\mu, \nu) - \mathcal{W}(\xi, \nu)| \leq \frac{r}{1-\zeta}$.

According to (Ding et al., 2021), the output coreset S is an r-cover for Q under the Wasserstein distance. This means that for any $\mu \in Q$, there exists $\xi \in S$ such that $W(\mu, \xi) \leq r$. Consequently, for any $\mu \in Q$, there exists $\xi \in S$ such that $|\mathcal{W}(\mu, \nu) - \mathcal{W}(\xi, \nu)| \leq \frac{r}{1-\zeta}$.

Let $g(\cdot) = \mathcal{W}(\mu, \cdot)$ and $f(\cdot) = \mathcal{W}(\xi, \cdot)$. Using Lemma B.5, we obtain

$$\left| \min_{\nu \in Q} \mathcal{W}(\mu, \nu) - \min_{\nu' \in Q} \mathcal{W}(\xi, \nu') \right| \leq \frac{r}{1-\zeta}, \tag{24}$$

which implies $|\mathcal{W}(\mu, C) - \mathcal{W}(\xi, C)| \leq \frac{r}{1-\zeta}$.

By setting $g(\cdot) = -\mathcal{W}(\cdot, C)$, we have $|\max_{\mu \in Q} \mathcal{W}(\mu, C) - \max_{\xi \in S} \mathcal{W}(\xi, C)| \leq \frac{r}{1-\zeta}$ according to Lemma B.4.

Setting $r = \mathcal{O}(\epsilon\Gamma)$, we obtain

$$|\mathsf{cost}(Q, C) - \mathsf{cost}(S, C)| \leq \epsilon \cdot \mathsf{cost}(Q, C). \tag{25}$$

Time complexity: Algorithm 4 induces a tree with a maximum height of $\mathcal{O}(\log \frac{R}{r})$. Constructing each layer requires time $\mathcal{O}(2^{2 \cdot \mathsf{ddim}} \cdot |Q| \cdot \mathcal{T})$, thus the total time complexity is $\mathcal{O}(2^{2 \cdot \mathsf{ddim}} \cdot |Q| \cdot \mathcal{T} \cdot \log \frac{R}{r})$.

Coreset size: After executing the Gonzalez algorithm $\mathcal{O}(2^{2 \cdot \mathsf{ddim}})$ rounds, the radius is reduced by half. This implies that the degree of the tree is at most $\mathcal{O}(2^{2 \cdot \mathsf{ddim}})$. Therefore, the coreset size, which corresponds to the total number of nodes in the tree, is $\mathcal{O}((\frac{R}{r})^{2 \cdot \mathsf{ddim}})$.

□

By setting $g(\cdot) = \mathsf{cost}(Q, \cdot)$ and $f(\cdot) = \mathsf{cost}(S, \cdot)$, we obtain the following corollary according to Lemma B.5.

Corollary B.6. *Algorithm 4 takes Q as its input and outputs the coreset S. The output S satisfies the following property*

$$\begin{aligned} &|\min_{C \in \mathcal{P}(\mathcal{X}), |C|=k} \mathsf{cost}(Q, C) - \min_{C' \in \mathcal{P}(\mathcal{X}), |C'|=k} \mathsf{cost}(S, C')| \\ &\qquad \leq \min_{C \in \mathcal{P}(\mathcal{X}), |C|=k} \epsilon \cdot \mathsf{cost}(Q, C). \end{aligned}$$

C. Full Experiments

This section demonstrated the effectiveness of our RWC-clustering problem and seeding algorithm, as well as the efficiency of the coreset method. All the experiments were performed on a server with 2.40GHz Intel CPU, 60GB RAM, and Python 3.12. We utilized the POT library (Flamary et al., 2021) to compute the Wasserstein distance (WD) (Bonneel et al., 2011) and unbalanced optimal transport (UOT) (Chizat et al., 2018; Frogner et al., 2015). The reported results are averaged over five runs.

We validated our methods on the following datasets.

i) *Geometric shapes* is a toy dataset designed by us, consisting of five geometric shapes. It is used to verify the advantages of our seeding algorithm and the RWC-clustering problem. Each shape is represented by a point set.

ii) *MNIST* (LeCun et al., 1998) is a well-known handwritten digit dataset. For each image, we extract the pixels with higher grayscale values (greater than 0.3) to form the corresponding point set. The number of points in each resulting image ranges from 26 to 281.

iii) *ModelNet10* (Wu et al., 2015) is a standard dataset containing 3D CAD models. Each CAD model is discretized and represented as a point set.

For the point set $\{x_i\}_{i\in[n]}$ corresponding to a specific data item in the above datasets, we represent it as a probability measure $\mu = \sum_{i=1}^{n} \frac{1}{n}\delta_{x_i}$ to construct the clean dataset Q^0. The noisy dataset Q is generated by adding clustered noise with ζ mass following a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ to each clean probability measure.

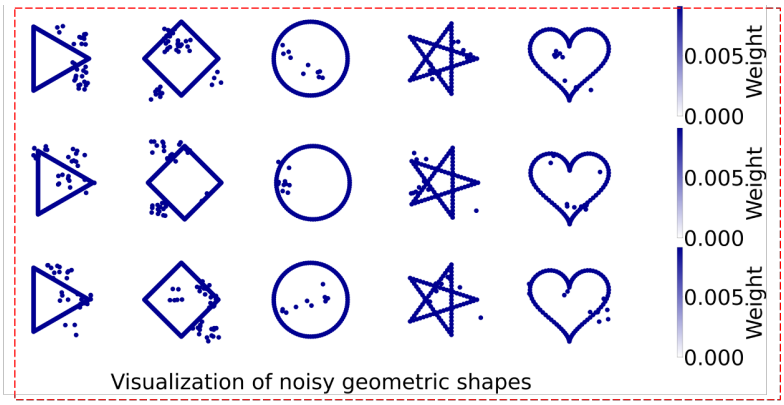


Figure 5: Visualization of noisy geometric shapes.

To evaluate the performance of our methods, we consider the following three criteria: i) Runtime: This includes the sampling time and the clustering time required to compute the k-center set C. ii) cost-cd: Defined as $\max_{\mu\in Q^0} \min_{\nu\in C} W(\mu,\nu)$, it quantifies the distance between the center set C and the original clean dataset Q^0. iii) cost-nd: Defined as $\max_{\mu\in Q} \min_{\nu\in C} \mathcal{W}(\mu,\nu)$, it evaluates the distance from center set C to the noisy dataset Q. The baselines for these three criteria are established using the results computed on the original full dataset for comparison.

Effectiveness of our method: To demonstrate the effectiveness of our seeding algorithm and the RWC-clustering problem, we conducted a series of experiments on the Geometric shapes dataset. We visualized the clean geometric shapes in Figure 6(a), and the noisy geometric shapes are in Figure 5.

Figure 6 validates the **effectiveness of our seeding algorithm** (Algorithm 1) for initialization. In this experiment, we first perform initialization and then apply Algorithm 2 as a post-processing step for clustering. Specifically, Figure 6(b) shows the clustering results with random initialization, while Figure 6(c) presents the results using our Algorithm 1 for initialization. Figure 6(b) shows poor denoising performance without using the seeding algorithm for initialization. In contrast, by using the seeding algorithm, the resulting centers in Figure 6(c) are more closely with the original five clean shapes. This implies the importance of a good initialization, and demonstrates the advantage of our seeding algorithm in providing a good starting point.

Figure 7 evaluates the **effectiveness of our RWC-clustering problem**. Specifically, Figure 7(a) displays the clustering

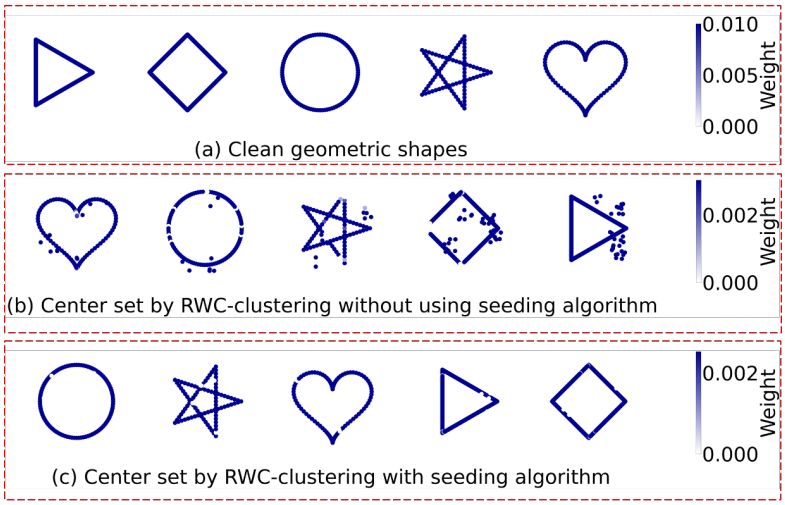


Figure 6: Effectiveness of our seeding algorithm.

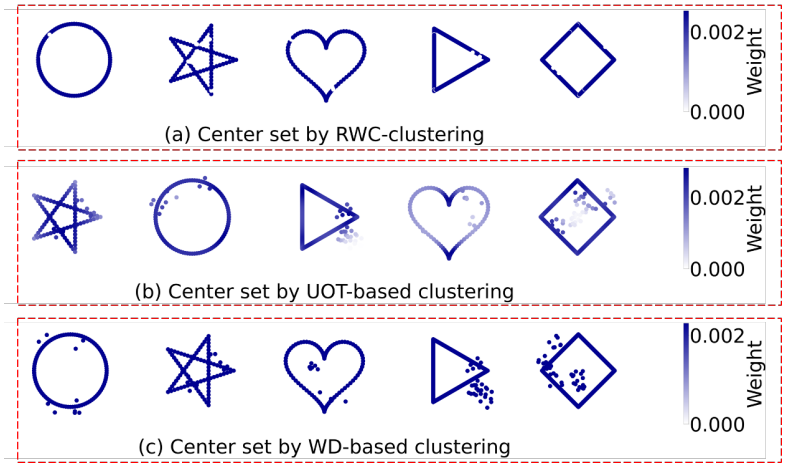


Figure 7: Comparing Our RWC-clustering with WD-Based clustering and UOT-Based clustering.

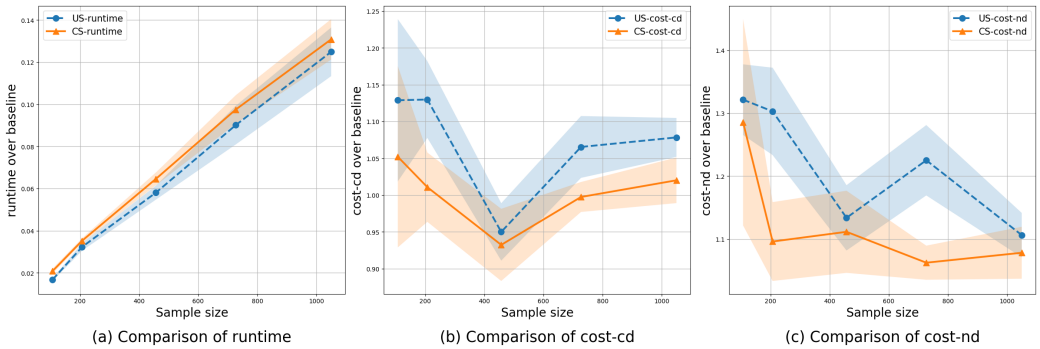
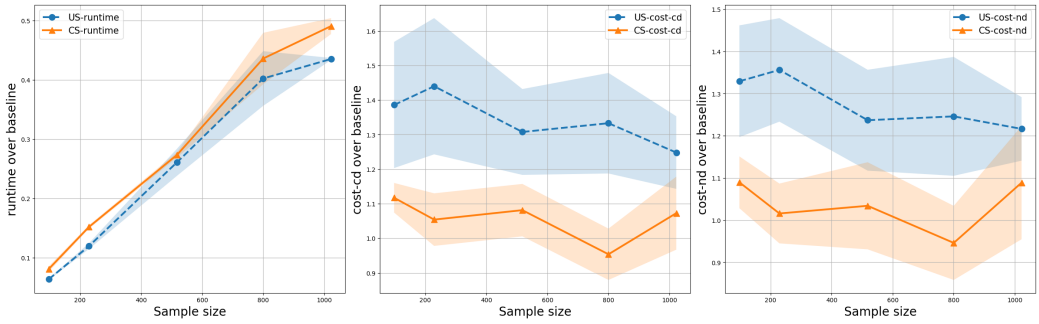


Figure 8: Comparison of the US method and our CS method across varying sample sizes on MNIST dataset.



(a) Comparison of runtime (b) Comparison of cost-cd (c) Comparison of cost-nd

Figure 9: Comparison of the US method and our CS method across varying sample sizes on ModelNet10 dataset.

results obtained using our approach, while Figure 7(b) shows results based on Unbalanced Optimal Transport (UOT), and Figure 7(c) presents clustering results using the classic Wasserstein distance (WD). All three methods follow the same framework, where seeding is used for initialization, followed by a local search refinement. Specifically, the UOT-based clustering algorithm is derived by replacing RWD with UOT in Algorithms 1 and 2. Since WD is a metric, thus the WD-based clustering can directly use the existing Gonzalez's algorithm (Gonzalez, 1985) along with the local search method (Lattanzi & Sohler, 2019; Choo et al., 2020).

Among these, the WD-based clustering exhibits the worst performance, showing almost no denoising capability. The UOT-based clustering outperforms the WD-based clustering; however, the inclusion of an entropy regularization term causes a diffusion effect that hinders noise removal, leaving some residual noise inescapably. In contrast, our RWC-clustering approach achieves the best denoising performance, outperforming the other two methods.

Table 4: Comparison of the US method and our CS method with varying values of k. We fix the sample size as 483, $\sigma = 1$, and $\zeta = 0.1$.

k	SM	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
10	US	$18.0_{\pm 1.8}$	$14.7_{\pm 0.9}$	$775.1_{\pm 49.6}$
	CS	$\mathbf{15.7}_{\pm 0.6}$	$\mathbf{13.8}_{\pm 0.8}$	$822.6_{\pm 81.4}$
20	US	$15.7_{\pm 0.6}$	$12.0_{\pm 0.8}$	$830.1_{\pm 16.2}$
	CS	$\mathbf{14.2}_{\pm 0.3}$	$\mathbf{11.9}_{\pm 0.2}$	$858.5_{\pm 15.4}$
30	US	$14.0_{\pm 1.4}$	$10.8_{\pm 0.5}$	$901.6_{\pm 62.1}$
	CS	$\mathbf{12.9}_{\pm 0.9}$	$\mathbf{10.7}_{\pm 0.7}$	$996.1_{\pm 75.0}$

Table 5: Comparison of the US method and our CS method using varying noisy intensity σ. We fix $k = 10$ and $\zeta = 0.1$.

σ	SM	SS	cost-nd(↓)	cost-cd(↓)	Runtime (↓)
1	US	474	$22.1_{\pm 0.7}$	$14.5_{\pm 1.3}$	$776.0_{\pm 40.6}$
	CS	474	$\mathbf{19.8}_{\pm 2.6}$	$\mathbf{13.6}_{\pm 0.4}$	$828.5_{\pm 44.9}$
2	US	479	$17.3_{\pm 1.1}$	$14.5_{\pm 0.9}$	$804.9_{\pm 77.6}$
	CS	479	$\mathbf{15.2}_{\pm 0.4}$	$\mathbf{13.7}_{\pm 0.7}$	$853.6_{\pm 38.7}$
3	US	457	$15.6_{\pm 1.5}$	$14.1_{\pm 0.6}$	$748.3_{\pm 42.7}$
	CS	457	$\mathbf{15.4}_{\pm 0.8}$	$\mathbf{13.8}_{\pm 0.8}$	$838.0_{\pm 39.1}$

Effectiveness of our coreset method: We show the effectiveness of our coreset (CS) method by comparing it against the uniform sampling (US) method.

Specifically, we first construct a coreset $S \subseteq Q$ and simultaneously generate a uniformly sampled subset S' of the same size $|S|$ from the full dataset Q as a baseline. We then run the clustering algorithm (i.e., Algorithms 1 and 2) on both subsets.

Table 6: Comparison of the US method and our CS method using varying mass ζ of noise. We fixed $k = 10$ and $\sigma = 1$.

ζ	SM	SS	cost-nd(↓)	cost-cd(↓)	Runtime (↓)
0.1	US	479	$17.3_{\pm 1.1}$	$14.4_{\pm 0.8}$	$695.9_{\pm 77.6}$
	CS	479	$\mathbf{15.2}_{\pm 0.4}$	$\mathbf{13.7}_{\pm 0.7}$	$741.6_{\pm 38.6}$
0.2	US	474	$22.1_{\pm 0.7}$	$14.5_{\pm 1.3}$	$726.1_{\pm 40.6}$
	CS	474	$\mathbf{19.8}_{\pm 2.6}$	$\mathbf{13.6}_{\pm 0.4}$	$778.5_{\pm 44.8}$
0.3	US	452	$33.4_{\pm 3.7}$	$\mathbf{15.9}_{\pm 3.7}$	$721.8_{\pm 22.9}$
	CS	452	$\mathbf{32.2}_{\pm 1.7}$	$16.2_{\pm 2.5}$	$757.5_{\pm 78.6}$

If the clustering algorithm is applied to the coreset S, we refer to it as the CS method. Conversely, if it is applied to the uniformly sampled subset S', we refer to it as the US method. Since the exact size of the coreset cannot be pre-specified, to ensure fairness between the two sampling methods (SM), we set the sample size (SS) of the US method to match that of the CS method. This results in somewhat irregular sample sizes, such as 479, 474 and 452 in Table 5.

Figure 8 illustrates the performance of our CS method on the MNIST dataset. Although our CS method is slightly more time-consuming compared to the US method, it remains significantly more efficient than processing the original dataset. Moreover, in terms of both cost-cd and cost-nd criteria, our CS method consistently outperforms the US method.

Figure 9 also shows that our CS method consistently surpasses the US method on both cost-cd and cost-nd criteria. Due to the randomness inherent of our algorithms and approximate nature of the derived k-center set, some fluctuations are inevitable. Overall, as the sample size increases, the performance improves. Particularly, for the primary metric cost-nd in this study, our CS method reflects this trend more clearly.

Ablation experiments: Taking the MNIST dataset as an example, we explore the effects of the parameter k, noise mass ζ and noise intensity σ on the performance of our coreset method.

Table 4 demonstrates that our CS method consistently outperforms the US method across different values of k under both the cost-cd and cost-nd criteria. Table 5 shows that under varying noise intensity levels σ, the CS method consistently achieves better performance compared to the US method. Table 6 illustrates that the CS method consistently outperforms the US method under the cost-nd criterion. Under the cost-cd criterion, the CS method generally performs better, except for a slight degradation when $\zeta = 0.3$. This minor degradation can be attributed to the stochastic nature of our algorithm, which is inescapable and reasonable.

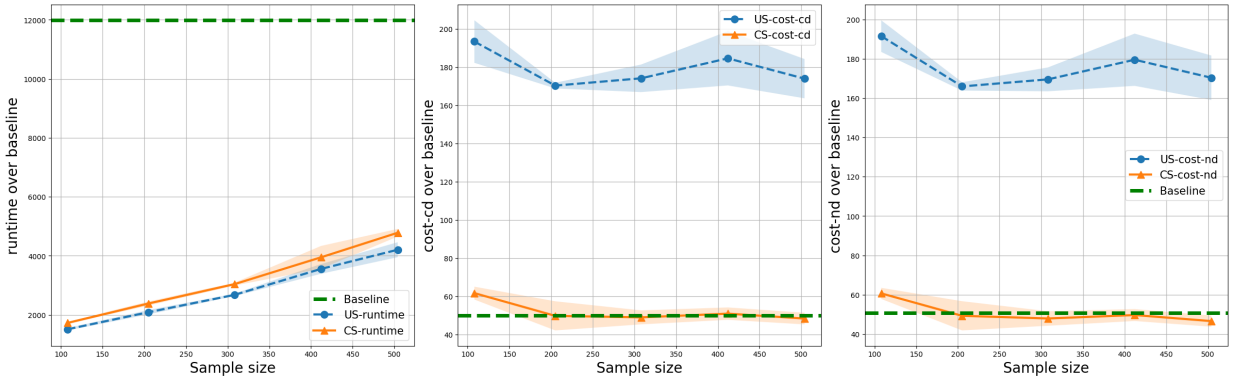
D. Experiments for review

D.1. Experiments for coreset

D.1.1. Experiments on S3DIS

The **S3DIS** (Armeni et al., 2016) is a large-scale indoor 3D point cloud dataset. It consists of 13 semantic categories, including ceiling, floor, wall, beam, column, window, door, table, chair, sofa, bookcase, board, and clutter. We first extract individual point cloud shapes from the dataset and then perform sampling on each shape, representing it with 300 points to form our point cloud dataset.

Efficiency of our coreset method: The green dashed line represents the results on the full dataset. From Figure 10, it can be seen that our CS method significantly outperforms the US method.



(a) Comparison of runtime (b) Comparison of cost-cd (c) Comparison of cost-nd

Figure 10: Comparison of the US method and our CS method across varying sample sizes on S3DIS dataset.

Results on different k: From Table 7, our CS method consistently has an advantage over the US method for different k.

Table 7: Comparison of the US method and our CS method with varying values of k on S3DIS dataset. We fix the sample size as 504, $\sigma = 1$, and $\zeta = 0.1$.

k	SM	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
10	US	$170.37_{\pm 11.29}$	$173.89_{\pm 10.28}$	$4205.60_{\pm 252.75}$
	CS	$\mathbf{46.53}_{\pm 2.83}$	$\mathbf{48.33}_{\pm 3.04}$	$4225.40_{\pm 129.21}$
20	US	$160.31_{\pm 0.38}$	$166.94_{\pm 0.07}$	$5160.80_{\pm 145.69}$
	CS	$\mathbf{27.77}_{\pm 2.80}$	$\mathbf{28.77}_{\pm 2.30}$	$5244.40_{\pm 86.73}$
30	US	$160.12_{\pm 0.43}$	$166.12_{\pm 1.71}$	$5958.20_{\pm 322.21}$
	CS	$\mathbf{24.49}_{\pm 5.45}$	$\mathbf{25.33}_{\pm 5.66}$	$5897.80_{\pm 209.25}$

Ablation experiments on τ: From Table 8, a smaller τ leads to faster computation and lower cost. Thus, we recommend using a relatively small τ.

Table 8: Comparison of our CS method using varying parameter τ of noise. We fix the sample size as 504, $\sigma = 1$, and $\zeta = 0.1$.

τ	SM	SS	cost-nd(↓)	cost-cd(↓)	Runtime (↓)
1	CS	504	55.15	60.03	3453.58
2	CS	504	50.42	52.05	3333.24
3	CS	504	50.70	51.38	3395.46
4	CS	504	49.65	50.48	3407.61
5	CS	504	**43.64**	**44.89**	4032.00
10	CS	504	46.53	48.33	4225.40
20	CS	504	49.00	51.02	5145.33
50	CS	504	57.75	58.37	11173.33
100	CS	504	53.51	57.49	31920.00

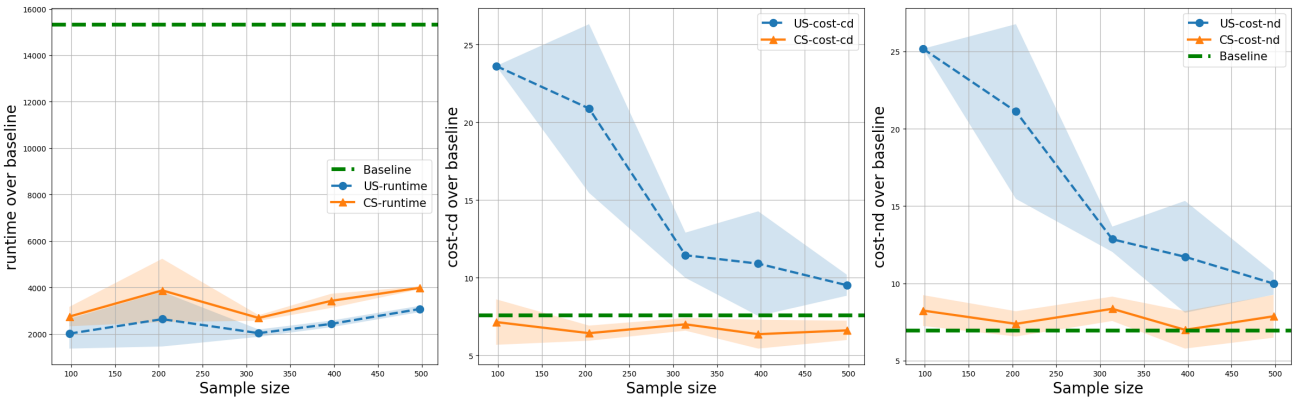
Selection of ζ: Each of our data items contains 0.1 mass of noise, but we run experiments with varying ζ. The results in Table 6 confirm that slightly overestimating ζ has only a minor impact, while underestimating it severely degrades the solution quality. Therefore, when the actual noise mass is unknown, we recommend setting ζ slightly larger than the expected noise mass.

Table 9: Comparison of our CS method with varying values of ζ on S3DIS dataset with 0.1 mass of noise. We fix the sample size as 504 and $\sigma = 1$.

ζ	SM	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
0.05	CS	$104.96_{\pm 11.80}$	$206.60_{\pm 6.81}$	$3991.60_{\pm 278.49}$
0.1	CS	$46.53_{\pm 2.83}$	$48.33_{\pm 3.04}$	$4225.40_{\pm 129.21}$
0.2	CS	$43.52_{\pm 52.93}$	$52.93_{\pm 7.41}$	$4148.60_{\pm 156.73}$
0.3	CS	$45.62_{\pm 3.32}$	$59.94_{\pm 3.96}$	$4237.60_{\pm 40.49}$

D.1.2. Experiments on KITTI

The **KITTI** dataset (Geiger et al., 2012) is a widely used benchmark for autonomous driving, containing 3D LiDAR scans. We use its 3D object detection subset, which contains point clouds for various categories such as *Pedestrian*, *Cyclist*, *Car*, *Van*, *Truck*, *Person_sitting*, *Tram*, and *Misc*. We extract object-level point cloud instances from the dataset and uniformly sample each to 300 points, forming our point cloud dataset.



(a) Comparison of runtime (b) Comparison of cost-cd (c) Comparison of cost-nd

Figure 11: Comparison of the US method and our CS method across varying sample sizes on KITTI dataset.

Efficiency of our coreset method: The green dashed line represents the results on the full dataset. From Figure 11, it can be seen that our CS method significantly outperforms the US method.

Table 10: Comparison of the US method and our CS method with varying values of k on KITTI dataset. We fix the sample size as 498, $\sigma = 1$, and $\zeta = 0.1$.

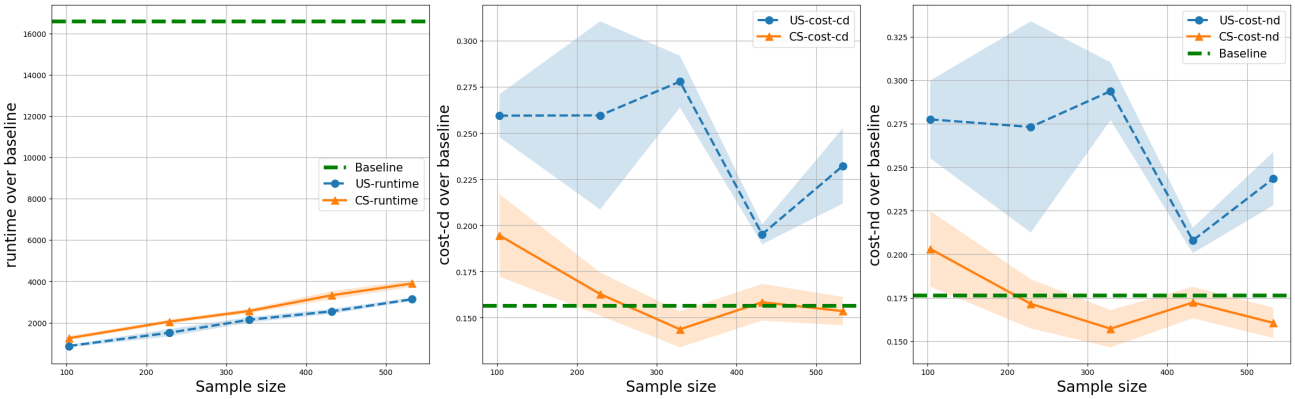
k	SM	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
10	US	$9.52_{\pm 0.68}$	$9.98_{\pm 0.72}$	$3074.89_{\pm 135.57}$
	CS	$\mathbf{6.62}_{\pm 0.62}$	$\mathbf{7.87}_{\pm 1.38}$	$3987.10_{\pm 62.02}$
20	US	$6.02_{\pm 0.72}$	$6.44_{\pm 0.47}$	$3345.67_{\pm 36.58}$
	CS	$\mathbf{3.58}_{\pm 0.29}$	$\mathbf{5.00}_{\pm 0.62}$	$4314.92_{\pm 66.38}$
30	US	$5.06_{\pm 0.00}$	$5.85_{\pm 0.14}$	$3525.40_{\pm 94.05}$
	CS	$\mathbf{2.90}_{\pm 0.38}$	$\mathbf{3.68}_{\pm 0.06}$	$4605.90_{\pm 125.72}$

Results on different k: From Table 10, our CS method consistently has an advantage over the US method for different k.

D.1.3. Experiments on ShapeNetCore

ShapeNetCore (Chang et al., 2015) is a dataset containing a large collection of 3D object models. It includes 55 categories, such as chairs, tables, cars, airplanes, and other common objects. We extract object-level point cloud instances from the dataset and uniformly sample each to 300 points, forming our point cloud dataset.

Efficiency of our coreset method: The green dashed line represents the results on the full dataset. From Figure 12, it can be seen that our CS method significantly outperforms the US method.



(a) Comparison of runtime (b) Comparison of cost-cd (c) Comparison of cost-nd

Figure 12: Comparison of the US method and our CS method across varying sample sizes on ShapeNetCore dataset.

Results on different k: From Table 11, our CS method consistently has an advantage over the US method for different k.

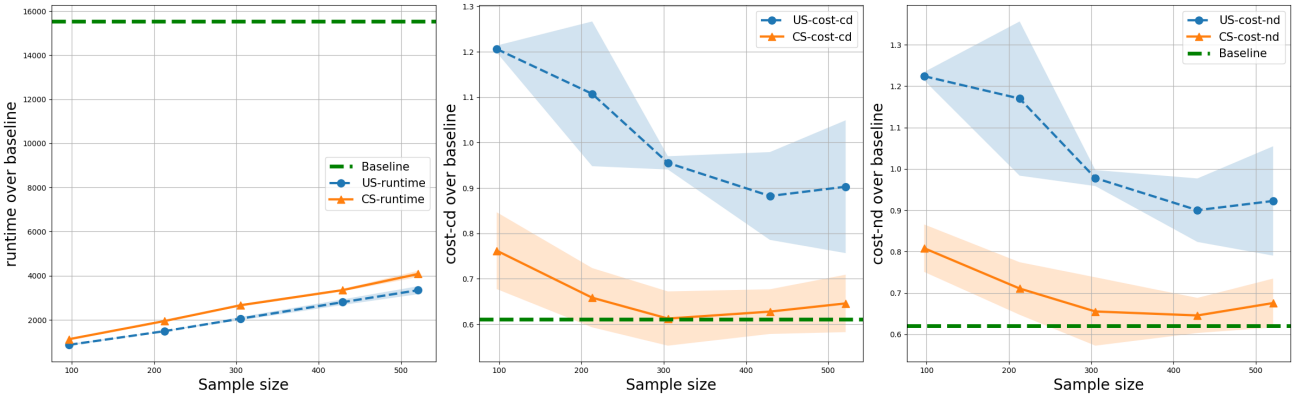
Table 11: Comparison of the US method and our CS method with varying values of k on ShapeNetCore dataset. We fix the sample size as 533, $\sigma = 1$, and $\zeta = 0.1$.

k	SM	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
10	US	$0.23_{\pm 0.02}$	$0.24_{\pm 0.02}$	$3132.53_{\pm 81.48}$
	CS	$\mathbf{0.15}_{\pm 0.01}$	$\mathbf{0.16}_{\pm 0.01}$	$3896.53_{\pm 166.72}$
20	US	$0.18_{\pm 0.03}$	$0.20_{\pm 0.03}$	$3712.89_{\pm 180.86}$
	CS	$\mathbf{0.13}_{\pm 0.01}$	$\mathbf{0.14}_{\pm 0.01}$	$4430.57_{\pm 59.18}$
30	US	$0.17_{\pm 0.01}$	$0.18_{\pm 0.01}$	$4476.98_{\pm 48.44}$
	CS	$\mathbf{0.11}_{\pm 0.01}$	$\mathbf{0.13}_{\pm 0.02}$	$5170.38_{\pm 17.25}$

D.1.4. Experiments on ScanObjectNN

ScanObjectNN (Uy et al., 2019) is a real-world 3D object classification benchmark, which is captured from real scans, making it more challenging than synthetic datasets such as ModelNet10. We extract object-level point cloud instances from the dataset and uniformly sample each to 300 points, forming our point cloud dataset.

Efficiency of our coreset method: The green dashed line represents the results on the full dataset. From Figure 13, it can be seen that our CS method significantly outperforms the US method.



(a) Comparison of runtime (b) Comparison of cost-cd (c) Comparison of cost-nd

Figure 13: Comparison of the US method and our CS method across varying sample sizes on ScanObjectNN dataset.

Results on different k: From Table 12, our CS method consistently has an advantage over the US method for different k.

Table 12: Comparison of the US method and our CS method with varying values of k on ScanObjectNN dataset. We fix the sample size as 521, $\sigma = 1$, and $\zeta = 0.1$.

k	SM	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
10	US	$0.90_{\pm 0.15}$	$0.92_{\pm 0.13}$	$3345.48_{\pm 179.84}$
	CS	$\mathbf{0.65}_{\pm 0.06}$	$\mathbf{0.68}_{\pm 0.06}$	$4082.18_{\pm 142.71}$
20	US	$0.73_{\pm 0.02}$	$0.73_{\pm 0.02}$	$3631.17_{\pm 181.59}$
	CS	$\mathbf{0.56}_{\pm 0.02}$	$\mathbf{0.57}_{\pm 0.02}$	$4477.70_{\pm 185.45}$
30	US	$0.72_{\pm 0.05}$	$0.72_{\pm 0.05}$	$4205.75_{\pm 214.16}$
	CS	$\mathbf{0.38}_{\pm 0.02}$	$\mathbf{0.39}_{\pm 0.03}$	$4934.63_{\pm 127.15}$

D.2. Experiments for seeding

Table 13 validates the **effectiveness of our seeding algorithm** (Algorithm 1) for initialization. In this experiment, we first perform initialization and then apply Algorithm 2 as a post-processing step for clustering. The experimental results show that our seeding algorithm consistently achieves better performance on these datasets.

Table 13: Comparison of seeding algorithm for initialization and random initialization on different datasets. We fix the dataset size as 500, $\sigma = 1$, and $\zeta = 0.1$.

Dataset	Initialization	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
S3DIS	random	$15.20_{\pm 2.20}$	$16.18_{\pm 3.75}$	$3658.67_{\pm 156.94}$
	seeding	$\mathbf{12.76}_{\pm 0.60}$	$\mathbf{13.06}_{\pm 0.48}$	$3762.06_{\pm 54.35}$
KITTI	random	$4.67_{\pm 0.45}$	$5.43_{\pm 1.20}$	$3680.96_{\pm 26.08}$
	seeding	$\mathbf{3.93}_{\pm 0.26}$	$\mathbf{4.95}_{\pm 0.64}$	$3765.64_{\pm 42.54}$
ShapeNetCore	random	$0.08_{\pm 0.01}$	$0.08_{\pm 0.01}$	$3595.80_{\pm 81.51}$
	seeding	$\mathbf{0.06}_{\pm 0.01}$	$\mathbf{0.06}_{\pm 0.00}$	$3904.79_{\pm 64.84}$
ScanObjectNN	random	$0.15_{\pm 0.02}$	$0.17_{\pm 0.02}$	$3323.02_{\pm 30.03}$
	seeding	$\mathbf{0.14}_{\pm 0.00}$	$\mathbf{0.15}_{\pm 0.01}$	$3439.52_{\pm 10.48}$

D.3. Experiments for RWC-clustering

To demonstrate the effectiveness of our seeding algorithm and the RWC-clustering problem, we conducted a series of experiments on the several datasets.

Tables 14 to 17 evaluates the **effectiveness of our RWC-clustering problem**. All three methods follow the same framework, where seeding is used for initialization, followed by a local search refinement. Specifically, the UOT-based clustering algorithm is derived by replacing RWD with UOT in Algorithms 1 and 2. Since WD is a metric, thus the WD-based clustering can directly use the existing Gonzalez's algorithm (Gonzalez, 1985) along with the local search method (Lattanzi & Sohler, 2019; Choo et al., 2020).

Among these, the WD-based clustering exhibits the worst performance. The UOT-based clustering outperforms the WD-based clustering; however, the inclusion of an entropy regularization term causes a diffusion effect that hinders noise removal, leaving some residual noise inescapably. In contrast, our RWC-clustering approach achieves the best denoising performance, outperforming the other two methods.

Table 14: Comparison of our RWC-clustering with UOT-based clustering and WD-based clustering on S3DIS dataset with 0.1 mass of noise. We fix the dataset size as 500 and $\sigma = 1$.

Method	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
RWC-clustering	$\mathbf{14.16}_{\pm 0.79}$	$\mathbf{14.52}_{\pm 0.76}$	$12588.32_{\pm 697.77}$
UOT-based clustering	$25.25_{\pm 2.10}$	$27.01_{\pm 2.89}$	$20833.29_{\pm 3096.17}$
WD-based clustering	$33.99_{\pm 5.68}$	$47.37_{\pm 1.18}$	$4120.79_{\pm 982.49}$

Table 15: Comparison of our RWC-clustering with UOT-based clustering and WD-based clustering on KITTI dataset with 0.1 mass of noise. We fix the dataset size as 500 and $\sigma = 1$.

Method	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
RWC-clustering	$\mathbf{3.23}_{\pm 0.36}$	$\mathbf{3.83}_{\pm 0.16}$	$3196.76_{\pm 533.97}$
UOT-based clustering	$10.93_{\pm 4.37}$	$14.01_{\pm 5.28}$	$1373.75_{\pm 97.39}$
WD-based clustering	$8.63_{\pm 0.49}$	$12.93_{\pm 0.89}$	$1173.18_{\pm 104.29}$

Table 16: Comparison of our RWC-clustering with UOT-based clustering and WD-based clustering on ShapeNetCore dataset with 0.1 mass of noise. We fix the dataset size as 500 and $\sigma = 1$.

Method	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
RWC-clustering	$\mathbf{0.14}_{0.00}$	$\mathbf{0.16}_{0.01}$	$2605.34_{22.91}$
UOT-based clustering	$0.59_{0.08}$	$0.67_{0.08}$	$1152.31_{218.31}$
WD-based clustering	$0.34_{0.02}$	$0.62_{0.03}$	$1236.69_{206.04}$

Table 17: Comparison of our RWC-clustering with UOT-based clustering and WD-based clustering on ScanObjectNN dataset with 0.1 mass of noise. We fix the dataset size as 500 and $\sigma = 1$.

Method	cost-nd(↓)	cost-cd(↓)	Runtime(↓)
RWC-clustering	$\mathbf{0.06}_{0.01}$	$\mathbf{0.06}_{\pm 0.01}$	$3395.96_{\pm 116.10}$
UOT-based clustering	$0.18_{0.01}$	$0.25_{\pm 0.00}$	$1131.50_{\pm 90.53}$
WD-based clustering	$1.15_{0.10}$	$1.76_{\pm 0.11}$	$1376.86_{\pm 206.49}$