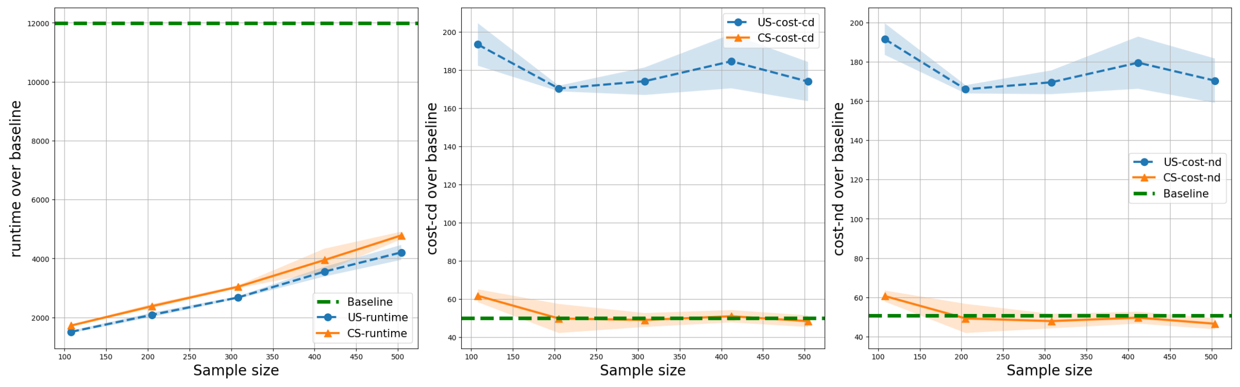
D. Experiments for coreset

D.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.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.

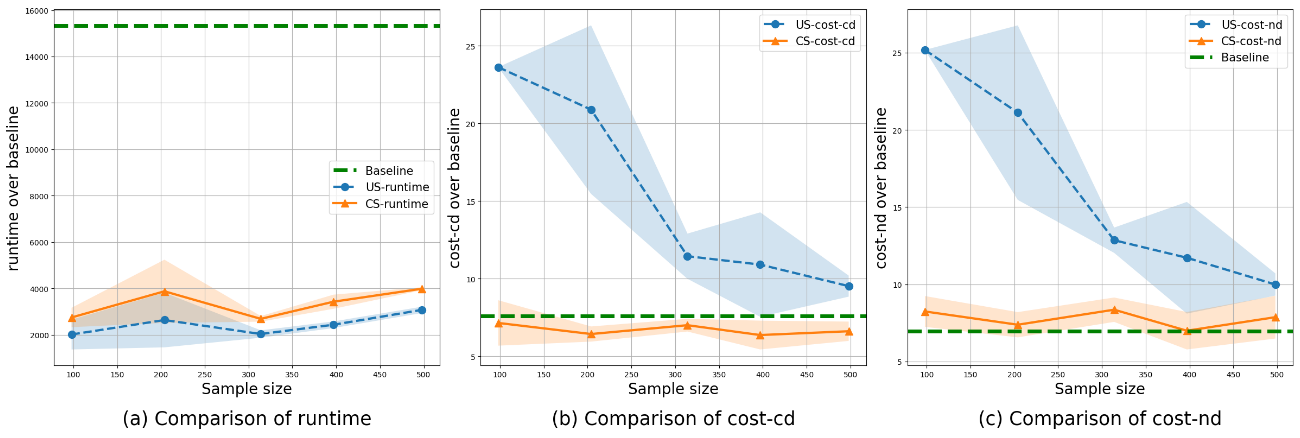


Figure 11: Comparison of the US method and our CS method across varying sample sizes on KITTI dataset.

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 $\zeta = 0.1$.

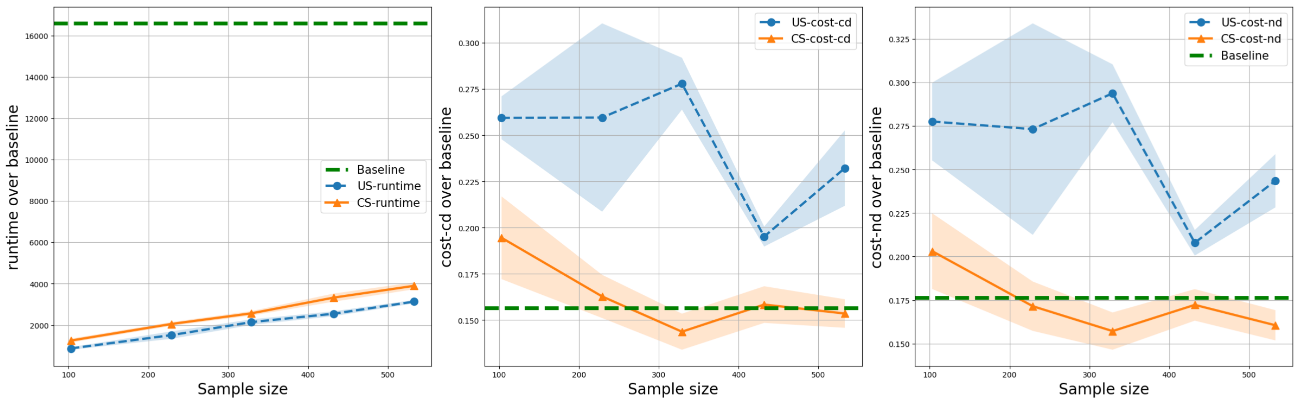
k	SM	cost-nd($\downarrow$)	cost-cd($\downarrow$)	Runtime($\downarrow$)
10	US	$9.52_{\pm 0.68}$	$9.98_{\pm 0.72}$	$3074.89_{\pm 135.57}$
	CS	$\mathbf{6.62}_{\pm 0.62}$	$\mathbf{7.87}_{\pm 1.38}$	$3987.10_{\pm 62.02}$
20	US	$6.02_{\pm 0.72}$	$6.44_{\pm 0.47}$	$3345.67_{\pm 36.58}$
	CS	$\mathbf{3.58}_{\pm 0.29}$	$\mathbf{5.00}_{\pm 0.62}$	$4314.92_{\pm 66.38}$
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.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.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.

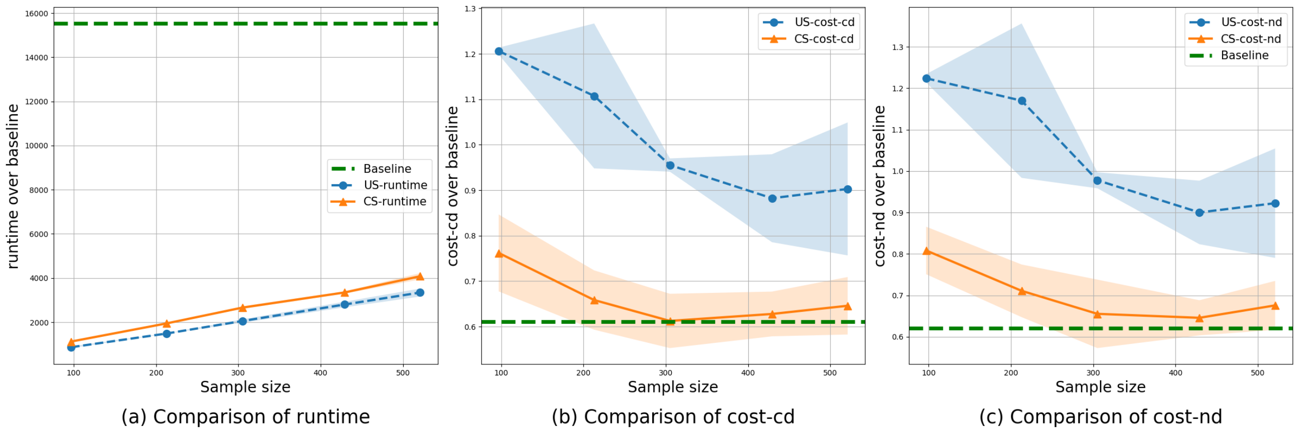


Figure 13: Comparison of the US method and our CS method across varying sample sizes on ScanObjectNN dataset.

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	$\mathbf{0.38}_{\pm 0.02}$	$\mathbf{0.39}_{\pm 0.03}$	$4934.63_{\pm 127.15}$

E. 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	$\mathbf{0.14}_{\pm 0.00}$	$\mathbf{0.15}_{\pm 0.01}$	$3439.52_{\pm 10.48}$

F. 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	$1.15_{0.10}$	$1.76_{\pm 0.11}$	$1376.86_{\pm 206.49}$