Pseudo-Code

Algorithm 1: Prunning based clustering

Input: *A*: the point set; *n_clusters*: the target number of clusters;

Output: *S*: the tree set;

Algorithm: KMeans: clusters generated by k-means; Delaunay: performs the triangulation and returns the network; Kruskal: generates the MST;

rough clustering by k-means

G ← *KMeans*(*A*);

S ← ∅, add *Kruskal*(*Delaunay*(*gi*)) into *S*, all *gi* ∈ *G*;

while length of *S* < *n_clusters* **do**

find the cluster with the highest STD in *S* as *t0*;

prun *t0* into *t1* and *t2*, where the higher STD of *t1* and *t2* is the lowest case;

S ← *S* \ t0; add *t1*, *t2* to *S*;

end while;

return *S*;

Algorithm 2: Attribute spatial association based sampling

Input: *A*: the point set; *r*: radius parameter; *min_r*: minimum radius;

Output: *S*: sampled set;

Algorithm: Random: randomly returns an element from the set; KDE: kernel-density estimate; Label: the label of point, processed by the clustering algorithm; Dist: the Euclidean distance between the given two points; Entropy: the entropy of the set;

S ← ∅; *T* ← ∅;

while *A* ≠ ∅ **or** *T* ≠ ∅ **do**

if *T* ≠ ∅ **then**

c ← *Random*(*T*); *T* ← *T* \ *c*; add *c* into *S*;

else

c ← *Random*(*A*); *A* ← *A* \ *c*; add *c* into *S*;

radius ← *Max*(*r* / *KDE*(*c*), *min_r*);

neighbors ← ∅;

add *Label*(*pi*) into *neighbors*, *Dist*(*c*, *pi*) ≤ *radius*, all *pi* ∈ (*A* + *T*);

radius ← (*radius* - *min_r*) / (1 + *Entropy*(*neighbors*)) + *min_r*;

disabled ← ∅; *active* ← ∅;

add *pi* into disabled, *Dist*(*c*, *pi*) ≤ *radius*, all *pi* ∈ (*A* + *T*);

add *pi* into *active*, *radius* < *Dist*(*c*, *pi*) ≤ 2 * *radius*, all *pi* ∈ (*A* + *T*);

A ← *A* \ *pi*, all *pi* ∈ (*disabled* + *active*);

T ← *T* \ *pi*, all *pi* ∈ (*disabled* + *active*);

add *pi* into *T*, all *pi* ∈ *active*;

end while;

return *S*;
