DOI: 10.1111/tgis.12463

WILEY Transactions in GIS

RESEARCH ARTICLE

HiSpatialCluster: A novel high-performance software tool for clustering massive spatial points

Yiran Chen[1,2] | Zhou Huang[1,2] | Tao Pei[3] | Yu Liu[1,2]

[1]Institute of Remote Sensing and Geographical Information Systems, Peking University, Beijing 100871, China

[2]Beijing Key Lab of Spatial Information Integration & Its Applications, Peking University, Beijing 100871, China

[3]State Key Laboratory of Resources and Environmental Information System, Institute of Geographical Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing 100101, China

Correspondence
Zhou Huang, Institute of Remote Sensing and Geographic Information Systems, Peking University, Beijing 100871, China.
Email: huangzhou@pku.edu.cn

Funding information
National Key Research and Development Program of China, Grant/Award No. 2017YFB0503602; National Natural Science Foundation of China, Grant/Award Nos. 41771425, 41625003.

Abstract

In the era of big data, spatial clustering is a very important means for geo-data analysis. When clustering big geo-data such as social media check-in data, geotagged photos, and taxi trajectory points, traditional spatial clustering algorithms are facing more challenges. On the one hand, existing spatial clustering tools cannot support the clustering of massive point sets; on the other hand, there is no perfect solution for self-adaptive spatial clustering. In order to achieve clustering of millions or even billions of points adaptively, a new spatial clustering tool—HiSpatialCluster—was proposed, in which the CFSFDP (clustering by fast search and finding density peaks) idea to find cluster centers and the DBSCAN (density-based spatial clustering of applications with noise) idea of density-connect filtering for classification are introduced. The tool's source codes and other resources have been released on Github, and experimental evaluation was performed through clustering massive taxi trajectory points and Flickr geotagged photos in Beijing, China. The spatial clustering results were compared with those through K-means and DBSCAN as well. As a spatial clustering tool, HiSpatialCluster is expected to play a fundamental role in big geo-data research. First, this tool enables clustering adaptively on massive point datasets with uneven spatial density distribution. Second, the density-connect filter method is applied to generate homogeneous analysis units from geotagged data. Third, the tool is accelerated by both parallel CPU and GPU computing so that millions or even billions of points can be clustered efficiently.

1 | INTRODUCTION

Spatial clustering, as a powerful technology for exploratory data analysis, has been widely applied to epidemic monitoring, geographic customer segmentation, crime hotspot analysis, land-use detection, seismicity research, and so on (Miller & Han, 2009). Spatial patterns of natural events and environmental features can be distinguished by means of spatial clustering, such as the atmospheric pressure patterns and sea surface temperature distributions in polar and oceanic regions (Birant & Kut, 2007; Tan, Steinbach, & Kumar, 2005), and spatial patterns of trajectories such as tropical cyclones (Lee, Han, & Whang, 2007). Clustering analysis of spatial patterns is significant in understanding global and local climate change and geoscientific modeling.

Therefore, spatial clustering, as a mainstream spatial analysis method is a very important research topic in GIScience (Miller & Han, 2009; Ng & Han, 2002; Pei, Gao, Ma, & Zhou, 2012). There is much research on spatial clustering: a number of clustering algorithms have been developed for spatial data. These algorithms can be classified into seven groups (Deng, Liu, Cheng, & Shi, 2011), including partitioning algorithms, hierarchical algorithms, density-based algorithms, graph-based algorithms, model-based algorithms, grid-based algorithms, and various combinational algorithms (see Table 1). In addition, the mainstream GIS software (e.g., ArcGIS) also provides a suite of spatial clustering functions, which is meaningful to many other scientific fields.

As a kind of clustering, spatial clustering can be achieved by applying the general clustering algorithm to two-dimensional point data. However, spatial clustering has its own particularity. It is a clustering method for spatial point sets and the similarity measure is based on geographic distance. Not all general clustering algorithms are suitable for spatial clustering. There are two difficulties in spatial clustering: clustering of classes with irregular spatial shapes and adaptive clustering for datasets with uneven spatial density distribution. The density-based clustering algorithm, represented by density-based spatial clustering of applications with noise (DBSCAN), can solve the first problem to some extent, but there is still no perfect solution for the second problem.

Particularly in the era of big data, spatial clustering becomes a more important tool for geo-data analysis (e.g., it is frequently used to discover "natural" human activity hotspots from geotagged data). Some scholars have used spatial clustering in mining and discovering city landmarks (Ji, Xie, Yao, & Ma, 2009), points of interest (Lee, Cai, & Lee, 2013, 2014 ; Yang, Gong, & U, 2011), functional regions (Yuan, Zheng, & Xie, 2012), spatial patterns in origin–destination mobility data (Guo, Zhu, Jin, Gao, & Andris, 2012), tourist destinations (Zhou, Xu, & Kimmons, 2015), and tourist areas (Vu, Li, Law, & Ye, 2015). Another fundamental function of spatial clustering in big geo-data research is to generate "natural" and homogeneous analysis units. For example, when analyzing taxi trajectories and traffic pollutants emissions with common units such as regular grids, administrative areas, or traffic districts, it is unavoidable to divide the hotspots of the real world into several areas. Owing to such divisions, the research unit is often heterogeneous, resulting in differences between analysis results and the actual phenomena. One way to address this problem is to use the spatial clusters of geotagged data as research units. Through spatial clustering of the geotagged data, such as social media check-in data and geotagged photos, the "natural" boundaries of human activities are generated. This means that relatively homogeneous research units can be obtained, which facilitate performing further geo-data analysis in various urban and geoscientific applications. Hence, spatial clustering is expected to play a fundamental role in big geo-data research.

When clustering big geo-data such as social media check-in data, geotagged photos, mobile phone data, and taxi trajectory points, the spatial density distribution of the point set is extremely uneven and the amount of data is huge. Hence, traditional spatial clustering algorithms are facing more challenges. On the one hand, existing spatial clustering tools cannot support the clustering of datasets with massive points. Tools with spatial clustering functions, such as ArcGIS, CLUTO (https://glaros.dtc.umn.edu/gkhome/views/cluto), and Weka (https://www.cs.waikato.ac.nz/ml/weka/), consist of many spatial clustering algorithms. The widely used Python-based package Scikit-learn also contains popular spatial clustering algorithms such as K-means and DBSCAN. However, these tools and software have difficulty supporting clustering on millions or even billions of points in the big data environment. On the other hand, there is no perfect solution for adaptively clustering big geo-data with an uneven

TABLE 1 Seven classes of clustering algorithms

Class	Method
Partitioning algorithms	K-means (MacQueen, 1967), CLARANS (Ng & Han, 2002)
Hierarchical algorithms	CURE (Guha, Rastogi, & Shim, 1998), AMOEBA (Estivill-Castro & Lee, 2002b)
Density-based algorithms	DBSCAN (Ester, Kriegel, Sander, & Xu, 1996), GDBSCAN (Sander, Ester, Kriegel, & Xu, 1998), OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999), SNN (Ertöz, Steinbach, & Kumar, 2003), CLNN (Pei, Zhu, Zhou, Li, & Qin, 2009), PDBSCAN (Kisilevich, Mansmann, & Keim, 2010), ACOMCD (Wan et al., 2012)
Graph-based algorithms	AUTOCLUST (Estivill-Castro & Lee, 2002a)
Model-based algorithms	EM (expectation-maximization), SOM (self-organizing feature map)
Grid-based algorithms	STING (Wang, Yang, & Muntz, 1997), WaveCluster (Sheikholeslami, Chatterjee, & Zhang, 1998)
Combinational algorithms	NN-Density (Pei, Zhu, Zhou, Li, & Qin, 2006), CSM (Lin & Chen, 2005), WNNEM (Pei et al., 2012)

spatial density distribution. Therefore, a high-performance and adaptive spatial clustering approach should be developed. The contributions of this study are as follows:

1. A novel high-performance and adaptive spatial clustering approach was proposed. First, the proposed method combines clustering by fast search and finding density peaks (CFSFDP)'s idea of finding cluster centers and DBSCAN's idea of density-connect filtering for classification. Second, this method is improved by GPU computing that enables clustering millions or even billions of spatial points. Clustering experiments were performed on massive taxi trajectory points and Flickr geotagged photos. The results illustrate that it achieves the clustering of classes with irregular spatial shapes and is an adaptive density-based clustering method for datasets with uneven spatial density distribution. Therefore, our method is especially suitable for clustering massive spatial point sets in the big geo-data environment.
2. Besides elaborating on the method, we have also implemented a high-performance spatial clustering tool named HiSpatialCluster and put its open-source code on Github (https://github.com/lopp2005/HiSpatialCluster). We believe that when a good idea becomes a practical software tool, it will greatly facilitate researchers and promote relevant scientific progress. In particular, currently the research work on big geo-data is rising rapidly, and spatial clustering plays a fundamental role in big geo-data mining. It is of scientific significance to do this practical work. As far as we know, except for HiSpatialCluster, there are no publicly available software tools that can support spatial clustering of billions of geotagged points.

The remainder of this article is organized as follows. Section 2 gives an overview of the algorithms. Section 3 introduces the software implementation and usage. Section 4 presents the application and evaluation of HiSpatialCluster on taxi trajectory points and Flickr geotagged photos in Beijing, China. Section 5 provides a summary and directions for future work.

2 | OVERVIEW OF THE ALGORITHMS

Assuming that the density of cluster center points is relatively high and the distance between the cluster center points may be larger than that between inner-cluster points, Rodriguez and Laio (2014) introduced the algorithm named CFSFDP. This algorithm calculates the density ρ_i for each point i and then searches for the nearest point j with higher density. The distance between point i and point j is denoted by δ_i. Based on the above assumptions, the cluster center points naturally have a large value of $\rho_i * \delta_i$. In other words, while the products of ρ_i and δ_i are calculated and ordered, the cluster center points are the top points in the sequence of products.

The clustering algorithm HiSpatialCluster is improved based on CFSFDP, combined with DBSCAN's density-connect filtering strategy and the GPU-based parallel computing technique that enables clustering of massive spatial points, especially on the scale of millions or even billions of points. The three main parts of the algorithm, namely finding cluster centers, classification, and parallel acceleration, are described as follows.

2.1 | Finding cluster centers

CFSFDP describes the method of finding cluster centers using two parameters, the density ρ_i and the distance δ_i from the nearest point with higher density.

2.1.1 | Density calculation

In a point dataset $DS = \{\boldsymbol{X}_i\}_{i=1}^{N}$, where $\boldsymbol{X}_i$ is the coordinate vector of point i, local density ρ_i can be represented by the cutoff density ρ_i^{cutoff} or Gauss kernel density ρ_i^{gauss}:

$$\rho_i^{cutoff} = \sum_{j=1}^{N} w_j \chi\left(dist\left(\boldsymbol{X}_i, \boldsymbol{X}_j\right) - cutoffthres\right) \quad \chi(x) = \begin{cases} 1 & if\, x < 0 \\ 0 & if\, x \geq 0 \end{cases} \tag{1}$$

$$\rho_i^{gauss} = \sum_{j=1}^{N} w_j e^{-\left(\frac{dist\left(X_i, X_j\right)}{\sigma}\right)^2} \tag{2}$$

where w_j is the weight of $\boldsymbol{X}_j$, $dist\left(\boldsymbol{X}_i, \boldsymbol{X}_j\right)$ is the distance between $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, $cutoffthres$ is the cutoff distance, $\chi(x)$ is the indicator function, and σ is the parameter in the Gauss convolution kernel.

The value of ρ_i^{gauss} is a float number, which can avoid equals in comparison between densities that may be caused by the integer value of ρ_i^{cutoff}.

Algorithm 1: Density calculation

Input: Point set, denoted by S

Output: Points with density

```
for X_i ∈ S do
    X_i.ρ ← 0
    for X_j ∈ S do
        if cutoff density is used then
            if dist(X_i, X_j) < cutoffthres then
                X_i.ρ ← X_i.ρ + w_j
        end if
        else if Gauss kernel density is used then
            X_i.ρ ← X_i.ρ + w_j e^{-(dist(X_i, X_j)/σ)^2}
        end if
    end for
end for
Output S
```

2.1.2 | Calculating distance from nearest higher-density point

After the definition of local density above, the parent point $\boldsymbol{X}_i^{parent}$ for each point i and the distance parameter δ_i can be defined as follows: $\boldsymbol{X}_i^{parent}$ is the nearest point to point $\boldsymbol{X}_i$ in the set $DS_i' = \{\boldsymbol{X}_j | (\boldsymbol{X}_j \in DS \wedge \rho_j > \rho_i)\}$ composed of all points with higher densities, and $\delta_i = dist\left(\boldsymbol{X}_i, \boldsymbol{X}_i^{parent}\right)$. Equally,

$$\delta_i = \begin{cases} \min\limits_{j:\rho_j > \rho_i} dist\left(\boldsymbol{X}_i, \boldsymbol{X}_j\right) & \text{if } \exists\, j{:}\rho_j > \rho_i \\ +\infty & otherwise \end{cases} \tag{3}$$

When $\boldsymbol{X}_i$ is the point with highest density in DS, there is no parent point $\boldsymbol{X}_i^{parent}$ and δ_i can be defined as $+\infty$.

Algorithm 2: Calculating distance from nearest higher-density point

Input: Point set with density, denoted by S

Output: Points with distance parameter

```
for X_i ∈ S do
    X_i.δ ← +∞
        X_i.parent ← null
            for X_j ∈ S do
                if X_j.ρ > X_i.ρ and dist(X_i, X_j) < X_i.δ then
                    X_i.δ ← dist(X_i, X_j)
                    X_i.parent ← X_j
                end if
        end for
end for
Output S
```

2.1.3 | Finding cluster centers

Based on the above assumptions, points with larger density and distance parameters are the cluster centers. After the calculation of density and distance, the scatterplot of $\rho_i - \delta_i$ can be drawn. In the scatterplot of $\rho_i - \delta_i$, the noise points are far from the other points and have low density, close to the δ_i-axis in the plot. The inner-cluster points are close to other points and have high density, generally close to the ρ_i-axis in the plot. Cluster centers appear at the top right of the plot. By ordering and plotting the product $m_i = \rho_i \delta_i$, the cluster centers could be determined by the natural breaks of the sequence, where the m_i values of cluster centers are larger.

This method shows robustness on clustering in areas with uneven density distribution. Taking the taxi trajectory points in Beijing, China as an example, the downtown has higher point density and the suburban point density is one order of magnitude lower. When generating the analysis units via spatial clustering, the distance parameter δ_i in the suburbs is greater than that downtown, while the density is lower, so the m_i values are relatively evenly distributed in space. The parameter δ_i, which is the distance between cluster centers, represents the area of the cluster; therefore, the area of the suburban cluster is greater than that of the downtown cluster, confirmed by the reality that it is suitable to use larger units for spatial analytics in suburbs because of the low density of human activities. In practice, it is difficult to find natural breaks in the very smooth curve of the plot of m_i, which means that it is impossible to automatically distinguish cluster centers by plotting $\rho_i - \delta_i$ or m_i. Therefore, in HiSpatialCluster

the number of clusters n is defined based on a priori knowledge, and the top n points in the m_i sequence are taken as cluster centers X_i^{center}.

In this step, the number of clusters n has a great effect on the clustering results. If the n value is set too high, the final cluster will be too small; otherwise, the cluster will be too large. Hence, the n value is set to determine the scale of the clustering results. For example, when clustering the Flickr geotagged photos to discover popular tourist attractions, some small-scale attractions such as Seventeen-Arch Bridge, Jade Belt Bridge, and West Causeway in the Summer Palace would be classified into separate clusters if the n value is set higher, while these sub-attractions would be merged into one cluster (i.e., the Summer Palace) if the n value is set lower. A feasible option is to determine the most suitable n via assessing the rationality of the distribution of output hotspots after inputting different n values.

2.2 | Classification

After cluster centers are determined, the remaining unclassified points are then classified based on their dependencies on the parent point X_i^{parent}. However, after classification, some output clusters might have an uneven inner density distribution, especially when the n value is set lower. In order to ensure the homogeneity of each cluster, an improved filter method is used for post-processing.

2.2.1 | Pre-classification

Assuming that DS_{center} is the set comprising all cluster centers, naturally the class of X_i should be the same as X_i^{parent}:

$$class\left(X_i\right)=\begin{cases} class\left(X_i^{parent}\right) & if\ X_i \notin DS_{center} \\ C_i & if\ X_i \in DS_{center} \end{cases} \tag{4}$$

where $class\left(X_i\right)$ is the class of X_i and C_i is the class of cluster center X_i^{center}.

Algorithm 3: Classification

Input: Point set, denoted by S

Output: Points with class tags

```
function classify(X_i):
    if X_i is class center of C_i then
        return C_i
    else
        return classify(X_i.parent)
    end if
for X_i ∈ S do
    X_i.classtag ← classify(X_i)
end for
Output S
```

2.2.2 | Filter method based on density-connect

Because the number of clusters is distinguished by a priori knowledge, an inappropriate setting would cause some problems. On the one hand, an overestimated cluster number may lead to fragmentation in similar regions, which affects the recognition of hotspots and the regularity of regions. On the other hand, if the number of clusters is

underestimated, regions with different characteristics may be merged together, which expands the internal heterogeneity of clusters, leading to trouble in describing the spatial patterns.

The following methods are presented to address these issues (1) For the problem of too many clusters, semantic features other than spatial coordinates (such as tags or text topics) can be used for class mergence. The semantic similarity between clusters can be defined, and thus similar clusters can be merged; (2) A filter method based on density-connect is proposed to resolve the problem of underestimated number of clusters, which results in internal heterogeneity. The number of clusters would not be changed after filtering; only the dense region near the cluster center is retained by the filter method, with the noise points and other regions removed, which eliminates the effects of noise and enables more effective discovery of hotspots.

Density-connect is defined in DBSCAN as follows (Ester et al., 1996):

Definition 1: (ε-neighborhood of a point) The *ε-neighborhood* of a point p, denoted by $N_\varepsilon(p)$, is defined by $N_\varepsilon(p) = \{q \in D | dist(p,q) \le \varepsilon\}$.

Definition 2: (directly density-reachable) A point p is *directly density-reachable* from point q if

1. $p \in N_\varepsilon(q)$ and
2. $|N_\varepsilon(q)| \ge MinPts$

where *MinPts* is the minimum number for an ε-neighborhood of a point in a cluster.

Definition 3: (density-reachable) A point p is *density-reachable* from a point q if there is a chain of points $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

Definition 4: (density-connected) A point p is *density-connected* to a point q if there is a point o such that both p and q are density-reachable from o.

If only the points density-connected to the cluster center points are retained, then the noise points and separated clusters are removed, which reduces the internal heterogeneity in the regions.

After introducing density-connect, the main changes are as follows: (1) The density defined in Section 2.1.1 and the density threshold, denoted by d_{thres}, are used instead of $|N_\varepsilon(q)| \ge MinPts$ in Definition 2; and (2) The ε-neighborhood becomes the minimum interval of the adjacent clusters. If the distance for the two closest points between two clusters is greater than ε, then the two clusters can be regarded as separate clusters.

By the definition of density threshold, denoted by d_{thres}, and the threshold of the distance between clusters, denoted by ε, the cluster could be filtered via density-connect. Only the points density-connected to the cluster center points are retained, which can reduce the internal heterogeneity of the cluster.

2.2.3 | Generating class boundaries

There are no explicit boundaries in spatial patterns described by discrete points. It is difficult to distinguish different clusters in visualization by original points. Hence, a boundary-generating algorithm is proposed for visualization. The boundaries generated by this algorithm are reference boundaries for assisting visualization and decision-making. Nonetheless, the reference boundaries provide some convenience for subsequent analysis and processing, allowing researchers to visually grasp the spatial patterns.

Algorithm 4: Generating class boundaries

Input: Point set with class tags, denoted by *S*

Output: Reference boundary polygons

1: *Delaunay Triangles*←*Generating Delaunay triangles in all points*

2: *Inner-class Triangles*←*Ø*

3: **for** *triangle* ∈ *Delaunay Triangles* **do**

```
    if three vertexes of triangle have the same class tags then
        Inner-class Triangles.append(triangle)
    end if
end for
Polygons←Merge Inner-class Triangles with the same class tags
Output Polygons
```

The Delaunay triangle is chosen instead of the Voronoi diagram because some locations in the study area do not belong to any clusters, where no phenomena are to be analyzed. For example, there should be no taxi trajectory points over the water area. The Voronoi diagram provides a global division, where any region belongs to a class. Therefore, it is not appropriate to use the Voronoi diagram to describe spatial patterns.

2.3 | Parallel acceleration

HiSpatialCluster supports accelerating calculations of the density and the distance from the nearest high-density point in Sections 2.1.1 and 2.1.2 by means of GPU-based parallel computing. The time complexity of these two steps is $O(n^2)$, while a simple dual-loop implementation is very time-consuming.

2.3.1 | CPU parallel acceleration

Because no write conflicts exist in calculating the density and distance from the nearest high-density point, the concurrency control is based on the division of points. The transmission of calculating tasks for threads and the reception of results are accomplished through the blocking queue.

Algorithm 5: CPU parallel acceleration

Input: Point set, denoted by *S*

Output: Results containing density or distance from nearest high-density point, denoted by *R*

```
Task Queue←generate task points from S
Result Queue←Ø
for thread do
    for Task Queue≠Ø do
        Task Point←Task Queue.get()
        Result←Ø
        for X_i ∈ S do
            Calculate density or nearest point (Task Point, X_i, Result)
        end for
        Result Queue.put(Result)
    end for
end for
transform result queue to array R
Output R
```

2.3.2 | GPU acceleration

Owing to the large differences in the CPU and GPU architectures, the CPU parallel acceleration algorithm cannot be directly migrated to GPU. Moreover, since GPU needs to control the action of each CUDA thread, some divergence exists between the CPU and GPU algorithms.

Algorithm 6: GPU acceleration

Input: ID of CUDA threads, denoted as *i*

Point dataset, denoted as *S*

Empty result array, denoted as *R*

Output: NULL

1: *Task Point*←*S[i]*
2: *Result*←*Ø*
3: **for** $X_i \in S$ **do**
4: *Calculate density or nearest point (Task Point, X_i, Result)*
5: **end for**
6: *R[i]=Result*

3 | SOFTWARE IMPLEMENTATION

The software architecture of HiSpatialCluster is shown in Figure 1. The tool contains three major calculating steps and two post-processing steps, and the functional codes are separated into individual modules. The tool depends on the Python packages of numpy and multiprocessing.dummy for CPU parallel acceleration, and a queue is used for data exchange. The Python packages of numba and cudatoolkit are used for GPU acceleration.

ArcGIS Python toolboxes are geoprocessing toolboxes created entirely in Python. A Python toolbox and the tools contained within look, act, and work in a similar manner to toolboxes and tools created in any other way. The ArcGIS Python toolbox allows prototype and fully functional geoprocessing tools to be quickly created (https://desktop.arcgis.com/en/arcmap/latest/analyze/creating-tools/a-quick-tour-of-python-toolboxes.htm). Python toolboxes are used as the UI (user interface) of HiSpatialCluster (Figure 2), and HiSpatialCluster with Python scripts can be run in a standalone environment.

The following three major steps are required for clustering spatial points:

- Step 1: Calculate the density and obtain the intermediate results described in Section 2.1.1 by the *Calculating Density Tool*. The tool supports both the Gauss kernel and the cutoff kernel.
- Step 2: Input the intermediate results of Step 1 and calculate the distance from the nearest high-density point described in Section 2.1.2 by the *Finding Nearest High-Density Points Tool*.
- Step 3: Input the intermediate results of Step 2, find the cluster center, and classify by the given number of clusters described in Section 2.2.1 by the *Finding Centers and Classification Tool*. The intermediate results of Step 2 can be reused, and the different numbers of clusters do not require recalculation of Steps 1 and 2.

Steps 1 and 2 are the most time-consuming in HiSpatialCluster. The intermediate results of the two steps can avoid recalculation caused by errors or crashes, and the CPU parallel and GPU acceleration can accelerate the calculation times.

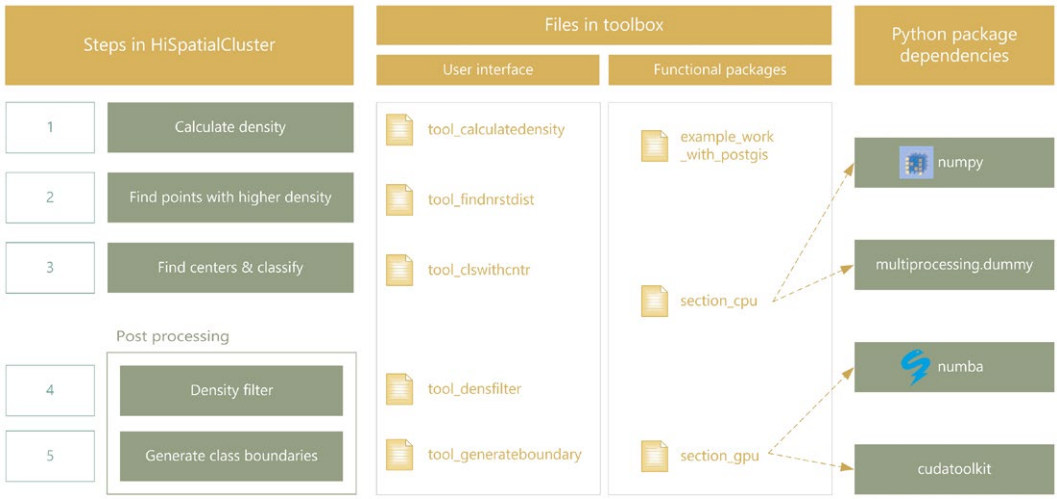


FIGURE 1 Software architecture of HiSpatialCluster.

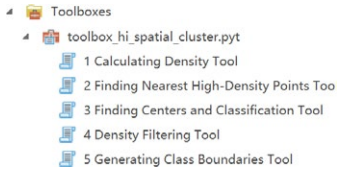
The two post-processing tools are the *Density Filtering Tool* and the *Generating Class Boundaries Tool*. The *Density Filtering Tool* can filter the clusters by the inputs of points and cluster centers and the given thresholds of density and distance. The *Generating Class Boundaries Tool* generates class reference boundaries described in Section 2.2.3 by the inputs of points.

4 | EXPERIMENTAL EVALUATION

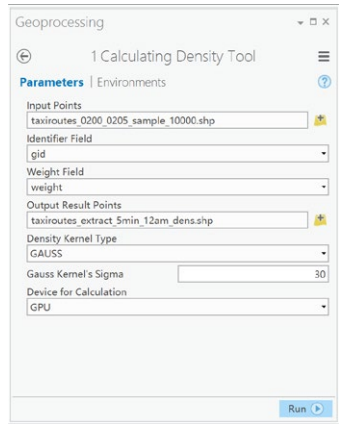
The experimental cases on big geo-data, such as taxi trajectory points and Flickr geotagged photos in Beijing, China, were presented for the evaluation of HiSpatialCluster. First, the tool's stability and reliability were verified via processing big geo-data with different volumes and discovering spatial patterns. Second, HiSpatialCluster was compared with traditional methods (i.e., K-means and DBSCAN). The *Grouping Analysis Tool* in ArcGIS was selected as an implementation of K-means, and a DBSCAN-based tool was developed based on PostGIS/PostgreSQL. Third, the effectiveness of the filter method based on density-connect, which enables generating homogeneous analysis units, was tested. Last, the elapsed time by the three different clustering approaches was presented.

4.1 | Test datasets

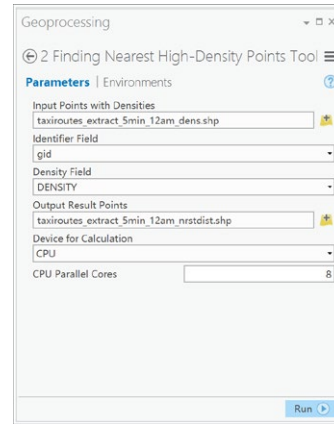
To perform clustering experiments, two typical datasets in the big geo-data environment, including taxi trajectory points and Flickr geotagged photos, were obtained from the Beijing Traffic Management Department and the Flickr online data access interface (https://www.flickr.com/services /api/), respectively. Taxi trajectory points were collected from June to August, 2013 at 10s intervals, with a total of approximately 2.85 billion points. There are taxi codes and timestamp information in addition to geographic location for each point. We also used Flickr geotagged photos in the area of Summer Palace, Beijing, China as the secondary test dataset, collected from January 1, 2005 to January 1, 2016. The descriptions of these two datasets are shown in Tables 2 and 3.t



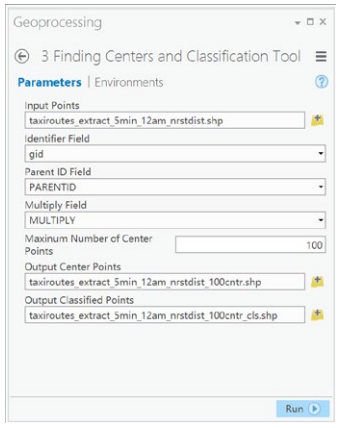
(a) Tool set in the ArcGIS framework



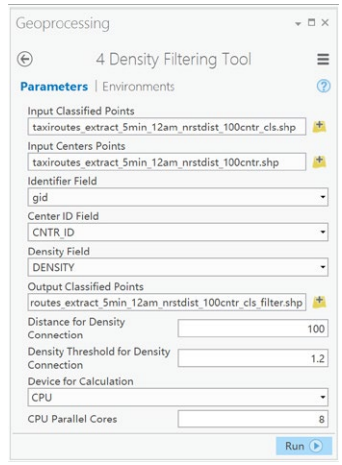
(b) Calculating density tool



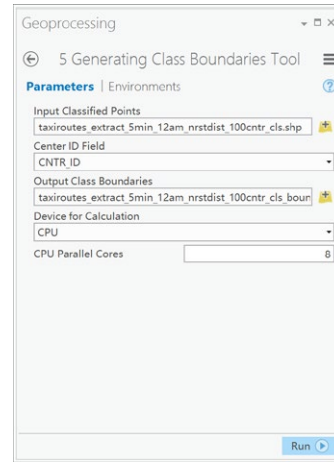
(c) Finding nearest high-density points tool



(d) Finding centers and classification tool



(e) Density filtering tool



(f) Generating class boundaries tool

FIGURE 2 Software UI of HiSpatialCluster.

4.2 | Clustering results

First, clustering experiments were performed on the dataset of taxi trajectory points. We chose 1,000 as the number of clusters to demonstrate the clustering output of massive trajectory points via HiSpatialCluster. In this case, the clustering results are similar to the scale of road segments, because the taxi trajectory points are mainly scattered on both sides of the road.

To reveal different spatial patterns by clustering results, the taxi trajectory data from 2:00–2:05 a.m and 12:00–12:05 a.m were chosen for clustering. The number of points from 2:00–2:05 a.m is approximately 400,000, and there are one million points from 12:00–12:05 a.m These two periods represent peak and trough hours for traveling, in which the clustering results might demonstrate clearly different spatial patterns. The taxi trajectory points were clustered through HiSpatialCluster and K-means, and visualized in the region of a part of Haidian, Beijing for comparison (see Figure 3). The algorithm in Section 2.2.3 was used to generate the class boundaries for visualization. In addition, the setting of the K-mean parameter k (i.e., the number of clusters)

TABLE 2 Descriptions on the dataset of taxi trajectory points

Time period	Total (May, 2013–July, 2013)	2:00–2:05 a.m every day	12:00–12:05 a.m every day
Spatial extent	42.021°N 114.951°E 39.003°N 118.010°E		
Number of points	2,846,277,995	406,524	1,008,352
Taxi count	33,841	16,138	25,928
Average number of points per taxi	84,107.38	25.19	38.89
Maximum number of points per taxi	765,786	545	529
Minimum number of points per taxi	1	1	1
Standard deviation of points per taxi	63,431.89	37.65	43.63

TABLE 3 Descriptions on the dataset of Flickr geotagged photos

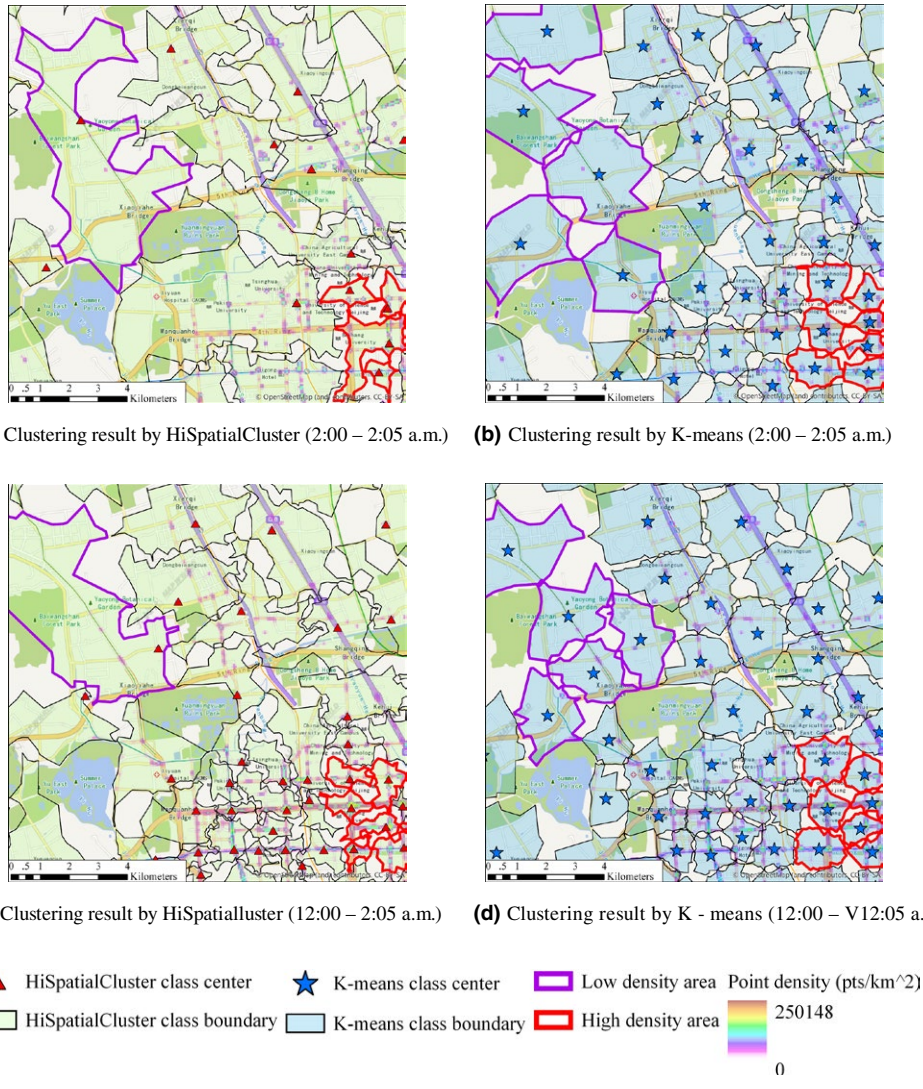
Time period	January 1, 2005–January 1, 2016
Spatial extent	40.003°N 116.259°E 39.979°N 116.282°E
Number of photos	7,424
User count	1,894
Average number of photos per user	3.92
Minimum number of photos per user	1
Maximum number of photos per user	117
Standard deviation of photos per user	5.89

can only depend on prior knowledge, which is also considered a weakness of K-means. For the convenience of comparison, we also chose 1,000 as the *k* value to perform K-mean clustering.

Clustering results for taxi trajectory points in different time periods illustrate different spatial patterns. Clustering results for the taxi trajectory points in trough and peak periods (2:00–2:05 a.m and 12:00–12:05 a.m) were visualized in the region of a part of Haidian, Beijing, as shown in Figures 3a and c. Because a low trough of taxi operations exists at 2:00 a.m, the overall density downtown is much lower than that at 12:00 a.m, while in suburban areas the density is relatively low and less variable at these two moments. It is observed that HiSpatialCluster tends to divide larger areas into clusters with lower density (in the suburbs) and smaller areas into clusters with higher density (downtown). In addition, the number of clusters at 12:00 a.m is significantly more than that at 2:00 am, especially downtown. In the real world, there are indeed more human activity hotspots in peak hours than in trough hours. The division in spatial patterns is well reflected through HiSpatialCluster.

4.3 | Comparison with K-means

HiSpatialCluster can find arbitrary clusters with irregular spatial shapes, and cluster adaptively on datasets with uneven spatial density distribution, namely clusters with high density have smaller areas and areas of clusters with low



(a) Clustering result by HiSpatialCluster (2:00 – 2:05 a.m.) **(b)** Clustering result by K-means (2:00 – 2:05 a.m.)

(c) Clustering result by HiSpatialluster (12:00 – 2:05 a.m.) **(d)** Clustering result by K - means (12:00 – V12:05 a.m.)

FIGURE 3 Clustering results on taxi trajectory points by HiSpatialCluster and K-means.

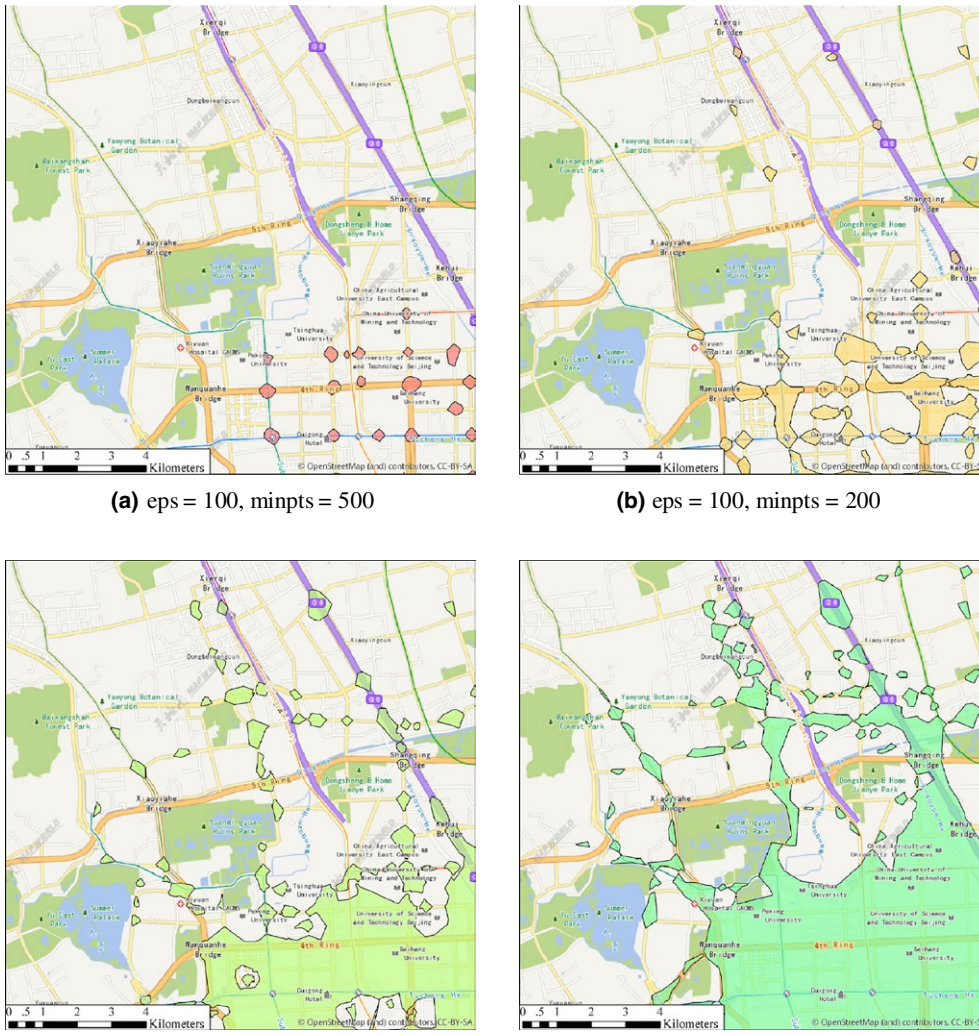
density are larger. K-means does not have these advantages. Comparing the results of HiSpatialCluster and K-means, as shown in Figure 3, K-means tends to result in an even distribution of oval-shaped clusters because of the rules of minimizing intra-class distance and maximizing inter-class distance. The centers are not fit to the density peaks. In fact, the northwest area (purple outline) is a relatively cold area with low density, which is the suburban area outside the Fifth Ring of Beijing; meanwhile the southeast area downtown (red outline) is of high density. Compared with K-means, HiSpatialCluster can cluster much more adaptively in these areas, and the cluster centers are close to the density peaks.

4.4 | Comparison with DBSCAN

As the current DBSCAN package (e.g., Scikit-learn) without GPU or distributed computing supports cannot cluster so many trajectory points directly, we developed a DBSCAN-based clustering tool based on PostGIS/PostgreSQL

with spatial indexes. Four numbers of parameter ϵ and six numbers of parameter *MinPts* for each ϵ were chosen. Some of the DBSCAN-based clustering results are shown in Figure 4, and others are illustrated in Appendix II. The results show that no matter how the parameters are set, DBSCAN cannot achieve adaptive clustering for point sets with uneven spatial density distribution. Figures 4c, d, g, and h show that the clusters cover a wide area with low values of parameter *MinPts*, resulting in the merging of different hotspots. Figures 4a, b, e, and f demonstrate how the high value of parameter *MinPts* leads to large non-classified areas. Compared with the DBSCAN results, it is observed that HiSpatialCluster enables clustering more adaptively for the taxi trajectory dataset and the output clusters are more evenly spatially distributed, whether in low-or high-density areas.

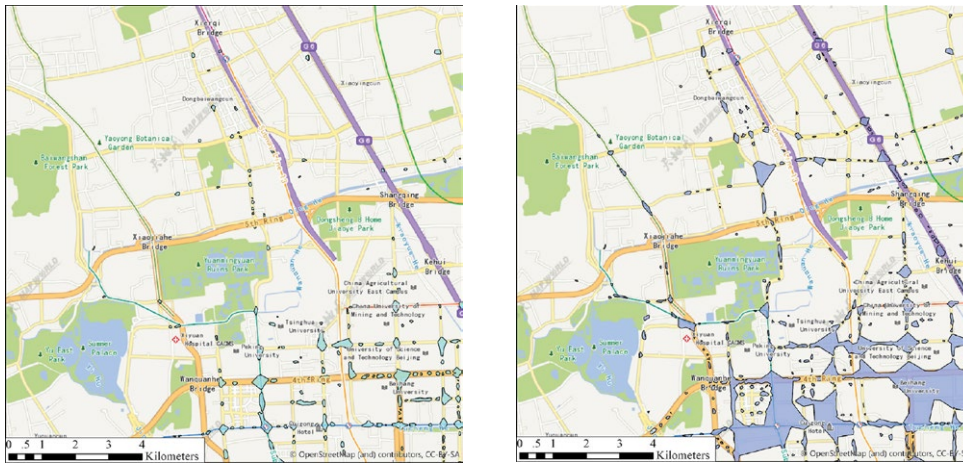
To better illustrate the advantages of HiSpatialCluster, the secondary test dataset (i.e., Flickr geotagged photos) was also used for clustering. DBSCAN and HiSpatialCluster were applied respectively to observe the differences in clustering results. As shown in Figure 5, three DBSCAN results under different parameter configurations



(a) eps = 100, minpts = 500 **(b)** eps = 100, minpts = 200

(c) eps = 100, minpts = 100 **(d)** eps = 100, minpts = 50

FIGURE 4 Clustering results by DBSCAN on the taxi trajectory dataset (12:00–12:05 a.m).



(e) eps = 30, minpts = 50 **(f)** eps = 30, minpts = 20

(g) eps = 30, minpts = 10 **(h)** eps = 30, minpts = 5

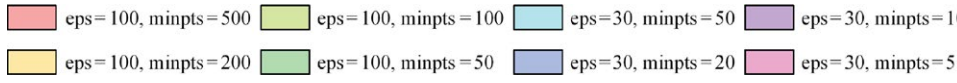
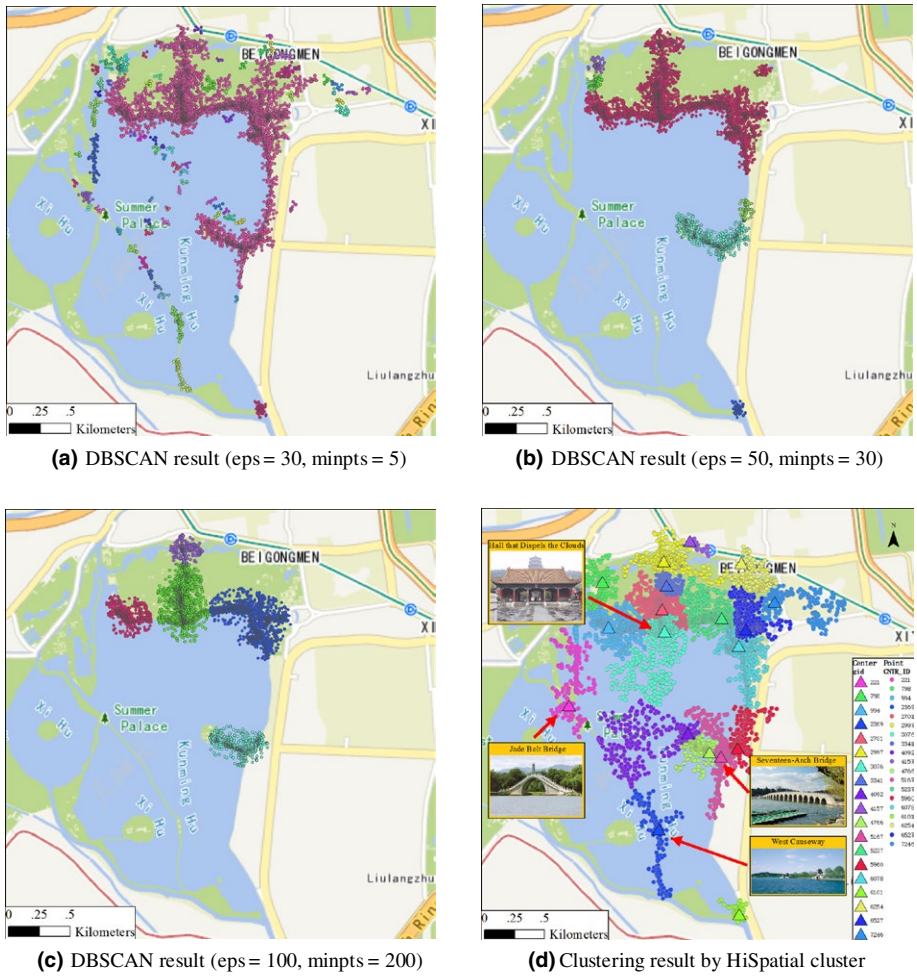


FIGURE 4 (*Continued*)

and a HiSpatialCluster result were obtained for comparison. In this figure, the points of the same category are rendered in the same color, that is, different colors represent different clusters. Figures 5a, b, and c show that the number of clusters decreases significantly with increasing parameter values. There are two main drawbacks of DBSCAN: (1) if the parameter values are set too low, a large number of real hotspots will be merged together; and (2) When the parameter values are set too high, only a small number of high-density clusters are reserved, which leads to large non-classified areas. These make it impossible for DBSCAN to achieve adaptive clustering. Figure 5d demonstrates that the clusters generated by HiSpatialCluster are much more evenly distributed in the global scope. Through spatial clustering, 19 hotspots (i.e., popular attractions) were discovered in the Summer Palace, including Seventeen-Arch Bridge, Jade Belt Bridge, Hall that Dispels the Clouds, West Causeway, and so on. In particular, the shapes of these clusters match the real boundaries of the attractions well. Therefore, it illustrates that HiSpatialCluster can not only cluster adaptively, but also discover arbitrary clusters with irregular spatial shapes.



(a) DBSCAN result (eps = 30, minpts = 5)

(b) DBSCAN result (eps = 50, minpts = 30)

(c) DBSCAN result (eps = 100, minpts = 200)

(d) Clustering result by HiSpatial cluster

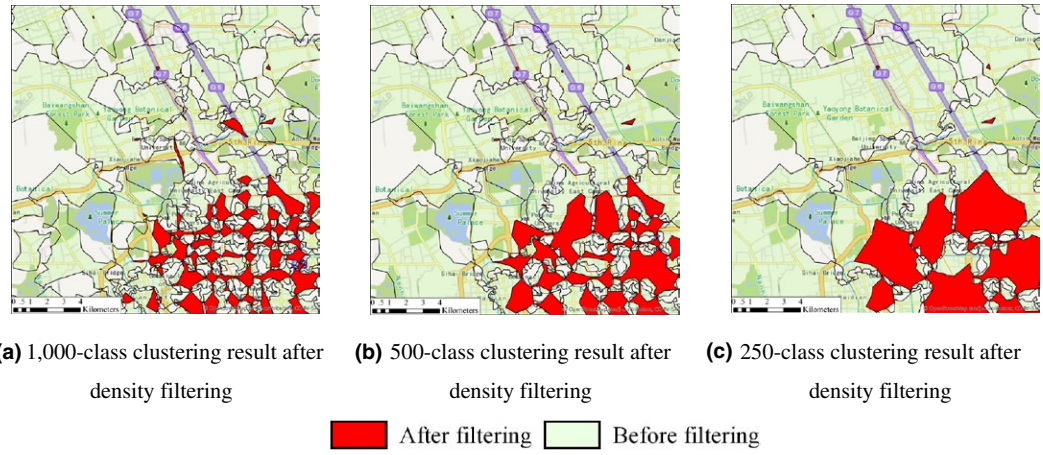
FIGURE 5 Clustering results by HiSpatialCluster and DBSCAN on the Flickr dataset.

4.5 | Evaluation of the filter method based on density-connect

Significantly different numbers of clusters were chosen for clustering taxi trajectory points for 12:00–12:05 a.m, and all results were filtered by density-connect. A total of 250 and 500 classes were chosen (Figures 6b and c) to compare with the former 1,000 classes (Figure 6a). The results demonstrate that the density-connect filter, avoiding the excessive internal heterogeneity in clusters, retains the dense areas. Therefore, it is suitable to use the density-connect filter for generating homogeneous analysis units.

4.6 | Elapsed time by different approaches

Seven trajectory point sets with different sizes were chosen to test the clustering efficiency by HiSpatialCluster, DBSCAN, and K-means. The test platform's software and hardware configurations are listed in Table 4. Besides



(a) 1,000-class clustering result after density filtering **(b)** 500-class clustering result after density filtering **(c)** 250-class clustering result after density filtering

FIGURE 6 Clustering results after density filtering

TABLE 4 Test platform

Platform	Content
Hardware	Intel Core i7-4770 32 GB memory 3 TB hard disk NVIDIA GTX 1080Ti*1
Software	Windows 10 64 bit ArcGIS Pro 1.4.1 with Python 3.5.3 64 bit CUDA 8.0

TABLE 5 DBSCAN-based clustering time on different point sets

Number of points			10 k	20 k	50 k	100 k	200 k	500 k	1 m
Time (s)	ε = 30	*MinPts* = 5	0.11	0.26	1.15	4.58	17.50	118.70	443.99
		MinPts = 10	0.09	0.24	1.09	4.41	18.20	123.42	451.52
		MinPts = 20	0.10	0.22	1.08	4.20	17.33	114.74	453.01
		MinPts = 50	0.09	0.22	1.10	4.38	17.42	113.86	453.55
		Average	0.10	0.24	1.11	4.39	17.61	117.68	450.52
	ε = 100	*MinPts* = 50	0.12	0.37	2.38	10.91	50.39	336.68	1,389.62
		MinPts = 100	0.12	0.35	2.31	11.11	51.52	332.19	1,396.38
		MinPts = 200	0.12	0.35	2.14	10.51	52.00	325.32	1,462.83
		MinPts = 500	0.13	0.36	2.25	10.36	52.78	327.13	1,427.11
		Average	0.12	0.36	2.27	10.72	51.67	330.33	1,418.99

our Python-based HiSpatialCluster, the *Grouping Analysis Tool* in ArcGIS was selected for implementation of K-means, and the DBSCAN-based clustering tool was developed through PostGIS/PostgreSQL. A UDF (user-defined function) named "*ST_ClusterDBSCAN*" was implemented in PostGIS/PostgreSQL, which enables DBSCAN clustering on geotagged data (Table 5). Then, the three approaches' clustering time for different point sets was compared. The number of test points varies from 10,000 to 1 million. When the number of test points exceeded

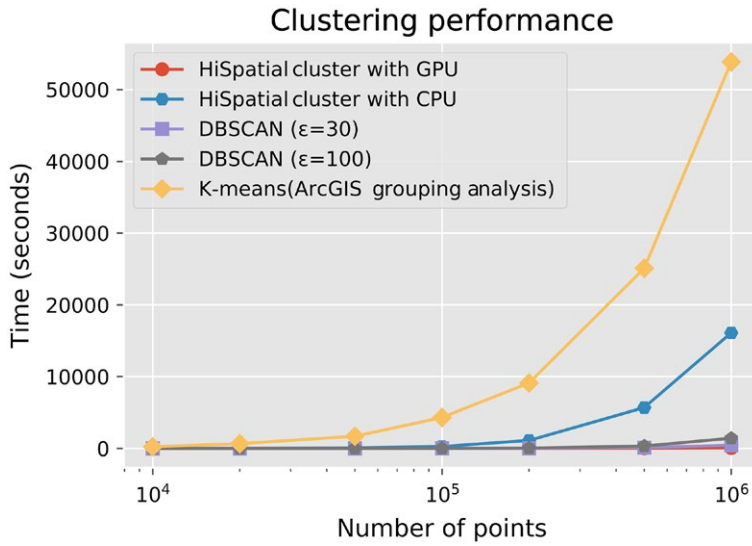


FIGURE 7 Clustering performance by different approaches

TABLE 6 Clustering time by different approaches

Number of points	10 k	20 k	50 k	100 k	200 k	500 k	1 m
K-means (ArcGIS Grouping Analysis)	3 min 51 s	11 min 3 s	28 min 16 s	1 hr 11 min 49 s	2 hr 31 min 54 s	6 hr 58 min 25 s	14 hr 57 min 36 s
DBSCAN (ε=30)	0.10 s	0.24 s	1.11 s	4.39 s	17.61 s	1 min 58 s	7 min 30 s
DBSCAN (ε=100)	0.12 s	0.36 s	2.27 s	10.72 s	51.67 s	5 min 30 s	23 min 39 s
HiSpatialCluster with GPU	1.32 s	2.02 s	3.22 s	5.78 s	10.34 s	28.33 s	62.33 s
HiSpatialCluster with CPU	7.42 s	27.6 s	1 min 27 s	4 min 27 s	18 min 32 s	1 hr 34 min 49 s	4 hr 28 min 34 s

1 million, the efficiency of K-means and DBSCAN started to be very low. Thus, only HiSpatialCluster was tested on very large (billions of) point sets.

The testing results reveal that, as shown in Figure 7 and Table 6, HiSpatialCluster has a short elapsed time throughout with GPU acceleration, and performs excellently on a small number of points with CPU parallel acceleration. The time complexity of HiSpatialCluster and DBSCAN is $O(n^2)$, rather than the $O(nmk)$ of K-means, where n is the number of points, m is the iteration time, and k is the number of clusters. As shown in Figure 8 and Table 7, DBSCAN performs best on smaller point sets due to the GIST spatial index in PostGIS; meanwhile the efficiency of HiSpatialCluster with GPU is beginning to exceed DBSCAN's with increasing numbers of points. With the support of the powerful computational capability provided by GPU computing, HiSpatialCluster with GPU has speed-up ratios near 1,000 for massive trajectory point sets. In addition, HiSpatialCluster with GPU works well on very large point sets. From Figure 9 and Table 8, it is observed that HiSpatialCluster could finish clustering 2.85 billion points in about 3 hr.

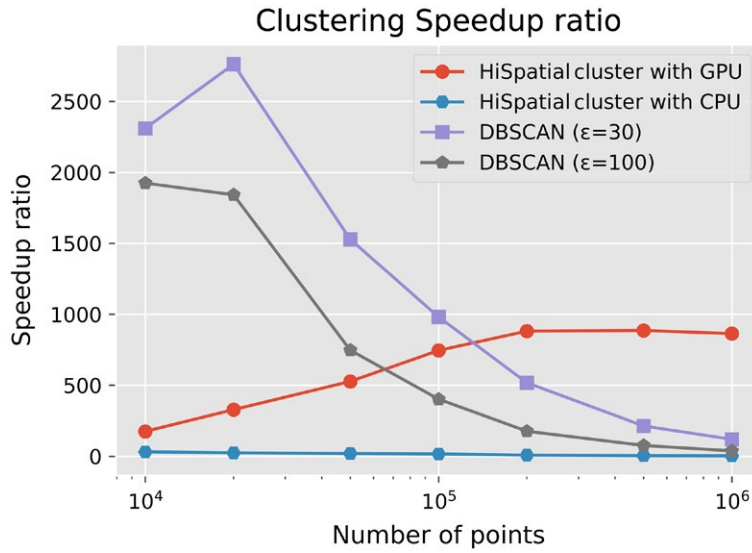


FIGURE 8 Clustering speedup ratios over K-means

TABLE 7 Speedup ratios over K-means

Number of points	10 k	20 k	50 k	100 k	200 k	500 k	1 m
HiSpatialCluster with GPU	175.0	328.2	526.7	745.5	881.4	886.2	979.6
HiSpatialCluster with CPU	31.1	24.0	19.5	16.1	8.2	4.4	3.8
DBSCAN (ε=30)	2310.0	2762.5	1527.9	981.5	517.5	213.3	119.5
DBSCAN (ε=100)	1,925.0	1,841.7	747.1	402.0	176.4	76.0	38.0

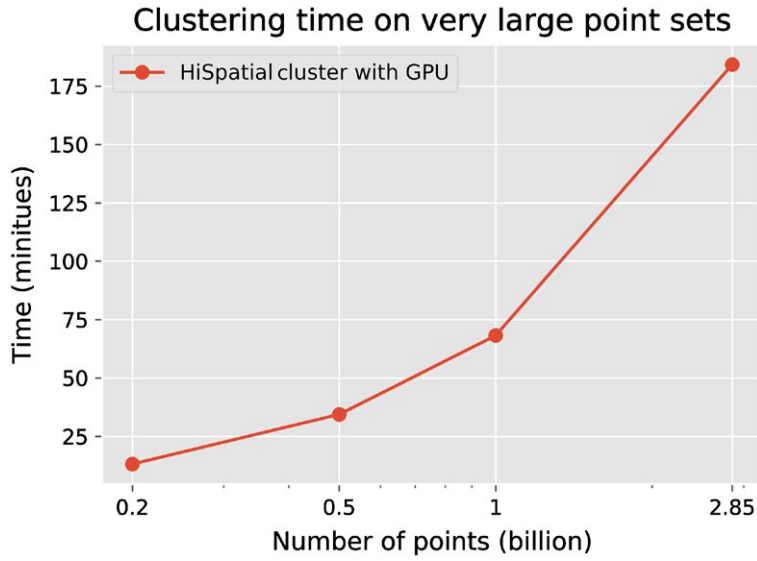


FIGURE 9 Clustering time on very large point sets by HiSpatialCluster with GPU

TABLE 8 Clustering time on very large point sets by HiSpatialCluster with GPU

Number of points (billion)	0.2	0.5	1	2.85
Time (min)	13.15	34.42	68.21	184.2

5 | CONCLUSIONS AND FUTURE WORK

In order to achieve clustering of big geo-data more effectively, a novel high-performance and adaptive spatial clustering approach was proposed. First, the proposed method combines CFSFDP's idea of finding cluster centers and DBSCAN's idea of density-connect filtering for classification. Second, this method is improved by GPU computing. Clustering experiments were performed on massive taxi trajectory points and Flickr geotagged photos. The results illustrate that it achieves discovering arbitrary clusters with irregular spatial shapes and is an adaptive density-based clustering method for datasets with uneven spatial density distributions. Furthermore, it enables clustering of very large spatial datasets (with billions of points) due to high-performance computational capabilities. In the era of big data, the density distribution of spatial point sets is extremely uneven and the amount of geo-data is huge. Therefore, the proposed method is especially suitable for clustering massive spatial point sets in the big geo-data environment.

Then, we implemented a high-performance spatial clustering tool named HiSpatialCluster; the tool's source codes and other resources have been released on Github. Since the research work on big geo-data is increasing rapidly, HiSpatialCluster, as a public spatial clustering tool, is expected to play a more important role in big geo-data mining. As far as we know, except for HiSpatialCluster there are no publicly available software tools that can support spatial clustering of very large datasets with billions of geotagged points.

Improvements in HiSpatialCluster will be made in the future, as follows:

(1) Currently, HiSpatialCluster is only able to cluster with linear distance. Geodesic distance and Manhattan distance will be added to support clustering in different distance metrics.

(2) Customized distance and distance correction by attributes will be supported by HiSpatialCluster to evolve into a more general spatial clustering tool.

(3) The current implementation of HiSpatialCluster is based on a single GPU. Multi-GPU implementation will be provided for larger spatial datasets in the future.

ACKNOWLEDGMENTS

We appreciate the detailed comments from the Editor and the five anonymous reviewers, which have helped us greatly improve the quality of the paper.

ORCID

Yiran Chen http://orcid.org/0000-0001-6626-5413

REFERENCES

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. *ACM SIGMOD Record*, *28*(2), 49–60.

Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, *60*(1), 208–221.

Deng, M., Liu, Q., Cheng, T., & Shi, Y. (2011). An adaptive spatial clustering algorithm based on Delaunay triangulation. *Computers, Environment & Urban Systems*, *35*(4), 320–332.

Ertöz, L., Steinbach, M., & Kumar, V. (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the 2003 SIAM International Conference on Data Mining*. San Francisco, CA: SIAM.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland, OR: ACM.

Estivill-Castro, V., & Lee, I. (2002a). Argument free clustering for large spatial point-data sets via boundary extraction from Delaunay diagram. *Computers, Environment & Urban Systems*, *26*(4), 315–334.
Estivill-Castro, V., & Lee, I. (2002b). Multi-level clustering and its visualization for exploratory spatial analysis. *GeoInformatica*, *6*(2), 123–152.
Guha, S., Rastogi, R., & Shim, K. (1998). CURE: An efficient clustering algorithm for large databases. *ACM SIGMOD Record*, *27*(2), 73–84.
Guo, D., Zhu, X., Jin, H., Gao, P., & Andris, C. (2012). Discovering spatial patterns in origin–destination mobility data. *Transactions in GIS*, *16*(3), 411–429.
Ji, R., Xie, X., Yao, H., & Ma, W.-Y. (2009). Mining city landmarks from blogs by graph modeling. In *Proceedings of the 17th ACM International Conference on Multimedia*. Beijing, China: ACM.
Kisilevich, S., Mansmann, F., & Keim, D. (2010). P-DBSCAN: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos. In *Proceedings of the First International Conference and Exhibition on Computing for Geospatial Research and Application*. Washington, DC.
Lee, I., Cai, G., & Lee, K. (2013). Mining points-of-interest association rules from geo-tagged photos. In *Proceedings of the 46th Hawaii International Conference on System Sciences*. Maui, HI: IEEE.
Lee, I., Cai, G., & Lee, K. (2014). Exploration of geo-tagged photos through data mining approaches. *Expert Systems with Applications*, *41*(2), 397–405.
Lee, J.-G., Han, J., & Whang, K.-Y. (2007). Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. Beijing, China: ACM.
Lin, C.-R., & Chen, M.-S. (2005). Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. *IEEE Transactions on Knowledge & Data Engineering*, *17*(2), 145–159.
MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, CA.
Miller, H. J., & Han, J. (2009). *Geographic data mining and knowledge discovery*. New York, NY: CRC Press.
Ng, R. T., & Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, *14*(5), 1003–1016.
Pei, T., Gao, J., Ma, T., & Zhou, C. (2012). Multi-scale decomposition of point process data. *GeoInformatica*, *16*(4), 625–652.
Pei, T., Zhu, A. X., Zhou, C., Li, B., & Qin, C. (2006). A new approach to the nearest-neighbour method to discover cluster features in overlaid spatial point processes. *International Journal of Geographical Information Science*, *20*(2), 153–168.
Pei, T., Zhu, A. X., Zhou, C., Li, B., & Qin, C. (2009). Detecting feature from spatial point processes using collective nearest neighbor. *Computers, Environment & Urban Systems*, *33*(6), 435–447.
Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, *344*(6191), 1492–1496.
Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining & Knowledge Discovery*, *2*(2), 169–194.
Sheikholeslami, G., Chatterjee, S., & Zhang, A. (1998). WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the 24th International Conference on Very Large Data Bases*. New York, NY: VLDB.
Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to data mining*. Boston, MA: Pearson Addison Wesley.
Vu, H. Q., Li, G., Law, R., & Ye, B. H. (2015). Exploring the travel behaviors of inbound tourists to Hong Kong using geo-tagged photos. *Tourism Management*, *46*, 222–232.
Wan, Y., Pei, T., Zhou, C., Jiang, Y., Qu, C., & Qiao, Y. (2012). ACOMCD: A multiple cluster detection algorithm based on the spatial scan statistic and ant colony optimization. *Computational Statistics & Data Analysis*, *56*(2), 283–296.
Wang, W., Yang, J., & Muntz, R. R. (1997). STING: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*. Athens, Greece: VLDB.
Yang, Y., Gong, Z., & U, L. H. (2011). Identifying points of interest by self-tuning clustering. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Beijing, China: ACM.
Yuan, J., Zheng, Y., & Xie, X. (2012). Discovering regions of different functions in a city using human mobility and POIs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Beijing, China: ACM.
Zhou, X., Xu, C., & Kimmons, B. (2015). Detecting tourism destinations using scalable geospatial analysis based on cloud computing platform. *Computers, Environment & Urban Systems*, *54*, 144–153.

How to cite this article: Chen Y, Huang Z, Pei T, Liu Y. HiSpatialCluster: A novel high-performance software tool for clustering massive spatial points. *Transactions in GIS*. 2018;22:1275–1298. https://doi.org/10.1111/tgis.12463

APPENDIX I: SOFTWARE AND DATA AVAILABILITY

Software name: HiSpatialCluster

Hardware requirements:

Any platform that supports Python. Video card with support of CUDA 7.5 or later is required for GPU enhancement.

Software requirements:

Python 2.6 or later (including Python 3), 32 bit for CPU parallel only and 64 bit with GPU enhancement.

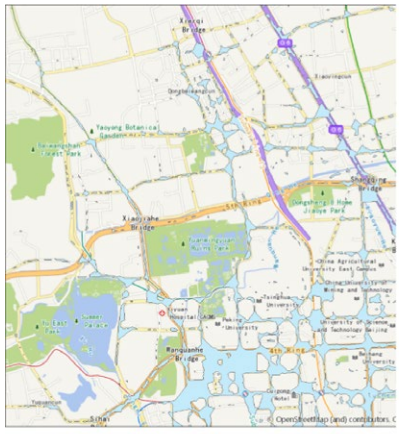
For user interface, ArcGIS Desktop 10.1 or later is required for CPU parellel only,

or ArcGIS Pro with GPU enhancement.

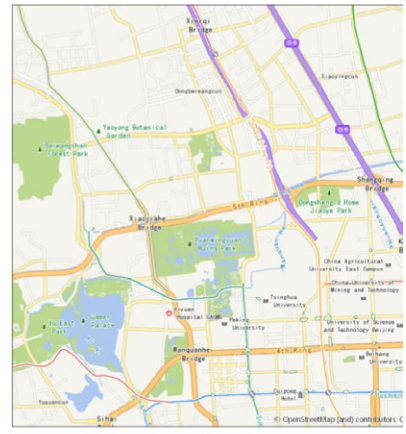
Program languages: Python

Availability: https://github.com/lopp2005/HiSpatialCluster

APPENDIX II: SUPPLEMENTARY CLUSTERING RESULTS BY DBSCAN ON THE TAXI TRAJECTORY DATASET (12:00-12:05AM)

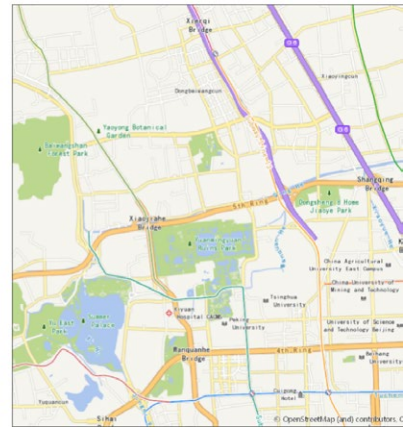


(1) eps =30 , minpts=2

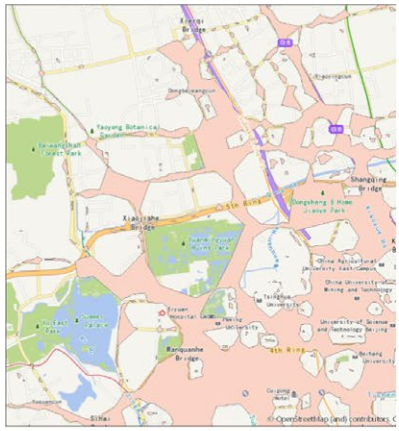


(2) eps =30 , minpts=100

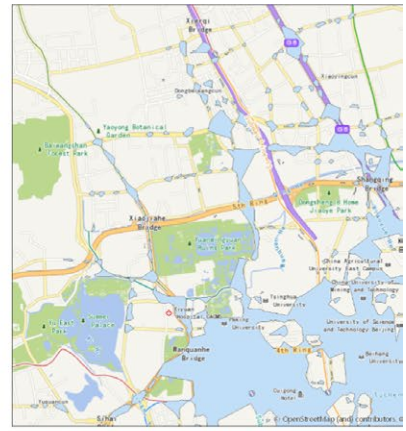
(3) eps = 100, minpts=20



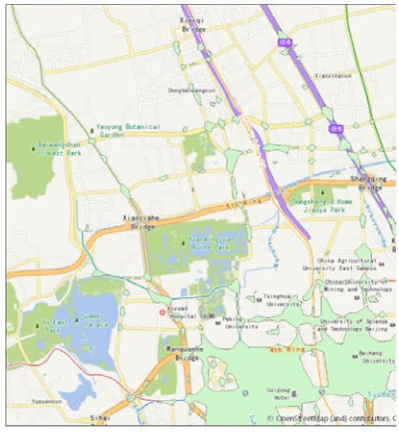
(4) eps = 100, minpts=1000



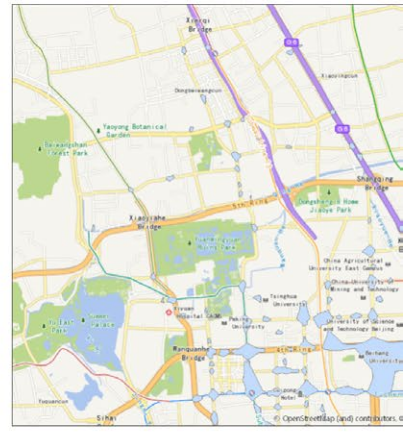
(5) eps =60 , minpts=10



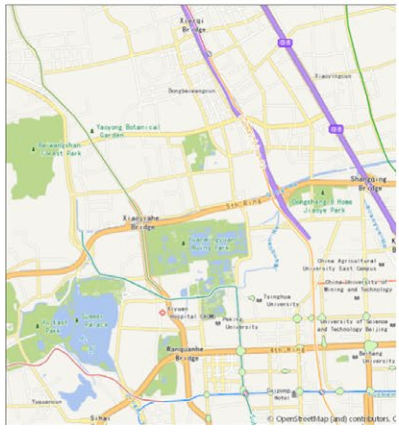
(6) eps =60 , minpts=25



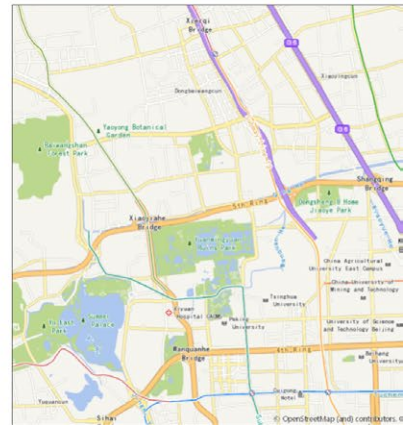
(7) eps =60 , minpts=50



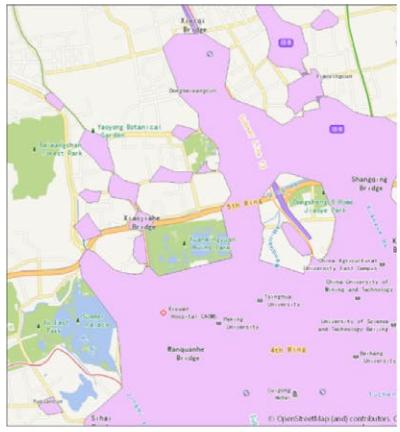
(8) eps =60 , minpts=100



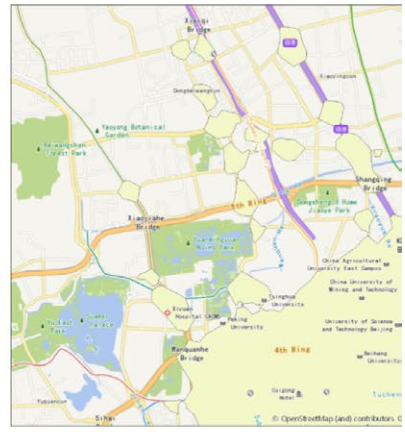
(9) eps =60 , minpts=250



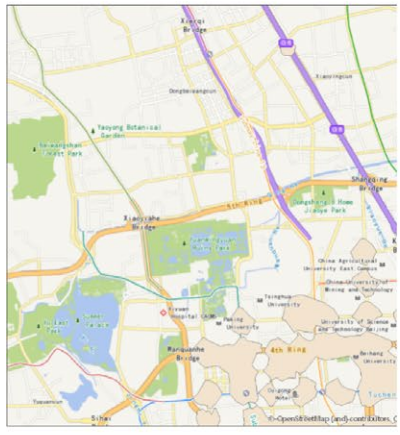
(10) eps =60 , minpts=500



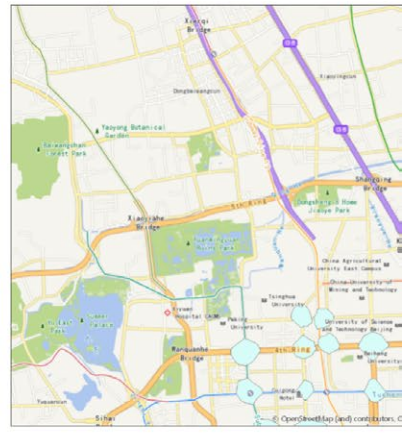
(11) eps = 200, minpts=100



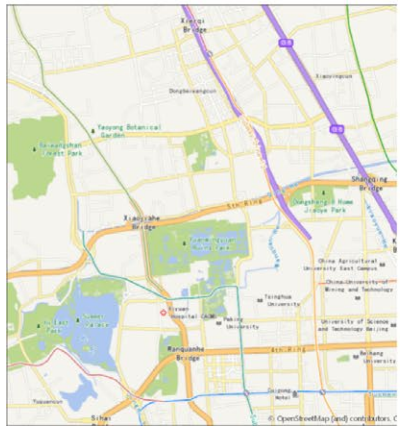
(12) eps =200 , minpts=250



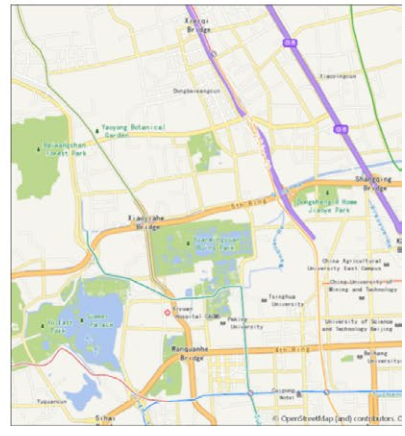
(13) eps = 200, minpts=500



(14) eps =200 , minpts=1000



(15) eps = 200, minpts=2500



(16) eps =200 , minpts=5000