Information Sciences 578 (2021) 37–63

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

On a clustering-based mining approach with labeled semantics for significant place discovery

Xinzheng Niu [a,*], Shimin Wang [a], Chase Q. Wu [c], Yuran Li [b], Peng Wu [a], Jiahui Zhu [a]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China
[b] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China
[c] Dept of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA

ARTICLE INFO

Article history:
Received 27 November 2020
Received in revised form 11 July 2021
Accepted 12 July 2021
Available online 15 July 2021

Keywords:
Spatio-temporal trajectory clustering
Semantic trajectory
Feature selection
Significant place discovery

ABSTRACT

With the rapid increase in GPS data collection through pervasive use of mobile devices, it has become an important problem to discover significant places of moving objects from complex spatial and temporal trajectories. This problem is challenging mainly because such trajectory data suffer from several issues including incompleteness, low quality, high redundancy, and oftentimes trajectory points do not follow Gaussian distribution. We propose a clustering-based method with temporal and spatial semantics, referred to as Stops and Moves of Trajectories using Attribute Selection (SMoTAS), whose technical advantages are multifold. Firstly, it improves data availability by using a self-adaptive algorithm to correct the deviation in traditional speed-based methods. Secondly, it improves place mining accuracy by filtering multi-label clustering results when there is a lack of detailed geographic data. Thirdly, it employs feature selection to exploit the core attributes of clustering and simplify the clustering results with Grubbs criterion. Experimental results on real-life datasets show that SMoTAS not only achieves substantial improvement of accuracy over existing methods in discovering significant places, but also exhibits superior adaptability to different trajectories and application scenarios.

1. Introduction

The pervasive use of portable and mobile devices with positioning capability has made the collection of GPS data far more convenient than ever before. As such mobile data continues to accumulate from massive users, it has become a practically very important problem to discover and exploit significant places. A significant place represents a geographical location where a moving object [14] stays for a relatively long time period in its trajectory, and usually carries valuable information in a wide range of applications. For example, a marketing team may make an appropriate advertising recommendation based on the user's favorite dining place; cellular service providers may allocate more communication resources at locations where people often gather; and zoologists may identify and protect animal habitats by analyzing their migration paths. Such significant places also facilitate the provisioning of personalized services [28]. One typical example is the system service provided by iPhone that sends a user-specific alert in the Calendar, Maps, and Photos Apps according to the significant locations that have been identified based on the trajectory data collected for the user.

* Corresponding author.
E-mail addresses: xinzhengniu@uestc.edu.cn (X. Niu), wang_shimin@std.uestc.edu.cn (S. Wang), chase.wu@njit.edu (C.Q. Wu).

https://doi.org/10.1016/j.ins.2021.07.050
0020-0255/

Significant place mining or discovery from trajectory data has been well investigated as a subject of clustering [4,26,42]. According to the definition of spatio-temporal data mining [11,13], significant place mining can be defined as a process of extracting implicit and useful information and knowledge from spatio-temporal data [34,15,12,22] with high-dimensional, and high-noise characteristics [19,2].

There exist a number of efforts in this field, but no algorithm for significant place mining is universally applicable to different environments. The existing methods for location mining based on trajectory model [31,36] can be divided into the following categories: partition-based clustering, hierarchical clustering, density-based clustering, and grid-based clustering. Among them, the most frequently used is the density-based approach [44,16,7,25], which is able to find clusters of any shape, but with a high computational cost with respect to the object density. The DBSCAN algorithm [16], one of the most common density-based algorithms, is robust to outliers, but suffers from high sensitivity to the choice of parameters (i.e., distance threshold ϵ and *minPts* neighbors) because a small difference in the parameters may lead to very different results. To address this issue, the OPTICS algorithm [29] generates an object processing sequence representing the density-based clustering structure of the data object. Later, inspired by DBSCAN, Jae-Gil Lee et al. [24] designed a trajectory clustering algorithm TRACLUS, which includes two phases: partition and group. It first partitions a trajectory into a set of line segments based on characteristic points, at which the object makes a sharp turn, and then groups similar line segments into a set of clusters. In the group phase, the traditional density-based point clustering algorithm DBSCAN is extended for segment-based clustering, which only considers the spatial attributes in distance calculation.

However, the methods that have evolved from traditional clustering algorithms can neither handle data deletion nor effectively use background information, and are incompetent in finding data relations and rules in a larger spatial space, hence ultimately affecting the accuracy of mining results. These problems are tackled in [18,3], where a semantic trajectory model [37] is constructed. Such a model, combined with background information, requires a large amount of context information as support, hence making it difficult for the algorithm to adapt quickly to different application scenarios.

In this paper, we propose a clustering-based method, referred to as Stops and Moves of Trajectories using Attribute Selection (SMoTAS), to automatically determine the underlying important features from the spatio-temporal semantics and context information in different application scenarios and discover significant places in trajectories based on such features. The proposed method consists of the following technical components:

- Construct a clustering model through the discretization of trajectory points and design a trajectory clustering algorithm for moving objects based on the trajectory velocity and direction information matching the road network space;
- Design an adaptive algorithm to refine the initial clustering results and remove abnormal clusters;
- Screen the initial clustering results using a variety of clustering attributes, select a subset of the most suitable attributes through feature selection for the given application scenario, and further improve the accuracy of results by applying Grubbs criterion [32] to eliminate bad clusters.

SMoTAS focuses on two aspects of object movement: velocity and direction, which are combined for the discovery of significant places. It corrects the anomalies caused by noisy data points, reduces the deviation between the initial and expected clustering results, and applies a secondary screening for higher accuracy and adaptability. The proposed method is evaluated with real-life trajectory data and the experimental results show that SMoTAS achieves substantial improvement of accuracy over existing methods in mining significant places, and also exhibits superior adaptability to different trajectories and application scenarios.

The rest of the paper is organized as follows. In Section 2, we conduct a survey of related work on significant location mining. In Section 3, we construct mathematical models for semantic trajectories and formulate the problem. In Section 4, we detail the design of SMoTAS. In Section 5, we present and analyze the experimental results for performance evaluation. In Section 6, we conclude our work and sketch a plan for future research.

2. Related work

Significant location/place mining or discovery in trajectories [31,36] with semantic information has been extensively studied in the literature. Semantic information refers to certain environmental characteristics that determine user behaviors, including computing scenario, user context, physical location, time period, and social surrounding, which facilitate the transformation from "computer centric" to "human centered". The acquisition, representation, and use of semantics provide a base for the mining of significant locations within a single trajectory, and enable the extraction, clustering, analysis, and visualization of significant locations in the trajectory.

We shall start with a brief introduction to several key concepts such as raw and semantic trajectories, followed by a survey of existing efforts on significant location mining in a single trajectory.

2.1. Raw trajectory and semantic trajectory

The concept of trajectory is derived from capturing the moving path of an object in a two-dimensional space over a period of time, and the direct tracking result of each moving object is its trajectory. In many practical applications, the trajectory

information needs to be consistent with the application scenario where the trajectory is generated. For example, for a tourist within a metropolitan area, it would be more useful if the trajectory is annotated with detailed information on each building and street to produce a semantic trajectory [36]. For illustration, we plot in Fig. 1a and b an example of a raw or original trajectory and a corresponding semantic trajectory, respectively, for a visual comparison.

2.2. Significant location mining in a single trajectory

Zhou et al. [44] designed a density clustering method similar to DBSCAN [16] to classify the primary locations of a user based on GPS data, without considering the concept of time. To address this problem, Birant et al. [7] proposed ST-DBSCAN algorithm, which is extended in the core object, noise, and adjacent clusters, and considered both time and space. However, these two aspects are separated in the evaluation function, hence undermining the space–time integrity. Lee et al. [25] proposed CB-DBSCAN (Clustering-Based DBSCAN), which considers both speed and distance, to discover significant places from GPS data. However, a small change to the minimum number of points and the given distance between adjacent points may cause some data to be considered as noise or assigned to a wrong cluster, hence resulting in a significant change in the clustering results.

Considering the above issue, researchers began to focus on the study of semantic trajectory. Particularly, Alvares et al. [1] proposed SMoT (Stops and Moves of Trajectories) algorithm to mine moving and stop areas from simple trajectories of spatial points, and designed an evaluation method that uses this semantic model to mine simple trajectory data. This algorithm is inspirational to many later models, but has limitations in the selection of trajectory datasets and spaces. Spaccapietra et al. [38] proposed a new semantic analysis model to interpret trajectory information, which divides a trajectory into moving regions and stop regions. A stop region where the object stayed longer than a threshold is an important semantic attribute of the trajectory. In different applications, stop regions may have different meanings, and the threshold for stay could also vary significantly. For example, in the trajectory of a passenger, a stop region could be a tourist spot, an airport, a hotel, and so on; while in the trajectory of animal migration, a stop region could be a habitat, a water source, a breeding site, and so on. These two methods perform clustering with an assumption that "the data has an intrinsic probability distribution", and attempt to best fit the data objects with a given model. Such a model could be a density function of data objects in the space distribution or some other descriptive functions, which may not be able to effectively cope with data loss and identify the user's unmarked points. Since real-world data does not always follow a well-defined distribution, this model may lack adaptability in practice.

Following the above work, Palma et al. [30] proposed a fast algorithm named CB-SMoT (Clustering-Based Stops and Moves of Trajectories). Compared with traditional algorithms, it solves the missing data problem in the track and introduces geographic semantic information, which facilitates the matching of real significant locations with places on the map. However, this method has several technical challenges, including the difficulty in determining an effective clustering radius, the high incompatibility in different application scenarios, and the risk of generating semantic errors.

In recent years, the research on semantic trajectory and significant location has made a rapid progress. Rocha et al. [33] proposed a direction-based DB-SMoT algorithm to explore significant locations of large-scale movement. Also, Le et al. [41] proposed a semantic-enhanced trajectory mining algorithm based on the *K*-Means algorithm, which does not require any knowledge about the domain, but only two parameters as input: the minimum number of points (*minPts*) and a given distance between adjacent points (*Eps*). In later years, the applications in this field started targeting specific objects and scenarios. For example, Bremond et al. [8] proposed a method to analyze the individual behavior using trajectory and video

Fig. 1. An example trajectory: (a) raw or original, (b) semantic.

information; Carboni and Bogorny et al. [10] made a breakthrough in the use of semantic trajectory to study car moving behaviors and significant locations; Lynen et al. [27] designed a placeless algorithm using continuous image stream matching technique. Recently, Bermingham and Lee designed a probability-based algorithm POSMIT (Probability of Stops and Moves in Trajectories) [6], which allows the user to better control the quality of stops and moves classification by filtering out entries with classification labels deemed too ambiguous for the application. Although having the potential to derive semantic trajectories and significant locations, these algorithms may suffer from low quality or efficiency. In addition, Hwang et al. [21] proposed an algorithm named Spatiotemporal Clustering-based Stops and Moves of Trajectories (STC-SMoT), which considers the *EPS* value and minimum stop duration to determine clusters with parameter sets derived through sensitivity analysis. STC-SMoT takes three main steps to impute time gaps, split data into base segments, and estimate states over a base segment. Although time gap imputation can solve the signal loss problem, it may incur a prohibitively long time to process if there are large signal losses in the trajectory. More recently, a semi-automatic approach named Particle Swarm Optimization based Optimal Parameter for Stop Points Detection (PSO-OPSPD) was proposed by Bandyopadhyay et al. [5], which employs particle swarm optimization, DBSCAN, and internal validity index *S_Dbw* to determine appropriate parameter values for clustering. The user input parameters of PSO-OPSPD are filtered by the internal validity index (i.e., *S_Dbw* and *Silhouette*) in advance, so it can produce a higher-quality clustering result. However, this approach is relatively time-consuming and highly dependent on the initial range of the parameter.

As discussed above, a clustering algorithm analyzes a set of data objects to identify places that satisfy similar conditions in the same class. Note that similarity conditions are typically distance-based and may not be able to determine locations of significance. Also, most of the existing significant place mining algorithms are of high time complexity with limited interpretability. In this paper, we propose a new mining algorithm combining semantic characteristics of moving objects to address these issues.

3. Problem formulation

In this section, we construct mathematical models for semantic trajectories, using stops and moves to integrate geographic information with trajectory sampling points. To facilitate the problem formulation and algorithm design for significant place discovery, we first provide the following definitions.

Definition 1. Raw Trajectory T_R: A raw trajectory T_R is represented by a list of spatio-temporal points sampled at certain time intervals: $\{p_0 = (x_0, y_0, t_0), p_1 = (x_1, y_1, t_1), \cdots, p_i = (x_i, y_i, t_i), \cdots, p_N = (x_N, y_N, t_N)\}$, where (x_i, y_i) denotes the latitude and longitude coordinates at time point $t_i, i = 0, 1, \cdots, N$, and $t_0 < t_1 < t_2 < \cdots < t_N$.

Definition 2. Labeled Trajectory T_L: A labeled trajectory T_L is a raw trajectory with labeled attributes such as speed and direction at each sampling point $p_i = (x_i, y_i, t_i, attr_1, attr_2, \cdots, attr_m)$, where m denotes the number of attributes.

Definition 3. Stop Region *SR*: A stop region *SR* is a four tuple $(R_S, t_i, t_{i+k}, \Delta_S)$, where $R_S = \{p_i, p_{i+1}, \ldots, p_{i+k}\}$ is a subtrajectory based on the labeled trajectory T_L, and $|t_{i+k} - t_i| \geqslant \Delta_S$, where Δ_S is a threshold that denotes the minimum time duration of R_S.

Definition 4. Moving Region *MR*: A moving region *MR* is a three tuple (R_M, t_j, t_{j+n}), where R_M is a maximal subtrajectory consisted of a set of contiguous spatio-temporal sampling points $\{p_j, p_{j+1}, \ldots, p_{j+n}\}$ in the labeled trajectory T_L.

There are four cases about a moving region in the labeled trajectory T_L:

(1) A moving region *MR* is a maximal contiguous subtrajectory of T_L between two temporally consecutive stop regions of T_L.
(2) A moving region *MR* is a maximal contiguous subtrajectory of T_L between the starting point of T_L and the first stop region of T_L.
(3) A moving region *MR* is a maximal contiguous subtrajectory of T_L between the last stop region of T_L and the last point of T_L.
(4) If there is no stop region *SR*, the labeled trajectory T_L itself is a moving region *MR*.

Note that each point in the trajectory is in either a moving region *MR* or a stop region *SR*.

Definition 5. Significant Place (SP): A significant place *SP* is defined as (R_{SP}, Δ_{SP}), where R_{SP} is a topologically closed polygon representing the location and shape of a place, and Δ_{SP} is the minimum amount of time the object must stay at the place. Specifically, R_{SP} contains a series of trajectory points $\left\{p_i, p_{i+1}, \ldots, p_{i+j}\right\}$, and the total time that the object stays in *SP* is $|t_{i+j} - t_i|$, which meets the condition $|t_{i+j} - t_i| \geqslant \Delta_S$.

Note that a significant place *SP* is generated on the basis of stop regions. However, a stop region may or may not be an actual significant place. Hence, to narrow down the search space, it is important to restrict the search within a limited set of stop regions. For instance, in a tourism application, it is not of interest to characterize the stop region of a tourist at a gas station, and hence gas stations are not considered as stop regions. Similarly, in a traffic application, we should rule out streets, traffic lights, viaducts, bridges, crossings, etc. from the set of stop regions. If a stop region SR_i precedes and intersects another one SR_j, SP is generated on the union of them; otherwise, SP is generated on the preceding stop region SR_i. Namely, for $\forall i,j \in [0,N]$, and $i < j$, $SP = \begin{cases} SR_i \cup SR_j, \text{if } SR_i \cap SR_j \neq \varnothing \\ SR_i, \text{if } SR_i \cap SR_j = \varnothing \end{cases}$, where SR_i and SR_j are two different Stop Regions.

In this work, we adopt a concept of *silhouette* [35] to evaluate the quality of significant place discovery, defined as follows:

Definition 6. Silhouette S: *silhouette* is a value based on the tightness and separation of clusters, and shows which objects lie well in the cluster. For each point i, its *silhouette* value $S(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$, where $a(i)$ is the average distance from i to all other points in the cluster it belongs to, and $b(i)$ is the minimum of the average distance from i to all points in the nearest cluster. The total *silhouette* S is calculated as $S = \frac{1}{N}\sum_{i=1}^{N} S(i)$, where N is the total number of points in trajectory T_L.

Obviously, *silhouette* S is in the range from -1 to 1, and the value increases as the trajectory in the cluster becomes denser. Therefore, a higher value of silhouette indicates a better quality of clustering.

Definition 7. Significant Place Discovery (SPD): Given a labeled trajectory T_L, our goal is to discover all possible significant places $\{SP_0, SP_1, \cdots, SP_m\}$ with the maximum *silhouette maxS*.

4. A clustering-based significant location mining algorithm: SMoTAS

The key idea of SMoTAS is as follows: Firstly, we adjust the initial parameter *Eps* based on the data of the entire trajectory. Secondly, we extract the corresponding velocity and moving direction, and obtain stop regions through clustering. Thirdly, we exploit the features to select core attributes for further screening and simplifying the results. To facilitate the explanation of our design, we provide several related theorems and definitions and properties as follows.

Definition 8. Trajectory Segment (TS): TS is a set of contiguous spatio-temporal points p in the labeled trajectory T_L, i.e., $TS = \left\{p_i, p_{i+1}, \cdots, p_j\right\}$, where $i \in [0,N], j \in [0,N], i \leqslant j$, and N is the total number of points in T_L.

Definition 9. Distance between two points $Dist(p_n, p_m)$: $Dist(p_n, p_m) = \sum_{i=n}^{m-1} Dist(p_i, p_{i+1})$, where $n < m$, and $Dist(p_i, p_{i+1})$ denotes the Euclidean distance between two neighbor points p_i and p_{i+1}.

Definition 10. Trajectory Direction Change: The direction change $DC(p_i)$ of labeled trajectory T_L at point p_i is represented by the changing angle from line segment $\overrightarrow{p_{i-1}, p_i}$ to $\overrightarrow{p_i, p_{i+1}}$.

Definition 11. *Eps* Neighborhood: Given a labeled trajectory $T_L = \{p_0, p_1, p_2, \ldots, p_N\}$, the *Eps* neighborhood of $p_i, i \in [0,N]$, is defined as $NEps(p_i) = \{p_k | Dist(p_k, p_i) \leqslant Eps \vee Dist(p_i, p_k) \leqslant Eps, \forall p_k \in T\}$.

Definition 12. Core Point: A point p_i is defined as a core point only if it satisfies both of the following two criteria: (1) $|t_m - t_n| \geqslant minTime$, where n is the last point of $NEps(p_i)$, m is the first one (the neighborhood is ordered by time), and *minTime* is the user-specified shortest duration; (2) $DC(p_i) \geqslant minDirChange$, where *minDirChange* is the user-specified minimum direction change.

SMoTAS consists of the following technical steps, as schematized in Fig. 2.

(1) *Eps* calculation and correction. When the distance distribution of locus points is not regular, this step corrects the original parameters by using bias feedback between the results and the expected ones.
(2) Stop region clustering. This step uses the modified *Eps* parameters to cluster the points in the whole trajectory, producing stop regions.
(3) Core attribute selection. To adapt to a wide variety of scenarios, the clustering results of the first step may contain a large number of attribute labels. This step employs a feature selection algorithm to choose the most suitable core attributes in the current situation.
(4) Final result output. Based on the core attributes from Step (3), this step removes "bad clusters" in the initial clusters with Grubbs to produce significant places.

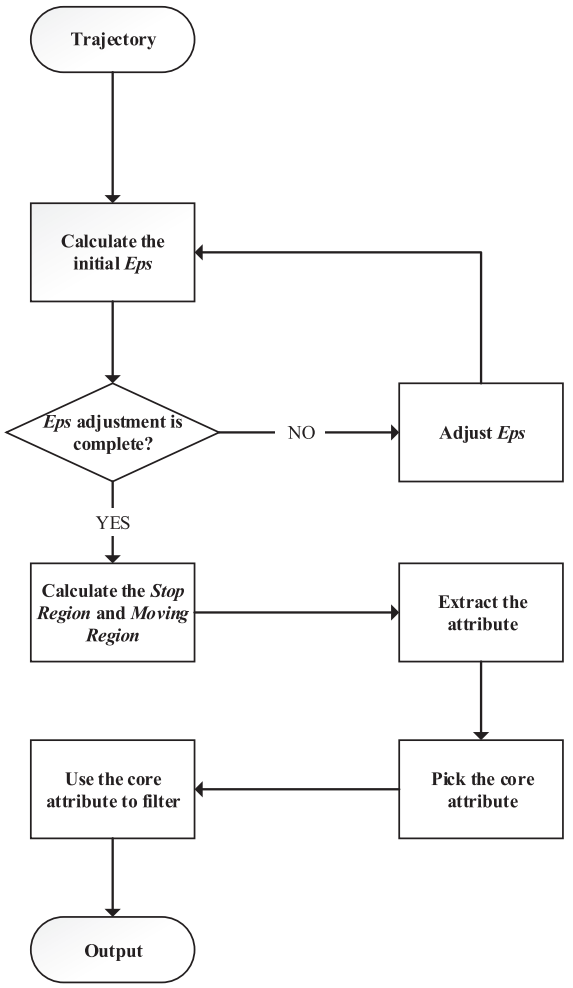

Fig. 2. The flowchart of SMoTAS.

4.1. Parameter Eps calibration

Parameter *Eps* is a positive real number threshold, which represents the maximum distance between point p and its nearest neighbors. This parameter is the key to clustering, but it is necessary to reference the data of the entire trajectory and the actual situation. We use the distance set of adjacent points p_i, p_{i+1} to represent trajectory, and use a quantile function to describe Gaussian curve:

$$F^{-1}(q,\mu,\sigma) = \mu + \sigma\sqrt{2}erf^{-1}(2q-1),$$

$$erf^{-1}(x) = \sum_{k=0}^{\infty}\frac{C_k}{2k+1}\left(\frac{\sqrt{\pi}}{2}x\right)^{2k+1},$$

where μ is the mean, σ is the standard deviation, $C_0 = 1, C_k = \sum_{m=0}^{k-1}\frac{C_m C_{k-1-m}}{(m+1)(2m+1)}$, and parameter $q, q \in [0,1]$, specified by the user according to the scenario, denotes the ratio of the total number of points in the cluster to the total number of points in the trajectory.

However, in practice, the distance distribution does not always follow Gaussian curve. Fig. 3 intuitively reflects the discrepancy between the actual data and the theoretical curve.

This discrepancy is mainly caused by the fact that the trajectory data of a moving object in practical applications has a large number of adjacent points (stationary states such as work, study, or rest) whose moving distances are almost zero and some points in high-speed motions (moving states such as cars and subways).

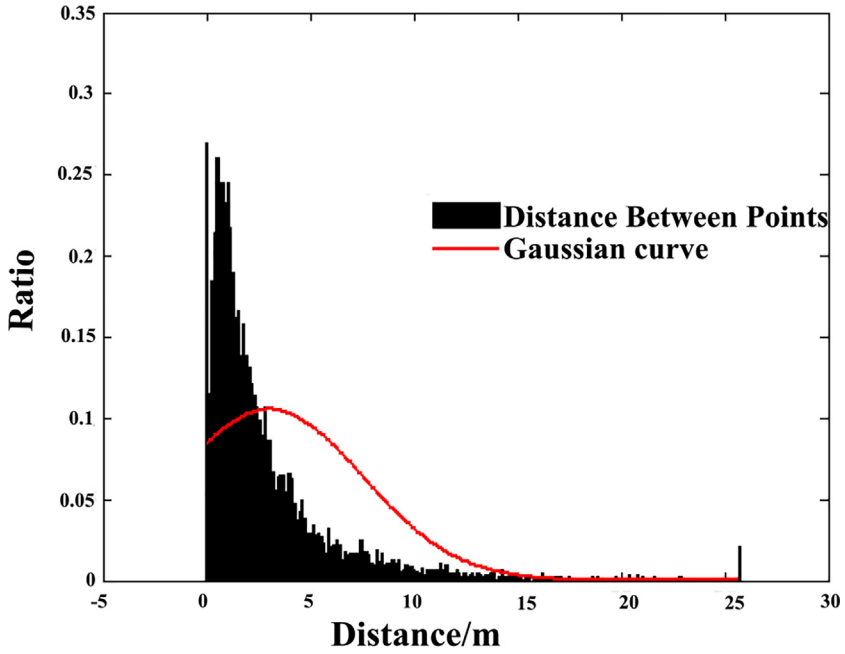

Fig. 3. Distance distribution compared with Gaussian curve.

Algorithm 1 EpsAdjustment

Require T, q, *minDirChange*, *minTime*, *Eps*
Ensure *newEps*
1: Set *newEps* = *Eps*;
2: $prev\_k = 1.0$;
3: **Repeat**
4: $k = calculateK(T, q, minDirChange, minTime, newEPS)$;
5: *change* = $k - prev\_k$;
6: a = adapt(k, q, *newEps*);
7: $newEps = a \cdot newEps$;
8: $prev\_k = k$;
9: **Until** $\left(abs(a-1) \leqslant \sqrt{2}\%\right)$ or $(abs(change) \leqslant e^2\%)$

Algorithm 2 calculateK

Require T, q, *minDirChange*, *minTime*, *EPS*
Ensure k
1: *Clusters* = *StopRegionClustering*(T, q, *minDirChange*, *minTime*, *EPS*);
2: $totalPointNumber = 0$;
3: **for** *stopRegion* in *Clusters*
4: *PointNumber* = the number of points in *stopRegion*;
5: $totalPointNumber = totalNumber + PointNumber$;
6: **end for**
7: n = the number of points in T;
8: $k = totalPointNumber/n$;

In this paper, we adopt an adaptive approach to adjusting clustering radius [45], and use actual results and expected deviations as the feedback to correct parameters. We design the algorithm $EpsAdjustment(E\text{-}ps, q, T)$ in Algorithm 2 to modify the value of *Eps*, where parameter k is the ratio of the total number of points in all stop regions under the current *Eps* computation to the total number of points, and is calculated by the function $calculateK(T, q, minDirChange, minTime, EPS)$, as detailed in Algorithm 2. Specifically, before calculating the parameter k, we first use Algorithm 3 (as detailed in Section 4.2) to identify stop regions based on the changes of speed and direction, and then obtain the total number of points in all stop regions (denoted as *totalPointNumber*). Lastly, we set n to be the total number of points in T, and then use the ratio of *totalPointNumber* to n to calculate the value of k. After obtaining k, we are able to calculate the value of parameter a, which denotes a multiplier of each adjustment made to *Eps*, and is calculated as: $adapt(k, q, newEps) = (1 + q - k)\sqrt{\frac{\mu+\sigma\sqrt{2}erf^{-1}(q)}{newEps}}$.

In Algorithm 1, two thresholds are set to determine the stop condition to obtain the best value of *EPS*. The decreasing value of a reflects the progress of adjustment, and the value of *change* reflects the impact of two adjacent adjustments on the clustering range: a small value means that the current adjustment has no significant change to the actual clustering. Our empirical study shows that setting the values of a and *change* to be $\sqrt{2}\%$ and $e^2\%$ yields good performance.

We use a set of real motion trajectories to illustrate the impact of the adjusted parameter *newEps* on clustering. In Figs. 4 and 5, where the horizontal axis represents the longitude and the vertical axis represents the latitude, we show the difference in clustering one area of the GPS trajectory data before and after *Eps* adjustment. We use a black square border to enclose a stop area, which is marked as α and α' in the upper right corner in Figs. 4 and 5, respectively. Figs. 6 and 7 provide a microscopic examination into the most signficantly different areas of α and α' in Figs. 4 and 5. The points in the stop region are marked in red while the points in the moving region are marked in black.

Figs. 8 and 9 show a comparison between before and after another cluster change in the same trajectory data, and the stop area is circled with a black square border, which is marked as β and β' in the upper left corner respectively. Figs. 10 and 11 provide a microscopic examination into the selected part of β and β'. In these plots, we observe that the modified parameter *Eps* finds more clusters, and better clustering results are obtained after adjusting the parameter.

In the above test, the initial parameter q is set to be 0.36, and *minTime* is set to be 100s. After the adjustment, the number of stops in the trajectory increases from 18 to 25, and the proportion of red-marked points increases from 0.24 to 0.32.

4.2. Stop region clustering based on velocity and direction

The initial clustering process is shown in Algorithm 3. The $linear\_neighbor(p, Eps)$ procedure determines the *Eps* neighborhood of point p. Every time when a core point p is obtained, it is expanded to a stop region. Specifically, it finds all core points in the current stop region, and then adds their *Eps* neighborhoods into the current stop region until it is no longer expandable, i.e., no new point is added.

Algorithm 3 StopRegionClustering

Require: T, q, *minDirChange*, *minTime*, *EPS*
Ensure: S
1: $S = \varnothing$;
2: **for all** unprocessed point p in T
3: $StopPoints = \varnothing$;
4: $Neighbors$=$linear\_neighbor(p, Eps)$;
5: **if** $isCorePoint(Neighbors, minTime, minDirChange)$ is true **then**
6: Place all points in the Neighbors in *StopPoints*;
7: **repeat**
8: **for all** each point p_i in *StopPoints*
9: **if** $isCorePoint(linear\_neighbor(p_i, Eps), minTime, minDirChange)$ is true **then**
10: $Neighbors$=$linear\_neighbor(p_i, Eps)$;
11: Place all points in the Neighbors in *StopPoints*;
12: **Until** the size of *StopPoints* is no longer expanded;
13: Set all point in *StopPoints* as processed;
14: Place all stops in S;

A stop region can be expanded to another stop region according to the core point's *Eps* neighborhood.

Initially, this region is the *Eps* neighborhood of point p. Then, after searching all points in the stop region, if there is a core point, we add the *Eps* neighborhood of this point to the stop region until no new point is added into the cluster.

In this work, the critical threshold of *Eps* is determined according to the user moving characteristics in practice. We calculate the instantaneous velocity of each position in the user trajectory, exclude the continuous noise segment with the speed above the threshold, and generate candidate sites. Finally, the same candidate sites are merged to produce a list of users' significant locations.

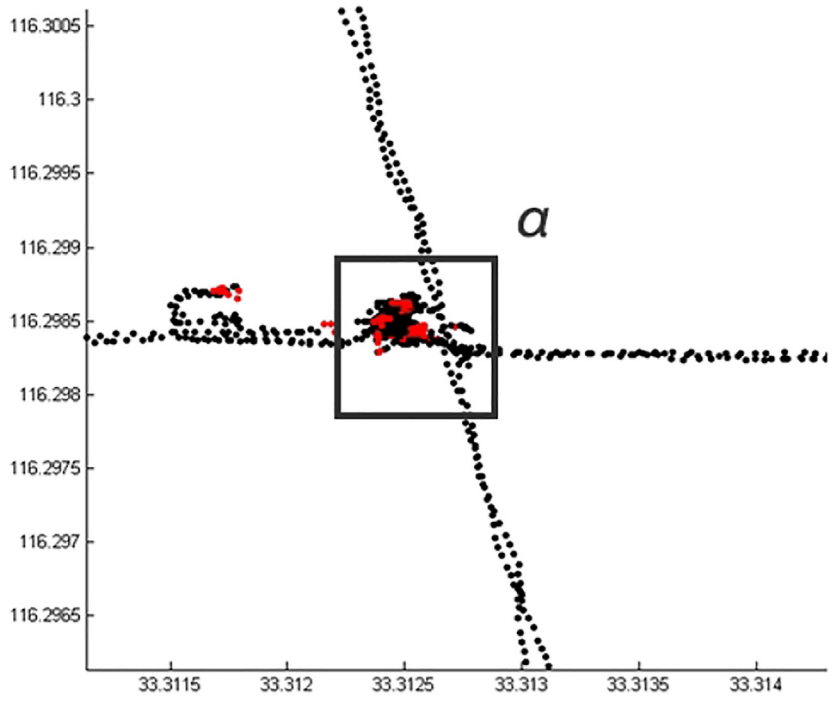

Fig. 4. Stop regions by *Eps*.

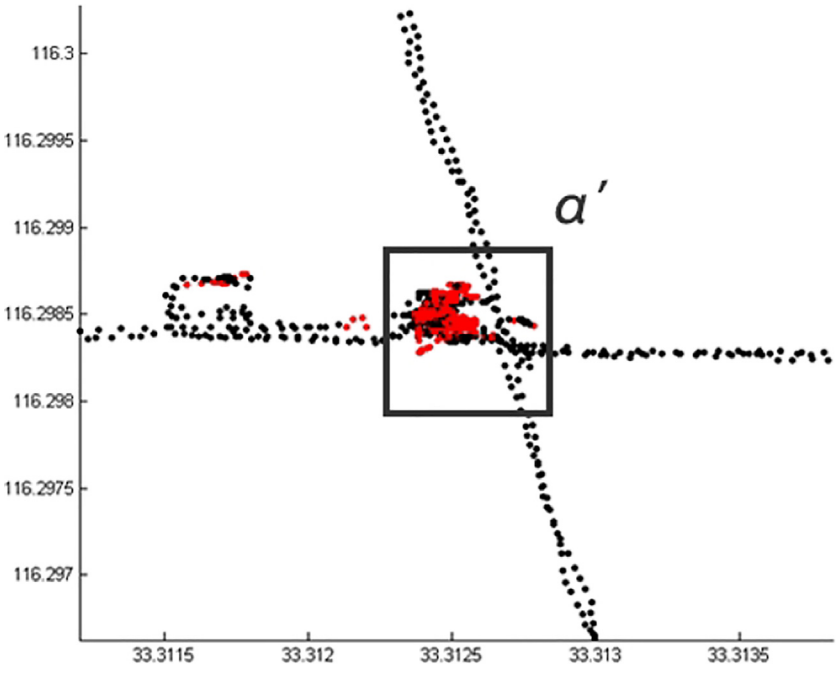

Fig. 5. Stop regions by *newEps* ($q = 0.36$, $minTime = 100s$).

Figs. 12 and 13 illustrate the process of creating a stop region. Firstly, we discover a core point and its *Eps* neighborhood represented by a red circle. Then, we discover two more core points in *Eps* neighborhood, and also add their *Eps* neighborhoods (two green circles) to the stop region to obtain the complete cluster (the outmost blue circle).

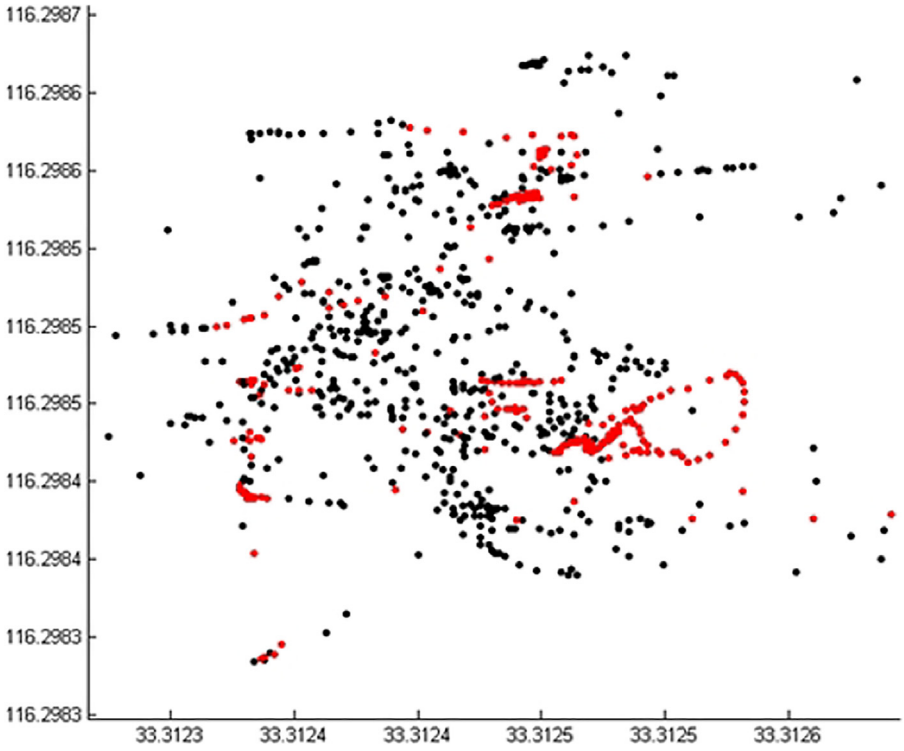

Fig. 6. A zoom-in view of the square area in Fig. 4.

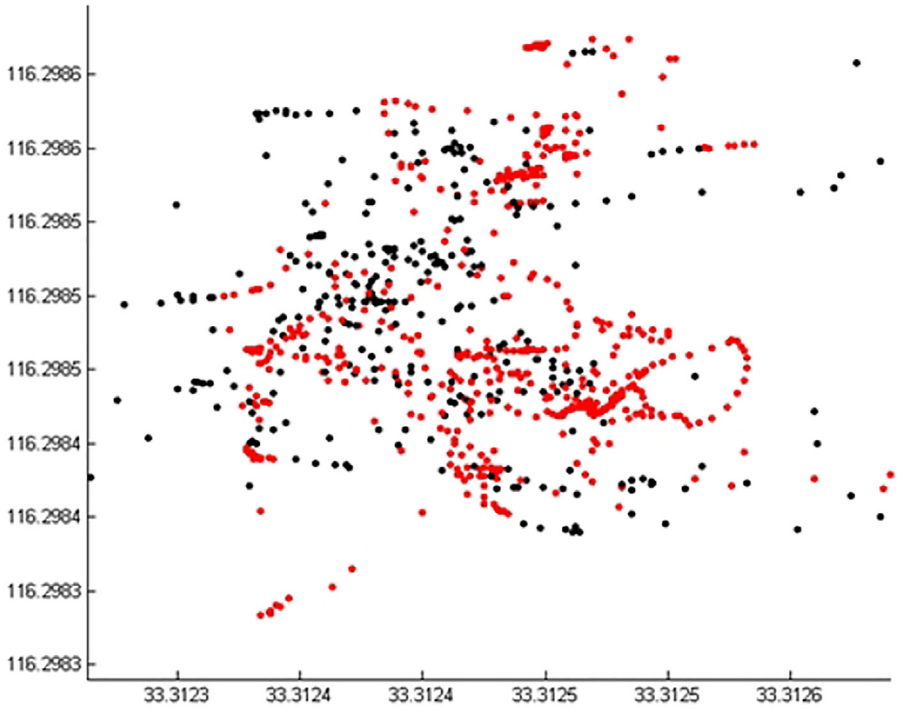

Fig. 7. A zoom-in view of the square area in Fig. 5.

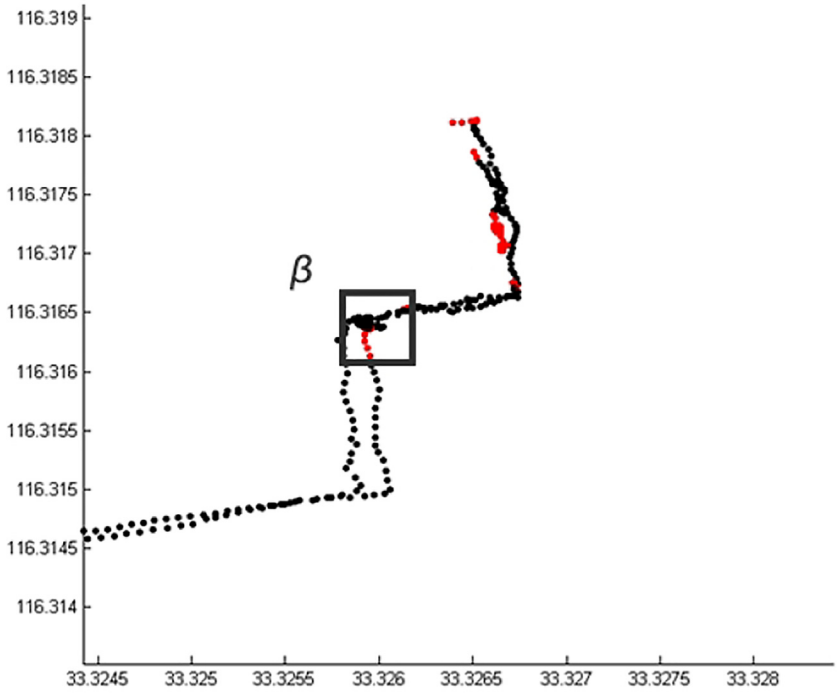

Fig. 8. Stop regions by *Eps*.

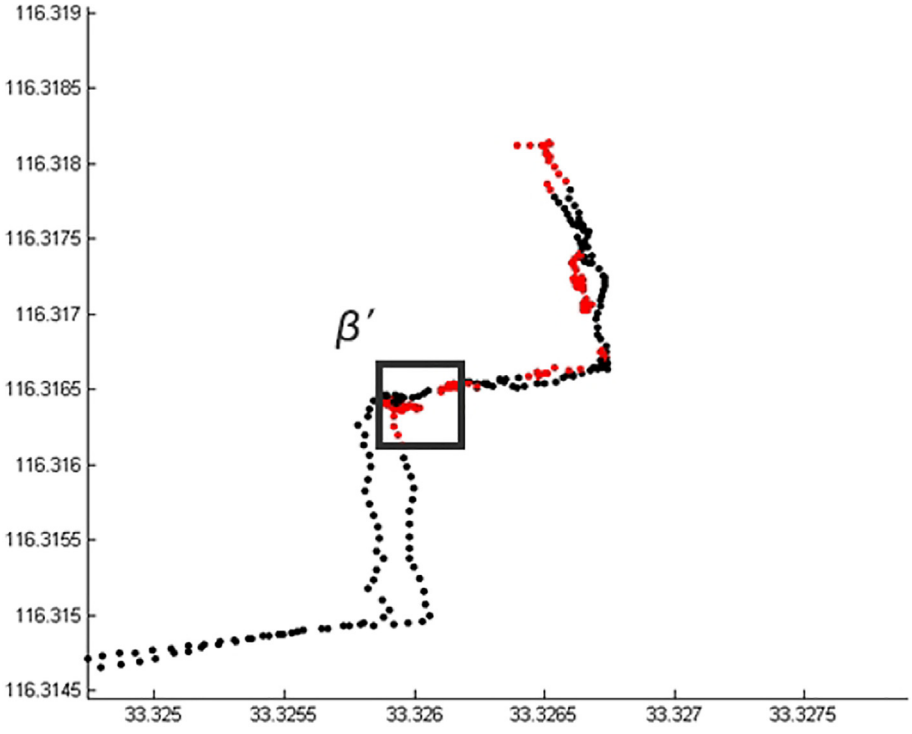

Fig. 9. Stop regions by *newEps* ($q = 0.36$, $minTime = 100s$).

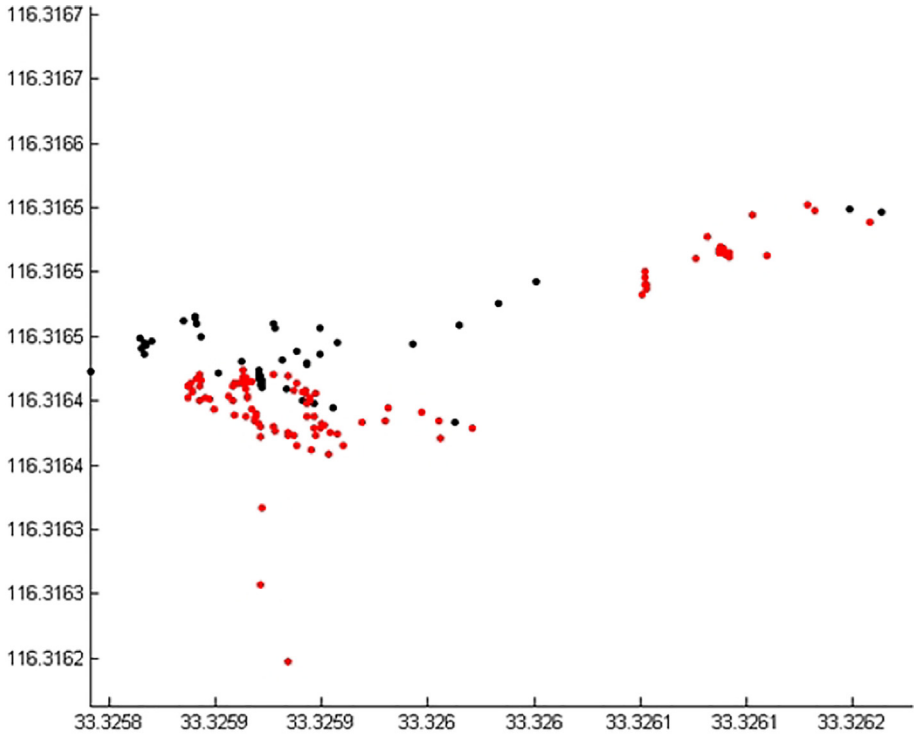

Fig. 10. A zoom-view of the square area in Fig. 8.

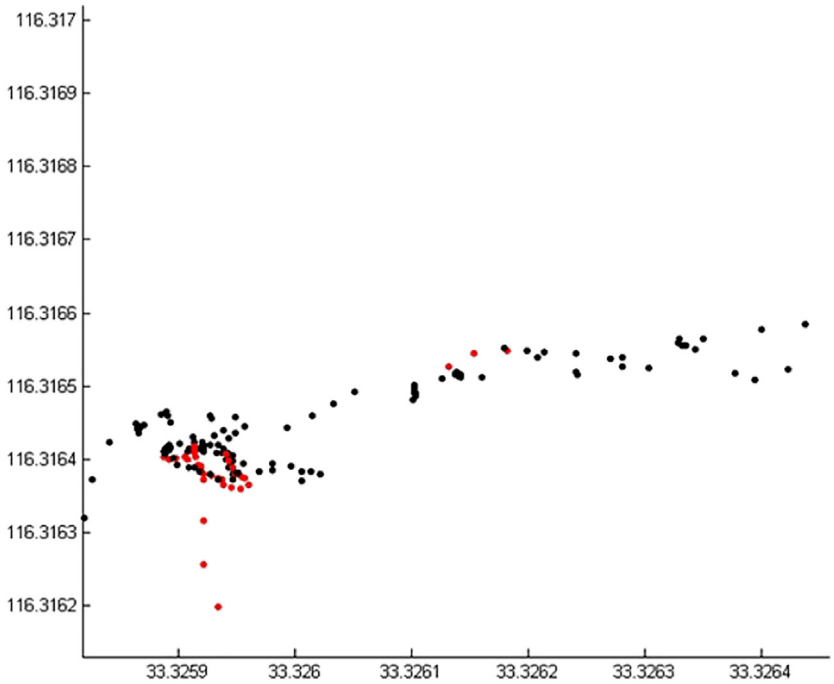

Fig. 11. A zoom-in view of the square area in Fig. 9.

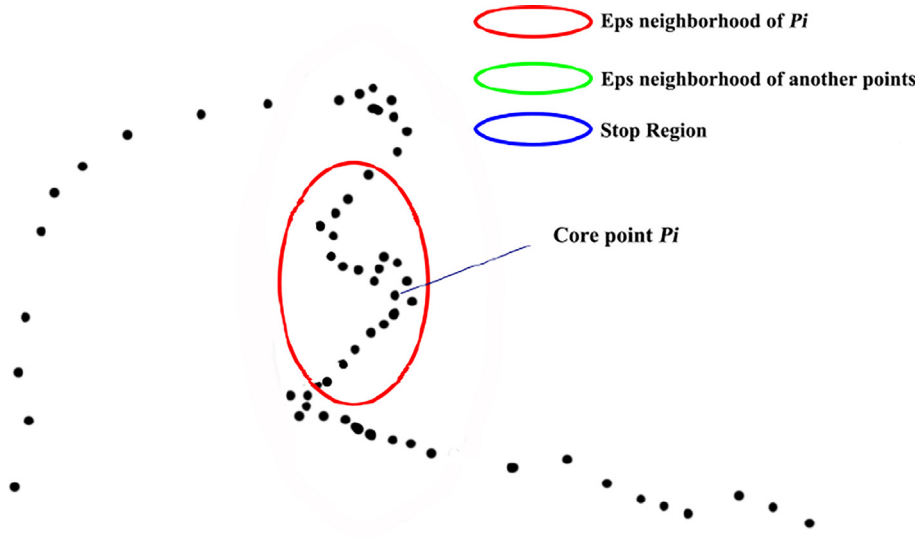

Fig. 12. Process of creating an *Eps* neighborhood.

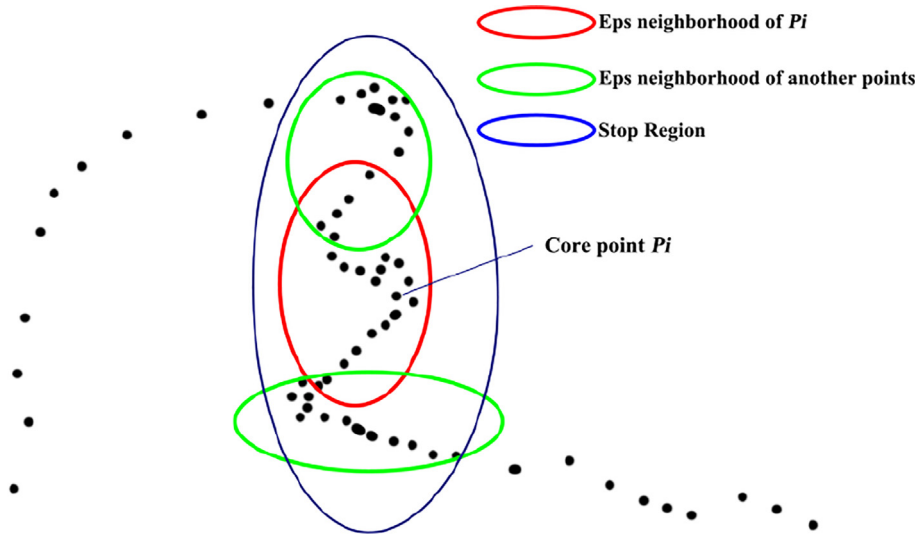

Fig. 13. Process of creating a stop region.

4.3. Initial significant place discovery

We use $InitialSignificantPlaceDiscovery(T, q, minDirChange, minTime)$, as detailed in Algorithm 4, to perform an initial discovery of significant places. We use S and M to denote the set of final stop and moving regions, respectively. After computing $originEps$ by quantile$(\mu(t), \sigma(t), a)$ using the mean distance and the standard deviation of the entire trace, the self-adaptive algorithm $EpsAdjustment(originEps, q, T)$ is called to modify the original $originEps$ and obtain $EpsAfterAjustment$. The method $StopRegionClustering(T, q, minDirchange, minTime, EpsAfterAdjustment)$ is then called to obtain the final results of stop regions. For those trajectory points that are not placed in any cluster (stop region), we generate a moving region and add them into M.

Algorithm 4 InitialSignificantPlaceDicovery

Require: T, q, *minDirChange*, *minTime*
Ensure: S, M
1: $M = \varnothing$;
2: *originEps* = quantile($\mu(t)$, $\sigma(t)$, a);
3: *EpsAfterAdjustment* = EpsAdjustment(*originEps*, q, T);
4: $S = StopRegionClustering(T, q, minDirchange, minTime, EpsAfterAdjustment)$;
5: n = the number of points in T;
6: **for** i from 1 to n **do**
7: **if** p_i is not in a stop region **then**
8: $MovingRegion = MovingRegion + p_i$;
9: **else**
10: $M = M + MovingRegion$;

Fig. 14 visually shows the approximate position of each significant location after trajectory clustering. The regions with high similarity and dense distribution of trajectory points are divided into a cluster. These low-speed regions represent significant locations of users.

4.4. Feature selection for extracting place attributes

The previous step generates stop regions, but some of them are mixed with "noise", which could lead to a low clustering accuracy. The key parameter *Eps* used in clustering is prone to interference by some bad points, which may increase or decrease the average distance in the trajectory. Some limiting conditions such as *minTime* alone are insufficient to guarantee the quality of clustering. Thus, it is very important to improve dynamic adjustment and remove abnormal clusters.

After obtaining the initial clusters based on speed, we extract a large amount of information, such as the number of points, start time, duration, average temperature, region, speed, and so on. In different applications, different parameters may have different functions. For example, when moving in a relatively small region such as malls or shopping centers, the duration plays a decisive role; while in a large-scale movement such as animal migration, the information of area, temperature and humidity is more significant for clustering. Thus, we consider different parameters to describe various attributes of clusters, depending on the application scenario.

However, the increase in the attribute dimension incurs a higher time cost of the algorithm. Therefore, to reduce the dimension of attributes, we employ the feature selection algorithm [23] to select the attributes of stops. The attribute with the largest weight value is selected as the core attribute in the current situation.

Algorithm 5 FeatureSelection

Require: D, m, δ
Ensure: T
1: Set all Weights to be 0;
2: $T = \varnothing$;
3: Separate D into D^+ (positive instances) and D^- (negative instances);
4: **for all** i = 1 to m **do**
5: Pick at random an instance $R \in D$;
6: Pick at random one of the positive instances closest to $R, Z^+ \in D^+$;
7: Pick at random one of the negative instances closest to $R, Z^- \in D^-$;
8: **if** R is a positive instance **then**
9: $Near-hit{=}Z^+$; $Near-miss{=}Z^-$;
10: **else**
11: $Near-hit{=}Z^-$; $Near-miss{=}Z^+$;
12: **for all** A=1 to N **do**
13: $W[A] = W[A] - diff(A, R, Near-hit)^2 + diff(A, R, Near-miss)^2$;
14: $Relevance = (1/m)W$;
15: **for all** A=1 to N **do**
16: **if** $Relevance[A] \geqslant \delta$ **then**
17: Insert A into T;

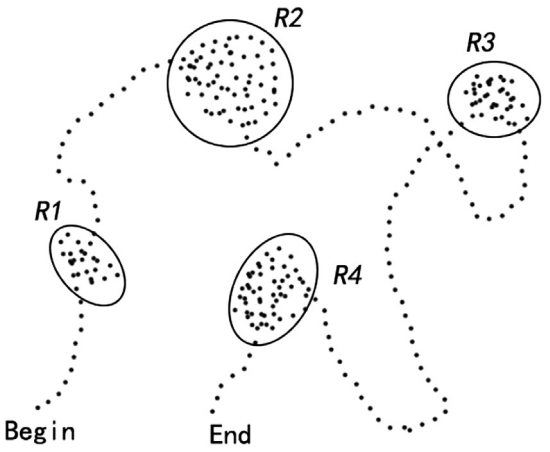

Fig. 14. A trajectory with four significant places.

The pseudocode is provided in Algorithm 4. Given training data D, sample size m, and a relevance threshold δ $(0 \leqslant \delta \leqslant 1)$, we divide the data D into two classes: one is D^+ (positive instances) and the other is D^- (negative instances). In this algorithm, D^+ is composed of a series of stop regions R_S, and D^- consists of a series of moving regions R_M. We suppose that the scale of every attribute A is either nominal (including boolean) or numerical (integer or real). The difference *diff* of attribute values between two samples R_1 and R_2 are defined as follows when R_1 and R_2 are both nominal:

$$diff(A, R_1, R_2) = \begin{cases} 0, & \text{if } R_1 \text{ and } R_2 \text{ are the same}, \\ 1, & \text{if } R_1 \text{ and } R_2 \text{ are different}. \end{cases}$$

When R_1 and R_2 are numerical, we have $diff(A, R_1, R_2) = (R_1 - R_2)/nu_k$, where nu_k is a normalization unit to normalize the values of *diff* into the interval $[0, 1]$, and we use $\sqrt{R_1^2 + R_2^2}$ as the normalization value of nu_k.

To repeat the calculation of weight for m times, we select a sample R randomly each time, find the nearest sample in the same class (Near-hit) and different class (Near-miss), and update all N attribute weights in attribute weight list W using *diff* function. Then, we calculate $1/m$ of W to obtain the average attribute weight list *Relevance*. Finally, we select those attributes whose average attribute weight in *Relevance* is above the given relevance threshold δ, and insert them into the final attribute weight list T. After calling $FeatureSelection(D, m, \delta)$ in Algorithm 4, we select the attribute with the highest average weight value. An attribute with the highest weight is able to distinguish the existing clustering differences, and thus is selected as the core attribute. Since there are not many stop regions produced in the test ($<$100), it is suitable to eliminate the "erroneous" clustering with Grubbs. Generally, the confidence interval is set to be 95%, i.e., $a = 0.05$.

After screening, it merges the stop regions that intersect with each other. The trajectory points in stop regions are sorted by time. If a stop region's end time is later than the next stop region's start time, we merge these two stop regions. We repeat the above steps until there is no intersection between any two stop regions, and produce the significant location of this trajectory as the final result.

5. Experiments and results

5.1. Experimental settings

For significant place discovery, a long trajectory distance with a high sampling rate of trajectory points would be beneficial as it may cover more significant places with a higher accuracy. We prepare and process two public datasets, denoted as *Dataset1* and *Dataset2*, to evaluate the performance of our method.

We first develop an experimental system that consists of a server and a client. The client is an Android-based mobile device [9], which collects the GPS information of the device in real time and transfers it to the server, which is equipped with Intel(R) Core(TM) i73612QMCPU@2.10 GHz and 8 GB memory. The GPS data is stored and managed by MySQL. We recruited 500 volunteers to participate in the experiment and collected a set of trajectory data on campus. We screened out a trajectory of a volunteer on a certain day to form *Dataset1*.

Dataset1: This trajectory data was collected by a volunteer from 8:17:14 to 10:18:19 in the morning of June 8, 2004. The dataset consists of logs of GPS data collected by the experimental participant at an interval of 1 s while moving on campus, and there are 7,080 position records consisting of latitudes and longitudes in the WGS84 reference system, dates, and time. The participant spends more time at some places (Library, Dormitory, and C Building) than others (the Second Teaching

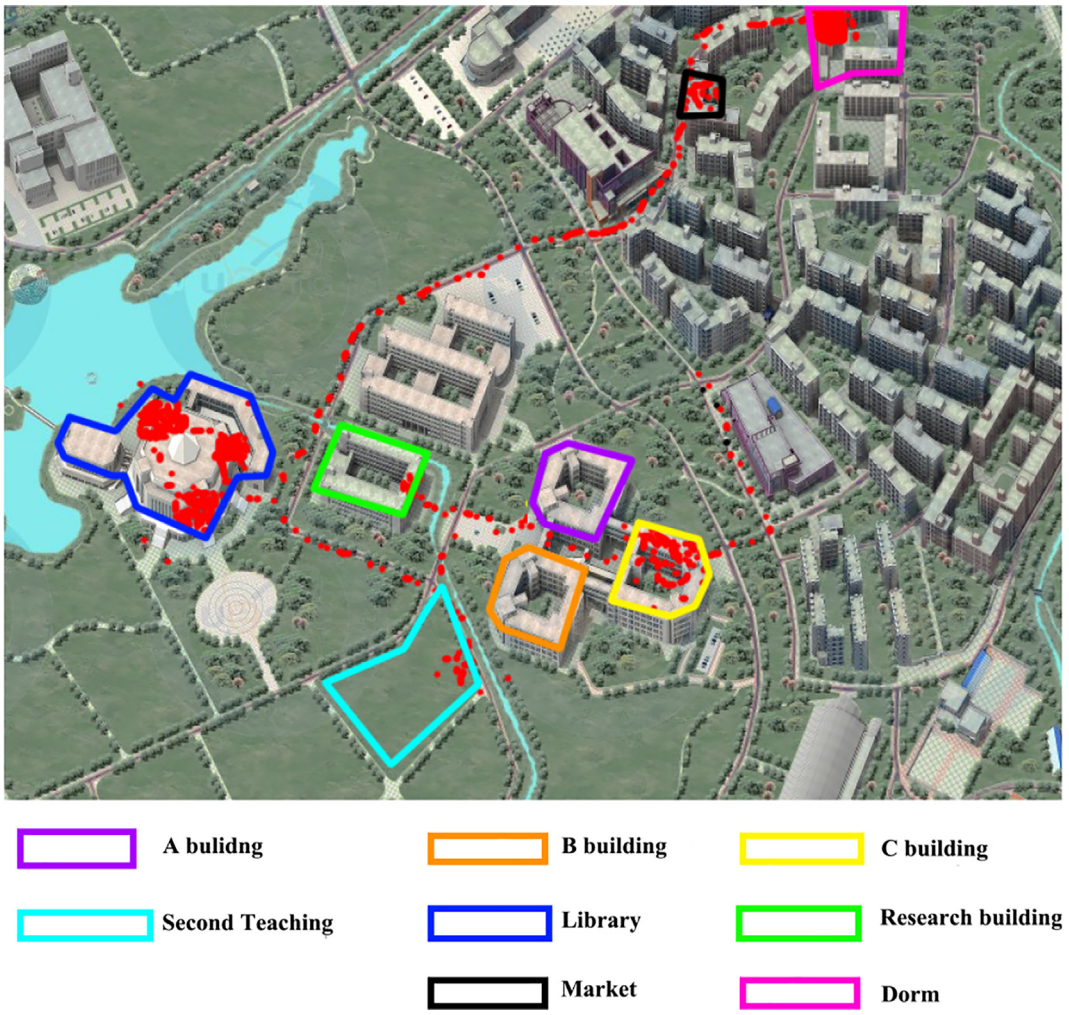

Fig. 15. Trajectory example and main buildings.

Building and Research Building), and the moving trace and main locations are plotted in Fig. 15. The dataset is publicly available in GitHub.[1]

To further evaluate the performance of the algorithms on significant places discovery, we conduct the following experiments on a small-scale dataset, GeoLife GPS Trajectories, which contains the trajectory data of 182 users from April 2007 to August 2012, and is also publicly available.[2] This dataset records not only the user's location at home or work, but also a wide range of outdoor activities, such as shopping, touring, hiking, and cycling. These trajectories are recorded by different GPS recorders and GPS phones, and have different sampling rates. Among them, 91.5% of the trajectories are recorded in a dense representation, for example, every 5–10 meters per point.

The initial distribution of trajectories by distance is illustrated in Fig. 16. Most of the trajectories are between 5 km and 100 km in distance, accounting for 72%, which are sufficiently long for significant place discovery. As illustrated in Fig. 17, the effective duration of 42% of the trajectories is longer than 1 h, which is conducive to studying the temporal feature for significant place discovery.

For most of the users in the same group, such as workers and students, the trajectories of their daily traces are of high similarity, because they follow almost an identical trace during the same time interval. Therefore, we conduct a comparative experiment of significant place discovery based on a single user's single trajectory data, which forms *Dataset2*.

Dataset2: The user performed the activities from 1:40:35 to 16:08:11 on June 21, 2009, and recorded a GPS data point every 5 s. The trajectory data contains 5,311 points, including longitude, latitude, and date/time.

[1] https://github.com/SuperShinningFinger/SMoTAS/blob/master/Dataset1.txt
[2] https://www.microsoft.com/en-us/download/details.aspx?id=52367

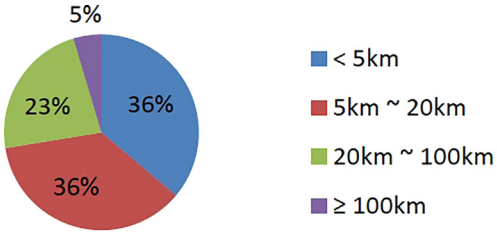

Fig. 16. Initial distribution of trajectories by distance.

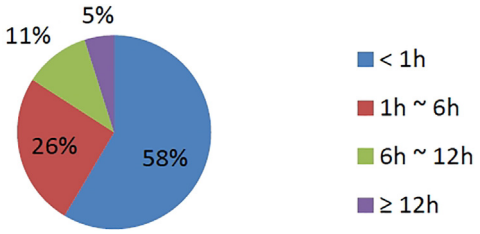

Fig. 17. Initial distribution of trajectories by duration.

We implement our method SMoTAS and five other algorithms, i.e., CB-DBSCAN, CB-SMoT, DB-SMoT, POSMIT, and TRACLUS for comparison in Python 2.7, and conduct experiments on a Windows workstation equipped with Intel Core i7 CPU@2.6 GHz and 16 GB of memory.

5.2. Experimental results

5.2.1. Results on Dataset1

In this subsection, we conduct a comparative experiment of SMoTAS and CB-DBSCAN based on *Dataset1* to evaluate the accuracy of significant place discovery. Table 1 lists the clusters in *Dataset1* produced by CB-DBSCAN, where the parameter q is used to compute *Eps*.

As shown in Table 1, CB-DBSCAN finds all the places the objects have stayed, but the Research Building and Second Teaching Building are not considered as significant locations, where the objects only stay for a short period of time. Table 2 tabulates the top six stops after the initial clustering in SMoTAS. Since the mobile devices used in the experiment cannot measure temperature and humidity, we consider only four attributes: $Start, Points, Number$, and *Speed*, where *Start* means the start time of trajectory, *Point* means the total number of points, *Time* means the time duration of the cluster, and *Speed* means the average speed of the cluster.

The next step of SMoTAS is to call $FeaturSelection(D, m, \delta)$ to compute the weight of each attribute. Fig. 18 plots the change of the values of each attribute. Table 3 tabulates the change of weight value and mean value. Note that only a portion of the values are listed in the table due to space limitation.

Table 3 shows that the duration has the highest weight and is selected as the core attribute to screen the initial clusters with Grubbs, we remove three clusters that are far from the normal value and tabulate the final results in Table 4. Fig. 19 plots the comparison of two algorithms, CB-DBSCAN and SMoTAS. We observe that SMoTAS removes two interfering items and discovers the actual interesting places of this trajectory.

Fig. 18 plots the changes of the eigenvalues of four attributes. We observe that the changes of the attributes' eigenvalues are not significant, and therefore, the mean value of the feature weights reflects the impact of the attributes on clustering.

We also evaluate the clustering quality of SMoTAS and CB-DBSCAN in terms of *silhouette*. As illustrated in Fig. 20, SMoTAS achieves a *silhouette* value of 0.913, while CB-DBSCAN achieves 0.702. These results show that SMoTAS achieves a higher quality in significant place discovery.

5.2.2. Results on Dataset2

To evaluate the performance and efficiency of our proposed method more thoroughly, we conduct comparative experiments between SMoTAS and the following sixdensity-based methods on *Dataset2*:

(1) CB-SMoT: Clustering Based Stops and Moves of Trajectories, finds significant places based on speed. This method mainly requires two user input parameters, namely, *Eps* and *minTime*, the same as *EPS* (**Definition 11**) and *minTime* (**Definition 12**) in SMoTAS. The ratio $Eps/MinTime$ gives the maximum average speed (speed limit) of the respective neighborhood.

Table 1
Dataset1 using CB-DBSCAN (q = 0.6, *minTime* = 200s).

No.	Name	Location(x)	Location(y)
1	C Building	30.758016	103.939801
2	Research Building	30.757223	103.937433
3	Second Teaching Building	30.756326	103.938135
4	Library1	30.756256	103.935083
5	Library2	30.756222	103.934494
6	Market	30.761564	103.938726
7	Dorm	30.762608	103.939478

Table 2
Dataset1 using SMoTAS (q = 0.6, *minTime* = 200s).

No.	Start	Points	Time	Speed
1	14	383	1920.0	0.38
2	14	99	495.0	0.28
3	14	71	355.0	0.56
4	14	88	446.0	0.43
5	15	82	410.0	0.44
6	15	138	706.0	0.42

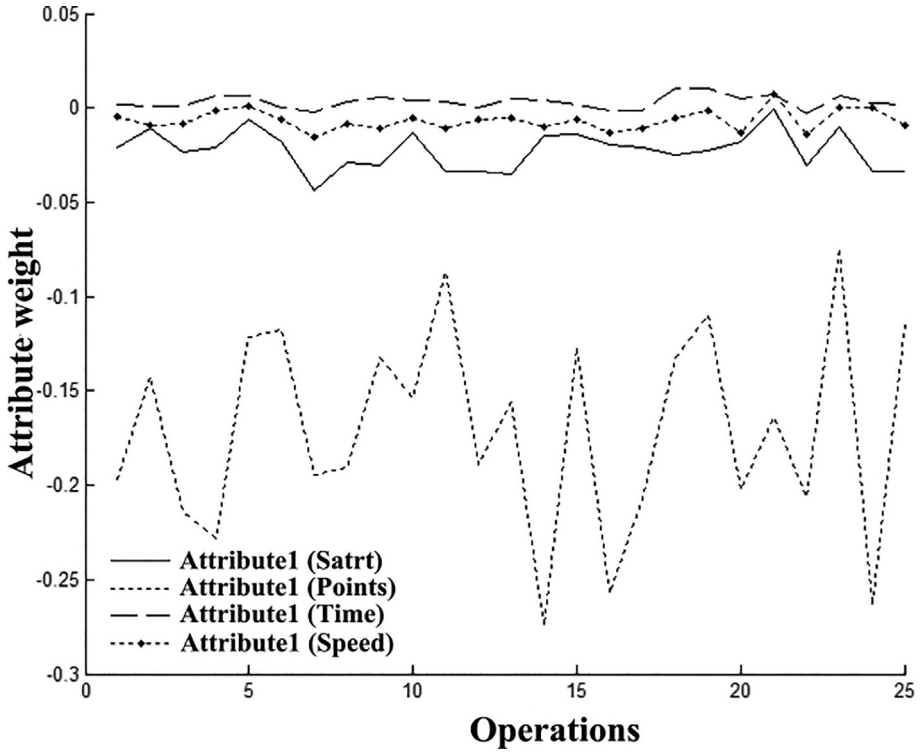

Fig. 18. Change of the values of each attribute (*Dataset1*).

(2) DB-SMoT: Direction-Based Stops and Moves of Trajectories, finds significant places based on direction and is sensitive to the amount of changes in direction. This method has three user-specified parameters *maxTol*, *minDirChange* and *minTime*. The maximal tolerance threshold *maxTol* specifies the maximum number of trajectory points with direction change less than the *minDirChange* threshold that can be found consecutively in a cluster, and the time threshold *minTime* determines the minimum time that a cluster must meet to become a stop region.
(3) POSMIT: Probability of Stops and Moves in Trajectories, which is the state of the arts, discovers possible significant places based on probability. POSMIT has 3 parameters of user input: the stop variance h_d and the index search bandwidth h_i can be obtained through a heuristic method introduced by the author, and the minimum stop probability threshold ϵ is

Table 3
Dataset1 using SMoTAS (q = 0.6, *minTime* = 200s).

Item	Start	Points	Time	Speed
weight	0.0063	0.0311	0.0084	0.0027
weight	-0.0023	-0.1122	0.0074	-0.0047
⋮	⋮	⋮	⋮	⋮
weight	0.0094	0.0702	0.0091	0.0036
weight	0.0070	-0.0435	0.0062	-0.0047
mean	0.0043	-0.0443	0.0087	-0.0036

Table 4
Dataset1 using SMoTAS (q = 0.6, *minTime* = 200s).

No.	Name	Location(x)	Location(y)
1	C Building	30.758016	103.939801
2	Library1	30.756256	103.935083
3	Library2	30.756291	103.934829
4	Library3	30.756224	103.934494
5	Market	30.761564	103.938726
6	Dorm	30.762608	103.939478

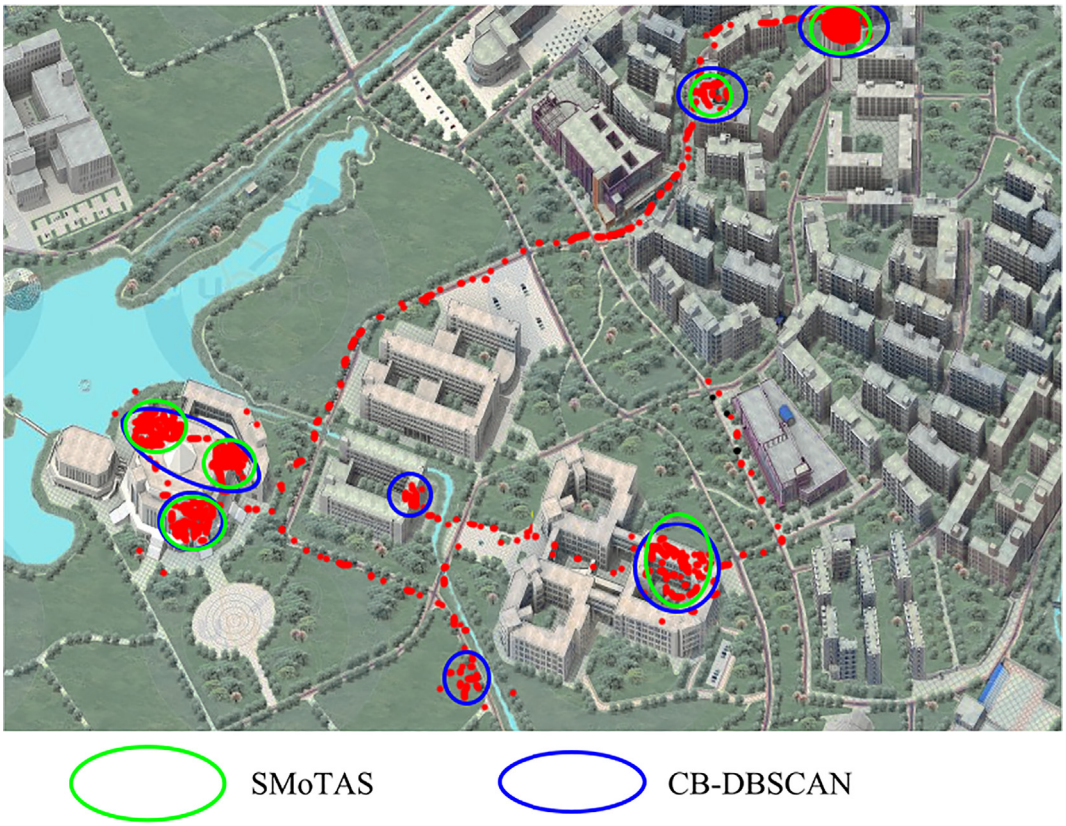

Fig. 19. Results of two algorithms in comparison (*Dataset1*).

specified by the user. Parameter h_d helps the user control how strict they wish to be when calculating stop probabilities, h_i controls the range of index numbers between the current entry and some neighbour entry, and ϵ controls which entries are classified as stop regions.

(4) TRACLUS: This is the only algorithm designed for multi-trajectory clustering at the beginning but is also suitable for significant place clustering in single trajectories. There are two parameters of user input, namely, the domain range

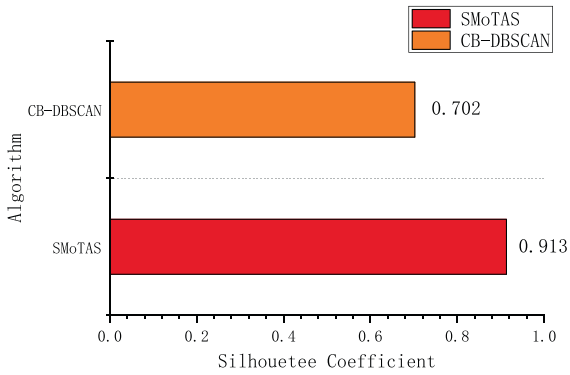

Fig. 20. Results of silhouette comparison (*Dataset1*).

parameter ϵ and the minimum number of line segment threshold *minLns*, which together determine the final clustering quality.

(5) STC-SMoT: Spatiotemporal Clustering-based Stops and Moves of Trajectories was proposed recently to address signal loss and signal noise by imputing gaps using state-dependent path interpolation, and is the state of the arts. This method mainly requires six user input parameters, namely, the spatial size of stop *SR*, the minimal velocity of move *MV*, the maximum duration of stop *MaxStopDur*, the minimum duration of stop *MinStopDur*, the sampling rates of trajectory *Interval*, and *SC* that captures the relationship between *Eps* and *Interval*, where *Eps* is defined in **Definition 11**.

(6) PSO-OPSPD: Particle Swarm Optimization based Optimal Parameter for Stop Points Detection, which is one of the most recently proposed methods, semiautomatically detects stops with a relatively high accuracy. This method has ten user-specified parameters, i.e., $E_{lb}, E_{ub}, m_{lb}, m_{ub}, w_{max}, w_{min}, c1, c2, N$, and *MaxIteration*. Here, E_{lb} and E_{ub} are the lower

Table 5
Parameter values for the seven methods used in the experiments based on *Dataset2*.

Method	Parameter	value
SMoTAS	q	0.30
	minTime	45.00
	minDirChange	30.00
	Eps	7.50
	EpsAfterAdjustment	16.36
CB-SMoT	*Eps*	7.50
	minTime	45.00
DB-SMoT	*maxTol*	7.00
	minDirChange	30.00
	minTime	45.00
POSMIT	h_d	7.82
	h_i	5.00
	ϵ	0.25
TRACLUS	*minLns*	5.00
	ϵ	10.00
STC-SMoT	*SR*	50.00
	MV	2.00
	MaxStopDur	60.00
	MinStopDur	45.00
	Interval	5.00
	SC	1.50
PSO-OPSPD	E_{lb}	5.00
	E_{ub}	10.00
	m_{lb}	2.00
	m_{ub}	5.00
	w_{max}	0.90
	w_{min}	0.40
	$c1$	2.00
	$c2$	2.00
	N	50.00
	MaxIteration	100.00

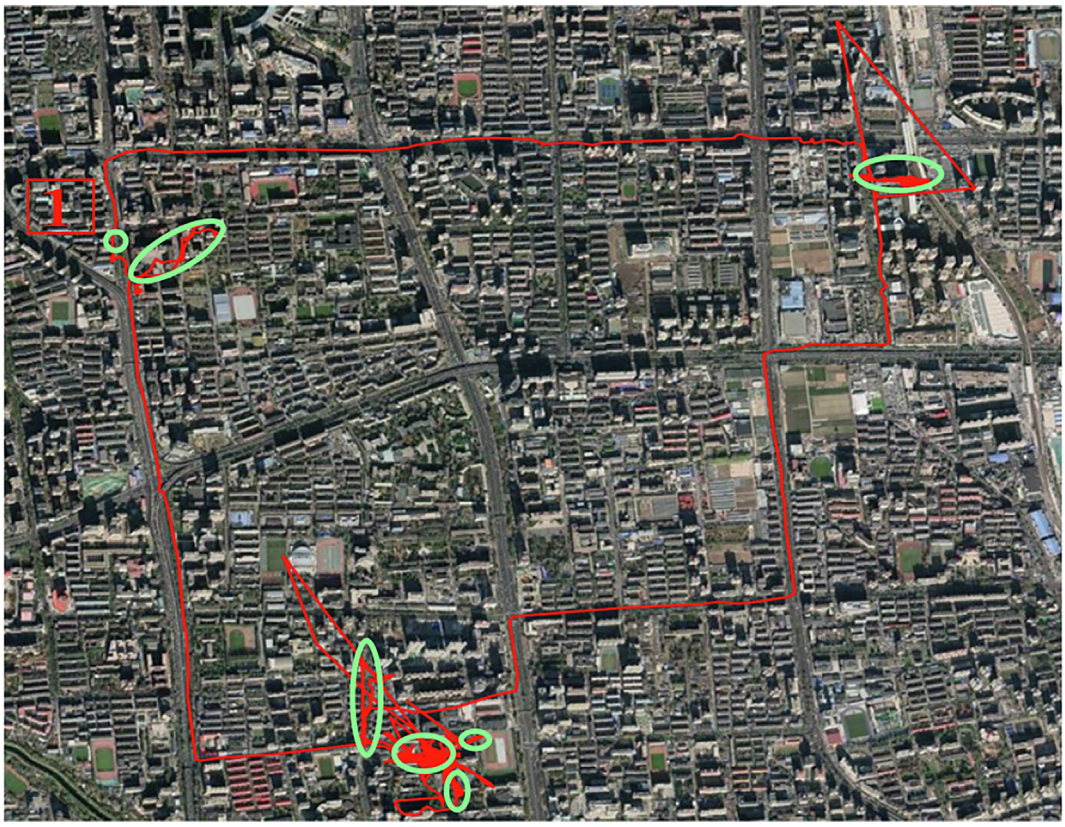

Fig. 21. Visualization results obtained by SMoTAS before feature selection (*Dataset2*).

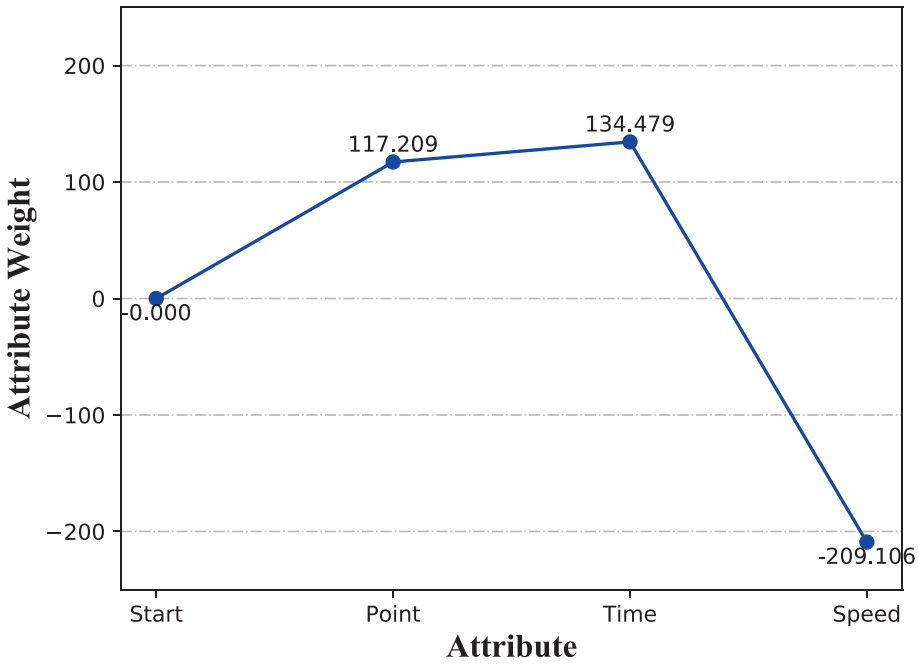

Fig. 22. Comparison of attribute weights (*Dataset2*).

and upper bounds of *EPS* range, respectively, where *EPS* is defined in **Definition 11**. Similarly, m_{lb} and m_{ub} are the lower and upper bounds of *MinPts* range, respectively, where *MinPts* is the parameter value of the minimum points. Particle Swamp Optimization used in the algorithm requires six parameters, i.e., maximum inertial weight w_{max}, minimum inertial weight w_{min}, cognitive coefficient $c1$, social coefficient $c2$, number of particles N, and maximum number of iterations *MaxIteration*.

All of the experiments are designed to vary algorithm parameters and measure the corresponding results. Table 5 summarizes the descriptions and settings of different parameters of each method. To make a fair comparison, we use optimal parameter values for all methods in the current experimental environment.

As shown in Table 5, the *Eps* parameter used by CB-SMoT is the initial *Eps* parameter of SMoTAS with a value of 7.50, while SMoTAS uses *EPSAfterAdjustment*, which is adjusted by Algorithm 1, with a value of 16.36. Particularly, *EPS* in STC-SMoT is calculated as $SC \times Interval$, and its value is 7.5 (i.e., 1.5×5), which is also the same as that in SMoTAS. In addition, the *minTime* value in CB-SMoT and DB-SMoT, and the *MinStopDur* value in PSO-OPSPD are the same as that in SMoTAS, which is 45.00. Similarly, the *minDirchange* value of DB-SMoT is the same as that in SMoTAS, which is 30.00. Moreover, the parameter values considered for the simulations of PSO-OPSPD include number of particles $N = 50.00$, range of inertial weight $[w_{min}, w_{max}] = [0.40, 0.90]$, cognitive coefficient $c1 = 2.00$, social coefficient $c2 = 2.00$, and $MaxIteration = 100.00$, which are consistent with the work by Bandyopadhyay et al. in [5].

The cluster visualization results obtained by SMoTAS before applying cluster filtering using the feature selection algorithm are shown in Fig. 21. Even without cluster filtering, our method finds seven clusters, which are marked with green circles of different sizes. The cluster labeled with 1 at the upper left corner is somewhat abnormal, as it resides in the middle of the road instead of inside the building. According to a close observation, the road that is labeled with 1 is not a traffic light intersection, so it should not be labeled as a significant place.

We select four attributes *Start*, *Point*, *Time* and *Speed* as alternatives and run Algorithm 5 to calculate the weight of each attribute. The results of weight calculation are shown in Fig. 22, where we observe that *Time* is the best attribute for feature selection with the largest weight, while *Speed* has the lowest value. Hence, *Time* is selected as the core attribute, and the filtered clusters are visualized in Fig. 23.

As illustrated in Fig. 23, the cluster circled with the green circle labeled with 1 at the upper left corner is filtered out. As shown in Fig. 24, CB-SMoT and POSMIT produce a large number of clusters, but most of them are very small, and the clusters

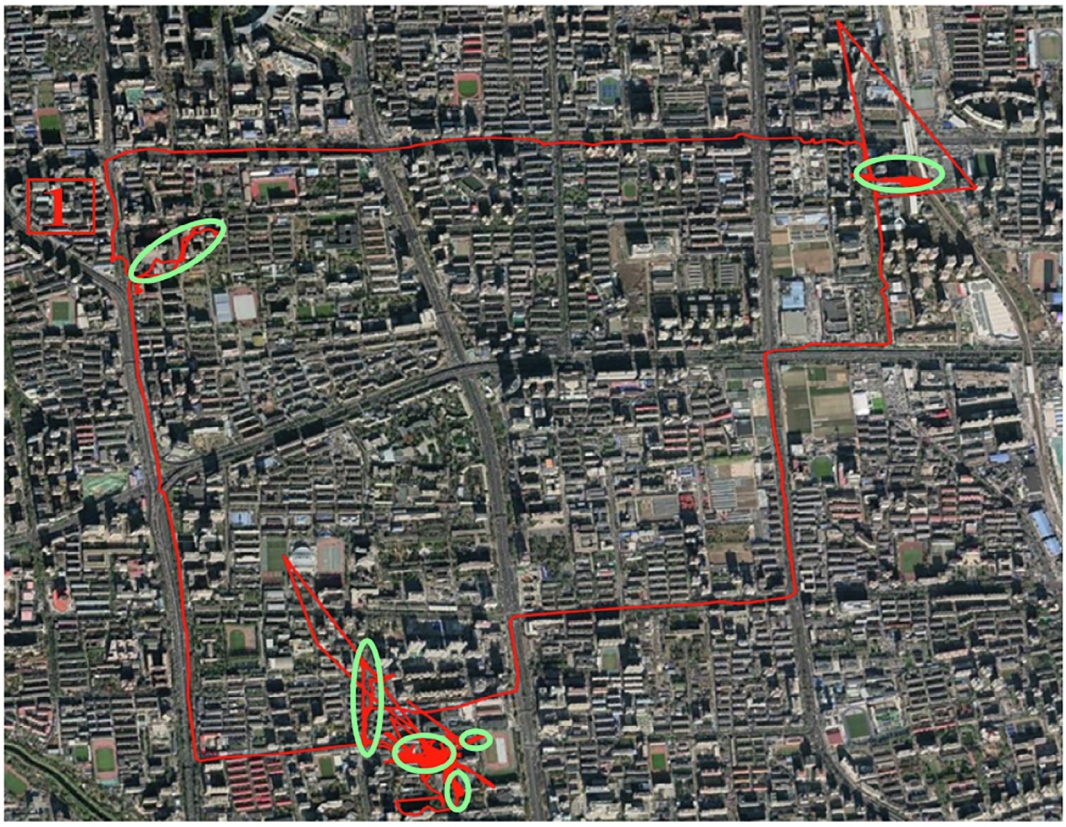

Fig. 23. Visualization results obtained by SMoTAS after feature selection (*Dataset2*).

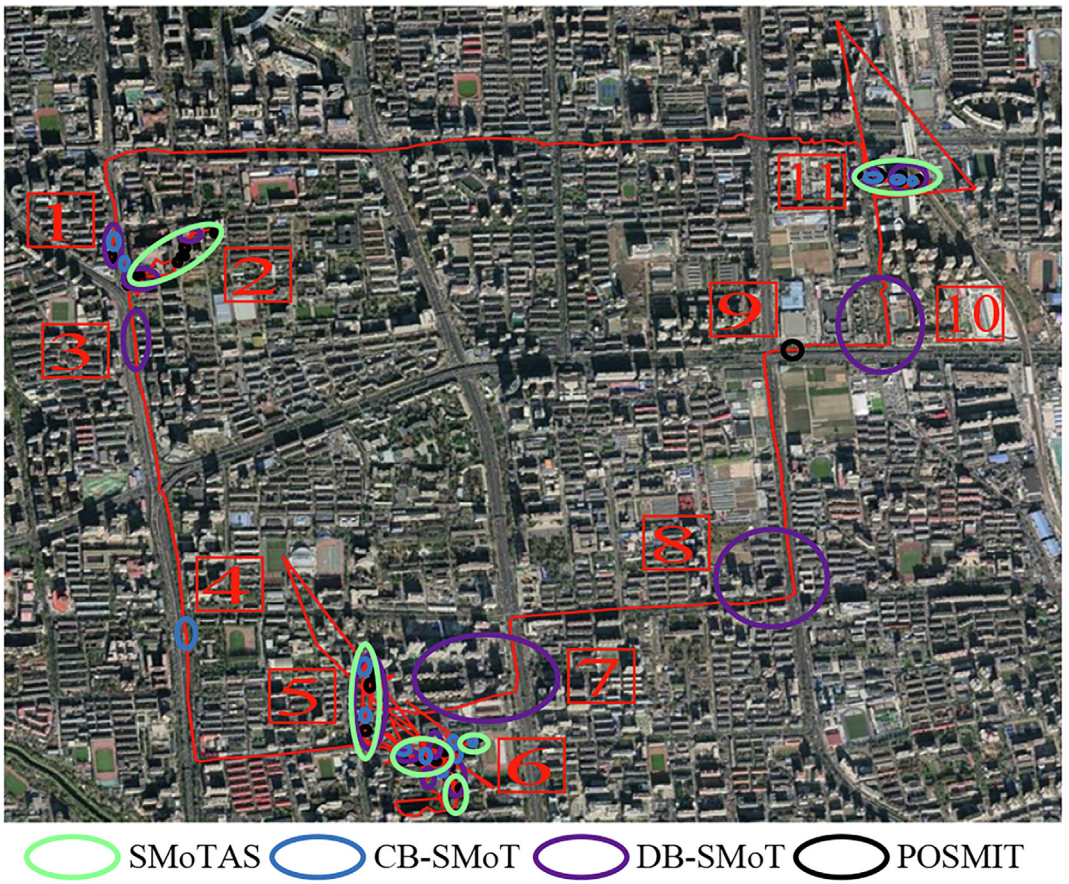

Fig. 24. Visualization results obtained by SMoTAS, CB-SMoT, DB-SMoT, and POSMIT (*Dataset2*).

that originally belong to the same significant place are divided into several nonadjacent clusters. For example, in the clusters labeled with 5, CB-SMoT and POSMIT obtain two small clusters separated by a certain distance (represented by blue circles and black circles, respectively), but the real significant place is represented by the green circle, where the user handles business. CB-SMoT and POSMIT also incur false alarms, such as the purple circled area labeled with 4 and the black circled area labeled with 9. These two areas are in the middle of the road and are not traffic signal intersections, so they are not significant places. This is more common in the clusters obtained by DB-SMoT. Since DB-SMoT is sensitive to the amount of direction change, it is more likely to form clusters in places where the turning angle is relatively large. For example, the areas circled by the purple circles labeled with 7, 8, and 10 in Fig. 24 are all formed at the corners, but these are only three ordinary corners, which are not sufficient to become significant places. In Fig. 25, we observe that some of the clusters obtained by TRACLUS, STC-SMoT, PSO-OPSPD, and SMoTAS are close in size and have intersections with each other, such as the clusters labeled with 2, 4, and 5. This visualized result intuitively shows that TRACLUS, STC-SMoT, and PSO-OPSPD obtain a better result than CB-SMoT, DB-SMoT and POSMIT. Moveover, the results of STC-SMoT and PSO-OPSPDT are aligned better with SMoTAS than TRACLUS. There are some other issues with TRACLUS. For instance, the blue cluster labeled with 1 is mistakenly reported as a significant place, and in the cluster labeled with 4, there is an oblong ellipse circled area in purple, which diagonally spans across multiple buildings and results in an excessively broad significant place. Similarly, for STC-SMoT and PSO-OPSPD, the circled area in purple and black in the cluster labeled with 1 is mistakenly reported as a significant place. Particularly, there is another cluster labeled with 3 that has been identified incorrectly, which shows that STC-SMoT produces a better result than PSO-OPSPD. In sum, these results qualitatively show that SMoTAS outperforms the six methods in comparison for significant place discovery.

Furthermore, we measure the *silhouette* value for a quantitative evaluation of clustering results. According to the definition, *silhouette* is in the range from -1 to 1, and the value increases as the trajectory in the cluster becomes denser. A higher silhouette value indicates a better clustering quality.

The silhouettes of seven different algorithms are plotted in Fig. 26. SMoTAS performs the best among all, and the rest are sorted from high to low as PSO-OPSPD, STC-SMoT, POSMIT, TRACLUS, CB-SMoT and DB-SMoT. PSO-OPSPD is the closest to SMoTAS, and STC-SMoT and POSMIT both achieve a relatively large silhouettes value. There is a gap between SMoTAS and CB-SMoT because the number of stop points in SMoTAS is larger while the number of stop regions in SMoTAS is less than CB-

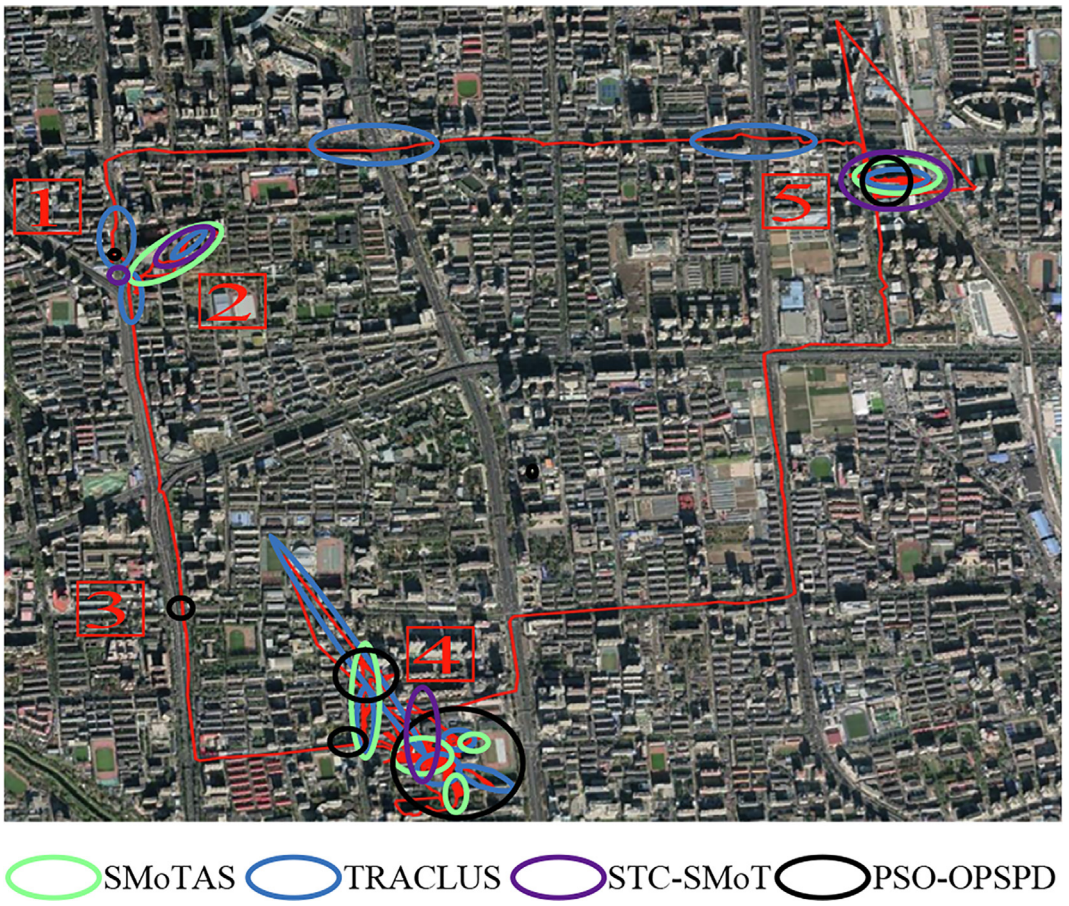

Fig. 25. Visualization results obtained by SMoTAS, TRACLUS, STC-SMoT, and PSO-OPSPD (*Dataset2*).

SMoT. Similar to CB-SMoT, the silhouette value of TRACLUS is also smaller than SMoTAS for the same reason. Moreover, as DB-SMoT is a direction-based clustering method while silhouette relies on distance, DB-SMoT has the worst performance. These results illustrate the superiority of SMoTAS over the other algorithms in comparison.

Finally, we compare the running time of seven methods, and the efficiency measurements are provided in Table 6. As shown in Table 6, CB-SMoT and DB-SMoT incur relatively shorter running time, which are 4.25s and 0.66s, respectively. However, their ability to discover significant places is quite limited as shown by the clustering visualization results and *silhouette* values, which are more important than running time in the problem of significant place mining. Similarly, POSMIT is the fastest, but its visualization result is unsatisfactory. Moreover, with relatively better results of significant place discovery, SMoTAS incurs a shorter running time of 11.39s than TRACLUS, STC-SMoT, and PSO-OPSPD, which are 12.54s, 14.65s, and 3201.23s, respectively. PSO-OPSPD, which traverses all possible parameter values in advance and iteratively searches for the optimal parameter values of *EPS* and *MinPts* based on the highest internal validity index (i.e., *S_Dbw* and *Silhouette*), achieves the second highest *silhouette* value with a relatively high-quality visualization result, but with the longest running time of 3201.23s.

In sum, among all the algorithms in comparison, SMoTAS produces the best visualization results and the highest *silhouette* value with a relatively shorter running time. These results indicate that SMoTAS has the best overall performance than all other algorithms in comparison.

6. Conclusion

We formulated a problem of significant place discovery from complex spatial and temporal trajectories and proposed a clustering-based mining approach, SMoTAS, with labeled semantics. SMoTAS combines a track clustering algorithm with feature selection to improve the accuracy of clustering and enhance the adaptability to noise-corrupted data without accurate geographic information. Adding multiple attributes and leveraging feature selection greatly extend the applicability of this algorithm to a wide variety of practical applications. Extensive experimental results on real-life trajectory data show that

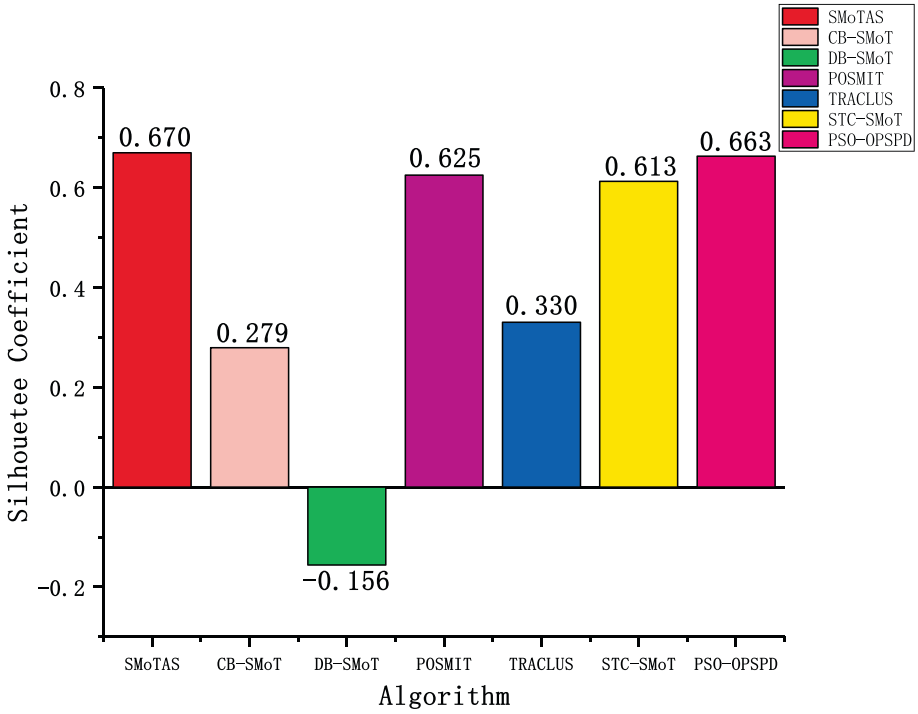

Fig. 26. Comparison of silhouette values (*Dataset2*).

Table 6
Efficiency performance analysis of seven methods in the experiments based on *Dataset2*.

Method	running time (s)
SMoTAS	11.39
CB-SMoT	4.25
DB-SMoT	0.66
POSMIT	0.58
TRUCLUS	12.54
STC-SMoT	14.65
PSO-OPSPD	3201.23

SMoTAS has better overall performance than the existing algorithms including PSO-OPSPD, STC-SMoT, POSMIT, TRACLUS, CB-DBSCAN, CB-SMoT, and DB-SMoT.

It is of our future interest to improve SMoTAS in various aspects, for example, by designing a more suitable calculation method for *Eps* parameter to make it adaptable to different scenarios and enhancing the adaptability to unmarked data. We also plan to extend SMoTAS to conduct real-time streaming data mining and use customized heat coefficients to detect specific significant places according to user needs. We will test our algorithm with more real-life trajectory data [20,40,39,17,43], and expand its application scope.

CRediT authorship contribution statement

Xinzheng Niu: Conceptualization, Methodology, Project administration, Funding acquisition. **Shimin Wang:** Writing - original draft, Resources, Formal analysis, Methodology. **Chase Q. Wu:** Methodology, Writing - review & editing, Supervision, Validation. **Yuran Li:** Investigation, Data curation, Visualization. **Peng Wu:** Formal analysis, Resources. **Jiahui Zhu:** Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is sponsored by the Science and Technology Planning Project of Sichuan Province under Grant No. 2020YFG0054, and the Scientific Research Project of State Grid Sichuan Electric Power Company Information and Communication Company under Grant No. SGSCXT00XGJS1800219. We would also like to thank Bowen Shi for his comments, which helped improve part of this research.

References

[1] L.O. Alvares, V. Bogorny, J. Fernandes de Macedo, B. Moelans, S. Spaccapietra, Dynamic modeling of trajectory patterns using data mining and reverse engineering, in: Twenty-Sixth International Conference on Conceptual Modeling ACS, 2007, pp. 149–154.
[2] G. Andrienko, D. Malerba, M. May, M. Teisseire, Mining spatio-temporal data, Journal of Intelligent Information Systems 27 (2006) 187–190, https://doi.org/10.1007/s10844-006-9949-3, URL:https://doi.org/10.1007/s10844-006-9949-3.
[3] S. Anekritmongkol, M.K. Kasamsan, The comparative of boolean algebra compress and apriori rule techniques for new theoretic association rule mining model, International Journal of Advancements in Computing Technology 3 (2010) 216–222.
[4] D. Ashbrook, T. Starner, Learning significant locations and predicting user movement with gps, in: Proceedings. Sixth International Symposium on Wearable Computers, Washington, DC, USA, 2002, pp. 101–108.
[5] M. Bandyopadhyay, Leveraging clustering validation index for detecting stops in spatial trajectory data: a semi-automatic approach, Journal of Spatial Science (2020) 1–21.
[6] L. Bermingham, I. Lee, A probabilistic stop and move classifier for noisy gps trajectories, Data Mining and Knowledge Discovery 32 (2018) 1634–1662.
[7] D. Birant, A. Kut, St-dbscan: An algorithm for clustering spatial–temporal data, Data & Knowledge Engineering 60 (2007) 208–221.
[8] F. Bremond, V. Bogorny, L. Patino, S. Cosar, G. Pusiol, G. Donatiello, Monitoring peoples behaviour using video analysis and trajectory clustering, Smart Cities 1 (2014) 46.
[9] T. Brinkhoff, A framework for generating network-based moving objects, GeoInformatica 6 (2002) 153–180.
[10] E.M. Carboni, V. Bogorny, Inferring drivers behavior through trajectory analysis, in: Intelligent Systems'2014, Springer International Publishing, Cham, 2015, pp. 837–848.
[11] M. Celik, Partial spatio-temporal co-occurrence pattern mining, Knowledge and Information Systems 44 (2015) 27–49, https://doi.org/10.1007/s10115-014-0750-2, URL:https://doi.org/10.1007/s10115-014-0750-2.
[12] Chandio, A.F., SHU, L., Cheng, C., Khawaja, A., 2007. Spatio temporal hazard mitigation modeling using gis and geospatial data mining techniques, in: Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science-Volume 6, Citeseer. pp. 659–662..
[13] A. Claudia, L.O. Arlindo, Temporal data mining: an overview, Proceedings of KDD Workshop on Temporal Data Mining (2001) 1–13.
[14] S. Dodge, R. Weibel, A.K. Lautenschütz, Towards a taxonomy of movement patterns, Information Visualization 7 (2008) 240–252.
[15] N. Eagle, A.S. Pentland, Reality mining: sensing complex social systems, Personal and Ubiquitous Computing 10 (2006) 255–268.
[16] M. Ester, H.P. Kriegel, J. Sander, X. Xu, et al, A density-based algorithm for discovering clusters in large spatial databases with noise, in: International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.
[17] S. Gao, K. Janowicz, H. Couclelis, Extracting urban functional regions from points of interest and human activities on location-based social networks, Transactions in GIS 21 (2017) 446–467.
[18] B. Goethals, M.J. Zaki, Advances in frequent itemset mining implementations: report on fimi'03, Acm Sigkdd Explorations Newsletter 6 (2004) 109–117.
[19] J. Han, Research frontiers in advanced data mining technologies and applications, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, Nanjing, China, 2007, pp. 1–5.
[20] W. Huang, S. Li, X. Liu, Y. Ban, Predicting human mobility with activity changes, International Journal of Geographical Information Science 29 (2015) 1569–1587.
[21] S. Hwang, C. VanDeMark, N. Dhatt, S.V. Yalla, R.T. Crews, Segmenting human trajectory data by movement states while addressing signal loss and signal noise, International Journal of Geographical Information Science 32 (2018) 1391–1412.
[22] S. Johnson, The handbook of geographic information science, Reference Reviews 22 (2008) 51.
[23] K. Kira, L.A. Rendell, et al, The feature selection problem: Traditional methods and a new algorithm, in: Aaai, California, San Jose, 1992, pp. 129–134.
[24] J. Lee, J. Han, K. Whang, Trajectory clustering: a partition-and-group framework, 2007, pp. 593–604.
[25] S. Lee, Y. Choi, S. Lim, J. Park, A spatio-temporal distance based clustering approach for discovering significant places from trajectory data, in: Intŕl Conference on Computer Science, Data Mining and Mechanical Engg., Bangkok, Thailand, 2015, pp. 130–134.
[26] L. Liao, D. Fox, H. Kautz, Extracting places and activities from gps traces using hierarchical conditional random fields, The International Journal of Robotics Research 26 (2007) 119–134, https://doi.org/10.1177/0278364907073775.
[27] S. Lynen, M. Bosse, R. Siegwart, Trajectory-based place-recognition for efficient large scale localization, International Journal of Computer Vision 124 (2017) 49–64.
[28] N. Marmasse, C. Schmandt, Location-aware information delivery withcommotion, in: P. Thomas, H.W. Gellersen (Eds.), Handheld and Ubiquitous Computing, Springer, Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 157–171.
[29] M. Nanni, D. Pedreschi, Time-focused clustering of trajectories of moving objects, Journal of Intelligent Information Systems 27 (2006) 267–289.
[30] A.T. Palma, V. Bogorny, B. Kuijpers, L.O. Alvares, A clustering-based approach for discovering interesting places in trajectories, in: Proceedings of the 2008 ACM Symposium on Applied Computing, Association for Computing Machinery, New York, NY, USA, 2008, pp. 863–868, https://doi.org/10.1145/1363686.1363886, URL:https://doi.org/10.1145/1363686.1363886.
[31] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M.L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, Z. Yan, Semantic trajectories modeling and analysis, ACM Computing Surveys 45 (2013), https://doi.org/10.1145/2501654.2501656, URL:https://doi.org/10.1145/2501654.2501656.
[32] B. Pinty, I. Andredakis, M. Clerici, T. Kaminski, M. Taberner, M.M. Verstraete, N. Gobron, S. Plummer, J.L. Widlowski, The application of grubbs criterion in determinant the gross error of measuring result, Woodworking Machinery (2006) 116.
[33] J.A.M. Rocha, V.C. Times, G. Oliveira, L.O. Alvares, V. Bogorny, Db-smot: A direction-based spatio-temporal clustering method, in: 2010 5th IEEE International Conference Intelligent Systems, IEEE, London, UK, 2010, pp. 114–119.
[34] J.F. Roddick, E. Hoel, M.J. Egenhofer, D. Papadias, B. Salzberg, Spatial, temporal and spatio-temporal databases-hot issues and directions for phd research, ACM SIGMOD Record 33 (2004) 126–131.
[35] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.
[36] S. Spaccapietra, C. Parent, Adding meaning to your steps (keynote paper), in: International Conference on Conceptual Modeling, Springer, Brussels, Belgium, 2011, pp. 13–31.
[37] S. Spaccapietra, C. Parent, M.L. Damiani, J.A. de Macedo, F. Porto, C. Vangenot, A conceptual view on trajectories, Data & Knowledge Engineering 65 (2008) 126–146.
[38] S. Spaccapietra, C. Parent, M.L. Damiani, J.A. de Macedo, F. Porto, C. Vangenot, A conceptual view on trajectories, Data & Knowledge Engineering 65 (2008) 126–146.

[39] E. Steiger, B. Resch, A. Zipf, Exploration of spatiotemporal and semantic clusters of twitter data using unsupervised neural networks, International Journal of Geographical Information Science 30 (2016) 1694–1716.
[40] E. Steiger, R. Westerholt, B. Resch, A. Zipf, Twitter as an indicator for whereabouts of people? Correlating twitter with uk census data, Computers, Environment and Urban Systems 54 (2015) 255–265.
[41] L.H. Tran, Q.V.H. Nguyen, N.H. Do, Z. Yan, Robust and hierarchical stop discovery in sparse and diverse trajectories, Epfl (2011).
[42] G. Yang, Discovering significant places from mobile phones – a mass market solution, in: R. Fuller, X.D. Koutsoukos (Eds.), Mobile Entity Localization and Tracking in GPS-less Environnments, Springer, Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 34–49.
[43] M. Yang, C. Cheng, B. Chen, Mining individual similarity by assessing interactions with personally significant places from gps trajectories, ISPRS International Journal of Geo-Information 7 (2018) 126.
[44] C. Zhou, N. Bhatnagar, S. Shekhar, L. Terveen, Mining personally important places from gps tracks, in: 2007 IEEE 23rd International Conference on Data Engineering Workshop, IEEE, Washington, DC, USA, 2007, pp. 517–526.
[45] Y. Zhou, D.Q. Miao, X.D. Yue, Rough k-means clustering based on self-adaptive weights, Computer Science 38 (2011) 237–241.