

Article

Road Congestion Detection Based on Trajectory Stay-Place Clustering

Qingying Yu ^{1,2,3}, **Yonglong Luo** ^{1,2,3,*} , **Chuanming Chen** ^{2,3} and **Xiaoyao Zheng** ^{2,3}¹ School of Geography and Tourism, Anhui Normal University, Wuhu 241002, China; ahnuyuq@ahnu.edu.cn² School of Computer and Information, Anhui Normal University, Wuhu 241002, China;
ccm1981@ahnu.edu.cn (C.C.); zxiaoao@ahnu.edu.cn (X.Z.)³ Anhui Provincial Key Laboratory of Network and Information Security, Wuhu 241002, China

* Correspondence: ylluo@ustc.edu.cn; Tel.: +86-0553-5910645

Received: 16 April 2019; Accepted: 4 June 2019; Published: 6 June 2019



Abstract: The results of road congestion detection can be used for the rational planning of travel routes and as guidance for traffic management. The trajectory data of moving objects can record their positions at each moment and reflect their moving features. Utilizing trajectory mining technology to effectively identify road congestion locations is of great importance and has practical value in the fields of traffic and urban planning. This paper addresses the issue by proposing a novel approach to detect road congestion locations based on trajectory stay-place clustering. First, this approach estimates the speed status of each time-stamped location in each trajectory. Then, it extracts the stay places of the trajectory, each of which is denoted as a seven-tuple containing information such as starting and ending time, central coordinate, average direction difference, and so on. Third, the time-stamped locations included in stay places are partitioned into different stay-place equivalence classes according to the timestamps. Finally, stay places in each equivalence class are clustered to mine the congestion locations of multiple trajectories at a certain period of time. Visual representation and experimental results on real-life cab trajectory datasets show that the proposed approach is suitable for the detection of congestion locations at different timestamps.

Keywords: trajectory; stay place; average direction difference; stay-place equivalence class; road congestion detection; stay-place clustering

1. Introduction

Traffic congestion has become a serious problem that hinders the rapid development of modern cities. If habitually congested road sections are detected in advance, vehicles can effectively avoid spending unnecessary time on the road, greatly increasing the travel efficiency. Utilizing modern technology to effectively identify road congestion locations is of great importance and has practical value in the fields of traffic and urban planning [1,2].

With the rapid development of positioning and storage technology, large amounts of trajectory data of moving objects can be produced. These trajectories contain abundant temporal and spatial information and can be used in many information services [3–6], such as location-based services, traffic monitoring, road planning, congestion prediction, user behavior analysis, and travel recommendations [7–11].

Moving and staying are the two basic states of locations in trajectory data. It is well known that stay point recognition is a critical method in mining information from trajectory data, and this is, therefore, the premise of frequent location discovery, interest point mining, etc. [12,13]. Stay point recognition plays a fundamental role in trajectory analysis. Stay points are a series of actual time-stamped locations extracted from the raw trajectory data. Each stay point denotes a geographic region where an individual

stays for a certain duration. On the one hand, stay points generally carry a particular semantic meaning, such as the shopping malls, restaurants, and cinemas, visited by an individual [14]. On the other hand, a vehicle is also in a state of staying when it is stuck in a traffic jam. These are two different kinds of staying states. The former is a meaningful stay, while the latter is a helpless and anxious waiting, unlike waiting for a normal signal. Therefore, the extraction of meaningless staying places based on trajectory data of vehicles can be used for road congestion detection. In this paper, we propose a methodology called Congestion Location Detection approach based on Stay-places Clustering (CLDSC) that allows the detection of congestion locations from the trajectory data of moving vehicles.

In summary, this paper makes the following contributions:

- (1) Trajectory stay model construction. By learning from the idea of stay points [14], we propose a novel trajectory stay model, which is adopted in stay place extraction and later research.
- (2) Trajectory stay places extraction. A stay place extraction algorithm is proposed based on the extraction of continuous low-speed locations and the calculation of average direction differences of each candidate stay place.
- (3) Equivalence classes partition and stay places clustering. An equivalence class partition algorithm is proposed, and a stay-place clustering algorithm based on equivalence classes is proposed to detect the optimal stay places, which can help detect road congestion locations.
- (4) Experimental Study. The proposed algorithm is tested on a real-life trajectory dataset. Visual presentation and experimental results show that our algorithm can detect road congestion locations with high accuracy.

The remainder of this paper is organized as follows. In Section 2, we introduce the related work. Section 3 introduces preliminary concepts and the problem definition. Section 4 presents a novel method for the detection of road congestion locations. Experimental results and analysis are presented in Section 5. Section 6 concludes the paper and provides future research directions.

2. Related Work

A large amount of research has been conducted on traffic congestion detection, which provides many useful insights in practical applications [10,15–17]. Detection methods are mainly realized based on the traffic flow of vehicles, road networks, or POIs (Points of Interests). Wang et al. [16] proposed a three-phase framework for exploring the traffic congestion correlation between road segments from multiple data sources including road networks, GPS trajectories of taxis, and POIs. This method focuses on traffic congestion correlation, and it is realized based on the congestion information on each road segment in each specific time slot. Anbaroglu et al. [15] presented a detection method for non-recurrent traffic congestion based on spatio-temporal clustering. Kong et al. [10] proposed a novel approach to estimate and predict urban traffic congestion using floating car trajectory data. This approach calculates traffic density based on traffic speed, traffic volume, and the length and lane number of the road section. Liu et al. [18] proposed a mobility-based clustering approach to detect crowded spots in city transportation. D’Andrea et al. [17] presented an expert system for the detection of traffic congestion and incidents from real-time GPS trajectories. Wang et al. [19] presented a locality-constraint distance-metric learning for traffic congestion detection to characterize the congestion level in different scenes. Therefore, a dataset that consists of 20 different scenes is constructed in their work. All the above-mentioned methods are implemented depending on both road network and vehicle information. The challenges of data collection and high computation complexity are becoming increasingly evident. The advantages and disadvantages of map matching algorithms directly affect the accuracy and reliability of the congestion detection results.

Meanwhile, many achievements have been made on the detection of stay points based on the trajectories of moving objects [14,20–22], which can be extended to many applications, including user behavior analysis and road planning. For example, Li et al. [23] first proposed a stay point detection algorithm based on the distance measure and time span between any two successive points.

Yuan et al. [20,24] proposed an improved stay point detection algorithm based on density clustering. Zheng et al. [14] proposed a hierarchical framework to uniformly model each individual's location history, which consists of three steps. The stay points are determined from the trajectory data in the first step. They presented a stay point detection algorithm based on two operations: checking the spatio-temporal constraint and expanding. On the one hand, the method ignores that, although the distance between two merged locations is small, the time interval may be very short; even if such locations and others form a time interval greater than the time threshold, it is not suitable for them to be merged together as a stop point. On the other hand, the selection of distance threshold is not easy due to the loss of signal in buildings. Do et al. [22] first used raw trajectory data as an input to detect stay points. Pavan et al. [25] proposed an approach to recognize important locations based on the detection of a candidate stay point and the mapping of its feature space. Stylianou et al. [26] presented a geometry-based method to extract stay points. The user's trajectory is converted to a two-dimensional discrete time series curve, and the stay point is converted to a local minimum of the first derivative of the curve.

Cao et al. [27] extracted semantic locations from GPS data and assigned significance to the extracted locations. Kami et al. [12] proposed an algorithm that extracts significant locations based on the method of probabilistically detecting peak locations in the density distribution of data points. They also used density information to score the extracted locations according to their importance. An et al. [28] proposed a method for measuring urban recurrent congestion evolution patterns based on grid divisions and GPS-equipped vehicle mobility data. The Grid Congestion Mode (GCM) is defined based on the average velocity of all trajectories within each grid. This method is vulnerable to individual noise locations.

The above methods have achieved good results in traffic congestion detection or stay point detection. However, the intrinsic characteristics of different kinds of stay points are rarely distinguished, and the relationship between abnormal staying and congestion is not considered. During the trip, if there is a large number of abnormal waiting in a certain position at a certain time, it indicates that there is congestion at this place at this time. Therefore, road congestion detection based on stay place extraction is reasonable and feasible.

In this paper, we extract stay places and propose a new approach called CLDSC to detect road congestions. The stay place recognition method based on the extraction of continuous low-speed points extraction can effectively avoid noise interference. In addition, this method considers the average direction difference within each candidate stay place, which can better serve the detection of road congestion locations.

3. Fundamental Concept and Problem Definition

In this section, we define the concepts, notations, and corresponding calculation equations used throughout the paper.

Definition 1 (Trajectory). *A trajectory is an ordered set of time-stamped locations, denoted as T:*

$$T = \{Tid, (t_1, x_1, y_1), (t_2, x_2, y_2), \dots, (t_n, x_n, y_n)\} \quad (1)$$

where $t_r < t_{r+1}$ for all $1 \leq r < n$, t_r is a timestamp, (x_r, y_r) is a location in \mathbb{R}^2 , (t_r, x_r, y_r) is a time-stamped location which means that a moving object is at location (x_r, y_r) at time t_r , Tid is the identifier of a trajectory representing a moving object, and n (also represented by $|T|$) is the number of time-stamped locations in the trajectory T .

Hereinafter, $TS = \{T_1, T_2, \dots, T_p\}$ is a dataset consisting of p trajectories, $T_i = \{i, loc_1^i, loc_2^i, \dots, loc_{n_i}^i\}$ represents the i -th trajectory in TS , loc_w^i is the w -th time-stamped location of the i -th trajectory ($1 \leq w \leq n_i$, $|T_i| = n_i$).

Definition 2 (Trajectory segment). Consider a trajectory T . A trajectory segment is defined as the line segment between any pair of adjacent time-stamped locations (loc_r, loc_{r+1}) in the trajectory T , $1 \leq r < |T|$.

Figure 1 shows an example including four trajectory segments Seg_{m_i} ($1 \leq i \leq 4$) and 5 locations loc_{m_j} ($1 \leq j \leq 5$).

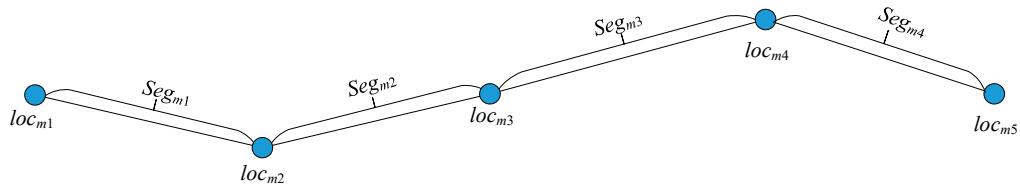


Figure 1. Trajectory segment example. Each point represents a location in the trajectory. There are four trajectory segments and five locations.

Definition 3 (Low-speed point). A low-speed point refers to a time-stamped location where a moving object is at rest, moves within a relatively small range for a long time, or moves at a very slow speed.

For example, when a user is resting, eating, visiting, shopping, waiting for traffic lights, or stuck in a traffic jam, the locations they pass are low-speed points. In this paper, low-speed points are extracted based on the speed threshold set by the experiments.

Definition 4 (Stay place). Stay place refers to a set consisting of at least three consecutive low-speed points that meet the time threshold and the average direction difference threshold requirements. The two thresholds are defined in specific applications. Consider a trajectory T , sn stay places of T can be represented as a set $SD = \{S_1, S_2, \dots, S_{sn}\}$, where $S_m = \{loc_{m_1}, loc_{m_2}, \dots, loc_{m_{|S_m|}}\}$ represents the m -th stay place in T , $1 \leq m \leq sn$.

Each stay place S_m is denoted as a seven-tuple $\langle sno_{S_m}, eno_{S_m}, st_{S_m}, et_{S_m}, x_{S_m}, y_{S_m}, dd_{S_m} \rangle$, where sno_{S_m} and eno_{S_m} respectively represent the No. of the first low-speed point and the No. of the last low-speed point in S_m , st_{S_m} and et_{S_m} represent the timestamp of the first low-speed point and the timestamp of the last low-speed point in S_m respectively, $\langle x_{S_m}, y_{S_m} \rangle$ represents the central coordinate of S_m , and dd_{S_m} is the average direction difference of S_m . The average direction difference will be described in detail in Section 4.3.1. The calculation method for x_{S_m} and y_{S_m} is as follows:

$$x_{S_m} = \frac{1}{|S_m|-1} \sum_{i=1}^{|S_m|-1} w_i * seg_x_i \quad (2)$$

$$y_{S_m} = \frac{1}{|S_m|-1} \sum_{i=1}^{|S_m|-1} w_i * seg_y_i \quad (3)$$

where $|S_m|$ represents the number of locations in S_m , $\langle seg_x_i, seg_y_i \rangle$ represents the central coordinate of the i -th trajectory segment $Seg_{m_i} = \langle loc_{m_i}, loc_{m_{i+1}} \rangle$ in S_m ($1 \leq i < |S_m|$), and w_i represents the weight of the coordinate (seg_x_i, seg_y_i) . The relevant calculation equations are as follows:

$$seg_x_i = \left(x_{loc_{m_i}} + x_{loc_{m_{i+1}}} \right) / 2 \quad (4)$$

$$seg_y_i = \left(y_{loc_{m_i}} + y_{loc_{m_{i+1}}} \right) / 2 \quad (5)$$

$$w_i = \frac{1/(v_i + \delta)}{\sum_{j=1}^{|S_m|-1} (1/(v_j + \delta))} \quad (6)$$

where v_i represents the average velocity of the trajectory segment Seg_{m_i} , and δ is the adjustment parameter used to avoid divide-by-zero error and smoothen the weight when the average velocity is very small. If v_i is not equal to 0, δ is 0, otherwise, δ is set as the standard deviation of average velocity.

Definition 5 (Stay point). *Stay point refers to a meaningful stay place for a purpose. It can be derived from stay place defined in Definition 4. Each stay point is denoted as a four-tuple $\langle \text{SPID}, \Delta t, x, y \rangle$, where SPID , Δt and $\langle x, y \rangle$ are the identifier, the duration, and the coordinate of the stay point, respectively.*

4. Road Congestion Location Detection Algorithm

4.1. Algorithm Framework

In this paper, we propose a road congestion location detection approach that helps users effectively avoid the wastage of time caused by congestion during trips. The framework of the proposed approach is shown in Figure 2.

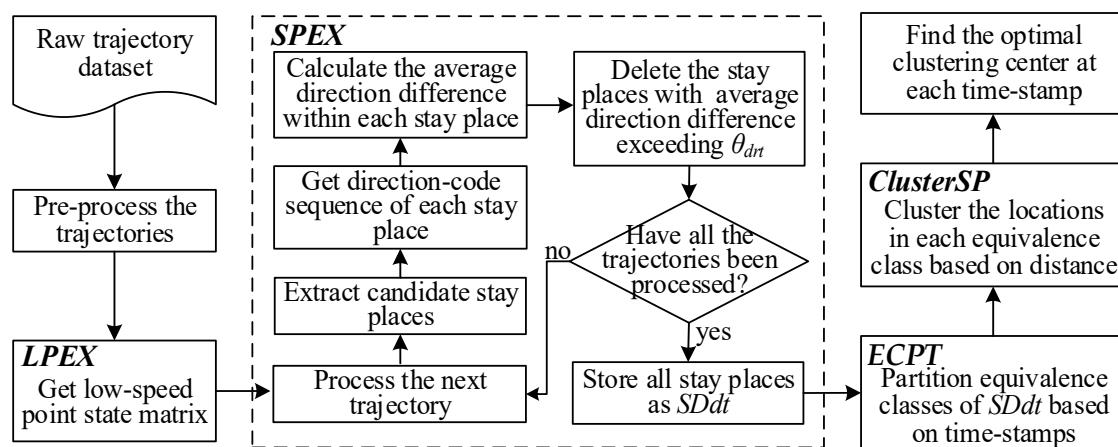


Figure 2. Schematic diagram of the proposed approach.

Before mining the trajectory dataset for potential knowledge, it is necessary to pre-process the trajectory dataset. The trajectory pre-processing phase in our algorithm specifically refers to formalization. The trajectory dataset TS is pre-processed so that each $T_i \in TS$ is in the form $T_i = \{i, (t_1^i, x_1^i, y_1^i), \dots, (t_n^i, x_n^i, y_n^i)\}$. Each trajectory does not necessarily consist of the same number of time-stamped locations. In the following algorithms, both trajectories and sub-trajectories conform to this form.

The trajectory pre-processing phase is pre-experimental preparation, which is not difficult to achieve. Therefore, the other steps of the detection approach shown in Figure 1 will be focused on. In particular, the proposed congestion location detection approach is denoted as CLDSC, which is implemented by four algorithms. These are LPEX (Low-speed Points Extraction), SPEX (Stay Places Extraction), ECPT (Equivalence Classes Partition), and ClusterSP (Clustering Based on Stay Places), which will be introduced later in Sections 4.2–4.4.

4.2. Low-Speed Point State Matrix Acquisition

Moving speed is an important indicator that reflects the state of moving objects. If the moving speed at some location is below the designated threshold, it is marked as a low-speed point. The low-speed point state matrix of trajectory dataset TS is acquired by Algorithm 1, denoted as LPEX. The pseudo code of the algorithm is given as follows:

Algorithm 1. LPEX: Low-speed Points Extraction

Input: $TS = \{T_1, T_2, \dots, T_p\}$ (the raw trajectory dataset), θ_s (the speed threshold)

Output: $LSPS$ (the low-speed point state matrix)

1. $tn \leftarrow \max\{|T_1|, |T_2|, \dots, |T_p|\};$
2. Initialize $LSPS \leftarrow zeros(p, tn);$
3. **for** $i \leftarrow 1$ to p **do**
4. **for** $j \leftarrow 2$ to $|T_i|$ **do**
5. **if** (there is the j -th location in T_i) **then**
6. $dist \leftarrow \text{Distance}(loc_{j-1}^i, loc_j^i);$
7. $v_{ij} \leftarrow dist / (t_j - t_{j-1});$
8. **if** ($v_{ij} < \theta_s$) **then**
9. $LSPS_{ij} \leftarrow 1;$
10. **end if**
11. **end if**
12. **end for**
13. **end for**
14. **return** $LSPS;$

$LSPS$ is a matrix with size $p \times tn$ (Line 2). If the j -th time-stamped location of the i -th trajectory is a low-speed point, its state is set to 1; otherwise, its state is set to 0 (Lines 3–13). The low-speed point state matrix $LSPS$ is the basis of stay places extraction.

The time complexity of Algorithm 1 is $O(p \cdot tn)$, which is used to calculate the speed status of each point in each trajectory, where p is the number of trajectories in TS , i.e., $p=|TS|$, tn is the maximum value of the set $\{|T_1|, |T_2|, \dots, |T_p|\}$, and $|T_i|$ represents the number of time-stamped locations in T_i . The space complexity of Algorithm 1 is also $O(p \cdot tn)$, which is mainly caused by storing the matrix $LSPS$.

4.3. Trajectory Stay Places Extraction Based on Average Direction Difference

4.3.1. Average Direction Difference of Stay Place

In the trajectory of a moving object, there may be different kinds of stay places, as defined in Definition 4. Stay places can be divided into 3 categories: (1) SP I is real stay place. At real stay places, moving objects are doing something meaningful, such as shopping, visiting, eating, etc. This kind of stay place is also regarded as a stay point, which is defined in Definition 5. (2) SP II is signal loss place. The data of certain time-stamped locations in some trajectories are missing because the signal of GPS devices may be blocked by buildings. For these stay places, there may be only a few time-stamped locations for a long period of time. If some data are missing due to poor indoor signals, it is very similar to the first class. (3) SP III is congestion stay place. Congestion stay place refers to the kind of stay place caused by traffic jams. This means that moving objects are waiting at congestion stay places for a long time due to traffic jams. This is a kind of stay place that makes people unhappy. Therefore, detecting congestion stay places plays an important role in the field of traffic management. For example, users can plan better routes to avoid congestion based on the detection results. In this paper, we focus on the third category of stay places. Figure 3 shows a trajectory including the three kinds of stay places, where S_1 and S_2 belong to SP I, S_3 belongs to SP II, and S_4 belongs to SP III.

In Class SP I or SP II area, a user is generally in a purposeful state of staying. As can be seen from the enlarged S_2 at the bottom of Figure 3, the directions of different trajectory segments in such stay places are constantly changing. By contrast, the directions of different trajectory segments are approximately unchanged in the Class SP III area. Therefore, we incorporate the concept of average direction difference into our approach.

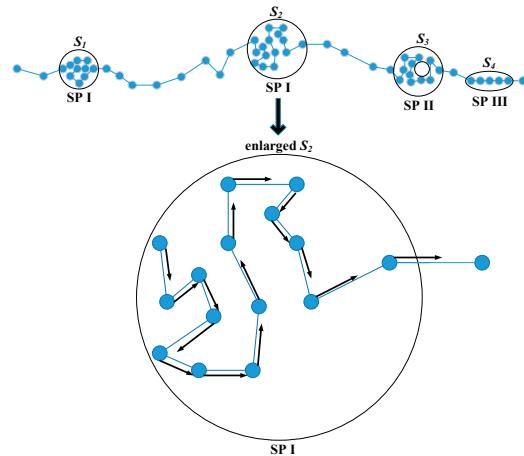


Figure 3. Three kinds of stay places within a trajectory. The direction change can be clearly seen in the enlarged S_2 .

Each trajectory segment has a direction code, which is calculated based on the trajectory direction-code sequence acquisition method proposed in our previous work [29]. At first, the 2D plane coordinate system is evenly divided into N regions according to the angle. Then, a specific direction code is assigned to each region, and the angle of each region is $2\pi/N$. Let N , the number of direction regions be 16, so the regions with continuous integer codes range from 1 to 16, as shown in Figure 4a. Consider a stay place with $SegN$ trajectory segments. Its average direction difference $Avedd$ is calculated as follows:

$$Avedd = \frac{\sum_{i=1}^{SegN-1} \min\{|dc_{i+1} - dc_i|, N - |dc_{i+1} - dc_i|\}}{SegN - 1} \quad (7)$$

where $SegN$ represents the number of trajectory segments within the stay place, and dc_i represents the direction code of the i -th trajectory segment.

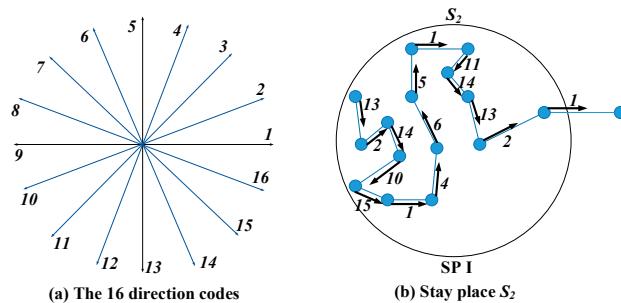


Figure 4. An example of the direction-code sequence within a stay place. (a) shows the partition of 16 direction code regions. (b) shows the direction code of each trajectory segment within the area of S_2 .

For example, Figure 4 displays an example of the direction-code sequence within the stay place S_2 shown in Figure 3. Figure 4a shows the $N (=16)$ direction codes introduced above, and Figure 4b shows the direction codes of different trajectory segments within S_2 . More specifically, the direction-code sequence of S_2 is [13,2,14,10,15,1,4,6,5,1,11,14,13,2,1]. The number of trajectory segments $segN$ within this stay place is 15. According to Equation (7), there are 14 (i.e., $segN-1$) direction difference values that can be calculated for this direction-code sequence. For example, the first one is $\min\{|dc_2 - dc_1|, N - |dc_2 - dc_1|\} = \min\{|2 - 13|, 16 - |2 - 13|\} = \min\{11, 5\} = 5$, the second one is $\min\{|dc_3 - dc_2|, N - |dc_3 - dc_2|\} = \min\{|14 - 2|, 16 - |14 - 2|\} = \min\{12, 4\} = 4$, the third one is $\min\{|dc_4 - dc_3|, N - |dc_4 - dc_3|\} = \min\{|10 - 14|, 16 - |10 - 14|\} = \min\{4, 12\} = 4$, and so on. All the

direction difference values are calculated in this way. They are 5, 4, 4, 5, 2, 3, 2, 1, 4, 6, 3, 1, 5, 1. Therefore, its average direction difference value $Avedd = \frac{5+4+4+5+2+3+2+1+4+6+3+1+5+1}{14} = 46/14 \approx 3.2857$.

In the rest of the article, if not stated, all the stay places we mentioned belong to the third category SP III.

4.3.2. Trajectory Stay Places Extraction

In this section, we introduce the method of trajectory stay place extraction based on the *LSPS* matrix and average direction difference.

The set of stay places in each trajectory is acquired by Algorithm 2, denoted as SPEX. Consider a trajectory dataset $TS = \{T_1, T_2, \dots, T_p\}$. First, for each trajectory, if at least three consecutive low-speed points meeting the duration requirement are found, they are regarded as one of the candidate stay places in this trajectory. Without loss of generalization, we consider that the trajectory T_i contains sn stay places ($1 \leq i \leq p$), which are successively denoted as $S_1^i, S_2^i, \dots, S_{sn}^i$. Each S_m^i ($1 \leq m \leq sn$) contains several low-speed points, which meet the duration requirement (Lines 3–14). Second, the seven-tuples for each stay place S_m^i are calculated (Lines 15–25). The specific meaning of each component in seven-tuple $\langle sno_{S_m^i}, eno_{S_m^i}, st_{S_m^i}, et_{S_m^i}, x_{S_m^i}, y_{S_m^i}, dd_{S_m^i} \rangle$ (Line 23) can be found in Definition 4.

The pseudo code of the SPEX algorithm is given as follows:

Algorithm 2. SPEX: Stay Places Extraction

Input: $TS = \{T_1, T_2, \dots, T_p\}$ (the trajectory dataset), *LSPS* (the low-speed point state matrix), θ_t (the time threshold), θ_{drt} (the direction difference threshold)

Output: $SDdt$ (the set of stay places in each trajectory)

```

1.   for  $i \leftarrow 1$  to  $p$  do
2.      $k \leftarrow 1; j \leftarrow 1; S_k^i \leftarrow \emptyset; SD^i \leftarrow \emptyset; SDdt^i \leftarrow \emptyset;$ 
3.     while ( $j < |T_i|$ ) do
4.       if  $LSPS_{ij} == 0$  then  $j \leftarrow j+1$ ; continue; end if
5.        $e \leftarrow j;$ 
6.       while ( $LSPS_{ie} == 1$  and  $e \leq |T_i|$ ) do
7.          $e \leftarrow e+1;$ 
8.       end while
9.       if ( $e-j > 2$  and  $loc_{e-1}^i.t - loc_j^i.t > \theta_t$ ) then
10.         $S_k^i \leftarrow S_k^i \cup \{loc_j^i, \dots, loc_{e-1}^i\};$ 
11.         $k \leftarrow k+1; S_k^i \leftarrow \emptyset;$ 
12.      end if
13.       $j \leftarrow e;$ 
14.    end while
15.    for  $m \leftarrow 1$  to  $k-1$  do
16.       $sno_{S_m^i} \leftarrow$  the No. of the first location in  $S_m^i$ ;
17.       $eno_{S_m^i} \leftarrow$  the No. of the last location in  $S_m^i$ ;
18.       $st_{S_m^i} \leftarrow$  the time stamp of the first location in  $S_m^i$ ;
19.       $et_{S_m^i} \leftarrow$  the time stamp of the last location in  $S_m^i$ ;
20.       $(x_{S_m^i}, y_{S_m^i}) \leftarrow$  the central coordinate of  $S_m^i$ ; // using Equations (2)–(6)
21.       $dd_{S_m^i} \leftarrow$  ComDirectDiff( $N, SLdcsl(sno_{S_m^i} : eno_{S_m^i} - 1)$ );
22.      if ( $dd_{S_m^i} < \theta_{drt}$ ) then
23.         $SDdt^i \leftarrow SDdt^i \cup \{(sno_{S_m^i}, eno_{S_m^i}, st_{S_m^i}, et_{S_m^i}, x_{S_m^i}, y_{S_m^i}, dd_{S_m^i})\};$ 
24.      end if
25.    end for
26.  end for
27. return  $SDdt$ ;

```

In Algorithm 2, the output result is $SDdt = \{SDdt^1, SDdt^2, \dots, SDdt^p\}$, where $SDdt^i$ contains all the stay places in the i -th trajectory ($1 \leq i \leq p$).

Note that the central coordinate $(x_{S_m^i}, y_{S_m^i})$ of S_m^i is calculated using Equations (2)–(6) (Line 20), and $dd_{S_m^i}$ is the average direction difference within S_m^i (Line 21), where ComDirectDiff is a function used to measure the amount of change in the direction of trajectory (or sub-trajectory). $SLdcs^i$ is a direction-code sequence of the i -th trajectory. In detail, the average value of direction-code differences between each two continuous trajectory segments within a stay place is obtained by calling Function ComDirectDiff, the return value of which is calculated based on Equation (7). Based on the comparison result between $dd_{S_m^i}$ and θ_{drt} , the final stay places are obtained (Lines 22–24).

The time complexity of Algorithm 2 depends on the following: (a) the time to extract the stay places of each trajectory, whose time complexity is $O(p \cdot tn)$; (b) the time to calculate and store the information of each stay place in each trajectory. From Line 21 of Algorithm 2, we know that the direction code sequence of each stay place is scanned to calculate the direction difference using Function ComDirectDiff. Therefore, the time complexity of this part is $O(p \cdot msn \cdot tn)$, where msn is the maximum value of the set $\{sn_1, sn_2, \dots, sn_p\}$, and sn_i represents the number of stay places in T_i . Stay places are derived from time-stamped locations (Lines 3–14), so $msn < tn$. Therefore, the total approximate time complexity of Algorithm 2 is $O(p \cdot tn^2)$. The space required by Algorithm 2 is $p \cdot msn \cdot 7$ because each stay place is recorded using a 7-tuple. Thus, the space complexity of Algorithm 2 is $O(p \cdot msn)$, which is mainly from storing the dataset $SDdt$.

4.4. Congestion Location Detection

4.4.1. Equivalence Classes Partition Based on Timestamps

Road congestion refers to traffic jams caused by a large number of vehicles, disorderly traffic or narrow roads at certain times. The locations visited frequently at different times are not congested locations. Such locations can constitute hot spots. Therefore, to detect road congestion locations, the time attribute must be considered. In this section, the time-stamped locations included in stay places are partitioned according to different timestamp values. For the detection of the same moment, we allow a difference of θ_t seconds before and after, where θ_t is a time threshold. The stay place equivalence class refers to a set consisting of several stay places at approximately the same time.

Figure 5 shows the densest area of stay places at 9:18 on 26 May 2008 in San Francisco. The results are detected by our algorithm based on the dataset $DS1$, which will be introduced in Section 5.2. Each blue point represents a location at that moment. The stay places in similar time periods can reflect that the current locations are in a congested state. Thanks to Bigemap, we can see from Figure 5 that this detected place is the parking lot next to the yellow cab co-op. The day is Monday, and staying in this area at this moment is normal. The detected result is rational and effective.



Figure 5. The densest area of stay places in San Francisco at 9:18 on 26 May 2008. The results are detected based on dataset $DS1$.

The set of stay place equivalence classes divided by timestamps is acquired by Algorithm 3, denoted as ECPT. Stay places whose start time and end time meet the specified range requirements are

divided into the same equivalence class (Lines 9–13). Different equivalence classes are finally sorted by start time (Line 22).

The pseudo code of the algorithm is given as follows:

Algorithm 3. ECPT: Equivalence Classes Partition

Input: $SDdt$ (the set of stay places in each trajectory), p (the number of trajectories), θ_t (the time threshold)
Output: $lscell$ (the set of stay place equivalence classes divided by time)

1. $ln \leftarrow 0;$
2. **for** $i \leftarrow 1$ to p **do**
3. **for** $k \leftarrow 1$ to $|SDdt^i|$ **do**
4. **if** ($st_{S_k^i} == -1$) **then** continue; **end if**
5. $ls \leftarrow \{st_{S_k^i}, et_{S_k^i}, x_{S_k^i}, y_{S_k^i}\};$
6. **for** $j \leftarrow 1$ to p **do**
7. **if** ($j == i$) **then** continue; **end if**
8. **for** $q \leftarrow 1$ to $|SDdt^j|$ **do**
9. **if** ($|st_{S_q^j} - st_{S_k^i}| \leq \theta_t$ and $|et_{S_q^j} - et_{S_k^i}| \leq \theta_t$) **then**
10. $l \leftarrow \{st_{S_q^j}, et_{S_q^j}, x_{S_q^j}, y_{S_q^j}\};$
11. $ls \leftarrow ls \cup l;$
12. $st_{S_q^j} \leftarrow -1;$
13. **end if**
14. **end for**
15. **end for**
16. $st_{S_k^i} \leftarrow -1;$
17. $ln \leftarrow ln + 1;$
18. $lscell^{ln} \leftarrow ls;$
19. **end for**
20. **end for**
21. $lscell \leftarrow \{lscell^1, lscell^2, \dots, lscell^{ln}\};$
22. Sort $lscell$ in ascending order by the start time;
23. **return** $lscell;$

The start and end time of each stay place of each trajectory are compared with those of all the other stay places. The stay places with similar start and end times are partitioned into the same equivalence class (Lines 2–20). Therefore, the time complexity of Algorithm 3 is $O(totalsn^2)$, where $totalsn$ represents the total number of stay places within the trajectory dataset.

4.4.2. Stay Places Clustering Based on Equivalence Classes

Existing methods mainly concentrate on the analysis of separated congestion locations and cannot provide time-ordered congestion sequence in periodic motions. Therefore, after the set of stay place equivalence classes divided by timestamps is obtained, we cluster stay places in each stay place equivalence class according to their central coordinates, to mine the aggregation locations of multiple trajectories during a certain period of time. The clustering process is implemented by Algorithm 4, denoted as ClusterSP. After this algorithm is executed, all the cluster centers and the optimal cluster center of each equivalence class are acquired, and the set of time and central coordinates of stay places belonging to the optimal cluster in each equivalence class is also obtained to be displayed visually on a real map. The notations used in the ClusterSP algorithm are listed in Table 1.

Table 1. Notations used in ClusterSP algorithm.

Notation	Description
$Idxcell^i$	The set of cluster labels in the i -th equivalence class
$Ccell^i$	The set of cluster centers in the i -th equivalence class
$sumDcell^i$	The set of the sum of distances between each point and its center within each cluster in the i -th equivalence class
$lscell^i$	The set of stay places in the i -th equivalence class
$ECts^i$	The two-tuple <start time, end time> in the i -th equivalence class
$EClocs^i$	The set of the central coordinates of each stay place in the i -th equivalence class
$ECcls^i$	The number of clusters in the i -th equivalence class

The pseudo code of ClusterSP algorithm is given as follows:

Algorithm 4. ClusterSP: Clustering Based on Stay Places

Input: $lscell$ (the set of stay place equivalence classes divided by time), K (the number of clusters)

Output: $Ccell$ (the set of cluster centers in each equivalence class), $BestC$ (the set of optimal cluster centers in each equivalence class), $OriBestLocs$ (the set of time and central coordinates of stay places belonging to $BestC$),

```

1.    $eqs \leftarrow$  the number of equivalence classes in  $lscell$ ;
2.   for  $i \leftarrow 1$  to  $eqs$  do
3.      $ECts^i \leftarrow$  time data from Row 1 and Columns 1–2 of the matrix  $lscell^i$ ;
4.      $EClocs^i \leftarrow$  location data from Columns 3–4 of the matrix  $lscell^i$ ;
5.      $[Idx, C, sumD] \leftarrow kmeans(EClocs^i, K)$ ;
6.      $Idxcell^i \leftarrow Idx$ ;
7.      $Ccell^i \leftarrow C$ ;
8.      $sumDcell^i \leftarrow sumD$ ;
9.   end for
10.   $Ccell \leftarrow \{Ccell^1, Ccell^2, \dots, Ccell^{eqs}\}$ ;
11.   $OriBestLocs \leftarrow \emptyset$ ;
12.  for  $i \leftarrow 1$  to  $eqs$  do
13.     $ECcls^i \leftarrow$  the number of clusters in  $EClocs^i$ ;
14.    if ( $ECcls^i == 1$ ) then
15.       $BestC^i \leftarrow <ECts^i, Ccell^i>$ ;
16.       $OriBestLocs \leftarrow OriBestLocs \cup lscell^i$ ;
17.    else
18.       $Minavg \leftarrow \text{Inf}; j_{min} \leftarrow -1$ ;
19.      for  $j \leftarrow 1$  to  $ECcls^i$  do
20.         $num \leftarrow$  the number of locations in the  $j$ -th cluster of  $Idxcell^i$ ;
21.         $avg \leftarrow sumDcell^i(j) / num$ ;
22.        if ( $avg < Minavg$ ) then
23.           $Minavg \leftarrow avg; j_{min} \leftarrow j$ ;
24.        end if
25.      end for
26.       $blocs^i \leftarrow$  the  $j_{min}$ -th center in  $Ccell^i$ ;
27.       $BestC^i \leftarrow <ECts^i, blocs^i>$ ;
28.       $xh \leftarrow \text{find}(Idxcell^i == j_{min})$ ;
29.       $OriBestLocs \leftarrow OriBestLocs \cup lscell^i(xh, :)$ ;
30.    end if
31.  end for
32.   $BestC \leftarrow \{BestC^1, BestC^2, \dots, BestC^{eqs}\}$ ;
33.  return  $Ccell, BestC, OriBestLocs$ ;

```

Lines 1–9 complete the process of clustering based on stay places in each equivalence class, where “kmeans” in Line 5 is a Matlab function used to implement k -means clustering. In the experiments, the parameter K is set as $|lscell^i|/3$ for the i -th stay place equivalence class $lscell^i$. This is set according to the number of stay places within each equivalence class. The experimental results on the four trajectory datasets (which will be introduced in Section 5.2) show that the maximum number of stay places in $lscell$ is 31. Lines 10–31 get the set of optimal cluster centers in each equivalence class and the set of time and central coordinates of stay places belonging to optimal clusters, where ($Idxcell^i == j_{min}$) in Line 28 is a Matlab function used to find the indexes of the elements with value j_{min} in $Idxcell^i$, and $lscell^i(xh,:)$ in Line 29 represents the data of all rows whose row numbers belong to the vector xh .

The time complexity of Algorithm 4 depends on the following: (a) the time taken for clustering within each equivalence class, whose time complexity is $O(eq \cdot Tkmeans)$, where $Tkmeans$ is the number of iterations in k -means clustering; (b) the time taken to calculate the optimal cluster centers, whose time complexity is $O(eq \cdot mECcls)$, where $mECcls$ represents the maximum value of the set $\{ECcls^1, ECcls^2, \dots, ECcls^{eq}\}$.

Figure 6 shows the hierarchical graph of optimal stay-place sequence detection based on the proposed approach. In Figure 6, the symbol Lt_i represents the set of stay places in the i -th time period ($1 \leq i \leq n$). (1) The top layer is the set of detected stay places $SDdt$ extracted using Algorithm 2. (2) The middle layer contains several sets of stay places in different time periods, which are stored in the set of stay place equivalence classes $lscell$ acquired by Algorithm 3. In each equivalence class of $lscell$, stay places are clustered according to their central coordinates. The clustering process is implemented by Algorithm 4. (3) The bottom layer shows the sequence of optimal stay places obtained based on the clustering results within different equivalence classes.

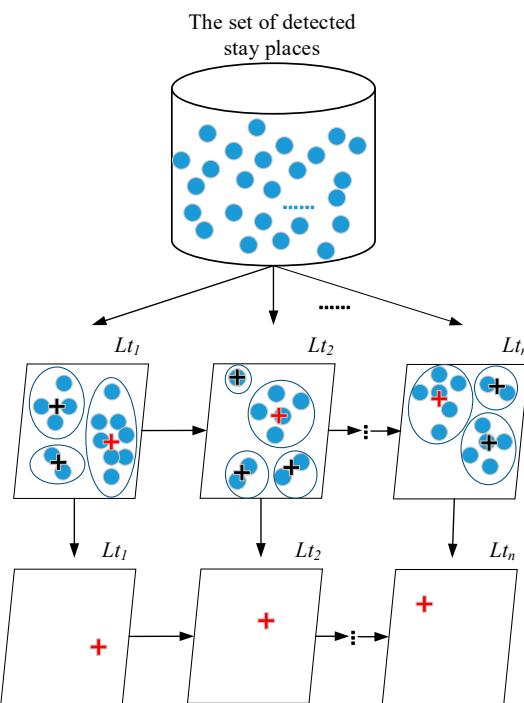


Figure 6. The hierarchical graph of optimal stay-place sequence detection.

5. Experiments

In this section, we perform a set of experiments to evaluate the performances of the proposed algorithms. We first present the experimental settings which consist of the introduction to the experimental environment, and some parameters we selected for the experiments. Then, the datasets and evaluation metrics used in the experiments are introduced. Finally, the visualization of the

experimental results is shown, and the proposed approach is evaluated in terms of average *precision*, average *recall*, average *F* and average *MissRate*.

5.1. Experimental Environment and Parameter Selection

The experiments are conducted with Matlab 8.3 on a PC with Intel (R) Core (TM) 2 Duo CPU 3.7 GHz and 8 GB of RAM. The operating system is Microsoft Windows 7. A real-life dataset is used to conduct our experiments.

According to Algorithms 1–4, our experimental process is specifically arranged as follows: (1) Pre-process the experimental datasets based on the proposed trajectory model. (2) Extract the low-speed points and stay places using Algorithm 1 and Algorithm 2, respectively. (3) Conduct equivalence class partitioning using Algorithm 3. (4) Detect the road congestion locations based on Algorithm 4. (5) Experimental results and related analysis.

The parameters used in Algorithms 1–4 include the speed threshold θ_s , the time threshold θ_t , and the direction difference threshold θ_{drt} . They are set specifically according to the related experimental results, which will be described later.

5.2. Dataset

The real-life dataset of cab moving trajectories collected from San Francisco in the United States [30,31] is used in our experiments. It contains GPS coordinates of approximately 500 cabs in the San Francisco Bay Area from May to June 2008. The locations in this dataset are very fine-grained because the average time interval between two consecutive locations is less than 10 s [31]. The format of each mobility trajectory file is as follows: each line contains latitude, longitude, occupancy, and time, where the occupancy is ignored in our experiments. As the trajectory of a cab over the course of an entire month can hardly be considered a single trajectory, we use the method in literature [7] to pre-process this dataset. In particular, the trajectory data between 25 May at 12:04 and 26 May at 12:04 is extracted by referring to Literature [7]. This dataset has a high concentration of locations. After a trajectory filtering process, we obtain 481 trajectories and 896 locations per trajectory on average. The longest trajectory contains 2483 locations. For ease of description, this dataset is denoted as *DS1*.

For contrast, the other three datasets are extracted from the real-life taxis trajectory dataset. These four datasets are extracted from the same time period of four consecutive weeks. Specifically, (1) the trajectory data between 18 May at 12:04 and 19 May at 12:04 is extracted as dataset *DS2*, which includes 492 trajectories and 967 locations per trajectory on average. The longest trajectory contains 2119 locations; (2) the trajectory data between 1 June at 12:04 and 2 June at 12:04 is extracted as dataset *DS3*, which includes 491 trajectories and 923 locations per trajectory on average. The longest trajectory contains 2030 locations; (3) the trajectory data between 8 June at 12:04 and 9 June at 12:04 is extracted as dataset *DS4*, which includes 499 trajectories and 883 locations per trajectory on average. The longest trajectory contains 2411 locations. The common characteristics of these four datasets are that they all record trajectories from 12:04 on Sunday to 12:04 on Monday.

5.3. Evaluation Metrics

Based on the reference stay points detected by the StayPointDetection algorithm [14,32], we evaluate the performance of the proposed approach in terms of the following statistical metrics [17,33]:

(1) *Precision*: the ratio between the number of correctly detected stay locations and the number of all the detected stay locations:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (8)$$

where *TP* (True Positive) represents the number of stay locations being truly detected, and *FP* (False Positive) represents the number of moving points being falsely detected as stay locations.

(2) *Recall*: the ratio between the number of correctly detected stay locations and the number of the actual stay locations.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (9)$$

where *FN* (False Negative) represents the number of stay locations being falsely detected as moving ones.

(3) *F measure*: the weighted harmonic mean of *precision* and *recall*.

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (10)$$

The range of *F* value is [0,1]. A higher value means that the algorithm has better performance.

(4) *MissRate*: the ratio between the number of undetected stay locations and the number of reference stay locations.

$$\text{MissRate} = \frac{FN}{TP + FN}. \quad (11)$$

In fact, the sum of *Recall* and *MissRate* is 1. The higher the *Recall* value, or the lower the *MissRate* value, the better the performance of the algorithm.

The values of the above metrics are calculated for each trajectory in the dataset. Therefore, for the whole trajectory dataset, the average values of each metric are adopted to evaluate the detection performance of our approach.

5.4. Experimental Results

In this section, both the experimental results and related analysis are given. A set of comparative experiments are firstly conducted to evaluate the performance of the proposed approach. Then, the detected congestion locations are visually displayed.

5.4.1. Detection Accuracy of Stay Points

We use the four metrics mentioned in Section 5.3 to evaluate the detection accuracy of stay points. To better illustrate the extraction results of trajectory stay points, we change the values of the corresponding parameters in the algorithms.

The stay points detected by the StayPointDetection algorithm [14,32] are used as a reference to verify the detection accuracy of our algorithms. The original time-stamped locations contained in the stay places detected by Algorithms 1 and 2 (LPEX and SPEX) are compared with the reference stay points. The extraction of stay places is the basis of our congestion detection approach. It is important to verify the accuracy of the extraction results.

On one hand, we change the values of parameters τ and δ in the StayPointDetection algorithm, where τ is the stay time threshold, and δ is the distance threshold. According to the literature [14], the distance threshold δ is set to 200 meters, and the stay time threshold τ is set to 15 minutes. We vary τ from 200 s to 900 s, and vary δ from 200 m to 900 m. In our proposed method, θ_t is set to 100 s, θ_s is set to 2.22 m/s, and θ_{drt} is set to 7. Figure 7 shows the comparison results of *F*, *Precision*, *Recall*, and *MissRate* values based on the change of parameters in the StayPointDetection algorithm. In Figure 7a, δ is set to 200 m or 400 m, and the average values of all trajectories in *DS1* for each metric are calculated in the case of different τ values. In Figure 7b, τ is set to 300 s or 900 s, and the average values of all trajectories in *DS1* for each metric are calculated in the case of different δ values.

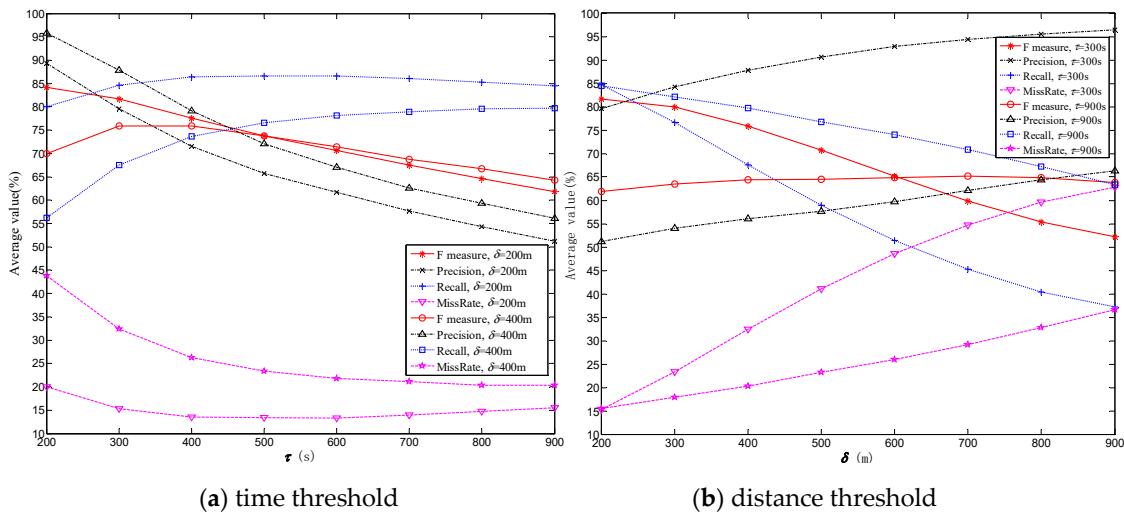


Figure 7. The running results on dataset *DS1*. In this figure, the parameters of the StayPointDetection algorithm are changed.

Meanwhile, we change the values of parameters θ_t , θ_s , and θ_{drt} in our algorithms. Specifically, we vary θ_t from 100 s to 1300 s, vary θ_s from 0.5 m/s to 4 m/s and vary θ_{drt} from 2 to 10. In StayPointDetection algorithm, δ is set to 200 m, τ is set to 300 s. Figure 8 shows the comparison results of *F*, *Precision*, *Recall*, and *MissRate* values based on the change of parameters in the LPEX and SPEX algorithms. In Figure 8a, the average values of all trajectories in *DS1* for each metric are calculated in the case of different θ_t values. In Figure 8b, the average values for each metric are calculated in the case of different θ_s values. In Figure 8c, the average values for each metric are calculated in the case of different θ_{drt} values.

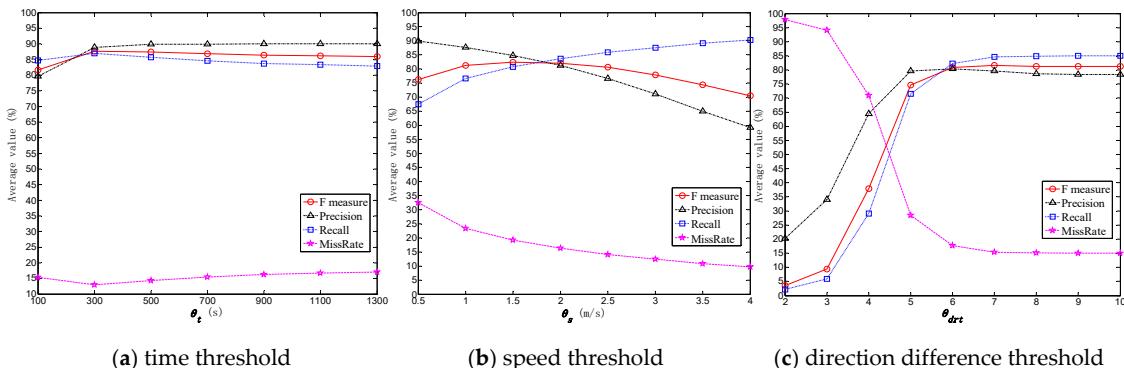


Figure 8. The running results on *DS1*. In this figure, the parameters of LPEX and SPEX algorithms are changed.

As can be seen from Figures 7 and 8, the locations within stay places detected by our algorithm have a high accuracy when referring to the results of the Algorithm StayPointDetection based on dataset *DS1*. The average values of *F* measure are more than 80% under specified parameters. This means that the stay places extracted by our method contain most of the stay points detected by the classic stay point detection algorithm StayPointDetection; the main difference is because of the deletion of stay places that exceed θ_{drt} . Note that the purpose of the two algorithms is not the same. StayPointDetection algorithm is used to detect the stay points that meet the specified time and distance requirement. The proposed algorithm SPEX is used to extract the stay places within which there is no large direction difference between different trajectory segments. Therefore, the detection results have some missing data when compared with reference results, which are reflected by the average *Recall* value and the average *MissRate* value. As shown in Figure 8c, the average *F* value increases as θ_{drt} increases, and the *F* value is substantially stable when θ_{drt} reaches 7. The smaller θ_{drt} is, the more candidate stay places

will be deleted, so the F value will be smaller, and vice versa. However, if θ_{drt} value is too large, the detected stay places most likely belong to the category of SP I instead of SP III. Therefore, we set θ_{drt} to 7 in the experiments.

5.4.2. Visual Display of the Detection Results

Using our approach, the optimal cluster centers of each equivalence class are obtained. Based on the detection results on *DS1* dataset, the most densely distributed areas of congestion are represented in Figure 9. This area mainly concentrates in the northeast part of San Francisco, which is a relatively populous area.

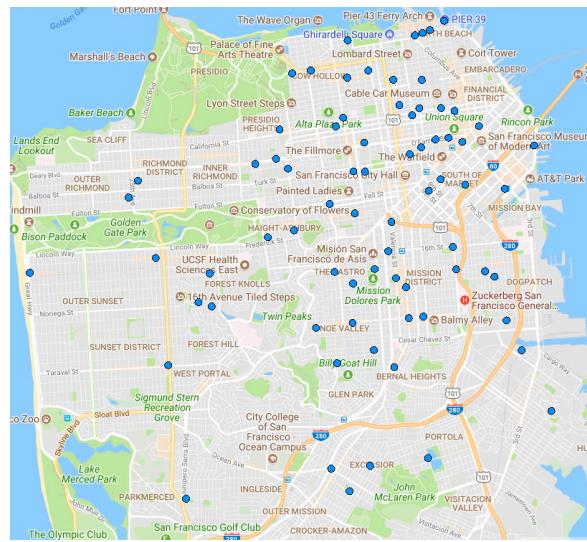


Figure 9. The most densely distributed area of detected stay places. The results are detected based on *DS1* dataset.

Congestion locations are of two types: recurrent ones and non-recurrent ones. The main difference between the two is whether the congestion event at the congestion location is accidental or recurring at a given time. Therefore, the detection of recurrent congestion relies on experiments using datasets with periodic time periods. Next, we use four datasets, *DS1–DS4*, from the same time period for the experiments on congestion location detection and visualize the results.

To facilitate the display of the optimal centers of stay places at different timestamps, we color them in chronological order. The coloring graphs on datasets *DS1–DS4* are shown in Figure 10. The clustering method is *k*-means.

As a comparison, the clustering method of DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is also used to conduct the same experiments. Figure 11 shows the corresponding running results.

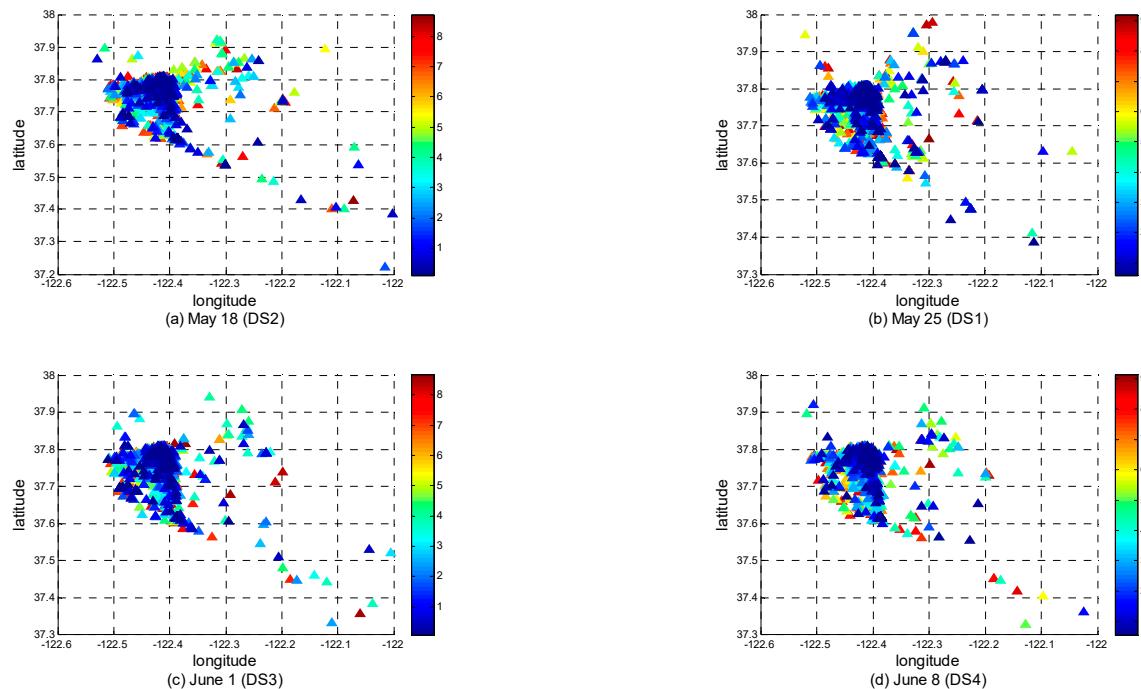


Figure 10. The time-sequential coloring graph of optimal stay centers. The k -means clustering method is used for datasets DS1–DS4.

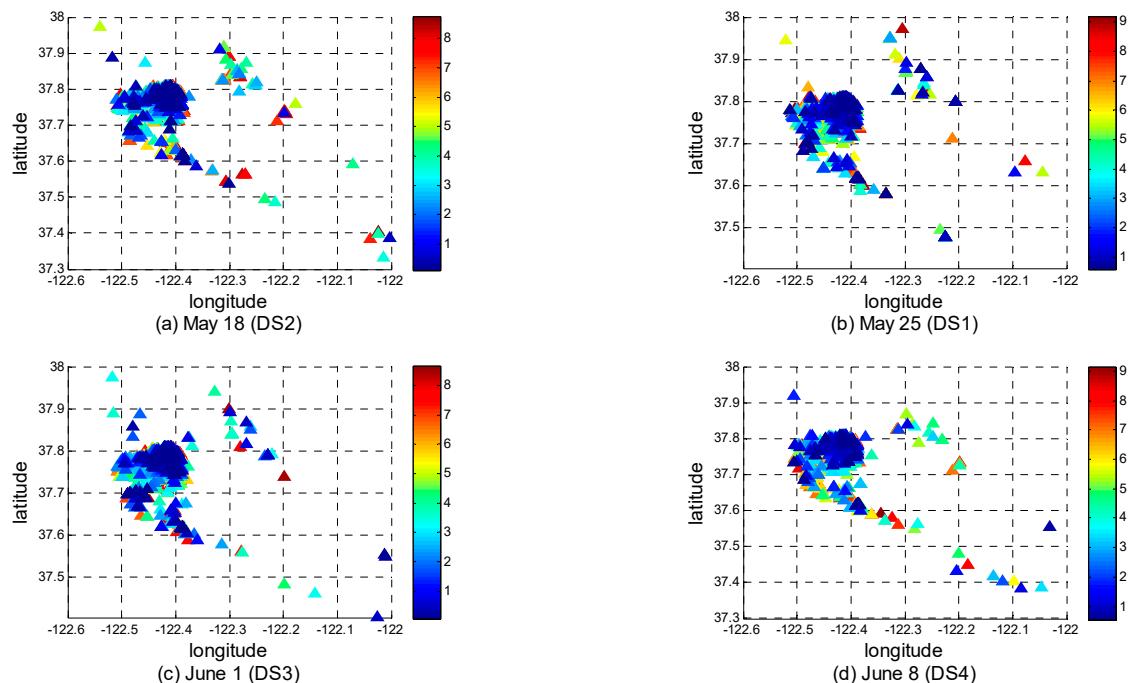


Figure 11. The time-sequential coloring graph of optimal stay centers. The DBSCAN clustering method is used for datasets DS1–DS4.

As can be seen from Figures 10 and 11, every Monday, taxis in San Francisco appear more concentrated in the latitude and longitude range $(37.7, -122.45, 37.8, -122.4)$. The results of the two clustering methods are similar in this respect. Just as with the distribution shown in Figure 9, this area is located in the northeast of San Francisco. Take one optimal center point in the running result of Figure 10a as an example. The time and location information of the optimal stay center is $(1211745933, 1211747557, 37.7832, -122.4345)$, where the first two numbers are, respectively, the start time and end

time of this stay center recorded with the Unix timestamps. After the Unix timestamps are converted to ordinary date and time, the start and end times of this stay place are 13:05:33 and 13:32:37 on 25 May 2008, respectively. The center of this stay place is close to Raymond Kimbell Playground, which is shown in Figure 12. From these detection results, we can identify recurrent congestion locations.

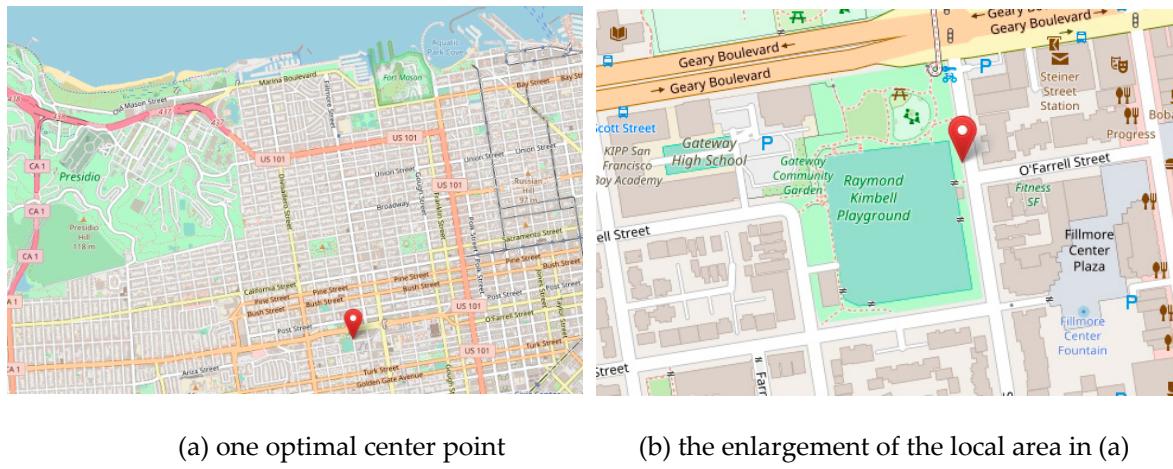


Figure 12. Example of one optimal center point in the running result of Figure 10a. (b) is the enlargement of the local area in (a).

In summary, the centers of congestion locations at different timestamps are obtained based on the detection results, which can be used as a guide for transportation supervision and personal travel route planning in San Francisco.

6. Conclusions

In this paper, we propose a novel approach for detecting road congestion locations based on trajectory stay-place clustering. Before the execution of our algorithms, a trajectory dataset is pre-processed according to Definition 1. There are four steps in this approach. First, the speed status of each point in each trajectory is calculated for obtaining the low-speed point state matrix. Second, the trajectory stay places extraction is implemented based on average direction difference. Third, the time-stamped locations included in the stay places are partitioned into different stay-place equivalence classes according to different timestamps. Finally, we cluster stay places in each equivalence class according to their central coordinates, to mine the aggregation locations of multiple trajectories during a certain period of time. Visual representations and simulated experimental results on real-life cab trajectory datasets show that the proposed approach can obtain effective and reasonable congestion clusters for vehicles data and provide worthwhile services for traffic periodic management system. This achievement is convenient enough to be applied in road planning and may be used in the aggregation analysis of other moving objects in the city. In future research, we plan to integrate the features of POI (Point of Interest) near the road, the drivers' travel purposes, and behavior features to mine deep causation of congestion formation and conduct travel recommendations.

Author Contributions: Conceptualization, Qingying Yu; Data curation, Chuanming Chen; Funding acquisition, Qingying Yu, Yonglong Luo and Xiaoyao Zheng; Methodology, Qingying Yu; Supervision, Yonglong Luo; Validation, Chuanming Chen; Visualization, Qingying Yu; Writing—original draft, Qingying Yu; Writing—review and editing, Yonglong Luo and Xiaoyao Zheng.

Funding: This research was supported by the National Natural Science Foundation of China (Grant Nos. 61702010 and 61672039), the Anhui Provincial Natural Science Foundation (Grant No. 1808085MF172), and the University Natural Science Research Program of Anhui Province (Grant No. KJ2017A327).

Acknowledgments: The authors would like to thank the reviewers for their useful comments and suggestions for this paper. This work was supported by the National Natural Science Foundation of China (Grant Nos. 61702010

and 61672039), the Anhui Provincial Natural Science Foundation (Grant No. 1808085MF172), and the University Natural Science Research Program of Anhui Province (Grant No. KJ2017A327).

Conflicts of Interest: The authors declare no conflict of interest.

References

- Wu, P.; Liu, K.; Hao, S.; Zhang, Q.; Tan, Y. Rapid traffic congestion monitoring based on floating car data. *J. Comput. Res. Dev.* **2014**, *51*, 189–198.
- Fang, H.; Hsu, W.J.; Rudolph, L. Mining user position log for construction of personalized activity map. In *International Conference on Advanced Data Mining and Applications*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 444–452.
- Hu, W.; Li, X.; Tian, G.; Maybank, S.; Zhang, Z. An incremental DPMM-based method for trajectory clustering, modeling, and retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1051–1065. [PubMed]
- Luo, W.; Tan, H.; Chen, L.; Ni, L.M. Finding time period-based most frequent path in big trajectory data. In Proceedings of the ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22–27 June 2013; pp. 713–724.
- Zheng, Y. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* **2015**, *6*, 29. [CrossRef]
- Lv, M.; Chen, L.; Xu, Z.; Li, Y.; Chen, G. The discovery of personally semantic places based on trajectory data mining. *Neurocomputing* **2016**, *173*, 1142–1153. [CrossRef]
- Domingo-Ferrer, J.; Trujillo-Rasua, R. Microaggregation- and permutation-based anonymization of movement data. *Inf. Sci.* **2012**, *208*, 55–80. [CrossRef]
- Cai, Y.; Wang, H.; Chen, X.; Jiang, H. Trajectory-based anomalous behaviour detection for intelligent traffic surveillance. *IET Intell. Transp. Syst.* **2015**, *9*, 810–816. [CrossRef]
- Zheng, K.; Zheng, Y.; Yuan, N.J.; Shang, S.; Zhou, X. Online discovery of gathering patterns over trajectories. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 1974–1988. [CrossRef]
- Kong, X.; Xu, Z.; Shen, G.; Wang, J.; Yang, Q.; Zhang, B. Urban traffic congestion estimation and prediction based on floating car trajectory data. *Futur. Gener. Comput. Syst.* **2016**, *61*, 97–107. [CrossRef]
- Gong, Y.; Chen, E.; Zhang, X.; Ni, L.M.; Zhang, J. Antmapper: An ant colony-based map matching approach for trajectory-based applications. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 390–401. [CrossRef]
- Kami, N.; Enomoto, N.; Baba, T.; Yoshikawa, T. Algorithm for detecting significant locations from raw GPS data. In *Discovery Science*; Springer: Heidelberg, Germany, 2010; pp. 221–235.
- Xiao, H.; Wang, W.J.; Zhang, X. Identifying the stay point using GPS trajectory of taxis. *Appl. Mech. Mater.* **2013**, *353–356*, 3511–3515. [CrossRef]
- Zheng, Y.; Zhou, X. *Computing with Spatial Trajectories*; Springer: New York, NY, USA, 2011.
- Anbaroglu, B.; Heydecker, B.; Cheng, T. Spatio-temporal clustering for non-recurrent traffic congestion detection on urban road networks. *Transp. Res. Part C Emerg. Technol.* **2014**, *48*, 47–65. [CrossRef]
- Wang, Y.; Cao, J.; Li, W.; Gu, T.; Shi, W. Exploring traffic congestion correlation from multiple data sources. *Pervasive Mob. Comput.* **2017**, *41*, 470–483. [CrossRef]
- D’Andrea, E.; Marcelloni, F. Detection of traffic congestion and incidents from GPS trace analysis. *Expert Syst. Appl.* **2017**, *73*, 43–56. [CrossRef]
- Liu, S.; Liu, Y.; Ni, L.; Li, M.; Fan, J. Detecting crowdedness spot in city transportation. *IEEE Trans. Veh. Technol.* **2013**, *62*, 1527–1539. [CrossRef]
- Wang, Q.; Wan, J.; Yuan, Y. Locality constraint distance metric learning for traffic congestion detection. *Pattern Recognit.* **2018**, *75*, 1339–1351. [CrossRef]
- Yuan, J.; Zheng, Y.; Zhang, L.; Xie, X.; Sun, G. Where to find my next passenger? In Proceedings of the 13th International Conference on Ubiquitous Computing, Beijing, China, 17–21 September 2011; pp. 109–118.
- Huo, Z.; Meng, X.; Hu, H.; Huang, Y. You can walk alone: Trajectory privacy-preserving through significant stays protection. In *International Conference on Database Systems for Advanced Applications*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 351–366.
- Do, T.M.T.; Gatica-Perez, D. Contextual conditional models for smartphone-based human mobility prediction. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, Pittsburgh, PA, USA, 5–8 September 2012; p. 163.

23. Li, Q.; Zheng, Y.; Xie, X.; Chen, Y.; Liu, W.; Ma, W. Mining user similarity based on location history. In Proceedings of the ACM Sigspatial International Conference on Advances in Geographic Information Systems, Irvine, CA, USA, 5–7 November 2008; p. 34.
24. Yuan, N.J.; Zheng, Y.; Zhang, L.; Xie, X. T-Finder: A recommender system for finding passengers and vacant taxis. *IEEE Trans. Knowl. Data Eng.* **2013**, *25*, 2390–2403. [[CrossRef](#)]
25. Pavan, M.; Mizzaro, S.; Scagnetto, I.; Beggiato, A. Finding Important Locations: A Feature-Based Approach. In Proceedings of the 16th IEEE International Conference on Mobile Data Management, Pittsburgh, PA, USA, 15–18 June 2015; pp. 110–115.
26. Stylianou, G. Stay-point Identification as Curve Extrema. *arXiv* **2017**, arXiv:1701.06276v1.
27. Cao, X.; Cong, G.; Jensen, C.S. Mining significant semantic locations from GPS data. *Proc. Vldb Endow.* **2010**, *3*, 1009–1020. [[CrossRef](#)]
28. An, S.; Yang, H.; Wang, J.; Cui, N.; Cui, J.; An, S.; Yang, H.; Wang, J.; Cui, N.; Cui, J. Mining urban recurrent congestion evolution patterns from GPS-equipped vehicle mobility data. *Inf. Sci.* **2016**, *373*, 515–526. [[CrossRef](#)]
29. Yu, Q.; Luo, Y.; Chen, C.; Wang, X. Trajectory outlier detection approach based on common slices sub-sequence. *Appl. Intell.* **2018**, *48*, 2661–2680. [[CrossRef](#)]
30. Piorkowski, M.; Sarafijanovic-Djukic, N.; Grossglauser, M. CRAWDAD dataset epfl/mobility(v. 2009-02-24). 2009. Available online: <http://crawdad.org/epfl/mobility/20090224> (accessed on 16 April 2019).
31. Piorkowski, M.; Sarafijanovic-Djukic, N.; Grossglauser, M. A parsimonious model of mobile partitioned networks with clustering. In Proceedings of the 1st International Conference on Communication Systems and NETworks, Bangalore, India, 5–10 January 2009; pp. 1–10.
32. Zheng, Y.; Zhang, L.; Ma, Z.; Xie, X.; Ma, W.-Y. Recommending friends and locations based on individual location history. *ACM Trans. Web.* **2011**, *5*, 1–44. [[CrossRef](#)]
33. Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*; Morgan Kaufmann: San Francisco, CA, USA, 2013.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).