

# Time-Dependent Popular Routes Based Trajectory Outlier Detection

Jie Zhu<sup>1</sup>, Wei Jiang<sup>1</sup>, An Liu<sup>1,2</sup>, Guanfeng Liu<sup>1,2</sup>, and Lei Zhao<sup>1,2</sup>(✉)

<sup>1</sup> School of Computer Science and Technology, Soochow University, Suzhou, China  
zjcomeon@gmail.com, jwpker@outlook.com

<sup>2</sup> Collaborative Innovation Center of Novel Software Technology  
and Industrialization, Nanjing, Jiangsu, China  
{anliu,gfliu,zhaol}@suda.edu.cn

**Abstract.** With the rapid proliferation of the GPS-equipped devices, a myriad of trajectory data representing the mobility of the various moving objects in two-dimensional space have been generated. In this paper, we aim to detect the anomalous trajectories from the trajectory dataset and propose a novel time-dependent popular routes based algorithm. In our algorithm, spatial and temporal abnormalities are taken into consideration simultaneously to improve the accuracy of the detection. For each group of trajectories with the same source and destination, we firstly design a time-dependent transfer graph and in different time period, we can obtain the top-k most popular routes as reference routes. For a pending inspecting trajectory in this time period, we will label it as an outlier if has a great difference with the selected routes in both spatial and temporal dimension. To quantitatively measure the “difference” between a trajectory and a route, we propose a novel time-dependent distance measure which is based on Edit distance in both spatial and temporal domain. The comparative experimental results with two famous trajectory outlier detection methods TRAOD and IBAT on real dataset demonstrate the good accuracy and efficiency of the proposed algorithm.

**Keywords:** Outlier detection · Time-dependent popular route · Trajectory pattern mining

## 1 Introduction

In recent years, the booming development of GPS-equipped portable devices has helped us gathering a huge amount of trajectory data. According to a report of a

---

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61073061, 61003044, 61232006, and 61303019, the Natural Science Foundation of Jiangsu Province of China under Grant No. SBK2015021685, Jiangsu Provincial Department of Education of China under Grant No. 12KJB520017, the Doctoral Fund of Ministry of Education of China under Grant No. 20133201120012, and Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu, China.

data research organization in China, there are about 66,000 taxis in Beijing and about 1,900,000 passengers each day. Each carry generates one trajectory and there are about 69 million trajectories in one single year. Such a big dataset can help us understanding the cabbies' driving behavior, the city's traffic condition and so on. On this background, extensive researchers are encouraged in trajectory pattern mining, such as life pattern mining [1, 2], popular routes discovering [3, 4], transportation mode mining [5].

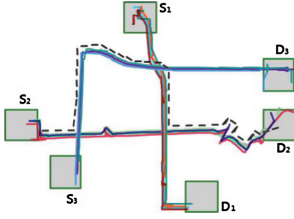
Trajectory outlier detection (TOD) is also a popular research topic in trajectory pattern mining. According to J. Han *et al.* [6], an outlier means a data object that is grossly different from or inconsistent with the remaining set of data. The trajectory outlier means a trajectory that has a great difference with most other trajectories in terms of some similarity metric.

Some TOD algorithms have been proposed. Each algorithm addresses certain aspects of abnormality. Among these TOD algorithms, the first impressive method is TRAOD (TRAjectory Outlier Detection) [7], which firstly splits a trajectory into many trajectory partitions and then compares each trajectory partition with its neighbors to determine whether it is an outlying portion or not. The main advantage of TRAOD lies in the ability to detect outlying sub-trajectories. But because of its sub-trajectory detection strategy, TRAOD has a high time complexity of  $O(n^2)$ . Moreover, the detected result of TRAOD may be influenced by irrelated trajectories because it detects outliers in the whole dataset, as shown in Fig. 1. Recent years, another impressive method is IBAT (Isolation Based Anomalous Trajectory detection) [8]. IBAT focuses on the test trajectory and tries to separate it from the reset trajectories by randomly selecting points solely from the test trajectory. IBAT is more efficient than TRAOD because IBAT does not need to partition the trajectory and the time complexity is  $O(n)$ . But IBAT has a same insufficiency with TRAOD: both of them do not have enough attention on the travel time (departure time, arrival time and ongoing time). TRAOD does not take the time constraint into account and IBAT just assumes the travel time of the trajectories to be detected are in the same time range but there is no in-depth analysis in IBAT.

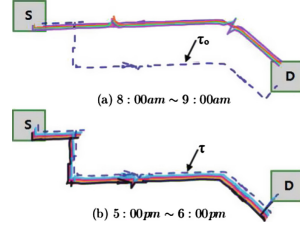
Taking the travel time into account can ensure more accurate detection result. Figure 2 shows an example of two groups of trajectories between two areas in different time.  $\tau_o$  and  $\tau_n$  are two trajectories that walk the same path. But  $\tau_o$  is an outlier while  $\tau_n$  is not because traffic condition changes over time. In other words, outliers' pattern is not static and usually changes with the time. To detect the time-dependent outliers, this paper proposes a novel TOD algorithm called *time-dependent popular routes based trajectory outlier detection* (TPRO).

TPRO detects outliers with the help of the popular routes. The popular routes represent the most trajectories' pattern, so it is a reasonable solution to detect outliers based on the popular routes. As we mentioned above, TPRO focuses on detecting the time-dependent outliers. So time-dependent popular routes are involved to achieve this goal.

TPRO does not partition the trajectories because when facing with a large dataset, efficiency is the first priority while sub-trajectory detection is time-consuming. Given a trajectory dataset, in order to eliminate the influence of



**Fig. 1.** A set of trajectories where  $S_i$  is the source area and  $D_i$  is the destination area. The  $S_2 \rightarrow D_2$  dashed curve is actually a trajectory outlier. But each subpart of it has enough closed neighbors because of being deceived by  $S_1 \rightarrow D_1$  and  $S_3 \rightarrow D_3$  trajectories. TRAOD cannot identify this kind of outlier.



**Fig. 2.** Two groups of trajectories which start from  $S$  and end at  $D$  in different time.  $\tau_o$  is an outlier in 8 : 00am ~ 9 : 00am because it has a great difference with other trajectories during this time. But the traffic condition changes when 5 : 00pm ~ 6 : 00pm.  $\tau_n$ , walking the same path with  $\tau_o$ , is a normal trajectory.

irrelevant trajectories, TPRO divides trajectories with the same source and destination (we call them relevant trajectories) into the same group. Then the dataset can be divided into many groups and detection is token group by group. During the detection, if a trajectory has a great difference with the popular routes during its travel time, this trajectory is classified as an outlier.

Despite that the meaning of the outlier is easy to understand, it is nontrivial to detect outlier based on the time-dependent popular routes. There are mainly two challenges in TPRO:

1. Each trajectory will be compared with its corresponding popular routes to judge whether it is an outlier or not. So given a trajectory (assume its departure time is  $t_s$  and arrival time is  $t_d$ ), TPRO should efficiently retrieve the corresponding popular routes during the time of  $t_s \sim t_d$ .
2. When calculating the difference between a trajectory and its corresponding popular routes, not only the spatial info but also the temporal info (departure time, arrival time and ongoing time) should be taken into account.

In response to the first challenge, a time-dependent transfer graph is constructed in TPRO. This graph records how many trajectories have passed through each road in different time. With the help of the time-dependent transfer graph, TPRO can efficiently retrieve the top- $k$  most popular routes in a user specified time range. And TPRO also puts forward the time-dependent edit distance to address the second challenge. The time-dependent edit distance not only takes the spatial distance into account but also considers the temporal distance.

The main contributions of this paper are as follows:

1. This paper presents a time-dependent popular routes based trajectory outlier detection algorithm, which takes both spatial and temporal abnormality into consideration and gives us a new solution in trajectory outlier detection.
2. We put forward an efficient popular routes query method in TPRO, which can efficiently retrieve the popular routes during a user specified time range.
3. We provide a real trajectory dataset in which the outliers have been labelled by user study.

The rest of this paper is organized as follows. A formal definition of our problem is given in Sect. 2. Section 3 gives a detailed statement of our solution and Sect. 4 shows our experiment's result. Section 5 gives a brief introduction of the related work. At last, a conclusion is given in Sect. 6.

## 2 Problem Definition

This part presents some prior definitions and gives a formal definition of the problem this paper focuses on.

**Definition 1** (*Raw Trajectory*). A raw trajectory  $\tilde{\tau}$  is a time-ordered sequence of sampled points:  $\tilde{\tau} = (\tilde{p}_1, \tilde{p}_2, \tilde{p}_3, \dots, \tilde{p}_x)$ . Each sampled point  $\tilde{p}_i$  is represented by  $\langle \tilde{l}_i, \tilde{t}_i \rangle$  where  $\tilde{l}_i$  is a geographic coordinate and  $\tilde{t}_i$  is the sampling time.

It is hard to find a common path from a group of raw trajectories because of the discrete sampled points. So this paper preprocesses the dataset and map each raw trajectory into the road network to get a mapped continuous trajectory.

**Definition 2** (*Road Network*). A road network is a directed graph  $G = (V, E)$  where  $V$  is a set of vertices representing road intersections and  $E$  is a set of edges representing road segments.

We use  $v_i$  to represent a certain vertex in  $G$ . If  $v_i$  and  $v_j$  are two endpoints of a certain edge, then we have  $\varphi(v_i, v_j) = 0$ . If the edge's direction is  $v_i \rightarrow v_j$ , then it can be denoted as  $e_i^j$ . Otherwise, the edge can be denoted as  $e_i^j$ .

**Definition 3** (*Mapped Trajectory*). A mapped trajectory  $\tau$  is a sequence of time-ordered road network locations. It can be denoted as  $\tau = (p_1, p_2, p_3, \dots, p_m)$ . Each road network location  $p_i$  is represented as  $\langle v_i, t_i \rangle$  where  $v_i$  a certain vertex in the road network and for all  $i \in \{1, 2, 3, \dots, m-1\}$  that  $\varphi(v_i, v_{i+1}) = 0$ . And  $t_i$  is the time  $\tau$  passes  $v_i$ .

Henceforth, we will only deal with the mapped trajectories. So for simplicity, we will drop the *mapped* qualifier. Thus trajectory in the rest of the article is short for mapped trajectory. After giving a definition of the trajectory, the time-dependent route is defined as follows.

**Definition 4** (*Time-Dependent Route*). A  $v_1 \rightarrow v_m$  time-dependent route is denoted as  $\gamma = (pf_1, pf_2, pf_3, \dots, pf_m)$  and each  $pf_i \in \gamma$  is represented as  $\langle v_i, \bar{t}_i, freq_i \rangle$  where  $v_i$  represents a certain vertex in the road network and for all  $i \in \{1, 2, 3, \dots, m-1\}$  that  $\varphi(v_i, v_{i+1}) = 0$ . Meanwhile,  $freq_i$  means how many trajectories have pass through  $v_i$  and  $\bar{t}_i$  is the average pass time.

For simplicity, the *time-dependent* qualifier will be dropped and route is short for time-dependent route in the rest of this paper. After giving a definition of the trajectory and the route, trajectory route distance function is put forward to indicate the difference degree between a trajectory and a route.

**Definition 5** (*Trajectory Route Distance Function*). A trajectory route distance function  $\delta(\tau, \gamma)$  is a formula that can give a difference score between  $\tau$  and  $\gamma$ .

Based on above definitions, we give a formal definition of the outlier and the problem this paper focuses on next.

**Definition 6** (*Outlier*). Given a trajectory  $\tau$ , a route set  $R = \{\gamma_1, \gamma_2, \dots, \gamma_k\}$ , a trajectory route distance function  $\delta$  and an anomalous score threshold  $\theta$ , we can calculate the trajectory's anomalous score

$$s_\tau = \sum_{i=1}^k w_{\gamma_i} \cdot \delta(\tau, \gamma_i) \quad (1)$$

where  $w_{\gamma_i}$  is the popularity weight of  $\gamma_i$  among the route set  $R$ . If  $s_\tau > \theta$ , then we say that  $\tau$  is a  $\theta$ -outlier on  $R$  and  $\delta$ .

**Problem:** Given a trajectory dataset  $T$ , a route distance function  $\delta$  and an anomalous score threshold  $\theta$ , we need to get a trajectory set  $T' = \{\tau_1, \tau_2, \dots, \tau_n\}$  that satisfies: for all  $\tau_i \in T'$ ,  $\tau_i$  is a  $\theta$ -outlier on its corresponding popular routes and  $\delta$ .

### 3 TPRO Algorithm

This section introduces how TPRO solves the problem proposed above. Given a trajectory dataset, to eliminate influence of irrelevant trajectories, TPRO first divides the trajectories into the many groups according to their source and destination. Then after trajectory grouping, the detection is taken for each group respectively. In each group, we firstly construct a time-dependent transfer graph from the trajectories. Then with the help of this graph, the time-dependent popular routes querying can be more efficient. At last, we use a time-dependent edit distance based trajectory route distance function to judge whether a trajectory is an outlier or not.

### 3.1 Dataset Grouping

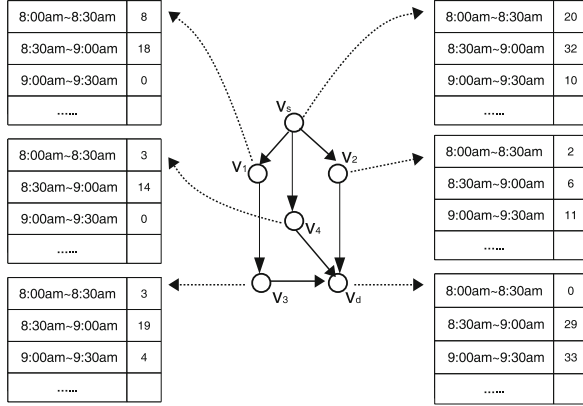
The source vertex and destination vertex of a trajectory  $\tau$  is represented as  $\tau.s$  and  $\tau.d$ . If we adopt the strategy that only trajectories starting at same vertex and ending at same vertex can be gathered into one group, we will find that each group has few trajectories. So we put forward the grid-equal-to relation to enlarge the particle size of source area and destination area.

**Definition 7** (*Grid-Equal-To Relation*). Given two number  $m, n$ , we can split the road network  $G$  into  $m \times n$  size-equal grids. For two vertices  $v_i, v_j$ , if  $v_i$  and  $v_j$  fall into the same grid, then we say that  $v_i$  is  $m$ - $n$ -grid-equal-to  $v_j$ . It can be denoted as  $o(G, m, n, v_i, v_j) = 1$ .

For two certain trajectories  $\tau_i$  and  $\tau_j$ , after given the grid number  $m$  and  $n$ , if  $o(G, m, n, \tau_i.s, \tau_j.s) = 1$  and  $o(G, m, n, \tau_i.d, \tau_j.d) = 1$ , they will be divided into the same group.

### 3.2 Construction of Time-Dependent Transfer Graph

After a certain group of trajectories with the same source and destination are mapped into the road network, we can get a subgraph of the road network (Fig. 3 shows an example of this subgraph). And for each vertex in this subgraph, we use a vertex frequency table (i.e. the table beside each vertex in Fig. 3) to record how many trajectories have pass through each vertex in different time range. This subgraph is called the time-dependent transfer graph (TTG).



**Fig. 3.** An example of TTG.  $v_s$  is the source and  $v_d$  is the destination. Each table beside the vertex  $v_i$  is called vertex frequency table of  $v_i$ .

Form Fig. 3, we can see that the vertex frequency tables in TTG are in the same time interval (30 min in this example). This time interval is called TTG time interval and is denoted as  $\Delta t$ .

With the help of these vertex frequency tables, we can easily estimate how many trajectories have pass through a certain vertex during a user specified time range. For example, we can infer that there are 26 trajectories (8 trajectories during  $8 : 00am \sim 8 : 30am$  and 18 trajectories during  $8 : 30am \sim 9 : 00am$ ) have passed through  $v_1$  during  $8 : 00am \sim 9 : 00am$ . In some cases, the use specified time range does not fully cover the vertex frequency table time ranges. Such as, what if we wan to know that how many trajectories have passed through  $v_1$  during  $8 : 10am \sim 9 : 00am$ . From the TTG, we can know that there are 18 trajectories have passed through  $v_1$  during  $8 : 30am \sim 9 : 00am$ , but we cannot infer how many trajectories during  $8 : 10am \sim 8 : 30am$  directly. In such situation, we multiply the trajectories number by the proportion of the covered time range. Thus, the trajectories during  $8 : 10am \sim 8 : 30am$  is

$$8 \times \frac{8 : 30am - 8 : 10am}{8 : 30am - 8 : 00am} = 8 \times \frac{20 \text{ min}}{30 \text{ min}} \approx 5$$

Obviously, more smaller the TTG time interval is, more accurate the inferred number is. But the space cost and time cost will increase.

And from the TTG in Fig. 3, we can also infer the average pass time of a certain vertex during a user specified time range. For example, there are 5 and 18 trajectories have passed through  $v_1$  during  $8 : 10am \sim 8 : 30am$  and  $8 : 30am \sim 9 : 00am$  respectively. So the average pass time during  $8 : 10am \sim 9 : 00am$  is

$$\frac{5 \times \frac{8:10am+8:30am}{2} + 18 \times \frac{8:30am+9:00am}{2}}{5 + 18} \approx 8 : 40am$$

### 3.3 Retrieving Time-Depended Popular Route

For a trajectory  $\tau$  to be tested, TPRO compares it with the popular routes during  $t_s \sim t_d$  ( $t_s$  represents the departure time and  $t_d$  represents the arrival time) to judge if it is an outlier. So this paragraph explains how to query the time-dependent popular routes with the help of TTG.

Assume that  $t_s = 8 : 00am$  and  $t_d = 9 : 00am$ , we should find the top-k most popular routes during this time. First of all, we can traverse the TTG and calculate each vertex's trajectories number and the average pass time during  $t_s \sim t_d$ . Then all possible routes during  $t_s \sim t_d$  are got as follows:

- $\gamma_1 = (\langle v_s, 8 : 33am, 52 \rangle, \langle v_1, 8 : 36am, 26 \rangle, \langle v_3, 8 : 41am, 22 \rangle, \langle v_d, 8 : 45am, 29 \rangle)$
- $\gamma_2 = (\langle v_s, 8 : 33am, 52 \rangle, \langle v_4, 8 : 40am, 17 \rangle, \langle v_d, 8 : 45am, 29 \rangle)$
- $\gamma_3 = (\langle v_s, 8 : 33am, 52 \rangle, \langle v_2, 8 : 38am, 8 \rangle, \langle v_d, 8 : 45am, 29 \rangle)$

Now that all routes have been got, we should judge which route is more popular. Inspired by Luo *et al.* [9], the route popularity and more-popular-than relation are proposed as follows.

**Definition 8** (*Route Popularity*). The popularity of a certain route  $\gamma = (pf_1, pf_2, pf_3, \dots, pf_m)$  can be represented as an ordered frequency sequence:  $\rho_\gamma = (freq_{j_1}, freq_{j_2}, freq_{j_3}, \dots, freq_{j_m})$ , where:

1.  $\{freq_{j_1}, freq_{j_2}, freq_{j_3}, \dots, freq_{j_m}\} \Leftrightarrow \{pf_1.freq, pf_2.freq, pf_3.freq, \dots, pf_m.freq\}$
2.  $freq_{j_1} \geq freq_{j_2} \geq \dots \geq freq_{j_m}$

**Definition 9** (*More-Popular-Than Relation*). For two routes  $\gamma$  and  $\gamma'$ , assume their popularity sequences are  $\rho_\gamma = (freq_{j_1}, freq_{j_2}, freq_{j_3}, \dots, freq_{j_m})$  and  $\rho_{\gamma'} = (freq'_{j_1}, freq'_{j_2}, freq'_{j_3}, \dots, freq'_{j_n})$ . If one of the following statements holds:

- $\rho_\gamma$  is prefix of  $\rho_{\gamma'}$ ,
- or there exists a number  $q \in \{1, 2, 3, \dots, \min(m, n)\}$  such that:
  1.  $freq_{j_x} = freq'_{j_x}$  for all  $x \in \{1, 2, 3, \dots, q-1\}$ , if  $q > 2$ .
  2.  $freq_{j_q} > freq'_{j_q}$

then we say  $\gamma$  is more-popular-than  $\gamma'$ , denoted as  $\gamma \succeq \gamma'$ .

According to Definitions 8 and 9, we have that  $\rho_{\gamma_1} = (52, 29, 26, 22)$ ,  $\rho_{\gamma_2} = (52, 29, 17)$  and  $\rho_{\gamma_3} = (52, 19, 8)$ . Obviously,  $\rho_{\gamma_1} \succeq \rho_{\gamma_2} \succeq \rho_{\gamma_3}$ . It means that  $\gamma_1$  is more popular than  $\gamma_2$  and  $\gamma_2$  is more popular than  $\gamma_3$ . Assume that  $k = 2$ , then the top-k most popular routes during 8:00am ~ 9:00am are  $\gamma_1$  and  $\gamma_2$ .

Luo *et al.* has proved that the selected popular routes by this method satisfy three key properties: suffix-optimal (i.e., any suffix of the popular route is also popular), length-insensitive (i.e., popular does not mean the shorter/longer the better), and bottleneck-free (i.e., popular routes should not contain infrequent vertices or edges) in [9].

### 3.4 Outlier Detection

After the top-k popular routes got, we compare the trajectory with each popular route. As we known, edit distance can represent two sequences' difference degree. But trajectory (or route) is not just vertex sequence, it also carries the temporal information. So we propose a time-dependent edit distance based trajectory route distance function to handle this problem.

Assume  $\tau_{-1} = (p_1, p_2, p_3, \dots, p_{m-1})$  is a sub-trajectory of  $\tau = (p_1, p_2, p_3, \dots, p_{m-1}, p_m)$  after remove the last point  $p_m$ . And  $\gamma_{-1} = (pf_1, pf_2, pf_3, \dots, pf_{n-1})$  is prefix of  $\gamma = (pf_1, pf_2, pf_3, \dots, pf_{n-1}, pf_n)$  after remove the last tuple  $pf_n$ . The trajectory route distance function in TPRO is defined as a recursive equation:

$$\delta(\tau, \gamma) = \text{Min} \begin{cases} \delta(\tau_{-1}, \gamma) + \text{delete\_cost}(p_m) \\ \delta(\tau, \gamma_{-1}) + \text{delete\_cost}(pf_n) \\ \delta(\tau_{-1}, \gamma_{-1}) + \text{replace\_cost}(p_m, pf_n) \end{cases} \quad (2)$$

where

$$\begin{aligned} \text{delete\_cost}(p_m) &= \begin{cases} 0.5, & m < 2 \text{ or } p_m.v \neq p_{m-1}.v \\ 0, & \text{otherwise} \end{cases} \\ &+ \begin{cases} 0.5, & m < 2 \text{ or } |p_m.t - p_{m-1}.t| > \Delta t \\ 0, & \text{otherwise} \end{cases} \end{aligned} \quad (3)$$



$$\begin{aligned} delete\_cost(pf_n) = & \begin{cases} 0.5, & n < 2 \text{ or } pf_n.v \neq pf_{n-1}.v \\ 0, & otherwise \end{cases} \\ & + \begin{cases} 0.5, & n < 2 \text{ or } |pf_n.\bar{t} - p_{n-1}.\bar{t}| > \Delta t \\ 0, & otherwise \end{cases} \end{aligned} \quad (4)$$

$$replace\_cost(p_m, pf_n) = \begin{cases} 1, & p_m.v \neq pf_n.v \\ 0, & otherwise \end{cases} + \begin{cases} 1, & |p_m.t - pf_n.\bar{t}| > \Delta t \\ 0, & otherwise \end{cases} \quad (5)$$

If there is only one vertex in  $\tau$  (or  $\gamma$ ), which means that  $m = 1$  (or  $n = 1$ ), then we have that  $\tau_{-1} = \phi$  (or  $\gamma_{-1} = \phi$ ). Assume that we use  $\tau.len$  and  $\gamma.len$  to represent the number of vertices of  $\tau$  and  $\gamma$ , then these two initial conditions in this recursive equation are

$$\delta(\tau, \phi) = \tau.len; \quad (6)$$

$$\delta(\phi, \gamma) = \gamma.len; \quad (7)$$

From Eqs. 3, 4 and 5, we can see that the *delete\_cost* or the *replace\_cost* can be broken down into the spatial cost and the temporal cost. If two vertices are different, the spatial cost is 0.5 (delete) or 1 (replace). Otherwise, the spatial cost is 0. When calculating the temporal cost, we have the aid of the TTG time interval  $\Delta t$  in Subsect. 3.2. If the time lag is larger than  $\Delta t$ , the temporal cost is 0.5 (delete) or 1 (replace). Otherwise, the temporal cost is 0.

The pseudo code of the time-dependent edit distance based trajectory route distance function is shown in Algorithm 1.

---

**Algorithm 1.** Trajectory Route Distance Function

---

**Input:** a trajectory  $\tau$ , a route  $\gamma$

**Output:** Distance between  $\tau$  and  $\gamma$

```

1: DECLARE int  $DP[0..\tau.len][0..\gamma.len]$ 
2: for  $i := 0$  to  $\tau.len$  do
3:    $DP[i][0] = i;$ 
4: end for
5: for  $j := 0$  to  $\gamma.len$  do
6:    $DP[0][j] = j;$ 
7: end for
8: for  $i = 1$  to  $\tau.len$  do
9:   for  $j = 1$  to  $\gamma.len$  do
10:     $DP[i][j] = \text{minimum}(\$ 
11:       $DP[i-1][j] + delete\_cost(\tau.p_i),$ 
12:       $DP[i][j-1] + delete\_cost(\gamma.pf_j),$ 
13:       $DP[i-1][j-1] + replace\_cost(\tau.p_i, \gamma.pf_j)$ 
14:     $);$ 
15:   end for
16: end for
17: return  $DP[\tau.len][\gamma.len];$ 

```

---

The trajectory route distance function can give a difference score between a trajectory and a popular route. But in most cases, there are more than one popular route between two areas. But it does not mean that each popular route has the same popularity degree. So we propose the popularity weight to represent how popular a route is among a set of routes.

**Definition 10** (*Popularity Weight*). Assume there is a route set  $R = \{\gamma_1, \gamma_2, \gamma_3, \dots, \gamma_k\}$  and for each  $\gamma_i = (pf_1, pf_2, pf_3, \dots, pf_m) \in R$ , we have

$$\gamma_i.sum = \sum_{a=1}^m \gamma_i.pf_a.freq \quad (8)$$

then the popularity weight of  $\gamma_i$  can be represented as

$$w_{\gamma_i} = \frac{\gamma_i.sum}{\sum_{b=1}^k \gamma_b.sum} \quad (9)$$

### 3.5 Time Complexity

The overall pseudo code of TPRO has been shown in Algorithm 2. From the pseudo code, we can see that the time complexity of TPRO is  $O(k \cdot n)$  where  $k$  is the number of popular routes used to detect outliers and  $n$  is the number of trajectories in the dataset. In most cases,  $k$  is a small number (less than 10), so we can use approximation  $O(n)$  for the time complexity.

## 4 Experiment Result

This section gives an exhibition of our experiment and the result. The first subsection gives an introduction to the experiment dataset and environment setting. The second subsection elaborates which criteria we use to evaluate our algorithm. Finally, we give an analysis on the experiment result.

### 4.1 Experiment Setting

The experiment is taken under a real-world dataset which contains 412,032 trajectories. This dataset is collected from around 10,700 taxis in BeiJing in 2012. We pick up about 1,300 trajectories from the dataset and asked volunteers to manually label whether each trajectory is abnormal or not. This labeled dataset is used to evaluate the accuracy of TPRO.

The road network in our experiment contains about 165,000 vertices and 226,000 edges. And the road network is split into  $120 \times 130$  grids in the grouping step<sup>1</sup>. Each grid's size is about  $1.5 \text{ km} \times 1.5 \text{ km}$ .

Our algorithm is implemented in cpp. The machine we use to accomplish the experiment has a quadcore Inter Core i5 CPU (3.2 GHz) and 8G memory. The operating system is Linux 3.13.0 x 86\_64 and the compiler is g++ 4.8.2.

<sup>1</sup> That's to say  $m$  is set to 120 and  $n$  is set to 130 in the grouping step.

**Algorithm 2.** TPRO

---

**Input:** road network  $G$ , dataset  $T$ , grid number  $m$  and  $n$ , popular routes number  $k$ , score threshold  $\theta$ , TTG time interval  $\Delta t$

**Output:** outlier set  $T'$

```

1: //DATASET GROUPING
2:  $grid = CreateGrid(G, m, n)$ ; // create  $m \times n$  grids on the road network
3:  $groups = \phi$ ;
4: for each  $\tau \in T$  do
5:    $srcGrid = grid.getLocatedGrid(\tau.s)$ ;
6:    $destGrid = grid.getLocatedGrid(\tau.d)$ ;
7:    $groups[srcGrid, destGrid].add(\tau)$ ; // add trajectory to corresponding group
8: end for
9: //DETECTING IN EACH GROUP
10: for each  $T^* \in groups$  do
11:    $ttg = CreateTTG(T^*, \Delta t)$ ; // construct time-dependent transfer graph
12:   for each  $\tau \in T^*$  do
13:      $s_\tau = 0$ ;
14:      $routes = GetTopKRoutes(ttg, k, \tau.t_s, \tau.t_d)$ ; // query popular routes
15:     for each  $\gamma \in routes$  do
16:        $s_\tau += w_\gamma \cdot \delta(\tau, \gamma)$ ; // compare trajectory with each route
17:     end for
18:     if  $s_\tau > \theta$  then
19:        $T'.add(\tau)$ ; // add to outlier set
20:     end if
21:   end for
22: end for
23: return  $T'$ ;

```

---

## 4.2 Evaluation Criteria

In practice, detection rate (the fraction of anomalous trajectories that are successfully detected) and false alarm rate (the fraction of normal ones that are predicted to be anomalous) are two important measures to evaluate the performance of an anomaly detection method. Obviously, a good outlier detection method should have a high detection rate and a low false alarm rate. After we plot the detection rate on y-axis and the false alarm rate on x-axis, we can get a curve called Receiver Operating Characteristic (ROC) [10] curve. The AUC [11] value is defined as the area under the ROC curve. For a randomly chosen normal trajectory  $\tau_n$  and a randomly chosen anomalous trajectory  $\tau_a$ , the AUC value is equal to the probability that  $s_{\tau_a} > s_{\tau_n}$ . Obviously, if the AUC value is close to 1, the outlier detection method is of high quality.

## 4.3 Results

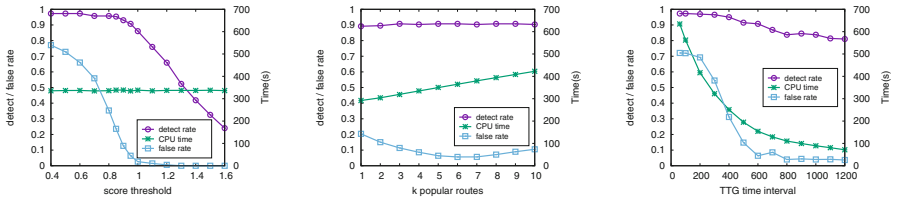
In this section, we first analysis how the parameters affect the experiment result. Then we give a comparison between TPRO, TRAOD and IBAT from the accuracy and efficiency.

**Varying Parameters.** There are mainly three parameters in the detecting step of TPRO: score threshold  $\theta$ , popular routes number  $k$  and TTG time interval  $\Delta t$ . So this paragraph elaborates how these three parameters affect the detection rate, false alarm rate and the process time<sup>2</sup>.

Figure 4(a) shows how  $\theta$  affects the detection rate, false alarm rate and process time when  $k = 5$  and  $\Delta t = 600$  s. As  $\theta$  increasing, which means that the detection criterion becoming more conservative, the detection rate and the false alarm rate will fall. But the process time is stable. When  $\theta \approx 1.0$ , we have a high detection rate and a low false alarm rate.

$k$  represents how many popular routes will be used to judge if a trajectory is an outlier. Of course, more popular routes are used, more accurate the result will be. But the process time will grow linearly because we must compare each trajectory with each popular route. Figure 4(b) shows how  $k$  affects the detection rate, false alarm rate and the process time when  $\theta = 1.0$  and  $\Delta t = 600$  s. It shows that as  $k$  increasing, the false alarm rate will fall and the process time will go up. But it has a small effect on the detection rate. When  $k = 5$ , we can have a high detection rate and a low false alarm rate. Meanwhile, the process time is acceptable.

Figure 4(c) shows how  $\Delta t$  affects the detection rate, false alarm rate and process time when  $k = 5$  and  $\theta = 1.0$ . If  $\Delta t$  is too small, TPRO will overstate the temporal cost when calculating the distance between a trajectory and a route. So the false alarm rate is very high and will fall as  $\Delta t$  increasing. But more smaller  $\Delta t$  is, more accurate the selected popular routes are. But the popular routes query time will be more longer. So as  $\Delta t$  increasing, detection rate and the process time will both fall, too. When  $\Delta t = 600$  s, we can have a low false alarm rate and a less process time. Meanwhile, the detection rate is acceptable.



**Fig. 4.** Detection rate, false alarm rate and time cost under varying  $\theta$  (Left),  $k$  (Middle) and  $\Delta t$  (Right)

**TPRO vs. TRAOD and IBAT.** This paragraph give a comparison between TPRO, TRAOD and IBAT. All of the three algorithms are tested in their best parameters, which are listed in Table 1.

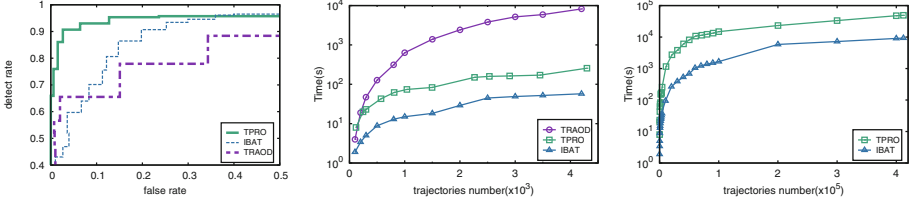
Figure 5(a) shows the ROC curves of TPRO, TRAOD and IBAT. For better illustration, the ranges of false alarm rate and detection rate are set to  $[0 \sim 0.5]$  and  $[0.4 \sim 1]$ . We can see that TPRO has a larger area under the ROC curve

<sup>2</sup> These three evaluating indicators are counted under the labeled dataset.

**Table 1.** Parameter setting of TPRO, TRAOD and IBAT

Algorithm	TPRO	TRAOD	IBAT
Parameters	$k = 5, \Delta t = 600 \text{ s}$	$D = 80, p = 0.95$	$m = 100, \psi = 256$

than TRAOD and IBAT. It means that TPRO has a better performance than TRAOD and IBAT in accuracy.



**Fig. 5.** ROC curves of TPRO, TRAOD and IBAT (Left). Efficiency under the small dataset (Middle) and large dataset (Right).

Figure 5(b) and (c) show the process time of TPRO, TRAOD and IBAT under different scale dataset. Because the time complexity of TRAOD is  $O(n^2)$ , which is very time consuming in larger dataset detection. So we only test it on the small dataset. From these two figures, we can see that the time cost of TPRO is between IBAT's and TRAOD's.

## 5 Related Work

Some related works are introduced in this part, which can be categorized into two groups. The first one focuses on trajectory outlier detection and the second one focuses on popular route mining.

**Trajectory Outlier Detection:** Some algorithms have been proposed to detect trajectory outlier, but each addresses certain aspects of abnormality. Lee *et al.* [7] put forward a group-and-detect framework and develop an algorithm called TRAOD. TRAOD splits a trajectory into various subparts (at equal intervals), then a hybrid of the distance-based and density-based approach is used to classify each subpart is abnormal or not. Chen *et al.* [8] propose an isolation based method, called IBAT. For a group of trajectories and a trajectory will be tested in this group, they randomly pick a point from the test trajectory and remove other trajectories which do not contain this point. This process is repeated until no trajectory is left or all the trajectories left contain all the points the test trajectory has. If the test trajectory is an outlier, this process will end very soon. And Li *et al.* [12] emphasis on historical similarity trends between data points. At each time step, each road segment checks its similarity with the other

road segments, and the historical similarity values are recorded in a temporal neighborhood vector at each road segment. Outliers are calculated from drastic changes in these vectors [13]. Guan *et al.* [14] use a feature vector, such as  $\langle \text{direction}, \text{speed}, \text{angle}, \text{location} \rangle$ , to represent a trajectory segment and detect the outliers according to these features. Mohamad *et al.* [15] take the speed and turn directions into consideration. If a trajectory has an sudden speed change or some unexpected turns, it is an abnormal trajectory. Recently, some studies have used learning methods to identify anomalous trajectories [16, 17]. But these methods usually need training data, which is inconvenient to label. There are also some works [18–20] have been done for the stream data detection or on-line detection.

**Popular Route Mining:** Finding the most desirable path has been a hot research topic for decades. Many works [21–23] have been done in finding the shortest/fastest path. But the popular route does not mean the shortest or fastest path. In most case, we prefer the most frequent path as the popular route. Lots of algorithms have been proposed for popular route searching. Zaiben *et al.* [3] introduce a transfer probability network to discover popular route from historical trajectories. They derive the probability of transferring from every significant location to the destination based on the historical trajectories, and the transfer probability is used as an indicator of popularity. The popularity of a route is defined as the product of transfer probabilities of all significant locations on the route. Luo *et al.* [9] also construct a network graph (called footmark graph) to mine frequent path. But they describe the edge frequency as the total number of trajectories passing through the edge. Then they define a descending edge frequency sequence to judge which path is more frequent. Another work, such as [4], aims at deriving routes from uncertain trajectory data.

## 6 Conclusions

In this paper, we propose a time-dependent outlier detection algorithm which is called TPRO. Given a trajectory dataset, we first divide the relevant trajectories into same group. Then for each group, we propose a time-dependent transfer graph to speed up querying time-dependent popular routes. We use a time-dependent edit distance to represent the difference score between a trajectory and a route. If a trajectory has a great difference with all of the selected popular routes, it’s an outlier. We evaluate our method on a real-world dataset. The experiment result shows that our method has a better performance than TRAOD and IBAT.

In the future, we plan to enhance our algorithm in two directions. Firstly, although TPRO is more efficient than TRAOD, but TPRO is slower than IBAT in about ten times. So we want to improve TPRO on efficiency. Secondly, we will improve TPRO for on-line outlier detection.

## References

1. Ye, Y., Zheng, Y., Chen, Y., Feng, J., Xie, X.: Mining individual life pattern based on location history. In: IEEE MDM, pp. 1–10 (2009)
2. Zheng, Y., Xie, X., Ma, W.-Y.: Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* **33**, 32–39 (2010)
3. Chen, Z., Shen, H.T., Zhou, X.: Discovering popular routes from trajectories. In: IEEE ICDE, pp. 900–911 (2011)
4. Wei, L.-Y., Zheng, Y., Peng, W.-C.: Constructing popular routes from uncertain trajectories. In: ACM SIGKDD, pp. 195–203 (2012)
5. Zheng, Y., Liu, L., Wang, L., Xie, X.: Learning transportation mode from raw GPS data for geographic applications on the web. In: WWW, pp. 247–256 (2008)
6. Han, J., Kamber, M., Pei, J.: *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, San Francisco (2006)
7. Lee, J.-G., Han, J., Li, X.: Trajectory outlier detection: A partition-and-detect framework. In: IEEE ICDE, pp. 140–149 (2008)
8. Zhang, D., Li, N., Zhou, Z.-H., Chen, C., Sun, L., Li, S.: iBAT: detecting anomalous taxi trajectories from GPS traces. In: ACM UbiComp, pp. 99–108 (2011)
9. Luo, W., Tan, H., Chen, L., Ni, L.M.: Finding time period-based most frequent path in big trajectory data. In: ACM SIGMOD, pp. 713–724 (2013)
10. Fawcett, T.: An introduction to roc analysis. *Pattern Recogn. Lett.* **27**(8), 861–874 (2006)
11. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.* **30**(7), 1145–1159 (1997)
12. Li, X., Li, Z., Han, J., Lee, J.-G.: Temporal outlier detection in vehicle traffic data. In: IEEE ICDE, pp. 1319–1322 (2009)
13. Gupta, M., Gao, J., Aggarwal, C., Han, J.: Outlier detection for temporal data. *Synth. Lect. Data Min. Knowl. Discov.* **5**, 1–129 (2014)
14. Yuan, G., Xia, S., Zhang, L., Zhou, Y., Ji, C.: Trajectory outlier detection algorithm based on structural features. *J. Comput. Inf. Syst.* **7**(11), 4137–4144 (2011)
15. Mohamad, I., Ali, M., Ismail, M.: Abnormal driving detection using real time global positioning system data. In: *Space Science and Communication (IconSpace)*, pp. 1–6. IEEE (2011)
16. Sillito, R.R., Fisher, R.B.: Semi-supervised learning for anomalous trajectory detection. In: BMVC, pp. 1–10 (2008)
17. Li, X., Han, J., Kim, S., Gonzalez, H.: Roam: rule-and motif-based anomaly detection in massive moving object data sets. In: SIAM SDM, pp. 273–284 (2007)
18. Yu, Y., Cao, L., Rundensteiner, E.A., Wang, Q.: Detecting moving object outliers in massive-scale trajectory streams. In: ACM KDD, pp. 422–431 (2014)
19. Bu, Y., Chen, L., Fu, A. W.-C., Liu, D.: Efficient anomaly monitoring over moving object trajectory streams. In: ACM SIGKDD, pp. 159–168 (2009)
20. Chen, C., Zhang, D., Castro, P.S., Li, N., Sun, L., Li, S., Wang, Z.: iBOAT: Isolation-based online anomalous trajectory detection. In: IEEE TITS(2013)
21. Gonzalez, H., Han, J., Li, X., Myslinska, M., Sondag, J.P.: Adaptive fastest path computation on a road network: a traffic mining approach. In: VLDB (2007)
22. Sacharidis, D., Patroumpas, K., Terrovitis, M., Kantere, V., Potamias, M., Mouratidis, K., Sellis, T.: On-line discovery of hot motion paths. In: ACM EDBT (2008)
23. Kanoulas, E., Du, Y., Xia, T., Zhang, D.: Finding fastest paths on a road network with speed patterns. In: IEEE ICDE, pp. 10–10 (2006)

Web Information Systems Engineering – WISE 2015

16th International Conference, Miami, FL, USA,

November 1-3, 2015, Proceedings, Part I

Wang, J.; Cellary, W.; Wang, D.; Wang, H.; Chen, S.-C.; Li,  
T.; Zhang, Y. (Eds.)

2015, XVIII, 622 p. 219 illus. in color., Softcover

ISBN: 978-3-319-26189-8