Easing Traffic Congestion: An Improved Clustering Method for Sharing Bike Station Deployment

Jian Kang, Weipeng Jing(✉), and Chengfang Zhao

College of Information and Computer Engineering,
Northeast Forestry University, Harbin, China
{laurelkang,weipeng.Jing,chengfangzhao}@outlook.com

Abstract. An excessive number of cars cause serious traffic jams. Fortunately, a new kind of environmentally friendly transportation service, sharing bikes, came into being. In the cities with shared bikes, deploying shared bikes stations purposefully will make a contribution to reducing the pressure of the traffic. We aim to draw support from sharing bikes to improve the bad traffic. To find the real problems of the current traffic. We make full use of history taxi trajectories to analyze current traffic condition. We design a traffic jam detection framework in this paper. It is called CF framework for short. Derived from the density-based clustering algorithm of inspiration, we propose a new clustering method (CF-Dbscan). The new method has successfully been applied to the trajectories clustering. To deal with errors of devices, a road network matching algorithm (CF-Matching) helps match GPS points to real road network accurately. The first experiment proves that our clustering algorithm performs better than DBSCAN in the field of trajectory clustering. We design another experiment to verify the effectiveness of our CF framework in the real scene. The results of the experiments prove that we can achieve the purpose of reducing traffic jam with our framework.

Keywords: Sharing bikes · DBSCAN · Clustering · Road matching
Traffic congestion

1 Introduction

Recently, the urban traffic is getting worse. As the population of the cities grows, the number of private cars and public transport also increases. Although it satisfies people's travel needs to some extent, it has also caused great pressure to traffic. Fortunately, the bicycles came back to the sight of the public in the form of sharing. In the last few years, shared bikes have been springing up across many big cities of China. Commuters scan QR code on shared bikes with their mobile phones to borrow them from a station. They only have to pay a certain amount of money to ride the bike anywhere. In addition to environmental friendliness,

I. Romdhani et al. (Eds.): CollaborateCom 2017, LNICST 252, pp. 296–308, 2018.
https://doi.org/10.1007/978-3-030-00916-8_28

the existence of sharing bikes can also improve traffic conditions by reasonably planning the station location.

By observing several cities with shared bicycles, it is not difficult to find that most of the shared bike stations [3] are deployed around personnel-intensive areas. There are some potential problems with such a deployment method. Firstly, bicycle resources are too much to be concentrated in personnel-intensive areas, people in other areas do not enjoy enough convenience so distribution is not balanced. Secondly, the environmentally friendly feature of sharing bicycles is not being used well. How to use shared bicycles to reduce traffic congestion is the core of this paper. With the development of location based service, GPS equipments are widely used by taxis [10]. We have collected a large number of taxi trajectory data. Information on traffic congestion of a city is hidden in the taxi trajectory data [11]. We apply the results of the analysis to shared bikes to find the most suitable station locations [5,13]. Our goal is to give pedestrians one more choice on traffic jams. This approach will reduce the number of vehicles on congested roads. As traffic congestion is mainly caused by motor vehicles. Bicycles travel on bike paths instead of occupying motor ways. This is a good way to divert pedestrians. The less vehicles in the motor ways, the more smooth the traffic will be.

Of course, it is not a compulsive act to choose a shared bike as a means of transportation. However, in the face of traffic congestion, shared bikes provide a new choice, which can guide people to choose other means of transportation to avoid the peak hours. There are always people who would rather ride a bike than wait for traffic jams in the car.

As we all know, there are always many trajectories generated by motor vehicles on the road at any time. In general, at certain intervals, the more trajectories on the road the more crowded the vehicles are. And we can identify traffic flow on a road as trajectories with similar temporal and spatial characteristics. We can find all the trajectories from the data set with similar temporal and spatial characteristics and treat them as on the same road segment. This conforms to the idea of clustering. Due to noise points in the data, we decided to use density-based clustering algorithm. We propose a framework (CF) to detect the traffic jam [2] in this paper. Trough taxi trajectory data mining to make traffic problems reveal. CF framework consists of two main phases, CF-Dbscan and CF-Matching. Compared with the traditional density-based clustering algorithm DBSCAN [1], the CF-DBSCAN is greatly improved. Traditional DBSCAN faces two potential problems in processing trajectory data. (1) The DBSCAN is an algorithm based on the location of points, it is too concerned about the location information of the point itself, so that the association between points and points is ignored. It may be well-suited for hashing irrelevant points, but it does not apply to trajectories that are closely related. Although a typical GPS trajectory consists of a series of points. Points belonging to the same trajectory are related to each other. The traditional DBSCAN will ignore some of the associations to make the clustering results unreasonable. (2) The traditional DBSCAN algorithm calculates the distance of every two points in the data set. There will be

huge computational costs on a large number of data sets. After fully considering the limitations of the original method, the density clustering idea was redesigned in this paper to make it more suitable for the trajectories [8,12]. We take into account the interrelationships between the points. So the CF-DBSCAN algorithm considers each trajectory as a whole [7,9], capturing the common parts (if any) [4] of the two trajectories. We have eliminated a lot of distance comparisons between points and points. Instead, we compare the macro features of the trajectories. The CF-DBSCAN algorithm improves the computing performance and makes the clustering result clearer. CF-Matching is a road network matching algorithm that can map the trajectory points to the real road network even at a low sampling rate (e.g., the time between two sampling points exceeds 2 min), thereby solving the GPS error problem. In this paper, the spatial evaluation function [6] is introduced to analyze the correlation of each sampling point sequence. With this framework a traffic congestion map is generated. It represents the most suitable places for the deployment of shared bicycle stations.

The remaining parts of the paper are arranged as follows: Sects. 2 and 3 elaborates the methodology of CF framework. Section 4 introduces the experiments and evaluation. Finally, we sum up this paper in Sect. 5.

2 Mining Trajectories

We elaborate the core stage of our methodology, CF-Dbscan. In this section. We first use a clustering algorithm to mine the trajectory data. We will describe the two phases of the clustering algorithm in two subsections.

2.1 Clustering Trajectories

The clustering algorithm proposed in this paper has two core steps. For easy description, we define the taxi track sequence Tr as a series of time-sequential GPS spatial points, Tr:$p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$. Each track has its unique identification code, Tr.id. Because the trajectory data is in the form of coordinate points on the two-dimensional plane in the data set. For trajectories, they have different shapes, lengths and geographical positions. So firstly, for any two trajectories to be compared, we need to standardize them as a preparation for formal clustering.

Get Public Parts. The task of CF-DBSCAN is to find congested roads within a specific period of time. In this paper, traffic congestion is defined in this way: if the number of trajectories on a road greater than the threshold during a fixed period of time, we regard this as a traffic jam. It is unscientific to use only the endpoints to calculate, because most of the trajectories do not strictly stop at a specific point from a specific point. For many taxis, the crowded road segments are not their entire journey, but only part of their whole trip. Figure 1 describes this situation.

For the situation shown in Fig. 1, we should not consider the overall trajectories, but should look at the trajectory split. For the trajectories, we should do

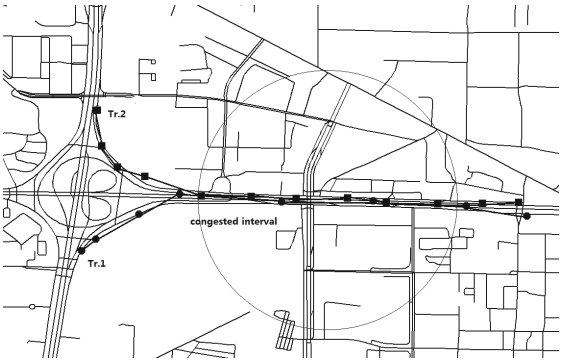

Fig. 1. The public parts of two tracks Tr.1, Tr.2

the following first. First of all, we need to know that the geographical coordinates are spherical coordinates. But for a city, for the convenience of computation, it can be abstracted as plane coordinates, and the error can be ignored. We regard longitude coordinates as X coordinates and latitude coordinates as Y coordinates. Then the trajectory is easily projected onto the X axis. We introduce four related variables $P1min, P1max, P2min, P2max$. They represent the endpoints of the projections (P1 and P2) above. To begin with, we have to figure out whether their projection has overlapping parts on the X axis. On X axis, if $P1max._x < P2min._x$ or $P2max._x < P1min._x$, obviously, their projections have no overlapping parts. We treat this situation as two trajectories without clustering conditions. So we do not need to do the remaining calculations. This judgment step can save a huge amount of calculation.

If the projection of the two tracks has an overlapping part, it means that two tracks have the initial condition of clustering, and only need further processing can we do cluster operation. In other words, they may have passed through the same road segments. We determine the common part of the original trajectory based on the projection boundary. The public section is used for a more thorough judgment.

CF-Dbscan. In this paper, the clustering algorithm is based on the public parts of two tracks. The traditional DBSCAN algorithm determines whether two candidates belong to the same cluster according to the spatial characteristics. But trajectory data contains more dimensions of information. The information needs to be considered as much as possible in order to achieve better clustering results. So DBSCAN is not suitable to directly use here. Therefore, we use the common parts of the trajectories to extend DBSCAN to an algorithm that is suitable for trajectory clustering.

We introduce a variable that represents spatial radius c_d. Variable c_t is used to represent temporal span. They represent the spatial-temporal feature of any two trajectories. But only the characteristics of spatial-temporal feature are not

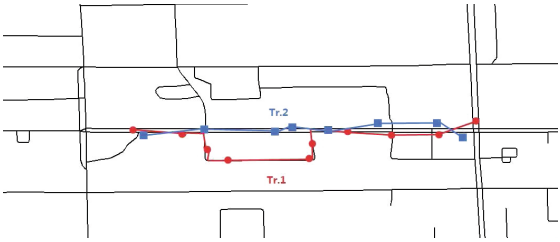

Fig. 2. A distribution situation of two trajectories

enough. Figure 2 describes the situation. The two trajectories have common parts and similar spatiotemporal characteristics. But in Fig. 2, they don't go through the same section of the road. Obviously, they should not be clustered into one cluster. To deal with this problem, we introduce a curvature index, which is represented by K. We define the curvature as the ratio of the actual distance from p_i to p_j and Euclidean distance between p_i and p_j. It is obvious that the actual driving distance is always greater than the Euclidean distance, so the curvature is always greater than or equal to 1.

The definition of curvature in this paper is shown as follow

$$K = \frac{L(p_i \rightarrow p_j)}{dist(p_i \rightarrow p_j)} \tag{1}$$

The more similar the shape of the two trajectories, the closer the result of the curvature will be. We set a threshold for the curvature gap ΔK. If the curvature difference satisfies the following formula, the result can be accepted.

$$|K_1 - K_2| \leq \Delta K \tag{2}$$

A set of neighborhood corresponding to a core trajectory is defined as $\mathrm{N}_{c_d,c_t,k}(\mathrm{Tr.i})$. The trajectories in this set are all trajectories that meet the clustering conditions with the core trajectories. For every common part, we call the left endpoint as the starting point, and the right endpoint is called the end point. This paper defines the clustering conditions according to the following rules. If a trajectory can be added to a neighbor, it needs to satisfy all conditions at the same time. (1) For the two cut public parts, the spatial distance between the starting points and the spatial distance between the end points should be within the threshold c_d. (2) For the two cut public parts, The temporal distance c_t between the two starting points should be within the threshold. (3) the absolute value of the curvature difference is within ΔK, i.e.,

$$\begin{aligned} N_{c_d,c_t,k}(Tr.i) = \{Tr.j \in \tau | dist(o_{T_i}, o_{T_j}) \leq c_d, dist(d_{T_i}, \\ d_{T_j}) \leq c_d, |t_{T_i} - t_{T_j}| \leq c_t, |K_i - K_j| \leq \Delta K\} \end{aligned} \tag{3}$$

where τ represents the set of tracks, dist$(\cdot\,,\cdot)$ denotes road net distance between two points. o_T and d_T denote starting point and end point, respectively. t_T

denotes the beginning timestamp of the public part, K is curvature while ΔK represents curvature threshold.

A capacity threshold named MinTrs is employed here to constrain the density of cluster. It denotes the lower bound of the number of trajectories in each cluster. If the number of trajectories belonging to a cluster exceeds MinTrs, i.e., $|N_{c_d,c_t,k}(Tr.i)| \geq$ MinTrs, then $Tr.i$ can be regarded as a core trajectory. Based on the number of existing shared bicycles we plan to deploy and the number of bicycle stations, we can flexibly adjust the above four thresholds, c_d, c_t, ΔK and MinTrs.

Considering the difference between one-way lanes and two-way lanes, we consider two-way lanes as two lanes and cluster them separately. Algorithm 1 gives a detailed description of the CF-Dbscan algorithm. Our CF-Dbscan algorithm required five parameters as input. A trajectories set τ and four threshold parameters, c_d, c_t, K and MinTrs. The procedure of GetNeighborhood (Line 5) helps find a neighborhood tracks set based on Eq. 3. The algorithm first randomly selects a trajectory from the set τ as the beginning. Then the procedure of GetNeighborhood can get the neighborhood set to the target trajectory. In the procedure of GetNeighborhood, to begin with, we cut two candidate trajectories on the basis of their projections on X axis (if any). If a track and multiple tracks have common parts, it will be cut many times. This will result in losing data. To avoid this phenomenon, in $N_{c_d,c_t,k}(Tr)$ original trajectories will be reserved instead of trajectories fragment. Although this approach will result in some redundancy, this is still a more comprehensive approach. Continue Algorithm 1. After obtaining the neighborhood set, it retrieves all unvisited trajectories within the neighborhood (line 12). In this way, the cluster is continuously expanded until it is saturated. Then one cluster is generated. In other words, we use open set M to reserve neighbor. The procedure will not stop until all trajectories of M are explored.

3 Clustering Result Matching

3.1 CF-Matching

After the execution of the CF-Dbscan, several clusters of trajectories are generated. To locate the location of the cluster accurately, first, find the core trajectory of the cluster. However, due to equipment errors and other external factors, they are not standardized. This error sometimes leads directly to wrong results and cannot be ignored. So in order to eliminate the error, we propose an algorithm to map the cores of these clusters to the real road network. We call our road network matching method as CF-Matching.

Get Candidates. The CF-Matching algorithm first traverses a trajectory and the road network around it, in order to determine the set of candidate points for each sampling point. Figure 3 describes the process. p_i denotes a sampling point of current trajectory, e represents road segments. c denotes the projection

Algorithm 1. CF-Dbscan for trajectories clustering

INPUT:

A set of trajectories τ

Four parameters introduced above c_d,c_t,K and MinTrs

OUTPUT:

A cluster set C={C_1,C_2,C_3,...,$C_{|C|}$}

```
Set c_id = 1
Initialize all trajectories in τ as unvisited
for each Tr ∈ τ do
  if Tr is unvisited then
    Mark T as visited
    GetNeighborhood(Tr)
    if |N_{c_d,c_t,k}(Tr)| < MinTrs then
      Mark Tr as a non-core trajectory
    else
      Mark Tr as a core trajectory
      Put T into C_{c_id}
      Put trajectories of neighborhood set of Tr into M, i.e.,M ← N_{c_d,c_t,k}(Tr)-{Tr}
      for each T_b ∈ M do
        if T_b is unvisited then
          Mark T_b as visited
          Put T_b into C_cid
          GetNeighborhood(T_b)
          if |N_{c_d,c_t,k}(Tr)| ≥ MinTrs then
            Insert N_{c_d,c_t,k}(Tr)-{Tr} into M
          end if
        end if
        if T_b is a non-core trajectory then
          Put T_b into C_{c_id}
        end if
        Remove T_b from M
      end for
      Increase c_id by 1
    end if
  end if
end for
```

points of p_i on e satisfying $c = argmin_{\forall c_i \in e}\text{dist}(c_i, p)$, $\text{dist}(c_i, p)$ returns the distance between p_i and any point on e. To speed up the search for candidate points, we employ a window that follows the sampling point p_i to cut the whole network to reduce the search range. After executing this step, the algorithm will get a set sequence, the next question is how to choose one of the most suitable points from each candidates set so that P: $c_1^{j_1} \rightarrow c_2^{j_2} \rightarrow \cdots \rightarrow c_n^{j_n}$ best matches T:$p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$.

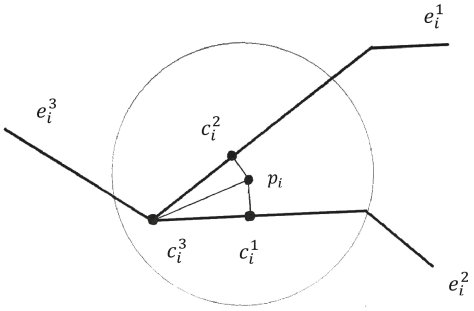

Fig. 3. Sampling point p_i and its candidate points

Correlation Analysis. Empirically, errors in GPS measurements can be reasonably described with a normal distribution $N(\mu, \sigma^2)$ of the distance between c_i^j and p_i. It indicates that, without considering the influence of other points, the probability that GPS observation p_i matches the candidate point c_i^j on the actual road network. We use a formula to describe the possibility $N(c_i^j)$ of c_i^j with regard to p_i:

$$N(c_i^j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i^j - \mu)^2}{2\sigma^2}} \tag{4}$$

$x_i^j = \text{dist}(c_i^j, p_i)$, it denotes the distance between c_i^j and p_i. In order to make the number as uniform as possible. Empirically, we use a zero-mean normal distribution with a standard deviation of 0.002 km.

Road network matching can not be completed only on the basis of possibility. In practice, we can't ignore the impact from the front and back sampling points, otherwise it may result in a wrong matching result. Figure 4 describes an example. p_{i-1}, p_i, p_{i+1} are the sequence of sampling points that belong to the same trajectory. c_i^1 and c_i^2 are candidate points of p_i. We may match p_i to c_i^2 if we only consider the observation possibility. Because they are closer in geographical location. But in fact, as depicted in Fig. 4, if we match p_i to c_i^2 it will lead to a deviation from the right route.

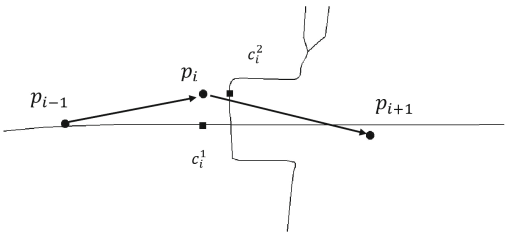

Fig. 4. An example that needs transmission probability

Actually, when moving to the destination, people tend to take a shorter route. This needs to take into account the linear correlation of the front and back points. Given two candidate points c_{i-1}^t and c_i^s of two neighboring trajectory points p_{i-1} and p_i, we define the possibility from c_{i-1}^t to c_i^s as the possibility of p_{i-1} to p_i on the real road network. We use the following formula to calculate the associated possibility.

$$V(c_{i-1}^t \rightarrow c_i^s) = \frac{d_{i-1 \rightarrow i}}{\omega_{(i-1,t) \rightarrow (i,s)}} \tag{5}$$

where $d_{i-1 \rightarrow i}$ is the Euclidean distance between p_{i-1} and p_i, $d_{i-1 \rightarrow i} = \text{dist}(p_i, p_{i-1})$. $\omega_{(i-1,t) \rightarrow (i,s)}$ returns the length of shortest path between c_{i-1}^t and c_i^s.

To calculate $\omega_{(i-1,t) \rightarrow (i,s)}$, we must first find shortest path. As the road network is a huge undirected map. Directly searching for the shortest path on it will bring exponentially increasing computational costs. So we employ a threshold to limit the search depth. In fact, if we can't find a path within this range, then the chance of the target point being the next candidate is very small. By combining Eqs. 4 and 5, we propose Eq. 6 for linear assessment.

$$F_s(c_{i-1}^t \rightarrow c_i^s) = N(c_i^s) * V(c_{i-1}^t \rightarrow c_i^s), 2 \leq i \leq n \tag{6}$$

where c_{i-1}^t and c_i^s are candidate points. They belong to sampling points p_{i-1}, p_i separately.

In all possible sequence of candidate points. We need to find a candidate point sequence with the highest total score according to Eq. 6. This candidate sequence is most likely the true path of this trajectory. The best match results follow the formula below:

$$P = argmax_{P_c} F(P_c), \forall P_c \in G_T'(V_T', E_T') \tag{7}$$

The result sequence of the calculated output is the sequence of candidate points with the greatest matching possibility. With CF-Matching, the core trajectory of each cluster can be accurately mapped to the real road network.

4 Experiments

We have prepared two groups of experiments to assess the performance of the CF framework. To begin with, we will describe a large-scale data set of actual taxi trajectories. Next we prove on our data set that our method is more suitable for the trajectory. Finally, we carried out field observations on the experimental results and made statistics and collation of the results, which proved that our station deployment method really contributed to improving the traffic condition.

4.1 Passenger Trajectories

The data set of taxi trajectory was collected in Beijing, China, in November 2011. A total of 12,000 taxis, a total of 230917631 GPS records. Each record contains nine fields. The fields associated with this study include the latitude, longitude, taxi ID, timestamp, and occupied status.

In this experiment, the travel pattern of pedestrians is fully taken into account. The way people travel on holidays is not like usual. For example, every working day has a similar peak. However, during the weekend, the peak hours in the morning may be somewhat later than usual, while the afternoon peaks last longer than usual. Compared to holidays, we tend to study regular working days. So we preprocess the data. In the use of shared bicycles to alleviate traffic congestion. We filter the data for the weekdays and the peak time data is extracted from the daily trajectories. Figure 5 shows the clustering result of CF-Dbscan. The dots in the figure indicate GPS points. From the diagram we can see that the shape of the crowded road is clearly shown in the result. Although there is still a little noise, it has little effect on the result.

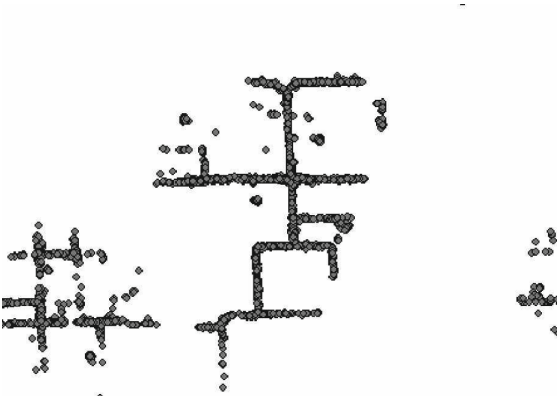

Fig. 5. Clustering effect shown in ArcGIS

We use the F-measure evaluation method to evaluate the performance of CF-Dbscan. We randomly select some roads, mark them to determine whether the tracks in the clustering result are the mark trajectories as the criterion. We randomly selected ten roads in Beijing to evaluate F-score of DBSCAN and CF-Dbscan. The contrast results are presented in Fig. 6. The higher the F-score value, the better the clustering effect. From Fig. 6, compared with DBSCAN, CF-Dbscan has an approximate 0.1 advantage over the average. This can prove that CF-Dbscan performs better than traditional DBSCAN on trajectory clustering.

4.2 CF Evaluation

To verify the actual effects of the CF framework. We need to do field observation and statistics. In order to allow pedestrians to travel in a regular and stable

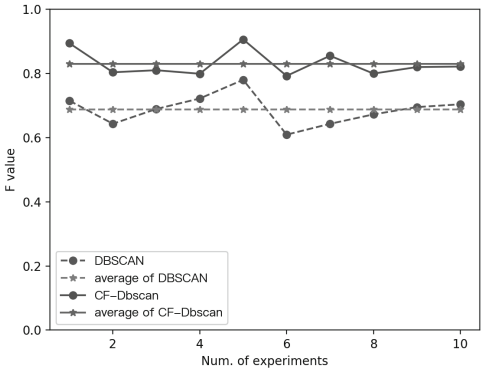

Fig. 6. Comparison of F values between original DBSCAN and CF-Dbscan

manner, we chose to conduct experiments for six consecutive days without holidays or special events. In order to control the influence of external factors, we chose a special experimental road, which has the following four characteristics. (1) This road is very crowded at peak time. (2) There is a relatively fixed taxi ride point. (3) There are no other types of parking spots near the taxi stand (e.g. bus stop). (4) Under completely non special circumstances, the number of moving cars is very stable during peak hours (here non special refers to non holidays, non special activities and no accidents). We set up a shared bicycle experiment point for observation. The reason why we did not choose an existing shared bicycle station for experiments is that we want to capture the changes in behavior of pedestrians when facing new options (shared bicycles). We count the number of pedestrians who take a taxi or choose a sharing bike. In this way, we can clearly see the respective changes in the number of pedestrians choosing the two modes of transport. The observation time is 17:00 to 18:00 points per day during the observation date (Fig. 7).

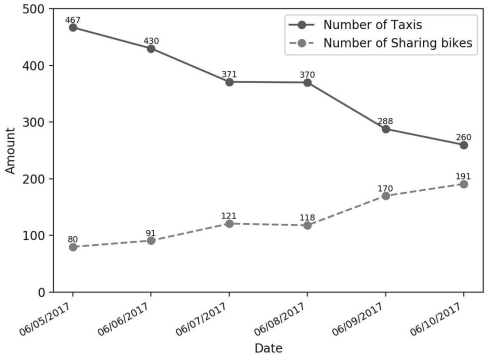

Fig. 7. Changes in the number of taxis and sharing bikes during the experiment

The solid line shows the number of taxis that change over time, and the lower dotted line shows the number of bicycles. From the chart we can easily see that there is a significant increase in the number of pedestrians choosing shared bicycles while the number of people taking taxis dropped obviously. The results can prove that the targeted placement of bicycles on traffic congested roads can relieve traffic pressure.

5 Conclusion

In this article, we propose a CF framework for the actual problems of traffic congestion. With the rise of shared bicycles, we hope to analyze the hidden congested sections from taxi trajectory data, and then use bicycles to solve the congestion problem. Reducing the number of vehicles on the road by allowing shared bikes to replace taxis to make traffic smoother. In order to find congestion, we use the improved clustering algorithm CF-Dbscan to cluster the trajectory data. Clustering results are mapped to real road network by road network matching algorithm, CF-Matching. We used two experiments to prove the performance of the framework itself and the effect in the actual scene.

Acknowledgement. The work described in this paper is supported by Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology (GDUPTKLAB201502) and Special Fund for Forest Scientific Research in the Public Welfare (201504307).

References

1. Bi, F.M., Wang, W.K., Chen, L.: Dbscan: density-based spatial clustering of applications with noise. J. Nanjing Univ. **48**(4), 491–498 (2012)
2. Ceder, A.: Public transit planning and operation: modeling, practice and behavior. Transport **30**(4), 448–450 (2015)
3. García-Palomares, J.C., Gutiérrez, J., Latorre, M.: Optimizing the location of stations in bike-sharing programs: a GIS approach. Appl. Geogr. **35**(1–2), 235–246 (2012)
4. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: ACM SIGMOD International Conference on Management of Data, pp. 593–604 (2007)
5. Li, Y., Luo, J., Chow, C.Y., Chan, K.L., Ding, Y., Zhang, F.: Growing the charging station network for electric vehicles with trajectory data analytics. In: IEEE International Conference on Data Engineering, pp. 1376–1387 (2015)
6. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate GPS trajectories. In: Proceedings of the ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, Seattle, Washington, USA, 4–6 November 2009, pp. 352–361 (2009)
7. Wang, W., Tao, L., Gao, C., Wang, B., Yang, H., Zhang, Z.: A C-DBSCAN algorithm for determining bus-stop locations based on taxi GPS data. In: Luo, X., Yu, J.X., Li, Z. (eds.) ADMA 2014. LNCS (LNAI), vol. 8933, pp. 293–304. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14717-8_23

8. Yan, L., Chow, C.Y., Lee, V.C.S., Li, Y., Zeng, J.: T2CBS: mining taxi trajectories for customized bus systems. In: Computer Communications Workshops, pp. 441–446 (2016)
9. Yuan, H., Qian, Y., Ma, B., Wei, Q.: From trajectories to path network: an endpoints-based GPS trajectory partition and clustering framework. In: Li, F., Li, G., Hwang, S., Yao, B., Zhang, Z. (eds.) WAIM 2014. LNCS, vol. 8485, pp. 740–743. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08010-9_80
10. Zhang, D., Li, N., Zhou, Z.H., Chen, C., Sun, L., Li, S.: iBAT: detecting anomalous taxi trajectories from GPS traces. In: International Conference on Ubiquitous Computing, pp. 99–108 (2011)
11. Zheng, Y., Liu, Y., Yuan, J., Xie, X.: Urban computing with taxicabs. In: International Conference on Ubiquitous Computing, pp. 89–98 (2011)
12. Zheng, Y., Yuan, N.J., Zheng, K., Shang, S.: On discovery of gathering patterns from trajectories. In: IEEE International Conference on Data Engineering, pp. 242–253 (2013)
13. Zhu, B., Xu, X.: Urban principal traffic flow analysis based on taxi trajectories mining. In: Tan, Y., Shi, Y., Buarque, F., Gelbukh, A., Das, S., Engelbrecht, A. (eds.) ICSI 2015. LNCS, vol. 9142, pp. 172–181. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20469-7_20