An Improved DBSCAN Algorithm to Analyze Taxi Pick-Up Hotspots

Wu Zheng[1], Yuan Cheng[1], Nan Li[2], Zizhen Wang[1(✉)], and Ao Li[1]

[1] College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China
2589377022@qq.com

[2] Information Center of State Administration of Science, Technology and Industry for National Defense, Harbin, China

Abstract. The DBSCAN algorithm is improved to analyze taxi pick-up hotspots. The time is divided into multiple time periods, and the load threshold and neighborhood radius are automatically extract. The accuracy of the DBSCAN algorithm for analyzing passenger areas is improved.

Keywords: Clustering · DBSCAN Algorithm · Pick-up Hotspots

1 Introduction

As the population grows, there is an increasing demand for taxis for residents to travel. However, there are many situations in which drivers are looking for passengers and passengers are waiting for drivers. Through the analysis of taxi passenger data, the hotspots in the city can be obtained. Therefore the efficiency of driver's pulling passengers can be improved that is a great significance for the urban traffic planning. In the past, the questionnaire survey method was mostly used in the investigation of hotspot areas [1]. This is not applicable to current large-area cities with large populations. At present, the unsupervised clustering method is used to cluster and analyze taxi trajectory data, which can extract hotspots more efficiently and accurately [3–5]. There are three main methods for clustering taxi trajectory data: the first is a clustering algorithm based on a grid, the second is a density-based clustering algorithm and the third is a partition-based clustering algorithm. Jiang H J [6] proposed an improved DBSCAN algorithm that takes into account the constraints of the road network and adds road topology relationships and road length information as constraints on the original basis. Xu [7] proposed a combination of the HIST algorithm and the K-means algorithm to calculate the traffic flow value in the region to extract passenger hotspots. Han Y. [8] optimized the parameters of the traditional DBSCAN algorithm. Zheng L.J. proposed a grid density-based GScan algorithm. Bao G.W. [9] used the A* path finding algorithm to select the passenger-carrying points in the neighborhood, which improved the speed of the DBSCAN algorithm. Liu P.P [10] added the R parameter to the FDBSCAN algorithm to limit the range of clusters.

A. Li et al. (Eds.): 6GN 2022, LNICST 504, pp. 180–195, 2023.
https://doi.org/10.1007/978-3-031-36011-4_16

Sun J. [11] obtained taxi hotspots through road feature information. Chakraborty S [12] proposed an incremental DBSCAN algorithm to dynamically cluster new data on the basis of the original clusters, which can better discover new clusters. Chen M et al. [13] proposed the P-DBSCAN algorithm which divides the data into regions and clusters each region separately and combines the results. Huang Z F [14] proposed an algorithm based on FGP-DBSCAN and improved it based on the P-DBSCAN algorithm which transforms the hard boundary problem of grid division into fuzzy grid partition.

Most of the existing passenger hotspot analysis algorithms have improved the accuracy and performance of the algorithm. However, the existing algorithms lack the analysis and processing of different time situations. A direct analysis of the data for the whole day does not provide a precise indication of the hot passenger pick-up areas at each time period. The DBSCAN algorithm is more suitable for processing taxi passenger data [15]. Because the algorithm can find clusters of arbitrary shapes and noise points are also regarded as unpopular areas for urban planning, the existing DBSCAN algorithm needs to adjust the input parameters. The problem of dividing the time is very complicated to manually adjust parameters for multiple time periods. This paper proposes an optimized DBSCAN algorithm (Auto-DBSCAN, A-DBSCAN for short). By dividing the time of the day every hour as a time period, more accurate hotspots at different times can be obtained. After extracting the passenger load situation in each time period, the driver's passenger load situation and the passenger's search situation are analyzed. The flow of people at different times is different. This results in different driving speeds of drivers and different times to find customers. The distance of seeking customers is determined by analyzing the time and speed of customer-seeking in each period. Thereby, the Eps parameter of DBSCAN is determined. The MinPts parameter of DBSCAN is determined by the average number of passengers carried in this period and the automatic processing of parameters is realized. This paper uses taxi trajectory data from New York City to optimize the processing results of the original DBSCAN algorithm with insufficient accuracy and difficult parameter adjustment. Analyzing the passenger loading situation in different time periods and performing a more accurate analysis of travel hotspots are helpful for urban construction planning.

2 Optimization of the DBSCAN Algorithm Through the Division of Time

2.1 Parameter Automation for Passenger-Carrying Analysis and Passenger-Seeking Analysis

DBSCAN is one of the most classic algorithms for density-based clustering. Eps is the neighborhood radius that defines the density and MinPts is the threshold that defines the core point [16]. An object is a core point if it contains more than MinPts' number of points within its radius Eps. An object is a boundary point if the number of points within its radius Eps is less than MinPts. However, when an object falls within the neighborhood of the core point, the object is ta boundary point. An object is a noise point if it is neither a core point nor a boundary point. Core points and boundary points can be regarded as hotspots, and noise points can be regarded as sparsely populated areas. Through the

distribution of these points, urban planning can be better carried out. It is helpful for drivers to carry passengers.

Passenger data need to be processed at different times; otherwise, it is difficult to obtain accurate results. The problem brought by the division of time is that different data are clustered many times and the input adjustment of parameters is difficult. It is important to consider the relationship between the Eps parameter and the MinPts parameter and the actual driver-passenger relationship. If the driver spends too much time looking for passengers, it will lead to unnecessary economic losses and environmental pollution. Such areas do not qualify as hotspots. In contrast, the driver can find the next passenger after driving the appropriate distance, which is the most beneficial result for both the passenger and the driver. Therefore, it is also in line with the actual customer-seeking situation by calculating the average customer-seeking distance of drivers during this period as the minimum radius of the neighborhood. The average search distance for a certain period is equal to the product of the average search time and the average travel speed for the period. The average customer-seeking distance during this period is shown in Formula 1:

$$Eps = avgVelocity * avgPickupTime \tag{1}$$

In the formula, avgVelocity is the average driving speed during the period, and avgVelocity is the average customer-seeking time during the period.

At the same time, the average number of passengers carried by drivers in this period is used as the threshold, which is also in line with the relationship between the flow of people and the number of passengers carried by drivers in this period. The threshold formula for this period is:

$$MinPts = len(data)/carCount \tag{2}$$

where carCount is the number of drivers for this period.

2.2 A-DBCSAN Algorithm Flow

Based on the above analysis, the algorithm will analyze and process the parameters according to the passenger load situation of each time period. In addition it realizes the automation of parameters. The specific algorithm flow is as follows:

Algorithm 1: This is the main algorithm of the A-DBSCAN algorithm. Time division is first performed on the incoming and outgoing passenger data. The passenger loading conditions at different times are recorded. The number of drivers is calculated, and then the threshold and neighborhood radius are determined to achieve automatic parameter extraction. Iteration is conducted over each point at that moment.

```
Input: D - Passenger Data.
1. time=[]
2. for i in range (24):
3.   time.append(D[D['pickup_datetime'].dt.hour.isin(np.arange(i,i+1))])
4.   data=time[i]
5.   carCount=set(len(D['medallion']))
6.   MinPts = len(data) / carCount
7.   Eps = avgVelocity * avgPickupTime /
8.   for each point P in dataset time[i]:
9.      if P is visited
10.        continue next point
11.    mark P as visited
12.    NeighborPts = regionQuery (P, eps)
13.    if sizeof (NeighborPts) < MinPts
14.           mark P as NOISE
15.    else
16.           C = next cluster
17.    expandCluster (P, NeighborPts, C, Eps, MinPts)
```

Algorithm 2: The expandCluster function merges adjacent clusters. The extended category core points are added first. The points in the core point neighborhood are traversed. If the point is a core point, the class is augmented.

Inputs: P—passenger points, NeighborPts—set of neighboring passenger points for a point P, C—new clustered passenger dataset.

```
1. add P to cluster C
2. for each point P' in NeighborPts
3.    if P' is not visited
4.       mark P' as visited
5.       NeighborPts'= regionQuery(P', eps)
6.       if sizeof(NeighborPts') >= MinPts
7.          NeighborPts = NeighborPts joined with NeighborPts'
8. if P' is not yet a member of any cluster
9.       add P' to cluster C
```

Algorithm 3: The regionQuery function calculates the neighborhood.
Input: P—passenger point, Eps—neighborhood radius.
Return all points within P's Eps-neighborhood.

3 Experimental Analysis

3.1 Data Analysis and Processing

The application of GPS enables the record keeping of a taxi's itinerary [17]. This paper uses taxi passenger data from February 1, 2010, to February 3, 2010, in New York City. A total of 11711 taxis were obtained from the taxi license plate number (hack_license), and

a total of 601821 pieces of data were generated during the period. First, the date format of the data is converted, and then unreasonably wrong data are cleaned and canceled. The taxi data trajectory points are shown in Table 1.

Table 1. Taxi data parameters.

Parameter	Value
hack_license	2010000001
pickup_datetime	2010-01-01 00:00:00
dropoff_datetime	2010-01-01 00:34:00
trip_time_in_secs	34.00
trip_distance	14.05
pickup_longitude	−73.948418
pickup_latitude	40.724590

3.2 Passenger Data Distribution

First, the passenger data from February 1, 2010 to February 3, 2010 are extracted, and there are 11,711 taxis carrying 601,821 passenger data. The three-day travel time distribution is shown (in Fig. 1):

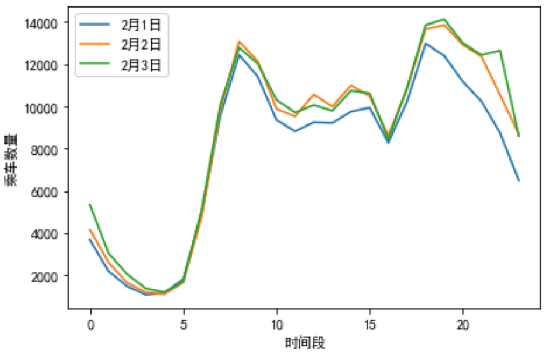

Fig. 1. Passenger time distribution.

The number of rides gradually decreases between 0:00 and 5:00. Then, it climbs quickly from 5:00 to 8:00. It drops slightly from 8:00 to 10:00 and flattens out until 16:00. After that, it climbs rapidly until 19:00. Finally, it drops rapidly until 24 o'clock. This is in line with the daily travel situation of most residents. The dataset covers the geographic space with a longitude range of [−74°03', −73°77'] and a dimensional range of [40°63', 40°85']. The map is shown. (in Fig. 1) (Fig. 2):

Fig. 2. Passenger area.

4 Comparison of Experimental Results

4.1 Division of Time and Parameter Automation

The original DBSCAN algorithm cannot divide the time and can only obtain the data results of the whole day. Without time division, the repeatability of data is too high. As a result, the distribution of ride points in some areas is too dense. The obtained clustering results combine the areas with a large amount of data and high repetition into a whole block. It is not possible to identify passenger hotspots at different times. Figure 3 shows the analysis of the total ride points without time division and some areas are too dense.

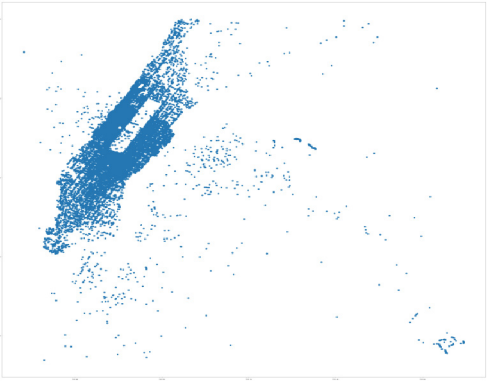

Fig. 3. Ride distribution without time division.

The results obtained by clustering are shown in Fig. 4 below. Regardless of how the parameters are adjusted, multiple time periods are mixed together. The clustering results of areas with high traffic and high repetition are not good. It is also difficult to pinpoint when drivers should pick up passengers in hotspots. Such a result cannot provide more detailed assistance for the driver to carry passengers.

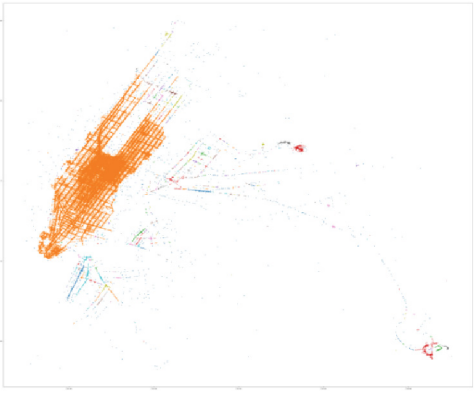

Fig. 4. Clustering results without time division.

Due to the huge passenger capacity of taxis, the time interval is divided into one-hour intervals. The passenger load situation is analyzed at 0:00-1:00 on February 1, 2010. At 0:00–1:00, the number of people and drivers is relatively small. The passenger load situation is shown in Table 2 below.

Table 2. Passengers at 0:00–1:00.

Parameter	Value
Number of taxis	2055
Number of passengers	3603
Average seeking distance	6.833 km
Average search time	19.5 min
Average number of pull loads	2
Time period selected	0:00–1:00

The distribution of passenger load points at 0:00–1:00 is shown in the figure below (Fig. 5).

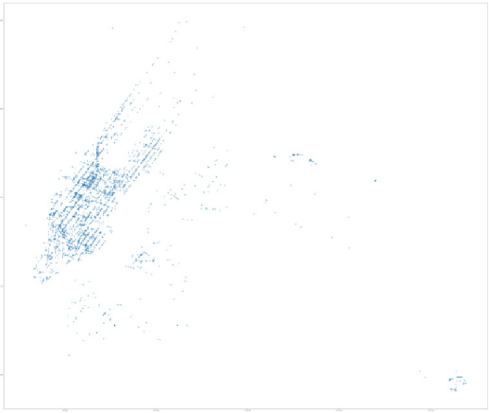

Fig. 5. Ride distribution at 0:00–1:00.

The clustering results of the A-DBSCAN algorithm are shown in the following figure (Fig. 6).

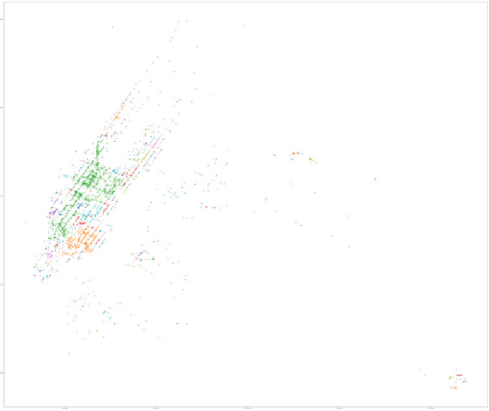

Fig. 6. A-DBSCAN clustering results at 0:00–1:00.

From 7–8 o'clock on the 1st is the peak travel time with the largest number of people and drivers. The passenger load situation is shown in Table 3 below. During peak travel times, the increase in foot traffic allows drivers to carry more passengers. Compared with the time when there is less traffic, the time to find customers is shorter. The distance to find customers is also shorter. This reflects the versatility of the A-DBSCAN algorithm for different times.

Table 3. Passenger load at 7–8 o'clock.

Parameter	Value
Number of taxis	9428
Number of passengers	3399
Average seeking distance	3.72 km
Average search time	14.1 min
Average number of pull loads	3
Time period selected	7:00–8:00

Table 4. Running time comparison

Algorithm	Running time
DBSCAN	0.5501
RL-DBSCAN	0.5366
P-DBSCAN	1.1928
A-DBSCAN	0.4463

Table 5. Davies–Bouldin index comparison

Algorithm	Davies–Bouldin Value
DBSCAN	0.9964
RL-DBSCAN	0.9962
P-DBSCAN	0.9987
A-DBSCAN	0.9988

The distribution of bus rides at 7:00–8:00 on the 1st is shown in the figure below (Fig. 7):

The clustering results of the A-DBSCAN algorithm are shown in the following figure (Fig. 8).

Through the above A-DBSCAN clustering results, it can be observed that the algorithm can obtain better clustering results in periods of high traffic and low traffic. This reflects the versatility of the A-DBSCAN algorithm and the realization of automation and conforms to the real passenger situation.

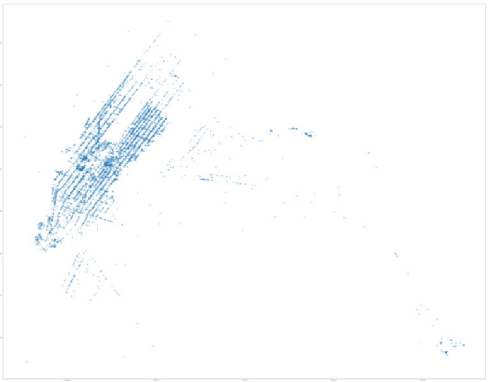

Fig. 7. A-DBSCAN clustering results at 7:00–8:00.

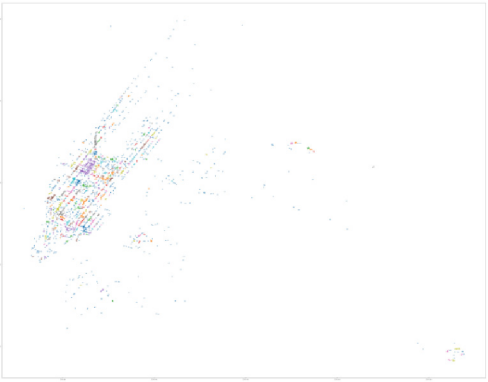

Fig. 8. A-DBSCAN clustering results at 7:00–8:00.

4.2 Comparison of Multiple Clustering Algorithms

Taking JFK International Airport as an example, a total of 1,103 drivers carried 1,829 passengers on the 1st. The distribution of rides is shown in the figure below (Fig. 9):

As seen from the above figure, some data repetitions are too high. The clustering results of the DBSCABN algorithm are shown in the following figure. Due to the high degree of data repetition without time division, the effect of clustering is not accurate. The average number of passengers picked up per driver is doubled here. The clustering results show many duplicate data as hotspots (Fig. 10).

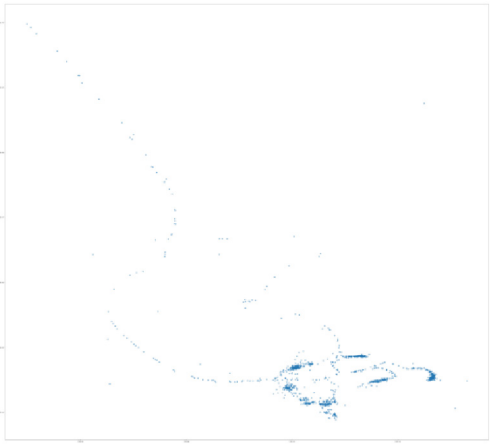

Fig. 9. JFK International Airport passenger distribution.

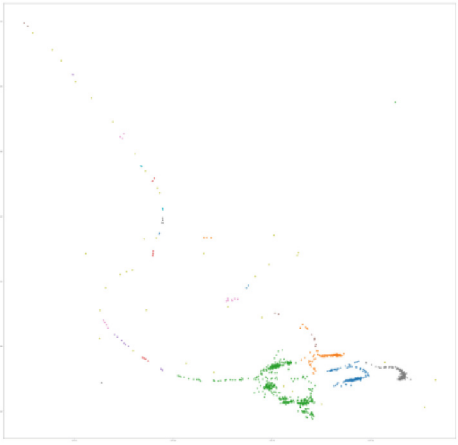

Fig. 10. DBSCAN algorithm clustering results.

The figure below shows the clustering results of the RL-DBSCAN algorithm for JFK Airport on the 1st. Although the RL-DBSCAN algorithm adds road topology constraints, the clustering results are better and fewer clusters are formed. Although the index is improved, it still does not meet the actual passenger load. The parameters also need to be debugged. Additionally, the time constraints are not considered. A large amount of data leads to the formation of dense travel hotspots. It does not correspond to the real situation which the actual average number of passengers carried is doubled (Fig. 11).

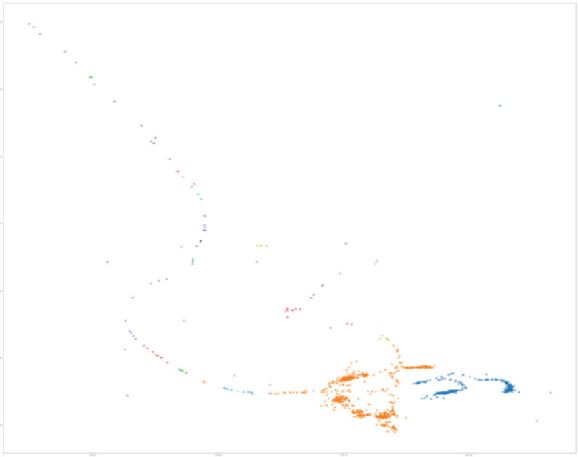

Fig. 11. RL-DBSCAN algorithm clustering results.

The figure below shows the clustering results of JFK Airport by the P-DBSCAN algorithm. The division of the region is more detailed than the results obtained by the original DBSCAN algorithm. However, the time is also not divided, which results in data that are too dense. The results obtained do not correspond to the actual passenger load situation. The P-DBSCAN algorithm will divide multiple regions and make the adjustment of parameters more complicated (Fig. 12).

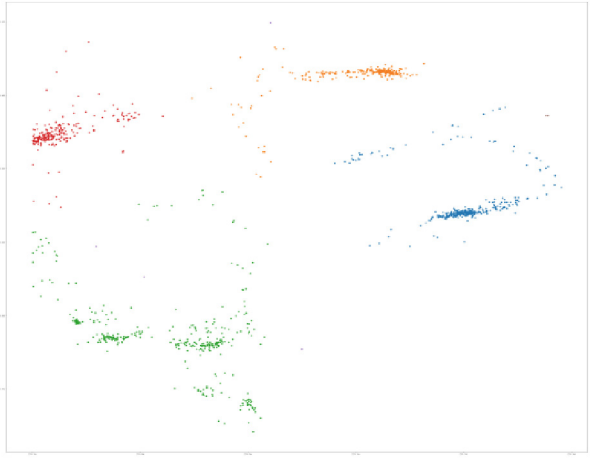

Fig. 12. P-DBSCAN algorithm clustering results.

Using the A-DBSCAN algorithm to divide the time, the passenger distribution of JFK Airport from 7:00 to 8:00 is shown in the figure below (Fig. 13):

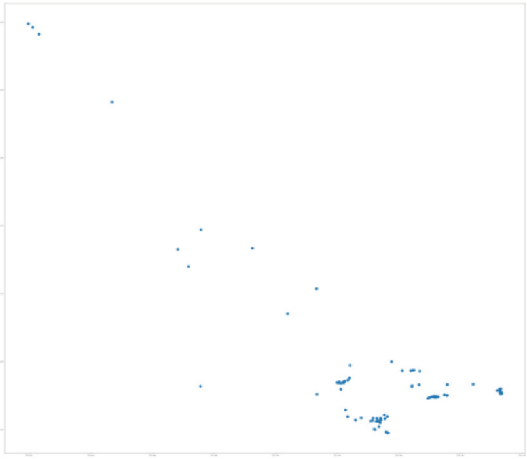

Fig. 13. Bus distribution at JFK International Airport from 7:00–8:00.

The clustering results of the airport between 7:00 and 8:00 by the A-DBSCAN algorithm are shown in the following figure. Through time division, a large number of data-intensive situations are avoided. The hotspot areas are also more accurate than those obtained from the traditional DBSCAN algorithms and RL-DBSCAN algorithms. These areas are in line with the driver's real passenger situation. At the same time, it also solves the difficult situation of manual parameter adjustment (Fig. 14).

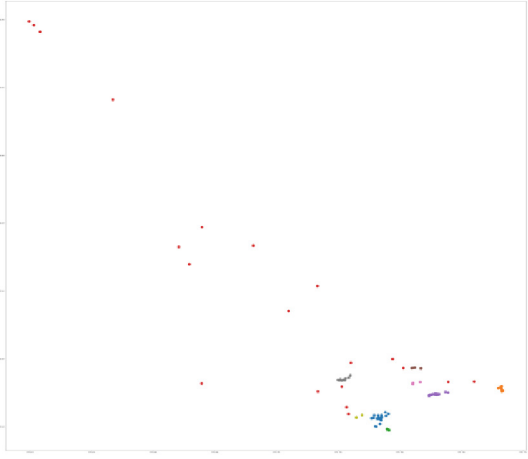

Fig. 14. A-DBSCAN algorithm clustering results.

4.3 Clustering Algorithm Performance Comparison

After the A-DBSCAN algorithm conducts time division, the data-intensive situation is reduced. The performance is also improved. The following table compares the running time of the A-DBSCAN algorithm and other algorithms for JFK Airport clustering. The running time of the A-DBSCAN algorithm is less than that of the DBSCAN, RL-DBSCAN and P-DBSCAN algorithms (Table 4).

Davies–Bouldin [18] and Silhoette [19] introduced an internal evaluation index of clustering algorithms. Different clustering algorithms are used to test the airport data. The Davies–Bouldin index is calculated by dividing the sum of the average distances within the class of any two categories and dividing the distance between the two cluster centers to find the maximum value. The smaller the index is, the better the clustering effect. The table below shows the comparison between the A-DBSCAN algorithm and other algorithms in terms of the Davies–Bouldin index (Table 5).

Comparing the above experimental results, the A-DBSCAN algorithm consumes less time. The Davies–Bouldin index results are better than those of the other algorithms. The results of several clustering algorithms in terms of the Silhoette index results are similar. The accuracy and performance of the A-DBSCAN algorithm are improved.

5 Conclusions

This paper uses New York City taxi trajectory data for analysis and processing. The traditional DBSCAN algorithm lacks time analysis and processing. This results in a large amount of data being too dense. The obtained clustering results cannot determine the passenger hotspots well. The traditional DBSCAN algorithm requires parameter adjustment that is very time-consuming and difficult in the case of multiple sets of experiments. In view of the above problems, this paper uses time division based on the DBSCAN algorithm to solve a large number of data-intensive problems and improve the accuracy of the experimental results. It analyzes the relationship between the distance of the driver to find passengers and the relationship between passengers to achieve parameter automation to solve the difficult problem of parameter input adjustment. First, the data format is dealt with. Then, duplicate data and unreasonable data are deleted. The distribution of residents' rides on multiple days is analyzed, and the results obtained reflect the regularity of travel. Then, the time is divided, and the passenger-carrying and passenger-seeking situations are analyzed at each time. The number of drivers, the passenger-seeking time, the passenger-seeking distance, and the average number of passengers carried by the drivers are extracted as the parameters of MinPts and Eps. The extracted parameters are in line with the real driver's passenger-seeking situation, and the algorithm can automatically extract the parameters. It solves the problem that multiple sets of data need a large number of input parameters to adjust parameters. The passenger loading situation is analyzed in different time periods. Through the experimental comparison and analysis, the versatility of the parameters extracted by the A-DBSCAN algorithm for different time periods is verified. Then, JFK Airport is analyzed in detail, and the results of various clustering algorithms are compared. The results show that the A-DBSCAN algorithm is more accurate and more in line with the real passenger situation. Finally,

the A-DBSCAN algorithm is compared with other algorithms in terms of time and clustering internal indicators, and both achieve better results, proving that the performance is also improved.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China under Grant 62071157, National Key Research and Development Programme 2022YFD2000500 and Natural Science Foundation of Heilongjiang Province under Grant YQ2019F011.

References

1. Shi, F., et al.: Sampling methods of resident trip investigation. J. Traffic Transport. Eng. **4**(4), 72–75 (2004)
2. Wang, L., et al.: Mining frequent trajectory pattern based on vague space partition. Knowledge-Based Systems 50. Complete, 100–111 (2013)
3. Yu, et al.: Urban computing: concepts, methodologies, and applications. ACM Trans. Intell. Syst. Technol. Spec. (2014)
4. Tang, J., et al.: Uncovering urban human mobility from large scale taxi GPS data. Phys. A **438**, 140–153 (2015)
5. Yuan, J., et al.: Discovering regions of different functions in a city using human mobility and POIs. ACM 186 (2012)
6. Cai, Y.W., Yang, B.R.: Improved DBSCAN algorithm for public bus station cluster. Comput. Eng. **34**(10), 190–192 (2008)
7. Xu, C., Zhang, A., Chen, Y.: Traffic congestion forecasting in shanghai based on multi-period hotspot clustering. IEEE Access (99), 1–1 (2020)
8. Han, Y., et al.: Exploring the temporal and spatial distribution of passengers based on taxi trajectory data. Periodical of Ocean University of China (2019)
9. Zheng, L., et al.: Mining urban attractive areas using taxi trajectory data. Computer Applications and Software (2018)
10. Bao, G.: Research and system implementation of taxi passenger hotspot recommendation method. North China University of Technology (2019)
11. Liu, P.: Research on hotspots mining of taxi passengers based on spatial clustering and Weka platform. Jilin University (2014)
12. Sun, J., Guan, C., Jinhong, M.: Exploiting optimization mechanism for pick-up points recommendations. In: International Conference on Computer Systems, Electronics and Control
13. Chakraborty, S.: Analysis and study of incremental DBSCAN clustering algorithm. Eprint Arxiv **1**(2), 2011 (2014)
14. Chen, M., Gao, X.D., Li, H.F.: Parallel DBSCAN with Priority R-tree. In: 2010 The 2nd IEEE International Conference on IEEE Information Management and Engineering (ICIME) (2010)
15. Huang, Z.: Mining and recommendation of customer-seeking areas based on taxi trajectories. Hangzhou Dianzi University (2020)
16. Yi, L.I., et al.: A weighted centroid localization algorithm based on DBSCAN clustering point density. J. Henan Univ. Sci. Technol. (Natl. Sci.) (2018)
17. Ester, M., et al.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. AAAI Press (1996)

18. Murakami, E., Wagner, D.: Can using global positioning system (GPS) improve trip reporting? Transport. Res. Part C Emerg. Technol. **7**(2–3), 149–165 (1999)
19. Rousseeuw, Peter J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987). https://doi.org/10.1016/0377-0427(87)90125-7