Distributed Density Peak Clustering of Trajectory Data on Spark

Yunhong Zheng[1], Xinzheng Niu[1](✉), Philippe Fournier-Viger[2], Fan Li[3], and Lin Gao[4]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China
2386100@qq.com

[2] School of Humanities and Social Sciences, Harbin Institute of Technology (Shenzhen), Shenzhen, Guangdong, China

[3] Key Laboratory of Fundamental Synthetic Vision Graphics and Image Science, Sichuan University, Chengdu, Sichuan, China

[4] Department of Computing Science and Technology, Southwest University of Science and Technology, Mianyang, Sichuan, China

Abstract. With the widespread use of mobile devices and GPS, trajectory data mining has become a very popular research field. However, for many applications, a huge amount of trajectory data is collected, which raises the problem of how to efficiently mine this data. To process large batches of trajectory data, this paper proposes a distributed trajectory clustering algorithm based on density peak clustering, named DTR-DPC. The proposed method partitions the trajectory data into dense and sparse areas during the trajectory partitioning and division stage, and then applies different trajectory division methods for different areas. Then, the algorithm replaces each dense area by a single abstract trajectory to fit the distribution of trajectory points in dense areas, which can reduce the amount of distance calculation. Finally, a novel density peak clustering-based method (E-DPC) for Spark is applied, which requires limited human intervention. Experimental results on several large trajectory datasets show that thanks to the proposed approach, runtime of trajectory clustering can be greatly decreased while obtaining a high accuracy.

Keywords: Trajectory data · Distributed clustering · Spark · DPC · Dense areas

1 Introduction

With the massive use of personal mobile devices and vehicle-mounted positioning devices, and the increasing demand for hidden information in trajectory data, trajectory data mining has emerged as an important branch of data mining. Due to several years of research, the trajectory mining process is now quite stable. It

H. Fujita et al. (Eds.): IEA/AIE 2020, LNAI 12144, pp. 792–804, 2020.
https://doi.org/10.1007/978-3-030-55789-8_68

consists of various sub-tasks such as classic trajectory acquisition and abnormal trajectory exploration.

Though trajectory data mining has many applications, most algorithms have been designed to deal with a small volume of data, and are thus unsuitable for applications where a large amount of data must be analyzed. In the data preprocessing stage, to protect the locality of each trajectory, a sub-trajectory segmentation strategy [1] and grid-based division [2] can be applied, but these techniques increase the amount of calculations. In the trajectory distance calculation stage, most studies directly calculate the distance between two sub-trajectories [3]. If the traditional trajectory division strategy was used in the previous stage, the amount of calculations will be further increased. Then, in the data mining stage, the most widely used model is clustering, using techniques such as k-means [4] and DBSCAN [5]. However, most clustering methods require human intervention. For example, k-means requires setting the number of clusters, while the *Eps* and *MinPts* parameters must be set for DBSCAN.

Considering these limitations of previous work, this paper proposes a novel approach for distributed trajectory clustering based on density peak clustering [6], named DTR-DPC (Distributed Trajectory-Density Peak Clustering). It is implemented on the Spark distributed computing platform to support the analysis of large trajectory data sets. The proposed method is optimized for fast calculations at all stages of the trajectory data mining process. In the trajectory partitioning and division stage, we propose to partition the data into dense and sparse areas, where sparse areas are divided into sub-trajectory segments, and entire trajectories are kept for dense areas. In the sub-trajectory similarity measurement stage, the proposed method replaces each dense area by a special trajectory called *ST* to reduce unnecessary calculations. For the clustering stage, a novel algorithm named E-DPC is applied, which requires limited human intervention and performs calculations in a distributed manner. An experimental evaluation was performed. Results show that thanks to the proposed approach, runtime of trajectory clustering can be greatly decreased while obtaining a high accuracy.

The rest of this paper is organized as follows. Section 2 surveys related work. Section 3 defines the trajectory data mining problem. Section 4 presents the propsed DTR-DPC approach. Then, Sect. 5 describes experimental results. Finally, a conclusion is drawn in Sect. 6.

2 Related Work

This section first reviews state-of-the-art techniques for trajectory data mining. Then it surveys related work on distributed trajectory clustering.

2.1 Trajectory Data Mining

Trajectory data mining models can be mainly grouped into three categories: pattern mining, classification, and clustering.

Pattern mining is a set of unsupervised techniques that are used for discovering interesting movement patterns in a set of trajectories [7]. Algorithms have been designed to extract several types of patterns such as gathering patterns [7], sequential patterns [8], and periodic patterns [9]. A gathering pattern represents the movements of groups of persons such as for celebrations, parades, protests and traffic jams [10]. A sequential pattern is a subsequence of locations that frequently appears in trajectories [8], while a periodic pattern is movements that periodically appear over time, found in trajectory data containing time information [9].

Classification consists of using supervised learning to predict the next location for a trajectory. But because trajectories generally do not have reliable labels, classification is rarely used. However, a classification based approach using Support Vector Machine (SVM) was applied for vehicle trajectory analysis and provided some good results [11].

Clustering is the most commonly used method for trajectory data mining. Most of the related clustering work fall in one of three categories: k-means-based trajectory clustering [12], fuzzy-based trajectory clustering [13], and density-based trajectory clustering [14,15]. Yeen and Lorita [12] proposed a k-means and fuzzy c-means clustering-based algorithm, which iteratively recalculates the centroids (means) of clusters in the trajectory dataset. Because k-means has the disadvantage that the number of clusters must be preset, researchers have gradually changed toward using fuzzy and density-based clustering. Nadeem et al. [13] first proposed an unsupervised fuzzy approach for motion trajectory clustering, which performs hierarchical reconstruction only in the region affected by a new instance, rather than updating all the hierarchies. Most recent trajectory clustering studies have focused on density-based clustering using algorithms such as DBSCAN [14] and density peak clustering (DPC) [15]. The biggest advantage of this type of algorithms is their high accuracy, while the biggest disadvantage is that they requires considerable human intervention.

2.2 Distributed Trajectory Clustering

An important limitation of most trajectory data mining algorithms is that they cannot handle large-scale trajectory data. To address this limitation, several parallel and distributed trajectory algorithms were proposed. Wang et al. [16] designed a Spark in-memory computing algorithm using data partitioning to implement parallelism for DPC and reduce local density calculations. Hu et al. [17] proposed a method for the fast calculation of trajectory similarity based on coarse-grained Dynamic Time Warping, implemented using the Hadoop MapReduce model to handle massive dynamic trajectory data with time information. Myamoto et al. [18] proposed a flight path generation method that uses a dynamic distributed genetic algorithm. This method differs from others in that the number of groups into which individuals are partitioned is not fixed, and hierarchical clustering is done using a distributed genetic algorithm.

Some of the most recent studies [19,20] are based on density-based clustering. Chen et al. [19] and Wang et al. [20] have used the DBSCAN and DPC algorithms, respectively. The former is implemented using Hadoop MapReduce for distributed computing, while the latter uses the Spark in-memory computing model. Compared with the former, the latter has higher computing efficiency because of the advantages of DPC's fast search strategy and Spark's memory-based operation. Though these algorithms are useful, efficiency can still be improved. Hence, this paper proposes optimizations for each stage of the trajectory data clustering process.

3 Problem Formulation

This section introduces preliminary definitions and then defines the studied problem of trajectory data clustering.

Definition 1. ***Trajectory (TR):*** *A trajectory TR_i is an ordered list of coordinate trajectory points collected at different time, denoted as $TR_i =< p_i^1, p_i^2, \ldots p_i^j, \ldots, p_i^{n_i} >$, where a superscript j is used to denote the j-th point of TR_i, and p_i^j is a pair containing a location l_i^j (longitude and latitude) and a time t_i^j.*

Definition 2. ***Sub-trajectory (STR):*** *A sub-trajectory STR of a trajectory TR_i is a list of consecutive and non-repeating points from that trajectory, denoted as $STR_i =< p_{STR_i}^{start}, \ldots, p_{STR_i}^j, p_{STR_i}^k, \ldots, p_{STR_i}^{end} >$, where $p_{STR_i}^{start}$ and $p_{STR_i}^{end}$ are the start and end point of STR_i, respectively.*

Definition 3. ***Trajectory Distance (TRD):*** *The trajectory distance (TRD) between two sub-trajectories STR_i and STR_j according to a distance metric, is denoted as $TRD_i^j = dist(STR_i, STR_j)$.*

Definition 4. ***Trajectory Clustering (TRC):*** *Given a trajectory distance matrix TRD, which specifies the distance between each sub-trajectory pair, TRC consists of dividing the sub-trajectories into clusters $C = \{C_1, C_2, \ldots, C_{N_C}\}$ based on the distance matrix TRD.*

This study uses the Davies-Bouldin Index (DBI) [21] to evaluate TRC accuracy. The DBI is defined as

$$DBI = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\overline{C_i}+\overline{C_j}}{dist(C_c^i, C_c^j)}\right) \qquad (1)$$

where $\overline{C_i}$ and $\overline{C_j}$ are the average distance inside clusters C_i and C_j respectively, C_c^i and C_c^j are the cluster centers of C_i and C_j, and k is the number of clusters.

In general, a lower DBI indicates a more accurate clustering. We define the distributed trajectory data clustering problem as follows:

Definition 5. ***(Problem Definition):*** *Let there be a set of trajectories* $TR = \{TR_1, TR_2, \ldots, TR_i, \ldots, TR_{n_{TR}}\}$*, and assume that each* TR_i *is divided into a set of sub-trajectories* $\{STR_1, \ldots, STR_i, \ldots STR_{n_{STR}}\}$*. Moverover, consider the distance matrix* TRD *between each* STR_i *and* STR_j*. The problem is to efficiently perform* TRC *using* TRD *to generate a clustering* C *having a low* DBI *value.*

4 A Distributed Density Peak Clustering Approach

To solve the defined problem, this section proposes a distributed trajectory density peak clustering algorithm for Spark, named DTR-DPC, which consists of three steps, namely, trajectory partitioning and division, sub-trajectory similarity calculation, enhanced density peak clustering. The proposed approach first partitions trajectory points into several dense or sparse grid areas. Then, approximate calculations are done in dense areas, while a traditional trajectory division strategy is applied in sparse areas. This dramatically reduce the amount of calculations. Secondly, the similarity between sub-trajectories is measured using three different methods. Finally, an enhanced density peak clustering (E-DPC) algorithm is called to cluster sub-trajectories. The two key advantages of E-DPC is that it reduces the need for human intervention and that calculations are distributed (using the Spark model).

An overview of the proposed approach is presented in Fig. 1.

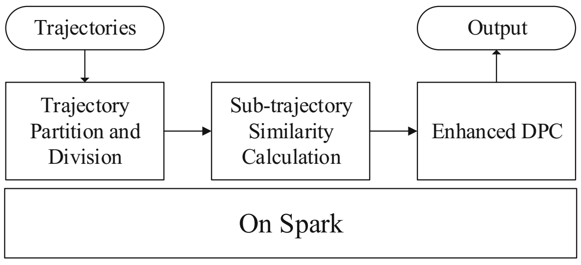

Trajectories
Output
Trajectory Partition and Division
Sub-trajectory Similarity Calculation
Enhanced DPC
On Spark

Fig. 1. Overview of the DTR-DPC approach

***Phase* 1** (trajectory partition and division): Data collection methods typically record a batch of trajectories TR containing many useless trajectory points which are continuously recorded in a small range and have no practical significance. To deal with this issue, the proposed method divides the area where trajectory points appear into several grid areas GA. The method assigns each point p_i^j to a different GA to form dense areas DA and sparse areas SA, for which different sub-trajectory partitioning strategies are applied. After that, trajectories are segmented into two different areas to form a sub-trajectory set STR.

***Phase* 2** (sub-trajectory similarity calculation): Then, using the trajectory division result of Phase 1, the similarity matrix (SM) of all sub-trajectory pairs is calculated using three different similarity measures, defined for various GA. The distance between two sub-trajectories in DA is denoted as $dist_{<da,da>}$, the distance between two sub-trajectories in SA is denoted as $dist_{<sa,sa>}$, while $dist_{<da,sa>}$ refers to the distance between a trajectory in a DA and one in a SA.

***Phase* 3** (enhanced density peak clustering): The SM matrix obtained in Phase 2 is used by an adaptive method to calculate the distance and density thresholds required by the DPC algorithm, to select some special points as center points. Then, each remaining point is assigned to its nearest center point to form a set of points called a cluster. Finally, the algorithm is optimized to run on Spark to reduce the runtime.

4.1 Trajectory Partition and Division

This first phase is performed in three steps. Firstly, a trajectory coordinate system (x, y) and grid area GA is built. Then, each point of the trajectory data is assigned to different grid areas to form dense areas and sparse areas based on a threshold Th_{NP} indicating a minimum number of points per area. Finally, different trajectory division strategies are applied for different area types to obtain the sub-trajectory set.

The trajectory coordinates system (x, y) is mainly used to map trajectory points to the two-dimensional coordinate axis. The X and Y axis values of a point denotes longitude and latitude, respectively. Furthermore, x-axis and y-axis scale values $|x_{scale}|$ and $|y_{scale}|$ are defined as:

$$|x_{scale}| = \frac{x_{max} - x_{min}}{NE} \tag{2}$$

$$|y_{scale}| = \frac{y_{max} - y_{min}}{NE} \tag{3}$$

where x_{max} and y_{max} represent the maximum longitude and latitude in all p_i^j, respectively, x_{min} and y_{min} represent the minimum longitude and latitude in all p_i^j, respectively, and NE is a number of equal parts, which needs to be set by the user.

Then, the method set grid lines at each x_{scale} and y_{scale} increment on the X and Y axis, respectively, to form GA, and each point p_i^j is assigned to different GA to form DA and SA, indicating that the number of points NP falling in an area is greater or smaller than the Th_{NP} threshold, respectively. An example of DA and SA partition diagram is shown in Fig. 2.

After partitioning trajectories, the algorithm divides trajectories for different areas to protect the locality of each trajectory and reduce the runtime. The algorithm scans all the trajectory segments of each entire trajectory TR_i, excluding all sub-trajectory segments that pass through DA, to save all the sub-trajectory segments that only pass through SA. The result is a set

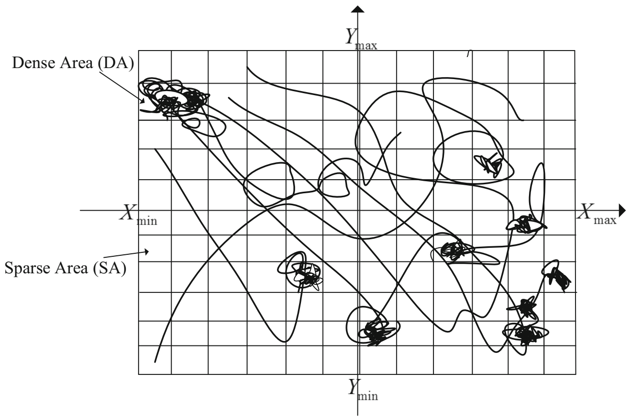


Fig. 2. A DA and SA partition diagram.

$SATR_i = \{SATR_i^1, \ldots, SATR_i^j, \ldots, SATR_i^n\}$, where $SATR_i^j$ represent the j-th sub-trajectory in TR_i, which only pass through SA.

All sub-trajectories that only pass through one dense area DA_i, are replaced by an abstract trajectory $DATR_i$, which is represented as:

$$DATR_i =< DA_i^{left_{lower}}, DA_i^{center}, DA_i^{right_{upper}} > \tag{4}$$

where $DA_i^{left_{lower}}$, $DA_i^{right_{upper}}$, DA_i^{center} represents the left lower corner point, right upper corner point, and the center point of DA_i's center area, respectively.

For each sub-trajectory $SATR_i^j$, based on the MDA trajectory division strategy [1], the algorithm adds corner angles to obtain a better sub-trajectory division. A corner angle θ is defined as:

$$\theta = \begin{cases} \pi - \alpha, & \alpha \leq \dfrac{\pi}{2}, \\ \alpha - \pi, & \alpha > \dfrac{\pi}{2}, \end{cases} \tag{5}$$

A schematic diagram of this trajectory division approach is shown in Fig. 3, where α is calculated as:

$$\alpha = arccos((a^2 + b^2 - c^2)/2ab). \tag{6}$$

Calculations made using the MDA strategy produce a trajectory division result $SASTR_k =< p_{SASTR_k}^{start}, \ldots, p_{SASTR_k}^{j}, \ldots, p_{SASTR_k}^{end} >$ of $SATR_i^j$. Hence the sub-trajectory set STR can be rewritten as:

$$\{DATR_1, \ldots, DATR_i, \ldots, DATR_{n_{DA}}, SASTR_1, \ldots, SASTR_j, \ldots, SASTR_{n_{SA}}\} \tag{7}$$

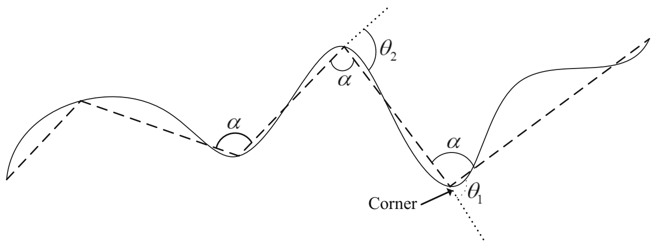


Fig. 3. The schematic diagram of trajectory division

where n_{DA} refers to the number of abstract sub-trajectories representing trajectories passing through DA and n_{SA} is the number of sub-trajectories obtained by dividing $SATR$.

4.2 Sub-trajectory Similarity Calculation

A sub-trajectory similarity measure is then applied to calculate the distance between each pair of trajectories. Because the proposed method handles two types of sub-trajectories ($DATR$ and $SASTR$), three types of distance are used: $dist^{<i,j>}_{<da,da>}$ between $DATR_i$ and $DATR_j$, $dist^{i,j}_{<da,sa>}$ between $DATR_i$ and $SASTR_j$, and $dist^{<i,j>}_{<sa,sa>}$ between $SASTR_i$ and $SASTR_j$.

For instance, two sub-trajectories $DATR_i$ and $DATR_j$ are shown in Fig. 4(a), and the distance $dist^{<i,j>}_{<da,da>}$ between two such sub-trajectories is calculated as:

$$dist^{<i,j>}_{<da,da>} = \beta dist_1 + \gamma dist2, \quad \beta + \gamma = 1, \tag{8}$$

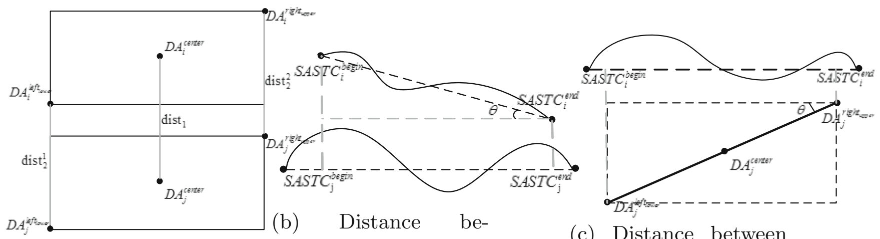
where β and γ are distance weights, $dist_1$ is the distance between DA^{center}_i and DA^{center}_j, and $dist_2$ represents the average distance between the remaining two points $DA^{right_{upper}}$ and $DA^{left_{lower}}$.

For example, two sub-trajectories $SASTR_i$ and $SASTR_j$ are depicted in Fig. 4(b). The distance $dist^{<i,j>}_{<sa,sa>}$ between two such sub-trajectories is calculated using the MDA strategy [1], which is defined as :

$$\begin{aligned} dist^{<i,j>}_{<sa,sa>} = \omega_\perp \cdot dist_\perp(SASTR_i, SASTR_j) + \\ \omega_\parallel \cdot dist_\parallel(SASTR_i, SASTR_j) + \omega_\theta \cdot dist_\theta(SASTR_i, SASTR_j) \end{aligned} \tag{9}$$

where $\omega_\perp$, $\omega_\parallel$ and ω_θ are the weights of three distance types. Those are $dist_\perp(SASTR_i, SASTR_j)$, $dist_\parallel(SASTR_i, SASTR_j)$ and $dist_\theta(SASTR_i, SASTR_j)$ of the MDA strategy [1], which are called vertical distance, parallel distance and angular distance.

Two sub-trajectories $DATR_i$ and $DASTR_j$ are shown in Fig. 4(c). The distance $dist^{<i,j>}_{<da,sa>}$ between two such sub-trajectories is also calculated using the MDA strategy [1], which treats $DA^{left_{lower}}_i$ as $p^{start}_{SASTR_j}$, and treats $DA^{right_{upper}}_i$ as $p^{end}_{SASTR_j}$.



(a) Distance between two sub-trajectories $SASTR_i$ and $SASTR_j$.

(b) Distance between two sub-trajectories $SASTR_i$ and $SASTR_j$.

(c) Distance between two sub-trajectories $DATR_i$ and $SASTR_j$.

Fig. 4. The three distance types.

4.3 Enhanced Density Peak Clustering

To improve the efficiency of clustering trajectory data and reduce the impact of artificially set parameters on the DPC algorithm, this paper proposes an enhanced DPC algorithm, called E-DPC. It uses the Log-Normal distribution to adaptively select a density threshold $th_{density}$ and a distance threshold $th_{distance}$. The Log-Normal distribution is used instead of other distributions because its shape is narrow and upward, which means that we may choose more cluster centers to offset side-effects of data reduction in dense areas.

To apply DPC, the algorithm then selects some special trajectories as center points. Remaining sub-trajectories are then assigned to their nearest center points to form clusters. This assignment of a trajectory is done by considering the local density ρ_i and the minimum distances from any other trajectory with higher density δ_i. The greater the ρ_i and δ_i are, the more likely a sub-trajectory is to be a center point. However, a drawback of this traditional approach is that the thresholds of ρ_i and δ_i need to be set by hand.

To address this issue, the proposed aproach performs Log-Normal distribution fitting on all the calculated ρ and δ sets, and takes the upper quantile of the Log-Normal distribution as the local density threshold $th_{density}$ and distance threshold $th_{distance}$. The Log-Normal distribution is shown in Fig. 5, and ρ's Log-Normal distribution is:

$$f(\rho;\mu;\sigma) = \frac{1}{\sqrt{2\pi}\rho\sigma} e^{-\frac{(\ln\rho-\mu)}{2\sigma^2}} \tag{10}$$

5 Experimental Evaluation

This section reports results of experiments comparing the proposed TDR-DPC approach with two distributed trajectory clustering algorithms from the literature, namely DBSCAN [19] and DPC [16]. A Spark platform was used, having

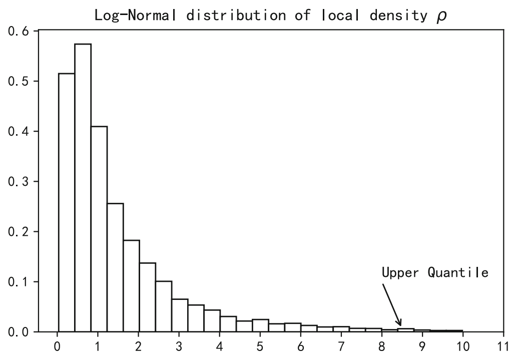


Fig. 5. The Log-Normal distribution of local density ρ

one master and five slaves, where each node is equipped with 8 GB of memory, a 500 GB hard disk and a Core i7@2.3 GHz CPU. The software settings are: Hadoop 2.7.7, Spark 2.4.4 and Python 2.7.11. The three compared algorithms were applied to two large-scale trajectory datasets. The first dataset (Gowalla) was obtained from a location-based social networking website where users share their locations by checking-in. It contains 6,442,890 check-ins with GPS information. The second dataset (GeoLife) consists of three years of trajectory data for 182 participants of the Geo-Life project. Both datasets are too big to be mined on a typical desktop computer.

An experiment was done to evaluate the influence of NE on the proposed method, and compare its performance to other methods. The parameter Th_{NP} was set to 1500, the distance weights β and γ were set to 0.6 and 0.4, respectively, and the distance weights $\omega_{\perp}$, $\omega_{\|}$ and ω_{θ} were set to 0.33. Then, the following algorithms were ran on each dataset: DBSCAN, DPC, DTR-STR($NE = 200$), DTR-STR($NE = 400$) and DTR-STR($NE = 800$). The DBI clustering accuracy and runtimes were recorded for each algorithm. This experimental design can not only verify the quality of the algorithm but also verify the impact of the partition size NE on the algorithm. Results in terms of clustering accuracy and runtimes are shown in Table 1 and Fig. 6.

Table 1. Clustering accuracy comparison (DBI)

Dataset	DBSCAN	DPC	DTR-DPC ($NE = 200$)	DTR-DPC ($NE = 400$)	DTR-DPC ($NE = 800$)
Gowalla	0.752	0.683	0.712	0.695	0.689
GeoLife	0.779	0.724	0.751	0.746	0.737

It can be observed in Table 1 that all DTR-DPC algorithms are slightly more accurate than the DPC algorithm, but both are less accurate than DBSCAN. These results are obtained because the DTR-DPC algorithm deletes

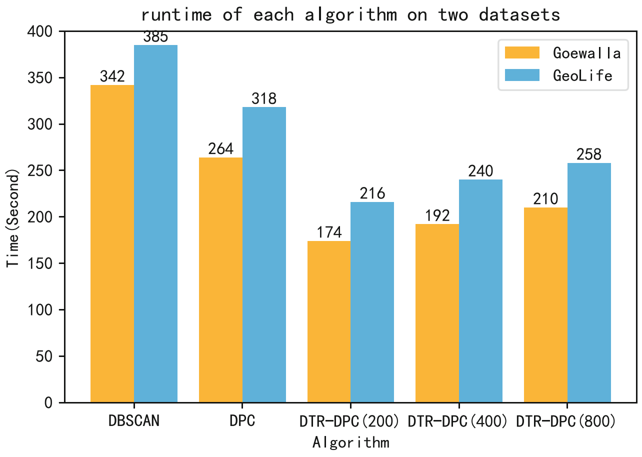


Fig. 6. Runtime comparison (s)

and abstracts numerous trajectory points. This has only a slight effect on clustering results because many points of a dense area are repeated stay points of the same trajectory, which has little effects on trajectory data mining. Moreover, it can also be observed that the larger the partition size NE of GA, the closer the accuracy is to that of the DPC algorithm. The reason is that the larger the NE is, the fewer points are deleted, and the greater the similarity is with the original data. In Fig. 6, the runtime of DTR-DPC is much less than those of the other two algorithms, and the larger the NE is, the longer the runtime, which was expected. In summary, NE is a value that needs to be reasonably set. If NE is set to a larger value, it may be faster but the accuracy may decrease. In future work, we will consider designing a methodology to automatically set NE. The experimental results reported in this paper have demonstrated that the proposed algorithm considerably reduces the runtime while preserving accuracy.

6 Conclusion

This paper proposed a distributed trajectory clustering approach based on density peak clustering on Spark, named DTR-DPC, which preprocesses data using a concept of dense and sparse areas, to reduce the number of invalid trajectories and calculations. Several different sub-trajectory distance measures are defined to compare sub-trajectories. Then, an enhanced E-DPC clustering algorithm is run on Spark to make large-scale trajectory data processing possible. Extensive experiments demonstrated that DTR-DPC guarantees almost constant clustering accuracy, while greatly reducing the runtime, and can perform large-scale trajectory data mining. In the future work, we will focus on finding the relationship between the number of equal parts NE and the size of the dataset to obtain maximum efficiency.

Acknowledgments. This research is sponsored by the Scientific Research Project of State Grid Sichuan Electric Power Company Information and Communication Company under Grant No. SGSCXT00XGJS1800219, and the Joint Funds of the Ministry of Education of China.

References

1. Lee, J.G., Han, J.: Trajectory clustering: a partition-and-group framework. In: Proceedings of ACM SIGMOD international conference on Management of data, pp. 593–604 (2007)
2. Wang, Y., Lei, P.: Using DTW to measure trajectory distance in grid space. In: IEEE International Conference on Information Science and Technology, pp. 152–155 (2014)
3. Lindahl, E., Hess, B., van der Spoel, D.: GROMACS 3.0: a package for molecular simulation and trajectory analysis. J. Mol. Model. **7**, 306–317 (2001). https://doi.org/10.1007/s008940100045
4. Huey-Ru, W., Yeh, M.-Y.: Profiling moving objects by dividing and clustering trajectories spatiotemporally. IEEE Trans. Knowl. Data Eng. **25**(11), 2615–2628 (2013)
5. Li, X., Ceikute, V.: Effective online group discovery in trajectory databases. IEEE Trans. Knowl. Data Eng. **25**(11), 2752–2766 (2013)
6. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**(6191), 1492 (2014)
7. Zheng, K., Zheng, Y.: On discovery of gathering patterns from trajectories. In: IEEE International Conference on Data Engineering, pp. 242–253 (2013)
8. Pei, J., Han, J.: Mining sequential patterns by pattern-growth: the prefixSpan approach. IEEE Trans. Knowl. Data Eng. **16**(11), 1424–1440 (2004)
9. Lin, T.H., Yeh, J.S.: New data structure and algorithm for mining dynamic periodic patterns. In: IET International Conference on Frontier Computing. Theory, Technologies and Applications, pp. 55–59 (2010)
10. Zheng, K., Zheng, Y.: Online discovery of gathering patterns over trajectories. IEEE Trans. Knowl. Data Eng. **26**(8), 1974–1988 (2014)
11. Boubezoul, A., Koita, A.: Vehicle trajectories classification using support vectors machines for failure trajectory prediction. In: International Conference on Advances in Computational Tools for Engineering Applications, pp. 486–491 (2009)
12. Choong, M.Y., Angeline, L.: Modeling of vehicle trajectory using k-means and fuzzy c-means clustering. In: IEEE International Conference on Artificial Intelligence in Engineering and Technology, pp. 1–6 (2018)
13. Anjum, N., Cavallaro, A.: Unsuoervised fuzzy clustering for trjectory analysis. IEEE Int. Conf. Image Process. **3**, 213–216 (2007)
14. Zhao, L., Shi, G.: An adaptive hierarchical clustering method for ship trajectory data based on DBSCAN algorithm. In: IEEE International Conference on Big Data Analysis, pp. 329–336 (2017)
15. Ailin, H., Zhong, L.: Movement pattern extraction based on a non-parameter clustering algorithm. In: IEEE 4th International Conference on Big Data Analytics, pp. 5–9 (2019)
16. Wang, N., Gao, S.: Research on fast and parallel clustering method for trajectory data. In: IEEE International Conference on Parallel and Distributed Systems. pp. 252–258 (2018)

17. Hu, C., Kang, X.: Parallel clustering of big data of spatio-temporal trajectory. In: International Conference on Natural Computation, pp. 769–774 (2015)
18. Miyamoto, S., Matsumoto, T.: Dynamic distributed genetic algorithm using hierarchical clustering for flight trajectory optimization of winged rocket. In: International Conference on Machine Leraning and Applications, pp. 295–298 (2013)
19. Chen, Z., Guo, J.: DBSCAN algorithm clustering for massive AIS data based on the hadoop platform. In: International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration, pp. 25–28 (2017)
20. Wang, N., Gao, S.: Research on fast and paraller clustering method for trajectory data. In: IEEE International Conference on Parallel and Distributed Systems, pp. 252–258 (2018)
21. Davies, D.L., Bouldin, D.W.: A cluster seperation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 224–227 (1979)