**Bloom Filter-based Index Structure of Massive Trajectory Data**

Zengjie Wanga, Wen Luoa,b,c, Linwang Yuana,b,c, Hong Gaoa, Fan Wua, Xu Hua, Zhaoyuan Yu a,b,c\*

*aMinistry of Education, Key Laboratory of Virtual Geographic Environment (Nanjing Normal University), Nanjing, China;*

*bState Key Laboratory Cultivation Base of Geographical Environment Evolution (Jiangsu Province), Nanjing, China;*

*cJiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing, China*

The correspondence author is Zhaoyuan Yu, email: yuzhaoyuan@njnu.edu.cn.

Abstract

Trajectory query and index commonly use the point as the minimum object, which may lead to performance issues for massive data, especially the precise query of trajectories. When creating an index, the overlap of index spaces cannot always be resolved. To this end, an indexing structure of trajectory is developed on the foundation of the Bloom Filter. We encode the different point sequences as the unique strings using the geo-grid, which transforms the precise query of trajectory into a string searching issue. Based on this index, it’s no need for additional I/O on the raw trajectory when querying a track. And relative to the original data, the memory occupied by the index is negligible. Many experiments suggest our method has good performance and stability, the complexity of time is near to  and it is hardly affected by grid size, original data size, number of points contained in each track. With such time advances, our method can be used for time-critical spatial computation tasks in applications of trajectory.

Keywords: Precise trajectory query, Geography big data, Bloom Filter, Travel record query

# Introduction

Trajectory data, such as taxi track data, motion track data, etc., are rapidly accumulated and widely been used in the area of road congestion detection[1], path planning [2], travel time prediction[3, 4], etc.. With the rapid growth of trajectory data, it has brought great suffering to index and query these data. So reshaping the index structure of trajectory is particularly important to improve query speed and efficiency.

The spatial index is mainly technology to improve the performance of the retrieval of trajectory, which is according to organize and manage spatial objects (points, lines, and polygons). In terms of improving retrieval efficiency, the current index structure has mainly made a series of studies in the following aspects, geo-space partition, minimizing coverage of retrieval region, and reducing search region and trajectory processing. When indexing massive trajectories, by dividing the space into different regions, each region is indexed separately, thereby improving retrieval efficiency. Index overlap is a problem often encountered in trajectory indexing, especially based on the tree structure, and the degree of overlap increases dramatically with the increase in the amount of trajectory. The overlap of index space inevitably results in an increase in the depth of the tree and the storage space, which leads to an increase in traversal time and query efficiency. Therefore, reducing the overlap rate of trajectory index, or even avoiding overlap completely, is an effective way to improve index efficiency. There are a large number of blank areas with no point objects in the geo-space. For example, most of the vehicle trajectories are only distributed in the road-networks. When constructing the index, exclude areas without data as much as possible, so as to reduce the retrieval time. There are a lot of noisy points and repeated recorded points in the trajectory, and these points need to be processed before the index is constructed, thereby improving the performance of the index.

Geo-space partition, that uses the grid to divide the whole region into sub-region. This includes SETI[5], SEB-Tree[6], etc. The SETI (Scalable and Efficient Trajectory Index) structure[5] proposed by V. Prasad Chakka et al. is that geo-space is partitioned into non-overlapping regions, and for each region, it builds a specific index which could be the R-Tree [7]. But in each sub-region, R-Tree will bring retrieval region overlap. SEB-tree index structure tries to represent trajectory segments by its start and ends timestamp within a spatial grid. Although space partition has reduced query time and has a good performance for regular spatial data, it hard to process the unregular and random trajectory.

The main access methods about minimizing coverage of retrieval space are R-Tree and its variants[7–9]. The R-tree index structure[7], proposed by Guttman, uses quadratics and linear algorithms to split spatial objects. But in this tree structure, pages (MBR, minimum bounding rectangle) overlap a lot. Greene's[10] split method, proposed by Greene, pages overlap much less than with Guttman's strategy. R\*tree[8], proposed by Beckmann, using the topological split algorithm to reduce pages overlap and the reinsertions further optimized the tree has a better performance for the index of spatial points. Bulk loaded R\*tree using Sort-Tile-Recursive (STR)[9]. For querying a single point, the leaf pages do not overlap at all, but it is hard to process the point sequence. R-tree and its variants do not use to index trajectory directly, but most indexing structures make it a core and important part.

Trajectory is randomly distributed in the whole Euclidean space in principle, but most vehicles or moving objects usually move on geo-networks(road), instead of on the whole spaces. Researchers have proposed many index structures based on fixed geo-networks. For objects constrained to move on fixed networks, Frentzos[11] proposed FNR-tree (Fixed Network R-tree), which is to construct a 1-dimensional R-trees on top of a 2-dimensional R-Tree to index trajectory segments. Li Guohui and Zhong Xiya[12] proposed an index structure call IMTFN (Indexing Moving objects Trajectories on Fixed Networks), which is based on the FNR-Tree and improves it. Victor Teixeira De Almeida and Ralf Hartmut Güting[13] proposed MON-Tree(Moving Objects in Networks Tree) index structure, which uses the top-level two-dimensional R-Tree and a hash table structure to index the road sections in the traffic network at the same time. The index based on the fixed network can reduce the overlap of index space and redundancy to some extent. However, the trajectory index based on the fixed-network still has the problem of index coverage, because most index structures had to maintain an R-tree as its core part.

Different from the index structures mentioned above, this paper proposes a direct track-oriented index structure, that is, using the Bloom Filter to index the trajectory. Although the Bloom Filter can be applied to a fast query of large-scale data, the key issue is how to query the track with the Bloom Filter. To our best knowledge, there is no existing solution to index the trajectory based on Bloom Filter. In the existing studies, Xiang et al.[16] proposed a method that combined Geohash with B+tree to encode the trajectory and to organize and query it. For the coding of the trajectory, it provides a train of thought for Bloom Filter.

In this paper, a direct track-oriented index structure is presented in this paper. The second part of this paper is the basic idea; the third part is the specific description and implementation of the index method; the fourth part is the explanation of the experimental results; the last part is the conclusion and discussion.

# Problem definition and basic idea

## Problem Definition

Fig.1 gives an illustration of a GPS trajectory. The Trajectory is the path of a body as it travels through space, which consists of plenty of points. While the track is a mark left by something that has passed along, such as the track of a ship, the track of a car. That is to say a trajectory is made up of a track or points sequence. For the original taxi’s trajectory data, we can extract a series of independent tracks from which each track can be represented as a single travel record, as depicted in the top part of Fig.1.



**Fig. 1** GPS trajectory and corresponding travel track or record

Lets  denote start of the trajectory, and  denote end time of the trajectory. The track of a GPS track is a sequence of points between the point  and the point . Put formally,

 (1)

The given GPS trajectory data set , where the travel record is represented as the space-time sequenceof the trajectory point, represents a single trajectory point, represented by tuple , where  represents car’s ID,  represents the time of the trajectory point  records,  represents the latitude coordinate of  coordinate, and  represents the longitude coordinate of , as depicted in the bottom part of Fig.1. The travel record  to be queried is represented as . The geographical grid is defined as  and the range is represented as , where  is the minimum spatial range of the coordinates of all trajectory points in the trajectory set , and the cell size in the geographic grid is represented as  in meters. The number of the grid is divided into longitude direction and latitude direction from the beginning of the upper right corner of the grid, which is represented as . For example, in Fig.2, based on the trajectory path represented by the geographical grid and show how to map trajectory sequence to Bloom Filter.  represents a travel records, each of them contain several trajectory points, such as trajectory  would be encoded as .



**Fig. 2** Schematic diagram of geographical grid representation of the trajectory

## Basic Idea

The general trajectory index takes spatial points as the minimum retrieval object, instead of the point sequence (travel record). But spatiotemporal trajectory is a sequence of records of the position and time of a moving object[20]. And each travel record is composed of a series of continuous spatiotemporal points. A sequence composed of a series of points, which can be regarded as a single independent object in the query process. The trajectory is irregular data in spatiotemporal distribution. To facilitate indexing, it’s necessary to reduce high-dimensional complex unstructured trajectory to low-dimensional and structured. So the geo-space is divided into regular cells by geo-grid and each gird has a unique independent encoding, the points located in the same cell can be regarded as the same point. The points in a travel record is stream data in time and space are very similar to text stream data, all of which are composed of a series of points/characters, the text string and the travel record have special meanings.

Bloom Filter has great advantages in string retrieval, and it represents a specific string through the combination of several bits, which makes the memory be utilized efficiently, and the query of Bloom Filter is implemented with some independent hash functions, and the time complexity is . This advantage is particularly obvious when dealing with big data. So each travel record will be encoded as a unique string, and each travel record is represented in the form of a string instead of point sequence. After coding and converting the travel record to a string, the travel record can be indexed by Bloom Filter. The judgment of different travel record would be changed from spatiotemporal logical judgment to Boolean decision. And using Bloom Filter can completely avoid index space overlapping problem.

This paper proposes an index structure of massive trajectory data based on Bloom Filter, using geo-Grid to code each travel record to form a unique travel record string, and then storing travel records in the bit array of Bloom Filter, so the index method of the travel record is constructed. For the retrieval of the travel records, as long as the Boolean judgment is carried out, a series of unnecessary processing operations between trajectories are not needed. The query method of trajectory points sequence (travel record) based on Bloom Filter mainly involves the creation of geographical grid, the segmentation of trajectories based on time and topology, the sub-trajectories coding, the creation of Bloom Filter, the query of trajectory points sequence and the updating of trajectory data, as shown in Fig. 3.

# Method

## Bloom Filter

The query time of Bloom Filter is under a constant range and the cost of storage space is small, so it has good practical value. Since it was put forward in 1970[21], it has been widely used in big data queries to improve query efficiency and reduce memory overhead. It is mainly used in the data dictionary, data judgment, collection and intersection, such as dictionary query[22], Spam Filtering system[23] and so on.

The Bloom Filter uses several bits to represent the elements in the collection. The essence of the algorithm is to map the string elements in the collection into a bit array through k independent hash functions. As depicted in the right part in Fig. 2,  it represents the encoded string of the trajectory point sequence (travel record). The travel record string obtains multiple addresses in the bit array through multiple hash operations and sets the corresponding value on the address to 1.

## The geographic grid and geocoding

The geographic grid (geo-Grid) is a unified and simple geospatial partition and positioning reference system. According to the unified rules, the area is continuously divided according to a certain latitude and longitude or distance to form regular polygons. Each polygon is called a grid or cell (a unit grid/cell) and gives unique coding to each cell. The geographical grid used in this method is regular. Other forms of the grid (irregular grid, Voronoi, etc.) can also be used. The selection of the trajectory of the study area also can be based on its spatial range (such as all the GPS trajectory data in Chaoyang district, Beijing, China) or its time range (such as the GPS trajectory data of 9: 00 to 11:00 in the morning). The cell size of the geographic grid and study area is customized by the real problem and users optionally. The geographical grid is created to code the points sequence. After the geo-grid is created, each grid has a unique code, and each travel record will get a unique code. Each of these travel records includes the vehicle ID, which can ensure that each travel record is unique. The trajectory geocoding process is shown in Fig. 2.

## Bloom Filter-based indexing and query process

Unlike the traditional index structure, the index table in the Bloom Filter is represented by a bit array. The elements calculated by hash only occupy a few bits of the bit array, and one element (travel record) is represented by the combination of several bits, and a bit can also be shared by multiple elements, as shown in Fig. 2. The Bloom Filter is created according to the string set formed after geo-coding. The creation of the Bloom Filter is mainly to construct its bit array, an array composed of bits, which is used to store the results of hash function calculation. After the k hash operations of each travel record string, the travel record string set will be mapped to the bit array of Bloom Filter, and the index of the travel record will be constructed.

The query method of the travel record based on Bloom Filter, that is, to judge whether the input travel record exists in the set of the current trajectory. Because the input travel record is the original coordinate point sequence, it is necessary to use the geo-Grid to encode the points. The results of the geocoding operation are mapped to the bit array one by one, and it is judged that if the result of the hash operation has one bit different from the bit array, it can be concluded that the input travel record is definitely not in the trajectory set, and if it is all the same with the bit array, it can be judged that the path is most likely to exist in the trajectory set. The whole indexing process is shown in Fig. 3.

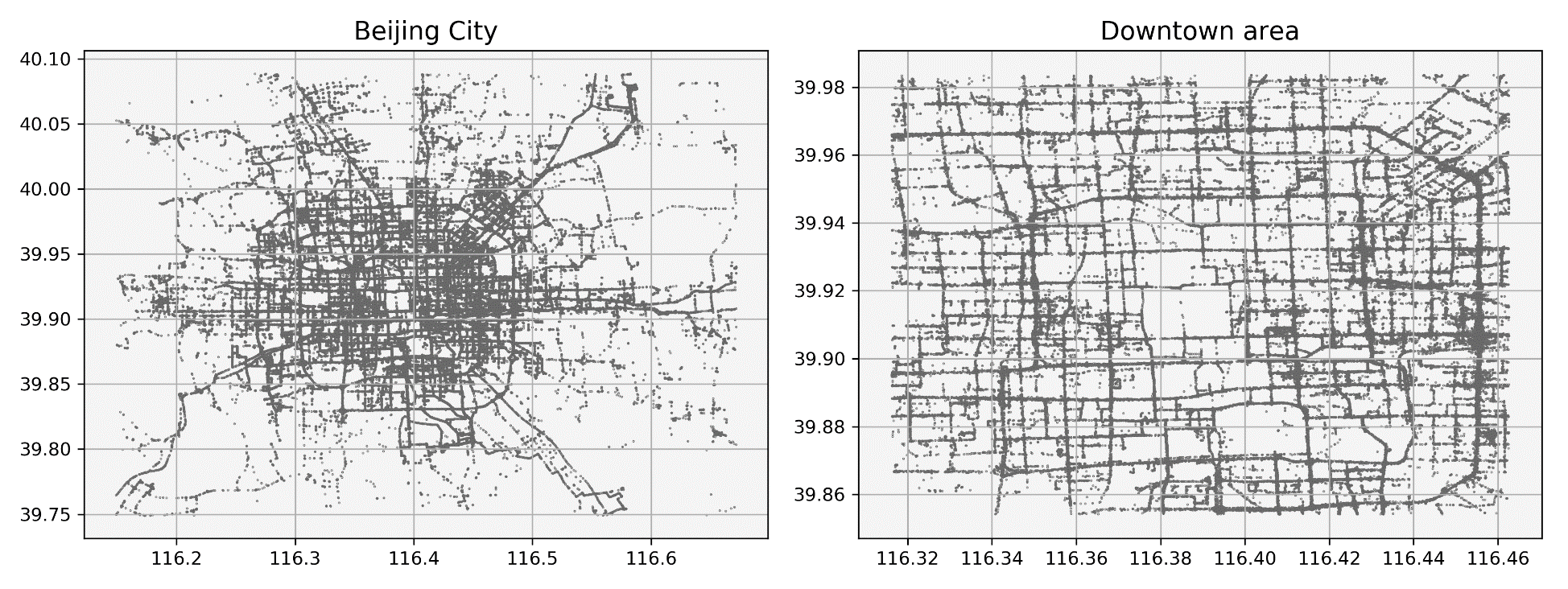


**Fig. 3** Frame Diagram of Index trajectory Point sequence of Bloom Filter

# Experiments and results

## Experiment description

The data used in this study come from the T-drive taxi trajectory data set provided by Microsoft Research Institute[24, 25], as shown in Fig. 4. Because each taxi trajectory is a long period of space-time point sequence, it needs to be segmented in order to simulate the travel records of the vehicle. According to the topological relationship, time threshold, and other segmentation methods, the travel records are formed by several points of one taxi trajectory. The segmented trajectory constitutes the vehicle travel record set. After the travel record is encoded according to the geographical grid, the travel record string set would be formed.



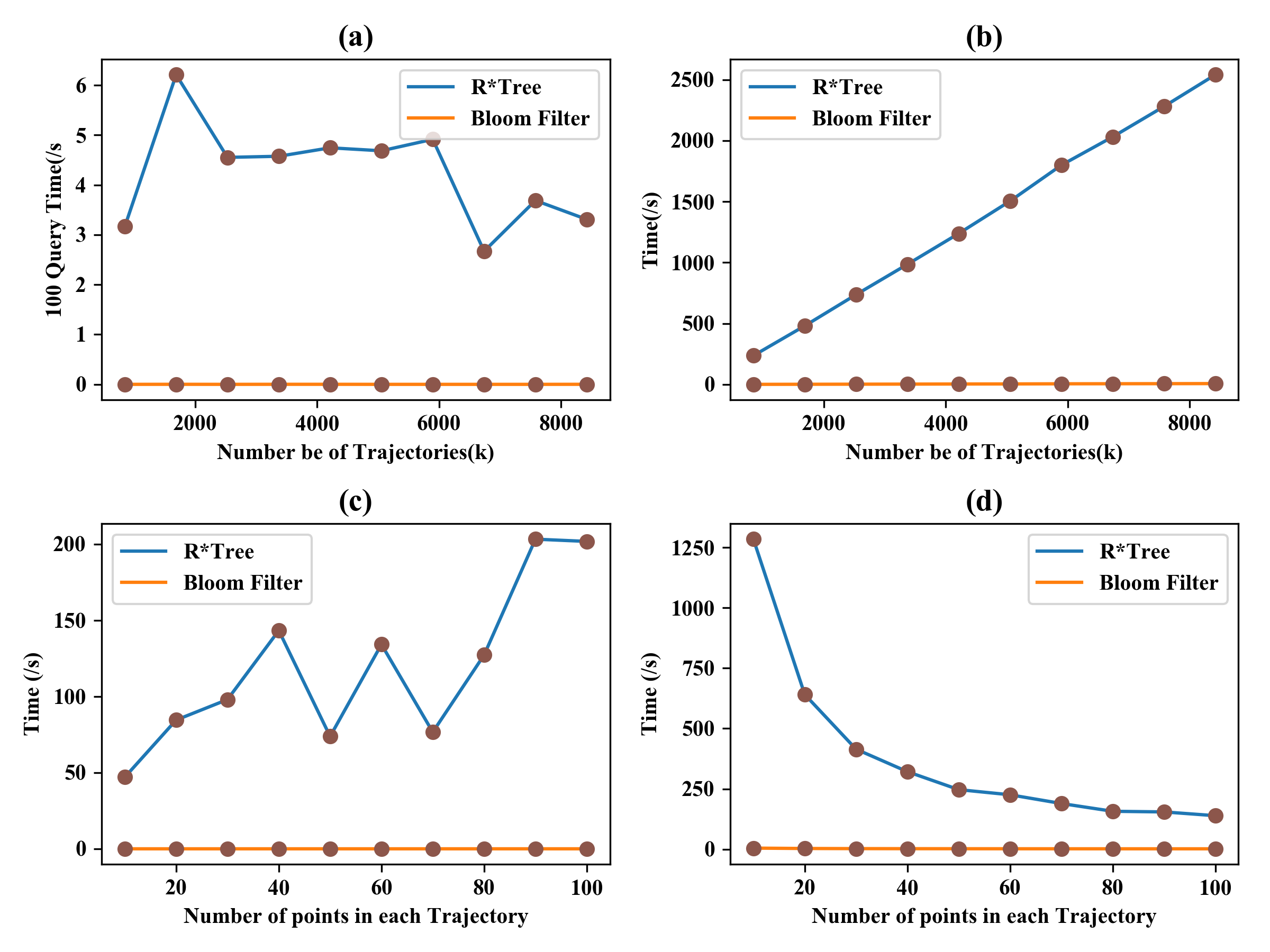
**Fig. 4** Trajectory data overview of study area (Beijing City and its Downtown area)

The experimental platform is Windows 10, the processor is Intel (R) Core (TM) i7-3770 CPU @ 3.40 GHz, running memory for 8GB, and the compiler version is MS VC++ 12.0. The third-party library of the Bloom Filter is ArashPartow/bloom[26]. The R\*tree index is Boost.Geometry.Index[27]. The experimental data are the 2008 Beijing taxi trajectory data[24, 25].

## Results

This method verifies the query efficiency of the method by comparing with the R\*tree index. For a precise query of a trajectory, the query time based on Bloom Filter is much fewer than the R\*tree index. Fig. 5(a) shows the comparison of the query time of R\*Tree and Bloom Filter with different amount of data. The horizontal axis represents the number of points and the vertical axis represents 100 query times. The number of points in each trajectory is about 5. We can see that the query time based on R \* Tree is much higher than the Bloom Filter, and it is more than 1000 times. For the R\*tree index, the query time complexity is O(log n), and the query mode of Bloom Filter is implemented by the hash function, and its time complexity is O(1), which also directly illustrates that the index based on Bloom Filter is negligible in query time. Similarly, there is a large disparity in the time consumed by the creation of these two types of indexes. The time consumed based on R\*Tree increases linearly with the increase in the amount of data. The time change for index construction based on the Bloom Filter is negligible, and the time consumption of the two is no longer on one level. As shown in Fig. 5(b), the horizontal axis represents the number of total points, and the vertical axis represents the cost of constructing the index. time.

In the case of the same trajectory, we tested the impact of different numbers of points included in each trajectory on index creation and query. The effect of tested as follows: The number of points included in each trajectory has basically no effect on the Bloom Filter. With the increase of the number of points in the track, the R\*tree index has an overall upward trend, as shown in Fig. 5(c), the horizontal axis represents the number of points in each track, and the vertical axis represents 1,000 query times. Fig. 5(d) shows the time consumed when creating the indexes, in which the two kinds of indexes have the same total number of track points, the number of track points in each track is different.



**Fig.5** Query time and construction time of Index of Trajectory based om Bloom Filter Index and R\*Tree

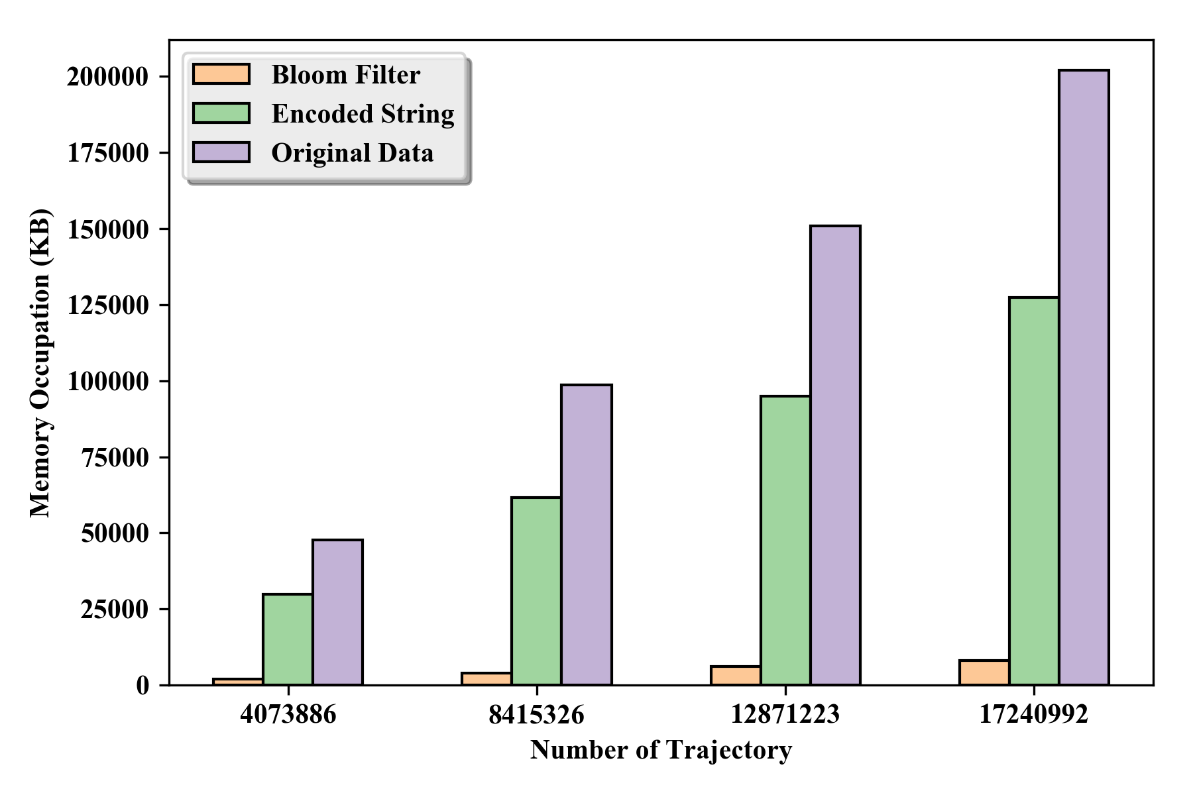
In order to further confirm the performance advantage of Bloom Filter for massive trajectories, the following tests were performed in this paper:

The experimental results show that the Bloom Filter has great advantages in memory consumption by using its bit array, such as Fig. 6 and Table 1, and Fig. 6 shows the memory occupation of the Bloom Filter, including the memory occupied by the encoded travel records, and the memory occupied by the original trajectory data. The transverse axis represents the number of points entered by each Bloom Filter, and the longitudinal axis represents the amount of memory occupied after processes, in KB. From the table and graph, we can conclude that compared with the original data, the memory consumption of the bit array in the Bloom Filter is much smaller than that of the original data. With the increase of the original data, the memory consumption of the bit array is linearly related to it, and the coding of the original data can also reduce the memory consumption.

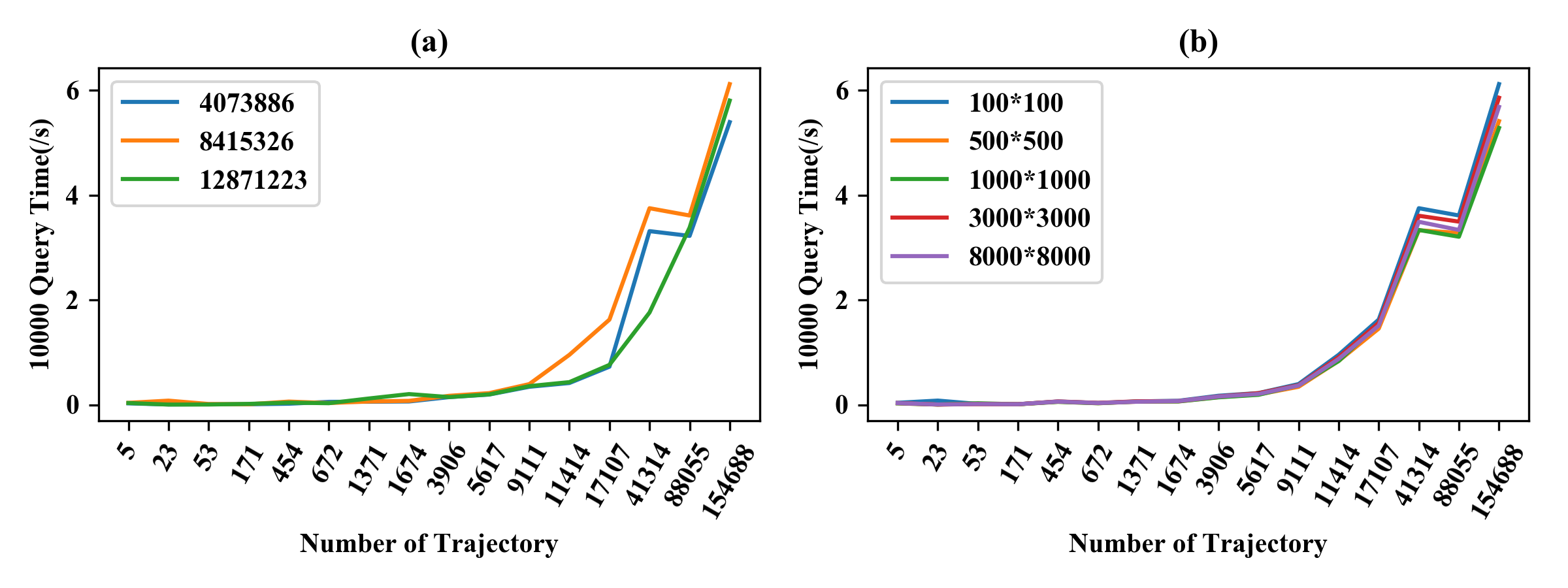
For the research of the query efficiency of this method, the experimental results are as follows: Fig. 7(a), the transverse axis represent the number of points contained in each retrieval travel record, the longitudinal axis represents the time consumed 10000 times of repeated query of the same travel record in second (S), the number of points in each travel record has little effect on the query time by the number of points in a single travel record. Moreover, the number of points in the different original trajectory set (4073886, 8415326, 12871223) has little effect on the retrieval time, which also satisfies the conclusion explained above, that is, the query time of Bloom Filter is determined by hash functions, and the time complexity is. In addition, the influence of grid size on retrieval time is also tested in this paper, such as Fig.7(b), transverse axis represents the number of query points each time, longitudinal axis represents the time consumed by 10000 query times in second, the five curves represent the time consumed by different grid sizes (100\*100, 500\*500,1000\*1 000,3000\*3000,8000\*8000, in meters) to query travel record. Through experiments, we can think that the grid size has little effect on the retrieval efficiency of travel records of different lengths. In addition, for Bloom Filter, there is a certain misjudgment rate for judging whether an element exists in the set. After many experiments, we find that the false positive probability caused by the index itself is always less than 0.000005, calculated by the formula (1).

**Table 1** Comparison of memory occupied by different point size

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Trajectory | Bloom Filter(KB) | Encoded String (KB) | Original Trajectory (KB ) |
| 4073886 | 1909 | 29830 | 47741 |
| 8415326 | 3944 | 61744 | 98617 |
| 12871223 | 6032 | 94975 | 150835 |
| 17240992 | 8080 | 127300 | 202043 |



**Fig. 6** Comparison of memory occupied by different trajectory data size



**Fig. 7** Comparison of the influence of querying different trajectory points on retrieval efficiency

# 5．Conclusion and discussion

This method verifies that the bloom Filter has a great advantage for the index of trajectory point sequence (travel record), both in time efficiency and memory occupancy. Moreover, the point sequence is regarded as the minimum retrieval granularity in this method, instead of indexing every single trajectory point, which further improves the efficiency of this method. In addition, the index using Bloom Filter has robust stability, and the query efficiency is hardly affected by the length of the input trajectory and the size of the grid. In the process of trajectory sequence query, the judgment of a certain travel record only needs to compare the hash results one by one, which saves lots of IO operations of the original data. In addition, an accurate trajectory query method based on Bloom Filter can effectively avoid a low recognition rate of traditional tree-oriented index structure and index overlapping.

Big data contains a great deal of information and is widely used. According to the different applications of big data, a suitable query method should be constructed. In the application of trajectory big data, this method provides a new model for the fast query of trajectory big data and also provides a new idea for the development of query models that should match the query results in different applications. Each travel record in trajectory data can be regarded as a whole in its time and space and semantics, so the storage mode of trajectory data can try to break through the traditional way of storing trajectory in series with ID of each point, which is not only convenient for the management of trajectory data but also saves a lot of unnecessary processing caused by index points, which also provides a new model for the management and the index construction of GIS data. For the track points contained in each travel record, different application selection methods are different. For those who come in contact with the 2019-nCoV patients, determine the precise time and place of their contact, and quickly generate the trajectory index based on the trajectory data recorded by devices, which can find out the contacted people in a short time.

At present, this work only distinguishes the travel records in the way of accurate matching and does not further verify the fuzzy query. The travel records are also simulated by segmented taxi trajectory, and there is no further processing of the merging, deletion, and simplification of the points in the travel records. The creation of the geographical grid also needs to be studied according to the temporal and spatial distribution of the points. For the Bloom Filter, the only disadvantage is that it has a certain misjudgment rate, which mean the Bloom Filter may misjudge elements that do not exist in the set as belonging to the trajectory set, but those elements that exist in the set must not be misjudged as not belonging to the set. Formula  represents the calculating method of the false positive rate,  denotes the number of input elements,  is the number of hash functions,  is the size of bits, and the false positive rate . But in the face of different regions and different size trajectory data, its influence needs further study. In this paper, we have to control it within acceptable limits, but for the different applications, it will be different, which needs to be further study.

At present, there are many shortcomings in the current work, and a lot of work is needed to improve the method. However, although this work is only the first step to verify the use of Bloom Filter to index travel records, it is also an important step. For later research, the mapping between the precise matching query and the fuzzy matching query is established to make it more extensive and practical. such as the fuzzy matching index of travel records, the fast query of suspicious trajectory in mass travel records, mobile trajectory behavior analysis, and so on. It provides an important theoretical basis and technical support.

# Acknowledgements

This work was supported by National Outstanding Youth Fund [41625004] and the National Science Fund of China [41971404]

# References:

1. Yu Q, Luo Y, Chen C, Zheng X (2019) Road congestion detection based on trajectory stay-place clustering. ISPRS Int J Geo-Information 8:. https://doi.org/10.3390/ijgi8060264

2. Chen Z, Shen HT, Zhou X, et al (2010) Searching Trajectories by Locations: An Efficiency Study. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, Proceeding. SIGMOD 2010

3. Jiang Y, Li X (2013) Travel time prediction based on historical trajectory data. Ann GIS 19:27–35. https://doi.org/10.1080/19475683.2012.758173

4. Wang Y, Zheng Y, Xue Y (2014) Travel time estimation of a path using sparse trajectories. Proc ACM SIGKDD Int Conf Knowl Discov Data Min 25–34. https://doi.org/10.1145/2623330.2623656

5. Chakka VP, Chakka VP, Everspaugh AC, et al (2003) Indexing Large Trajectory Data Sets With SETI. CIDR Conf Innov Data Syst Res. https://doi.org/10.1.1.12.182

6. Song Z, Roussopoulos N (2003) SEB-tree: An Approach to Index Continuously Moving Objects. In: Chen M-S, Chrysanthis PK, Sloman M, Zaslavsky A (eds) Mobile Data Management. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 340–344

7. Guttman A (1984) R-trees: A Dynamic Index Structure for Spatial Searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data. ACM, New York, NY, USA, pp 47–57

8. Beckmann N, Kriegel H-P, Schneider R, Seeger B (1990) The R\*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data. ACM, New York, NY, USA, pp 322–331

9. Leutenegger ST, Lopez MA, Edgington J (1997) STR: a simple and efficient algorithm for R-tree packing. In: Proceedings 13th International Conference on Data Engineering. pp 497–506

10. Greene D (1989) An implementation and performance analysis of spatial data access methods. In: [1989] Proceedings. Fifth International Conference on Data Engineering. pp 606–615

11. Frentzos E (2003) Indexing Objects Moving on Fixed Networks. In: Hadzilacos T, Manolopoulos Y, Roddick J, Theodoridis Y (eds) Advances in Spatial and Temporal Databases. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 289–305

12. Li G (2006) Indexing Moving Objects Trajectories on Fixed Networks. J Comput Res Dev 43:. https://doi.org/10.1360/crad20060509

13. Brunette W (2017) Extended abstract: Building mobile application frameworks for disconnected data management. In: MobiSys 2017 PhD Forum - Proceedings of the 2017 Workshop on MobiSys 2017 Ph.D. Forum, co-located with MobiSys 2017. pp 15–16

14. Su H, Zheng K, Zeng K, et al (2015) Making sense of trajectory data: A partition-and-summarization approach. Proc - Int Conf Data Eng 2015-May:963–974. https://doi.org/10.1109/ICDE.2015.7113348

15. Kong K, Tu L, Xu J, et al (2017) Trajectory query based on trajectory segments with activities. Proc 3rd ACM SIGSPATIAL Work Smart Cities Urban Anal UrbanGIS 2017 2017-January: https://doi.org/10.1145/3152178.3152180

16. Xiang L, Wang D, Gong J (2017) Organization and Efficient Range Query of Large Trajectory Data Based on Geohash. Geomatics Inf Sci Wuhan Univ

17. Zhang Z, Qi X, Wang Y, et al (2019) Distributed top-k similarity query on big trajectory streams. Front Comput Sci 13:647–664. https://doi.org/10.1007/s11704-018-7234-6

18. Morse MD, Patel JM (2007) An efficient and accurate method for evaluating time series similarity. Proc ACM SIGMOD Int Conf Manag Data 569–580. https://doi.org/10.1145/1247480.1247544

19. Wan Y, Zhou C, Pei T (2017) Semantic-geographic trajectory pattern mining based on a new similarity measurement. ISPRS Int J Geo-Information 6:. https://doi.org/10.3390/ijgi6070212

20. Li X, Han J, Kim S, Gonzalez H ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets. In: Proceedings of the 2007 SIAM International Conference on Data Mining. pp 273–284

21. Bloom BH (1970) Space/Time Trade-Offs in Hash Coding with Allowable Errors. Commun ACM 13:422–426. https://doi.org/10.1145/362686.362692

22. Bender MA, Farach-Colton M, Goswami M, et al (2018) Bloom filters, adaptivity, and the dictionary problem. In: Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS. pp 182–193

23. Yan J, Cho PL (2006) Enhancing Collaborative Spam Detection with Bloom Filters. In: 2006 22nd Annual Computer Security Applications Conference (ACSAC’06). pp 414–428

24. Zheng Y, Xie X, Ma W-Y (2009) Mining Interesting Locations and Travel Sequences From GPS Trajectories. In: Proceedings of International conference on World Wide Web 2009, Proceeding

25. Yuan J, Zheng Y, Zhang C, et al (2010) T-drive: Driving Directions Based on Taxi Trajectories. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, New York, NY, USA, pp 99–108

26. Partow A (2018) bloom. https://github.com/ArashPartow/bloom

27. boost::geometry::index::rtree. https://www.boost.org/doc/libs/1\_65\_1/libs/geometry/doc/html/geometry/reference/spatial\_indexes/boost\_\_geometry\_\_index\_\_rtree.html