XStar: a software system for handling taxi trajectory big data

Xiang Li¹, Joseph Mango^{1,2}, Jiajia Song¹ and Di Zhang^{1*}

Abstract

Advances in positioning and communicating technologies make it possible to collect large volumes of taxi trajectory data, quickly providing a complete picture of the ground traffic systems and thus being applied to different fields. However, there are still challenges for data users to handle such big data. In view of this, we have developed a software system named XStar to deal with trajectory big data. Its core is a scalable index and storage structure. Based on it, raw data can be saved in a more compact scheme and accessed more efficiently. A real taxi trajectory dataset is employed to demonstrate its performance. In general, XStar facilitates processing and analyzing trajectory data affordably and straightforwardly. Since its release in Jan. 2019, it has received downloads of over 4000 by May 2021. More analytical functions are being developed.

Keywords: XStar, Software, Trajectory, Taxi, Big data

1 Introduction

With the dramatic advances of positioning and communicating technologies and devices, more and more trajectory data from various moving objects can be collected, recorded and applied. Since both spatial and temporal information are inherent in trajectories, they can be called spatio-temporal big data. Among them, taxi trajectory data are the most applicable and accessible. Due to the highly intensive and nearly-full-time driving activities of taxis, the collected taxi trajectory data can quickly provide a complete picture of traffic dynamic on road networks in both space and time as long as enough taxis are involved. Furthermore, compared to other data sources, such as smart card data, mobile phone data or private car trajectory data, taxi trajectory data can protect the privacy of passengers or data providers to a great extent and meanwhile keep valuable information within data undamaged.

Raw taxi trajectory data are usually recorded discretely. In most cases, each record consists of taxi identification,

sampling time, instantaneous location and loading status (i.e. carrying passengers or not) and other information. By this means, it is possible to classify the driving motivations of taxis by discovering the instants when their loading statuses are switched between “loaded” and “unloaded” along the timeline. For example, if two successive records of one taxi have different loading statuses, it means the beginning or the ending of a passenger’s trip occurs between the two records. Eventually, it is possible to extract either complete trips of passengers with designated origins and destinations or cruising trajectories of unloaded taxis, and both of them can be employed as data sources to support traffic study or other objectives.

To date, many efforts have been devoted to explore the merit of taxi trajectory data and try to apply to various aspects, such as developing trajectory data mining methods (Dodge et al., 2009; Izakian et al., 2016; Liu & Karimi, 2006; Pfoser & Theodoridis, 2003; Zhou et al., 2017), inferring residents travel characteristics and patterns (Hu et al., 2014; Kang et al., 2015; Liu et al., 2012; Torrens et al., 2012), discovering spatio-temporal features of traffic flow (Ge et al., 2010; Liu & Ban, 2012; Wang et al., 2017; Wei et al., 2012; Zheng et al., 2011), and predicting travel time (Chen & Rakha, 2014; Jiang & Li, 2013; Xu et al., 2018b). Understandably, there is

* Correspondence: zhangdee@stu.ecnu.edu.cn

¹Key Laboratory of Geographic Information Science (Ministry of Education) and School of Geographic Sciences, East China Normal University, Shanghai, China

Full list of author information is available at the end of the article

always a tradeoff between data volume or coverage and the difficulty of data accessing. Limited data volumes refer to limited coverages, and they require less complicated techniques to process and access. However, from a practical perspective, the data volumes or coverages increase significantly, and in turn more efficient and complicated data processing and accessing methods are needed. For instance, the whole dataset employed in this paper includes more than 3 billion disordered records from about 13,000 taxis for thirty days covering a road network of 2500 km².

Before focusing on data analysis and application, it is a great challenge for data users to process large volumes of raw data. It is not straightforward to retrieve needed information from such a large number of records with respect to specific spatial and temporal query conditions. Most data users resort to commercial database management systems (DBMS) such as Oracle, MySQL or SQL Server because they usually have enough capacity to deal with large volumes of data and do not require programming skills of users. However, these DBMSs are inadequate to index big spatio-temporal data and handle queries with combined spatial and temporal conditions, and it is often time-consuming to complete data insertion (including building index structures) and evaluate queries. Other data users employ distributed or parallel big-data computational frameworks, such as Hadoop (Dittrich et al. 2012), Spark (Zahara et al. 2016) or Storm (Toshniwal et al. 2014), to process data. These frameworks usually require high-level programming skills and costly computational resources, which are not equipped or afforded by most data users. Moreover, because they are initially designed for big data in general, it is hard to tailor them for specific applications based on taxi trajectory data.

Behind the DBMSs mentioned above and big-data computational frameworks, data index structures are key to the acceleration of accessing large volumes of data. Since taxi trajectory data are also a kind of spatio-temporal data, we have examined spatio-temporal data index structures in the literature. A detailed review is given in the third section. Based on them, we design a scalable index structure for taxi trajectory data and develop a software system named XStar, to implement the proposed index structure and a set of functions to support its applications. XStar, as a standalone system, can read raw data, process them, conduct various analysis, visualize and export results in different ways. More important, all functions are developed from the bottom up. There is no need for users to install other commercial GIS software. Our motivation is to facilitate manipulating taxi trajectory data for general users with no requirements of either programming skills or powerful computational devices. The target platform of running

the software is a single personal computer, such as a laptop with memory as less as 8GB.

The rest of this paper is organized as follows. In the next section, we introduce the characteristics and applications of taxi trajectory data. A literature review on existing indexing structures for spatio-temporal data is given and their applicability for taxi trajectory data is discussed in the third section. The software is introduced in the fourth section, which includes methodology, instructions for downloading and installing it, functions, and applications based on a real taxi trajectory dataset. The last section concludes the paper.

2 Taxi trajectory data

2.1 Data features

Raw taxi trajectory data physically consist of sampled positioning records of taxis. Records are generated, transferred, and saved at a fixed or variable sampling frequency. Each record usually includes taxi identifier, sampling time, location and loading status. Table 1 gives part of sample records, where location is expressed as a pair of coordinates, and loading status can be one of two values: *loaded* and *unloaded*. Records in this table are not sorted by either taxi identifier or sampling time.

If sorting records by taxi identifiers and sampling time as the primary key and the secondary key, respectively, and plotting their locations on road networks, we may visualize the trajectories of taxis. For example, positioning records of one taxi from sampling time t_1 to t_{22} are illustrated in Fig. 1, where successive records are connected with dashed lines. Different symbols are used to reflect loading statuses of the taxi.

A trajectory of one taxi consists of *trips* of taxi passengers and *cruises* for searching passengers alternately. As shown in Fig. 1, From t_1 to t_6 , the taxi cruises and then carries passengers to a specified destination on t_7 . The passengers get off on t_{15} , and other passengers of this taxi start their trip on t_{18} .

According to the above examples, related variables are defined in the following way. Let $D_k = \{[t_i, l_i, s_i] \mid 1 \leq i \leq n\}$ denote a set of n positioning records of taxi k , where t_i , l_i and s_i are the i th sampling time, location and loading status, respectively. s_i equals *loaded* or *unloaded*. D_k is ordered by time, i.e., $t_i < t_{i+1}$. We define a trip or a cruise as a set of $\{[t_i, l_i] \mid p \leq i \leq q\}$, where $p, q \in [1, n]$. The set is a trip, if $s_{p-1} = \text{unloaded}$, $s_j \mid p \leq j \leq q-1 = \text{loaded}$, and $s_q = \text{unloaded}$, and otherwise, the set is a cruise. For example, in Fig. 1, the set of records from t_7 to t_{15} is a complete trip, and it is a complete cruise from t_{15} to t_{18} . It is noted that the last record in a trip or a cruise is always the first record belonging to a cruise or a trip. That means the trajectory of a taxi could also be a list of $\{Trip, Cruise, Trip, Cruise, \dots\}$ or $\{Cruise, Trip, Cruise, Trip, \dots\}$.

Table 1 Example records in raw taxi trajectory data

Taxi identifier	Sampling time	Location	Loading status
...	...	...	...
23,807	May 1, 2019 19:38:18	(478,988, 341,545)	Unloaded
23,390	May 1, 2019 02:55:08	(388,335, 26,389)	Unloaded
10,827	May 1, 2019 23:23:38	(333,718, 173,365)	Unloaded
20,043	May 1, 2019 04:49:49	(357,767, 38,562)	Unloaded
49,023	May 1, 2019 01:12:08	(303,755, 247,965)	Unloaded
21,941	May 1, 2019 17:37:56	(720,453, 179,758)	Loaded
15,042	May 1, 2019 02:28:59	(458,723, 272,005)	Unloaded
27,615	May 1, 2019 00:24:06	(389,823, 098167)	Unloaded
14,891	May 1, 2019 12:05:59	(406,283, 304,457)	Unloaded
14,230	May 1, 2019 03:46:56	(518,303, 23,779)	Loaded
15,990	May 1, 2019 19:51:53	(369,808, 258,588)	Unloaded
17,564	May 1, 2019 03:41:43	(558,867, 210,453)	Unloaded
...	...	...	...

2.2 Data applications

2.2.1 Overview

The taxi trajectory data discussed in this paper are records of taxi movement occurring during a historical period, such as yesterday, last week, etc. Accordingly, applications to them focus on queries on the past instead of the present (e.g., real-time traffic situation) or future (e.g., travel time estimation). Queries on the past target taxis with specific identifiers or

aggregate measures meeting conditions without taxi identifiers. As aforementioned, we employ taxi trajectory data to discover underlying traffic dynamics or patterns rather than keep track of individual taxis. Furthermore, queries on individual taxis may violate the privacy of taxi drivers, and thus they should not be conducted unless there are specific application purposes. Therefore, aggregate queries are mostly adopted in practical applications.

Each aggregate query has explicit spatial and temporal query conditions. For example, in the query “How many trips have happened from district *A* to district *B* during rush hours yesterday?”, “from district *A* to district *B*” and “rush hours yesterday” are spatial and temporal conditions, respectively. A temporal condition could be either an instant (e.g. 10 am) or an interval (e.g. 10 am–1 pm) according to the temporal length it refers to, while a spatial condition maybe the whole study area, one or a group of spatial analysis units, paths constrained by underlying road networks, points on road segments, etc. The combination of temporal and spatial conditions may be used to suit various query objectives. Some of them are introduced in the following parts, and typical query examples are given in Table 2. A combination of these queries may satisfy most application purposes. Most of these queries can be evaluated with XStar.

2.2.2 Origin-destination analysis

Origin-destination analysis targets trips of taxi passengers. Each trip has an origin and a destination. The origin-destination analysis is used to reveal the intensity of connectivity between locations. The result of an origin-destination analysis is often a matrix, in which origins and destinations are enumerated as labels of rows and columns, respectively. Each element of the matrix is the number of trips between a specific pair of origin and destination.

In space, origins or destinations are usually represented as small polygons, such as traffic analysis zones, etc. Spatial query condition of an origin-destination analysis may include one/many origins and one/many destinations. Especially, a one-many origin-destination analysis (i.e., one origin and many destinations) equals a destination analysis discovering the distribution of destinations of trips from the same origin. Similarly, a many-one origin-destination analysis indicates the spatial sources of travelers targeting the same destination. Temporal query conditions could be either instants or intervals. As

shown in Table 2, an instant query is usually applied to ongoing trips, while an interval query is to completed trips.

2.2.3 Distribution analysis

Distribution analysis is used to present spatial distributions of locations of taxis based on their loading statuses. The whole study area is split into several spatial analysis units. With distribution analysis, the number of taxis meeting conditions in each unit is calculated. The spatial query condition of a distribution analysis may be all or some analysis units. If the objective of a distribution analysis is to demonstrate the number of loaded or unloaded taxis in each unit, the temporal condition is usually an instant, while if its objective is to summarize the number of trips starting or ending in each unit, then an interval query condition is adopted.

Based on the distribution analysis results, various density or clustering analysis can be conducted, for example, finding hot spots of passengers getting on taxis during specific periods.

2.2.4 Network-constrained analysis

The movement of taxis are mostly constrained by road networks. It is possible to align taxi trajectories with road segments through map-matching algorithms. Therefore, network-constrained analysis can be conducted. Spatial query conditions in network-constrained analysis are usually composed of network locations, network components or network paths. Accordingly, we can classify network-constrained analysis into three categories: location-based, component-based, and path-based analysis. Network locations refer to locations associated with road networks, such as street addresses. The location-based network-constrained analysis is often used to analyze profiles of traffic flows passing through one location toward different directions. A network component could be a road segment or an intersection, while a network path consists of a series of network components. Component-based or path-based network-

Table 2 Typical queries on taxi trajectory data

Type	Example
Instant origin-destination analysis	Calculate the number of ongoing trips at 10 am yesterday for each pair of origin and destination
Interval origin-destination analysis	Calculate the number of trips starting and/or ending between 10 am and 2 pm yesterday for each pair of origin and destination
Instant distribution analysis	Calculate the number of loaded/unloaded taxis for each one of given spatial analysis units at 10 am yesterday
Interval distribution analysis	Calculate the number of locations where passengers getting on/off taxis in each one of given spatial analysis units between 10 am and 2 pm yesterday
Location-based network-constrained analysis	Calculate the volume/average speed of taxi traffic flow passing through the 2nd checkpoint of Road <i>A</i> between 10 am and 11 am yesterday.
Component-based network-constrained analysis	Calculate the volume/average speed of taxi traffic flow traveling on Road Segment <i>B</i> at 10 am or between 10 am and 11 am yesterday.
Path-based network-constrained analysis	Calculate the volume/average speed of taxi traffic flow traveling from Points <i>C</i> to <i>D</i> along Road Segments <i>E</i> , <i>F</i> , and <i>G</i> at 10 am or between 10 am and 11 am yesterday.

constrained analysis focuses on summarized or average measures of traffic flow related to one component or one path. In applications, the location-based network-constrained analysis is usually accompanied by an interval query condition, while other two types of network-constrained analysis could be applied with temporal conditions, either instants or intervals.

3 Related work

The core of XStar is a data indexing approach. In this section, we review existing works on spatio-temporal data indexing. Most of them are derived from R-tree (Guttman, 1984). Initially, R-tree is only used to index two-dimensional spatial data. Many variants of the R-tree have been proposed to deal with temporal information. Some methods try to handle spatial domain as a primary key while taking temporal domain as a secondary issue. For example, Xu et al. (1990) developed RT-tree to index spatio-temporal data by combining an R-tree and a TSB-tree (Lomet, 1989) for spatial and temporal, respectively. Since the two trees are separated, it is hard to search the spatial and temporal information simultaneously. Other structures merge spatial and temporal domains into a single high-dimensional R-tree, such as 3D R-tree (Theodoridis et al., 1996) and Trails-tree (Mahmood et al., 2018). These approaches support combined spatial and temporal query conditions, but evaluating time slice queries may have to search all tree entries and hence time-consuming.

Some other R-tree-based methods adopt overlapping and multi-version structures. They build separate R-trees for different time instances to keep all spatial data at each instance. For example, MR-tree (Yang et al., 2009) performs perfectly for time slice queries. However, it is inadequate for time window queries because of duplicated tree entries. HR+ -tree (Tao & Papadias, 2001). On the other hand, where parent nodes have only entries to those that belong to the parent's timestamp, but a node may have multiple parents. Since 3D R-tree performs well for queries with long time interval and is inferior in time slice queries while overlapping B-trees (Burton et al., 1990) do well in time slice queries, MV3R-tree (Tao & Papadias, 2000) uses a 3D R-tree to process time window queries and an MVR-tree to process time slice queries. And SMO-index (Romero et al., 2012) uses a sequence of snapshots and movement logs to support both timestamp and time interval queries.

Instead of splitting time domain into instances, other methods partition spatial domain into small zones which are indexed with a B-tree and records in each zone are indexed with an R-tree, such as SEIT (Chakka et al., 2003), SEB-tree (Song & Roussopoulos, 2003), MTSB-tree (Zhou et al., 2005), et al. Most of them are used to preserve the trajectories of moving objects.

To reduce the cost of updating data, time parameterized R-tree (TPR-tree, in short) (Saltenis et al., 2000) and its descendants, such as Bottom-up Updates (Lee et al., 2003), TPR*-tree (Tao et al., 2003), HTPR*-tree (Fang et al., 2011) and Lazy Update R-tree (Kwon et al., 2002), et al., have been proposed.

Those R-tree based index structures depend on Minimum Boundary Rectangle (MBR), which may cause a lot of dead space and have heavy workloads of updates. Due to the high frequency for the requirements of updates, some index structures transform spatio-temporal data into higher-dimensional space, such as BDual tree (Yiu et al., 2008), STRIPES (Patel & Chakka, 2004) and MB-index (Elbassioni et al., 2003), or lower-dimensional space, such as B^x-tree (Jensen et al., 2004), B^y-tree (Chen et al., 2008a) and ST²B-tree (Chen et al., 2008b). For example, in B^x-tree, the position of one spatio-temporal record in the structure is treated as a point along a space-filling curve with equal time intervals. The main drawback is that rectangular range queries in the primal space are always transformed into polygonal range queries in transformed space, which requires much more complex algorithms (Mahmood et al., 2019).

When applied to taxi trajectory big data distributed within a fixed spatial and temporal range, the above tree-based index structures become inefficient. First, it is expensive to maintain index structure, and second, the overlapping area among entries increases quickly. Thus, by evaluating all these challenges in the XStar, we design a new index structure that partly was derived from our two previous works. The first one is a mixed spatio-temporal index structure (Li & Lin, 2006), and the second one is a cube-based high-dimensional index structure (Xu et al., 2018a). Please refer to the next section for more details.

4 Software

4.1 Methodology

Based on existing works, from a practical perspective, we develop the software called XStar. Its storage and indexing structures are demonstrated in Fig. 2. Data are organized into four levels. A list of taxis' identifiers is located at the top level. Each taxi identified with a unique number (i.e. *ID*) may have several trajectories, and each trajectory consists of trips and cruises, alternatively. Each trajectory, trip, or cruise has at least three properties: start time, duration, and a pointer to data at the next level. Locational records (i.e. *Location_x*) of each trip or cruise are sited at the bottom level. The hierarchical structure can help organize and locate locational records efficiently.

Locational records at level 4 are not duplicate of raw trajectory data, as shown in Table 1. Instead, only locations (i.e. coordinates) are recorded while taxi identifiers

and sampling time are discarded, and thus these records are resampled or interpolated results from raw data based on a fixed temporal granularity given by users. Since locational records of each trip or cruise are physically saved in an ordered list sequentially, it is possible to quickly calculate the position of a locational record in the list at any temporal instant without fetching each record from the first one. Besides the basic structure, a B-Tree is created to index identifiers of taxis. An R-Tree is used to index 2-tuple temporal properties based on our previous work (Li & Lin, 2006) to facilitate evaluating queries with identifiers or temporal conditions.

The number of locational records in the software might be smaller than the number of raw sampling points if the temporal granularity is larger than the original sampling interval and vice versa. Even when the amount is huge, with the above structure, it is still very fast to locate a taxi at any time. Furthermore, since the properties of taxi identifiers and sampling time are not physically saved in locational records, we employ other measures to reduce the length of bytes for saving coordinates in the software (e.g. replacing 64-bit double numbers with 32-bit decimal numbers), and thus, storage space is saved to a great extent. It is crucial for most computers equipped only limited memory and hard disk capacity.

It is noted that there is indeed a lack of spatial indexing approaches in the above structure. Each locational record can be accessed only through the route from taxi to its trajectory and trip or cruise. The reason is given below. Because each query on taxi trajectory data must be accompanied with temporal conditions, we may reduce searching space quickly along temporal dimension, and then, with the proposed method of retrieving locational records, it is still very quick to enumerate each taxi in order to meet spatial query conditions.

we adopt the following steps to implement the above structure. First, figure out identifiers of taxis from raw data. Then, sort all raw records by taxi identifier (primary) and sampling time (secondary). Third, split records of each taxi into one or more trajectories according to the difference of sampling time between continuous records. If the difference is larger than a given threshold, then the two records belong to two trajectories, and vice versa. Fourth, regroup records in each trajectory according to their loading statuses to figure out each record group's type (i.e. trip or cruise). Fifth, for records in each trip or cruise, generate resampled locational records with identical interval according to a given temporal granularity. Sixth, build auxiliary B-Tree and R-Tree index structures. Since the input might be very huge during the implementation, we pay attention to every detail to reduce computational time and storage

costs. For example, it is usually impossible to sort more than 100 million records as a whole with a personal computer because either memory might be overflowed or the operation is extremely time-consuming. In XStar, we develop a split-and-conquer approach to the above problems. This approach makes it possible to deal with a data file as large as 80GB with a computer equipped with 8GB main memory and 200GB free storage space in hard disks. To further accelerate data processing, XStar supports parallel computations as long as the user's computers are equipped with multiple-core processors.

4.2 Download and install

The software is a Microsoft Windows application. Users can scan the Quick Response (in short, QR) Code in Fig. 3. The code leads to a WeChat account, "Big Data Lion" which is owned and maintained by the first author of this paper. Follow it and search articles on "XStar". Then, there is a detailed instruction on how to download, install, and use the software. Personal users can freely use XStar for non-commercial objectives. Commercial users may have to contact the first author for permissions and licenses.

After downloading the software package, run "xstar.exe" to start it. Basically, XStar can be executed on any computer with Microsoft Windows operation system. There are no requirements for CPU performance, memory size, or hard disk space. In some cases, the program may automatically prompt users to download and install ".NET Framework", which, released by Microsoft, is a runtime to support the running of XStar.

Fig. 3 QR code of WeChat account "Big Data Lion"

4.3 Functions and applications

4.3.1 Main portal

A 10GB taxi trajectory data file consists of more than 100 million locational records were collected from more than 13 thousand taxis running 24 h and used in this section to demonstrate the main functions and applications of XStar. The original sampling frequency is about 10 s per point. The computer used to run the XStar software is a laptop equipped with an i7 processor, 8 GB RAM, and 100 GB free hard disk storage space.

Figure 4 is the XStar's main portal to other functions classified into eight groups. The left four steps are used to process raw taxi trajectory data. The three modules, namely A, B, and C, provide various data analysis functions. The "Tool box" includes a set of tools to support other modules. In the rest of this section, we briefly introduce the four data processing steps and the three data analysis modules.

4.3.2 Data processing

Instead of a single operation, XStar splits raw data processing into four steps. Users are required to follow the sequence to finish processing data. Some steps are time-consuming operations, and thus saving intermediate results between steps may help users experiment with different parameters with no need to start over from the beginning.

Raw taxi trajectory data are usually saved in a text file, as shown in Fig. 5a, b, and c. Step 1 (Fig. 5a) is used to define or decode the raw data file structure to figure out the required fields, such as taxi identifier, time, location, loading status, etc. It is not a time-consuming operation. Users are required to select designated fields, and the software saves all settings in a raw-data-structure (in short, RDS) file for future steps. According to an RDS file created in the last step, step 2 (Fig. 5b) extracts taxi identifiers from the raw data file. The function explores all locational records to figure out the number of taxis and identify them in a list, saved in an object-identifier (in short, OID) file. Step 3 (Fig. 5c) sorts locational records by taxi identifiers as the primary key and sampling time as the secondary key. Sorted records are saved in a sorted-data (in short, SDT) file. The file includes all required information from the raw data file in a more compact format, and thus its size is usually much smaller than the raw one. Users can sort all locational records or part of them in one raw data file by defining a period of interests. It is especially valuable for large raw data files. Users can split the large one into several small SDT files with different periods and conduct analysis based on them respectively if their computers' capacities are too limited to deal with the raw data as a whole. Step 4 (Fig. 5d) builds index structures based on an SDT file. A temporal granularity must be given to

Fig. 4 The main interface of XStar

determine the interval between interpolated locations in the index structure. This means, for the same SDT file, users can create different index structures with different temporal granularities. The index structure is saved in an index-data (in short, IDT) file.

Along with the above steps, users can define other parameters, such as the number of parallel threads, thresholds of recognizing and removing wrong locational records, type of coordinate conversion, etc.

A summary of the data processing time and the resultant file size for the employed dataset is given in Table 3. It is noted that the entire processing time is less than 20 min, the SDT file size is only one-tenth of the raw data file size, and the IDT file size varies with the given temporal granularity denoted with tg . It takes users comparatively less time to create different IDT files with different temporal granularities based on the same SDT with respect to different application purposes. SDT or IDT file can be disseminated and applied in the following analytical modules independently. By this means, XStar facilitates sharing data between users and makes it possible to analyze a large data file with affordable personal computers.

4.3.3 Data analysis

Module A is a group of data analysis methods for visualizing trajectories. As shown in Fig. 6, after reading and opening an SDF file, the module can give a list of taxis' identifiers. Users can select one or several taxis by their identifiers, customize the time interval of interests, and then the trajectories meeting the above conditions will

be displayed in the map panel immediately. With different taxis, their trajectories are displayed with different colors too, and the legend is located in the top-left corner. Sampling points and trajectories are labelled, and the labels can be customized. Results can be saved as ESRI Shapefiles or images.

Module B and Module C focus on instantaneous analysis and interval analysis of trajectories, i.e., temporal condition in data queries is an instant or an interval, respectively. The input of both functions is an IDT file. Analytical results can be exported as ESRI Shapefiles or images.

In Module B, after opening an IDT file, the location of each taxi at the currently selected time instant is displayed as a symbol of car in the map panel as shown in Fig. 7. The symbol is labelled with the taxi's identifier. Different loading statuses correspond to different symbols. When users change the value of the current instant, the taxis' locations are updated immediately. In animation mode, the current instant can change automatically, and the map panel can present the movement of taxis in a movie fashion.

Instead of instantaneous locations, Module C usually targets trips or cruises happening within a given interval. It provides much more data analysis functions than Module B. Figure 8 illustrates the origin distributions of all trips meeting temporal conditions. Each red dot represents an origin of a trip. Similar analysis can be conducted to figure out the distribution of trips' or cruises' destinations. By sliding the two buttons in the top panel, users can easily define the time interval range.

(a) Step 1: Define file structure

(b) Step 2: Extract identifiers of taxis

(c) Step 3: Sort locational records

(d) Step 4: Build index structure

Fig. 5 Four steps for processing raw taxi trajectory data**Table 3** Computational time and file size for processing raw data

	Output	Computational time	File size
Step 1: define file structure	RDS file	Up to users	< 1 KB
Step 2: extract identifiers	OID file	312 s	121 KB
Step 3: sort records	SDT file	507 s	1034 MB
Step 4: build index structure	IDT file ($t_g = 10$ s)	253 s	1245 MB
	IDT file ($t_g = 20$ s)	122 s	640 MB
	IDT file ($t_g = 30$ s)	98 s	432 MB

Fig. 6 Visualizing selected trajectories

Fig. 7 Distribution of taxis at a given time instant

Fig. 8 Origin distribution of trips within a given time interval

Besides temporal conditions, Module B and Module C support queries based on the area of interests (in short, AOI). For instance, users can create an origin AOI, and then all trips or cruises departing from this AOI can be selected, as shown in Fig. 9a. If users change it to a destination AOI, then the distribution of the origins of all trips or cruises targeting this AOI is illustrated in Fig. 9b. According to these figures, it is possible to discover the pattern of traffics associated with this AOI. Users can also define multiple origin or destination AOIs. By this means, trips or cruises falling into these AOIs can be selected. For example, in Fig. 9c, there are one origin AOI and one destination AOI, and all trips, whose origins and destinations falling into the two AOI, respectively, are visualized. It is noted that they adopt different network paths.

The XStar can also generate aggregated results based on analysis zones in space. Any polygon GIS layer (e.g. Shapefile) can be loaded as analysis zones, or otherwise, the software can generate regular analysis zones. For example, in Fig. 9d, each analysis zone is a hexagon. The number on each hexagon is an aggregated analytical result, such as the sum of origins falling into the polygon. Analysis zones are also needed in OD analysis. As shown in Fig. 9e, there is a line between any two analysis zones. Label on the line indicates the number of trips between these two zones. In profile analysis, lines intersecting with road segments are regarded as analysis zones. Trips or cruises passing through these lines are recorded, and

their total number and average speed are displayed on these lines, as shown in Fig. 9f.

In XStar, it is convenient to combine temporal conditions and spatial conditions for retrieving needed information and conducting more complicated analysis, such as meeting analysis (i.e., to find overlapping trajectories in space and time), stay point analysis (i.e., to extract locations where taxis keep static for a while), etc. Except for taxis, trajectories from other moving objects (e.g., human beings, private vehicles, ships, airplanes, etc.) may also be processed and analyzed with XStar. More functions are being developed.

5 Conclusion

This paper presents a software system named XStar developed to deal with large volumes of taxi trajectory data. To the best of our knowledge, it is currently the only software targeting taxi trajectory data processing and analysis. Its core is a scalable index and storage structure. Based on it, raw data can be saved in a more compact scheme and be accessed more efficiently. It neither requires professional programming skills nor expensive computational devices to use. Further, it facilitates users processing and analyzing trajectory data in affordable and straightforward manners. Its design consists of four data processing steps and three data analysis modules. We have introduced its main features and examined its performance based on a real taxi trajectory dataset. Generally, the XStar fulfills most types of analysis on trajectory data and can export its results into

Fig. 9 Some examples of data analysis with XStar

other systems. The software was released in Jan. 2019, and by May 2021 has received more than 4000 downloads. Future research is focused on developing more functions to improve the current version.

Funding

This work is partially supported by the projects funded by the National Natural Science Foundation of China (Grant Number: 41771410) and the Ministry of Education of China (Grant Number: 19JZD023).

Availability of data and materials

Not applicable.

Declarations

Competing interests

No conflict of interest exists in the submission of this manuscript.

Author details

¹Key Laboratory of Geographic Information Science (Ministry of Education) and School of Geographic Sciences, East China Normal University, Shanghai, China. ²Department of Transportation and Geotechnical Engineering, University of Dar es Salaam, Dar es salaam, Tanzania.

Received: 19 May 2021 Accepted: 2 July 2021

Published online: 28 July 2021

References

- Burton, F. W., Kollias, J. G., Matsakis, D. G., & Kollias, V. G. (1990). Short note: Implementation of overlapping b-trees for time and space efficient representation of collections of similar files. *The Computer Journal*, 33(3), 279–280. <https://doi.org/10.1093/comjnl/33.3.279>.
- Chakka, V. P., Everspaugh, A., & Patel, J. M. (2003). Indexing large trajectory data sets with SETI. *CIDR*, 75, 76.
- Chen, H., & Rakha, H. A. (2014). Real-time travel time prediction using particle filtering with a non-explicit state-transition model. *Transportation Research Part C: Emerging Technologies*, 43, 112–126. <https://doi.org/10.1016/j.trc.2014.02.008>.
- Chen, N., Shou, L. D., Chen, G., & Dong, J. X. (2008a). Adaptive indexing of moving objects with highly variable update frequencies. *Journal of Computer Science and Technology*, 23(6), 998–1014. <https://doi.org/10.1007/s11390-008-9185-0>.
- Chen, S., Ooi, B. C., Tan, K. L., & Nascimento, M. A. (2008b). ST2B-tree: A self-tunable spatio-temporal B+-tree index for moving objects. In *Proceedings of the 2008 ACM SIGMOD international conference on management of data* (pp. 29–42).
- Dittrich, J., & Quiané-Ruiz, J. A. (2012). Efficient big data processing in Hadoop MapReduce. *Proceedings of the VLDB Endowment*, 5(12), 2014–2015. <https://doi.org/10.14778/2367502.2367562>.
- Dodge, S., Weibel, R., & Forootan, E. (2009). Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of

- moving objects. *Computers, Environment and Urban Systems*, 33(6), 419–434. <https://doi.org/10.1016/j.compenvurbsys.2009.07.008>.
- Elbassioni, K., Elmasry, A., & Kamel, I. (2003). An efficient indexing scheme for multi-dimensional moving objects. In *International conference on database theory* (pp. 425–439). Berlin, Heidelberg: Springer.
- Fang, Y., Cao, J., Wang, J., Peng, Y., & Song, W. (2011). HTPR*-tree: An efficient index for moving objects to support predictive query and partial history query. In *International conference on web-age information management* (pp. 26–39). Berlin, Heidelberg: Springer.
- Ge, Y., Xiong, H., Tuzhilin, A., Xiao, K., Gruteser, M., & Pazzani, M. (2010, July). An energy-efficient mobile recommender system. In proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 899–908).
- 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 (pp. 47–57).
- Hu, Y., Miller, H. J., & Li, X. (2014). Detecting and analyzing mobility hotspots using surface networks. *Transactions in GIS*, 18(6), 911–935. <https://doi.org/10.1111/tgis.12076>.
- Izakian, Z., Mesgari, M. S., & Abraham, A. (2016). Automated clustering of trajectory data using a particle swarm optimization. *Computers, Environment and Urban Systems*, 55, 55–65. <https://doi.org/10.1016/j.compenvurbsys.2015.10.009>.
- Jensen, C. S., Lin, D., & Ooi, B. C. (2004). Query and update efficient B+-tree based indexing of moving objects. In proceedings of the thirtieth international conference on very large data bases-volume 30 (pp. 768–779).
- Jiang, Y., & Li, X. (2013). Travel time prediction based on historical trajectory data. *Annals of GIS*, 19(1), 27–35. <https://doi.org/10.1080/19475683.2012.758173>.
- Kang, C., Liu, Y., & Wu, L. (2015, June). Delineating intra-urban spatial connectivity patterns by travel-activities: A case study of Beijing, China. In 2015 23rd international conference on Geoinformatics (pp. 1–7). IEEE.
- Kwon, D., Lee, S., & Lee, S. (2002). Indexing the current positions of moving objects using the lazy update R-tree. In proceedings third international conference on Mobile data management MDM 2002 (pp. 113–120). IEEE.
- Lee, M. L., Hsu, W., Jensen, C. S., Cui, B., & Teo, K. L. (2003, January). Supporting frequent updates in r-trees: A bottom-up approach. In proceedings 2003 VLDB conference (pp. 608–619). Morgan Kaufmann.
- Li, X., & Lin, H. (2006). Indexing network-constrained trajectories for connectivity-based queries. *International Journal of Geographical Information Science*, 20(3), 303–328. <https://doi.org/10.1080/13658810500432570>.
- Liu, X., & Karimi, H. A. (2006). Location awareness through trajectory prediction. *Computers, Environment and Urban Systems*, 30(6), 741–756. <https://doi.org/10.1016/j.compenvurbsys.2006.02.007>.
- Liu, Y., Wang, F., Xiao, Y., & Gao, S. (2012). Urban land uses and traffic ‘source-sink areas’: Evidence from GPS-enabled taxi data in Shanghai. *Landscape and Urban Planning*, 106(1), 73–87. <https://doi.org/10.1016/j.landurbplan.2012.02.012>.
- Lomet, D., & Salzberg, B. (1989). Access methods for multiversion data. *ACM SIGMOD Record*, 18(2), 315–324. <https://doi.org/10.1145/66926.66956>.
- Mahmood, A. R., Aly, A. M., Kuznetsova, T., Basalamah, S., & Aref, W. G. (2018). Disk-based indexing of recent trajectories. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 4(3), 1–27. <https://doi.org/10.1145/3234941>.
- Mahmood, A. R., Punni, S., & Aref, W. G. (2019). Spatio-temporal access methods: A survey (2010–2017). *Geoinformatica*, 23(1), 1–36. <https://doi.org/10.1007/s10707-018-0329-2>.
- Patel, J. M., Chen, Y., & Chakka, V. P. (2004, June). STRIPES: An efficient index for predicted trajectories. In proceedings of the 2004 ACM SIGMOD international conference on management of data (pp. 635–646).
- Pfoser, D., & Theodoridis, Y. (2003). Generating semantics-based trajectories of moving objects. *Computers, Environment and Urban Systems*, 27(3), 243–263. [https://doi.org/10.1016/S0198-9715\(02\)00023-6](https://doi.org/10.1016/S0198-9715(02)00023-6).
- Romero, M., Brisaboa, N., & Rodríguez, M. A. (2012). The smo-index: a succinct moving object structure for timestamp and interval queries. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems* (pp. 498–501).
- Šaltenis, S., Jensen, C. S., Leutenegger, S. T., & Lopez, M. A. (2000). Indexing the positions of continuously moving objects. In *Proceedings of the 2000 ACM SIGMOD international conference on management of data* (pp. 331–342).
- Song, Z., & Roussopoulos, N. (2003). SEB-tree: An approach to index continuously moving objects. In *International conference on Mobile data management* (pp. 340–344). Berlin, Heidelberg: Springer.
- Tao, Y., & Papadias, D. (2000). *MV3R-tree: A spatio-temporal access method for timestamp and interval queries* (Vol. 6). Technical Report HKUST-CS00.
- Tao, Y., & Papadias, D. (2001, July). Efficient historical R-trees. In proceedings thirteenth international conference on scientific and statistical database management. SSDBM 2001 (pp. 223–232). IEEE.
- Tao, Y., Papadias, D., & Sun, J. (2003). The TPR*-tree: An optimized spatio-temporal access method for predictive queries. In proceedings 2003 VLDB conference (pp. 790–801). Morgan Kaufmann.
- Theodoridis, Y., Vazirgiannis, M., & Sellis, T. (1996, June). Spatio-temporal indexing for large multimedia applications. In proceedings of the third IEEE international conference on multimedia computing and systems (pp. 441–448). IEEE.
- Torrens, P. M., Nara, A., Li, X., Zhu, H., Griffin, W. A., & Brown, S. B. (2012). An extensible simulation environment and movement metrics for testing walking behavior in agent-based models. *Computers, Environment and Urban Systems*, 36(1), 1–17. <https://doi.org/10.1016/j.compenvurbsys.2011.07.005>.
- Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., ... & Ryaboy, D. (2014, June). Storm@ twitter. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data (pp. 147–156).
- Wang, J., Wang, C., Song, X., & Raghavan, V. (2017). Automatic intersection and traffic rule detection by mining motor-vehicle GPS trajectories. *Computers, Environment and Urban Systems*, 64, 19–29. <https://doi.org/10.1016/j.compenvurbsys.2016.12.006>.
- Wei, L. Y., Zheng, Y., & Peng, W. C. (2012). Constructing popular routes from uncertain trajectories. In proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 195–203).
- Xu, X., Han, J., & Lu, W. (1990). RT-tree: An improved R-tree indexing structure for temporal spatial databases [C]. In *The international symposium on spatial data handling* (pp. 1040–1049). Zurich: SDH.
- Xu, T., Li, X., & Claramunt, C. (2018a). Trip-oriented travel time prediction (TOTTP) with historical vehicle trajectories. *Frontiers of Earth Science*, 12(2), 253–263. <https://doi.org/10.1007/s11707-016-0634-8>.
- Xu, T., Zhang, X., Claramunt, C., & Li, X. (2018b). TripCube: A trip-oriented vehicle trajectory data indexing structure. *Computers, Environment and Urban Systems*, 67, 21–28. <https://doi.org/10.1016/j.compenvurbsys.2017.08.005>.
- Yang, Y., Papadopoulos, S., Papadias, D., & Kollios, G. (2009). Authenticated indexing for outsourced spatial databases. *The VLDB Journal*, 18(3), 631–648. <https://doi.org/10.1007/s00778-008-0113-2>.
- Yiu, M. L., Tao, Y., & Mamoulis, N. (2008). The B dual-Tree: indexing moving objects by space filling curves in the dual space. *The VLDB Journal*, 17(3), 379–400. <https://doi.org/10.1007/s00778-006-0013-2>.
- Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache spark: A unified engine for big data processing. *Communications of the ACM*, 59(11), 56–65. <https://doi.org/10.1145/2934664>.
- Zheng, Y., Liu, Y., Yuan, J., & Xie, X. (2011, September). Urban computing with taxicabs. In proceedings of the 13th international conference on ubiquitous computing (pp. 89–98).
- Zhou, P., Zhang, D., Salzberg, B., Cooperman, G., & Kollios, G. (2005, November). Close pair queries in moving object databases. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems* (pp. 2–11).
- Zhou, Y., Zhang, Y., Ge, Y., Xue, Z., Fu, Y., Guo, D., Shao, J., Zhu, T., Wang, X., & Li, J. (2017). An efficient data processing framework for mining the massive trajectory of moving objects. *Computers, Environment and Urban Systems*, 61, 129–140. <https://doi.org/10.1016/j.compenvurbsys.2015.03.004>.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.