Multiple Routes Recommendation System on Massive Taxi Trajectories

Yaobin He, Fan Zhang*, Ye Li, Jun Huang, Ling Yin, and Chengzhong Xu

Abstract: This paper presents a cloud-based multiple-route recommendation system, $x$Go, that enables smartphone users to choose suitable routes based on knowledge discovered in real taxi trajectories. In modern cities, GPS-equipped taxicabs report their locations regularly, which generates a huge volume of trajectory data every day. The optimized routes can be learned by mining these massive repositories of spatio-temporal information. We propose a system that can store and manage GPS log files in a cloud-based platform, probe traffic conditions, take advantage of taxi driver route-selection intelligence, and recommend an optimal path or multiple candidates to meet customized requirements. Specifically, we leverage a Hadoop-based distributed route clustering algorithm to distinguish different routes and predict traffic conditions through the latent traffic rhythm. We evaluate our system using a real-world dataset (>100 GB) generated by about 20,000 taxis over a 2-month period in Shenzhen, China. Our experiments reveal that our service can provide appropriate routes in real time and estimate traffic conditions accurately.

Key words: route recommendation; route clustering; traffic prediction; cloud computing

1 Introduction

Finding the shortest route, in terms of driving time, is a subject of intense interest in the field of smart cities. It also has great practical significance on a global scale. The selection of fast driving routes mostly depends on route attributes such as distance, speed limits, traffic lights, and other traffic restrictions. At present, commercial online maps provide a service of finding the fastest driving path based on the speed constraint of each road segment. However, the real travel time of a driving route is obviously different from the result estimated using speed constraints[1]. In fact, it depends even more on the time-dependent traffic flow on the route. For example, a short route may cost a driver more time during rush hour or when there is a traffic accident. Hence, the route selection should consider traffic conditions in real time, along with route attributes.

Real-time traffic is sensed in several ways: (1) via taxicabs equipped with Global Positioning System (GPS) devices traversing metropolitan areas, (2) by official video cameras and sensors on the roads, (3) from some special investigators, etc. The first of these is considered the best for collecting real-time and fine-grained traffic information practically, reliably, and cost-efficiently. Furthermore, the traffic information reports appear constantly at regular intervals (e.g., approximately 0.5–2 min[1–3]). A huge volume of GPS trajectories is thus generated over time. Its size makes it a big-data knowledge-mining challenge for researchers.

In this paper, we present a route recommendation system, $x$Go. It can enable users to choose suitable paths based on knowledge discovery from a large number of taxi trajectories without the help from Geographic Information Systems (GIS) in advance. We adopt a cloud-based solution to handle the huge amount of traffic data generated by the taxicabs (about 2 GB every day), and to respond to a large number of queries in real
time. Our system continuously analyzes the correlation of routes and computes real-time traffic conditions. When client requests arrive, xGo employs taxi-driver routing strategies and provides optimal paths to the online platform. The contributions of this paper are summarized as follows.

- By mining massive taxi trajectory datasets and traffic patterns in clusters, we propose a real-time route recommendation system that gives best-fitting paths according to customized requirements and real-time traffic conditions. Furthermore, it can accurately predict future traffic conditions around these routes to support the recommendation. We implement it as a real public system service to demonstrate its accuracy and efficiency.

- We propose a clustering algorithm to distinguish different routes from customized searching results without the assistance of GIS information. In order to meet the requirement of near-real-time response, we design an efficient storage strategy for massive datasets of spatio-temporal trajectory data. Experiments reveal that it can improve indexing and searching performance explicitly.

The rest of this paper is organized as follows. Section 2 is an overview of our system. Then we address our storage strategy and indexing in Section 3. Next, we propose a route clustering algorithm and dynamic recommendation strategy in Sections 4 and 5, respectively. We evaluate the performance of our system in Section 6. Related work is presented in Section 7. Finally, we conclude the paper in Section 8.

2 System Overview

We propose xGo as our online route recommendation system solution that meets main requirements, including real-time capability, accuracy, low computational complexity, and scalability. It comprises three components: Big Data Analyzer, Taxi Information Server, and Intelligent Devices, as shown in Fig. 1.

Big Data Analyzer runs on a cloud service platform based on Hadoop\(^4\). HBase\(^5\) is embedded into the system as our database. We refine our storage strategy for massive data in order to reduce searching time. Also, we include route clustering and recommendation algorithms in this component to achieve availability and real-time capability.

Our Taxi Information Server provides interfaces to translate user requests into commands to the back-end servers, and report the results to users.

Passengers can use Intelligent Devices (e.g., smartphones) to communicate with the cloud service platform to inquire about online driving routes and traffic information. Each query should include the positions of the start and destination point, and a user can specify these positions by pointing at the map on the screen of a smartphone.

3 Storage Strategy on Spatial Trajectories

The storage strategy in Big Data Analyzer determines its searching performance. We choose HBase, a NoSQL database, to store the massive taxi trajectory datasets for its good scalability. HBase is a key-value database on Hadoop that is suitable for random, real-time read/write access to big data\(^5\). Also, it is good at storing unstructured data. Specifically, GPS records can be regarded as unstructured data because different trajectories have different GPS points and different lengths.

3.1 Indexing

The indexing solution can be illustrated as follows. We divide the map averagely into 4×4 sub-zones recursively. A balanced tree with 16 branches is constructed to represent all the sub-zones obtained at different steps by the tree nodes. We index the nodes at each level of the tree from 1 to 16 in a fixed order. Each node also inherits the index from its parent node as the prefix. This partition and indexing schemes are illustrated in Fig. 2. Our strategy of partitioning is similar to that of Geohash\(^6\). A main difference between our method and Geohash is that we make 16 divisions for each zone, which correspond to the
geographical distribution of Shenzhen.

The map of Shenzhen can be divided into $2^{32}$ zones by this indexing, which are represented by a complete 8-level tree. It is considered that the degree of precision is enough since each leaf zone covers no more than $100 \text{m}^2$. The index of each zone is thus a 32-bit Boolean string. Every group of four digits in the string from left to right indicates the index of the zone in a distinct tree-level, from which we can infer its geographical location at the corresponding precision. To store a GPS point, we construct its key as the index of the zone that this point falls in. The lexicographical order of the keys thus provides a step-wise proximity rank for GPS points. To search the closest neighbors of a given point, we first obtain the key of this point and try to retrieve the records with the same key. If no record is found, we mask the last 4-digits of the key and search with the remaining key, until records with matching keys are found.

### 3.2 Storage Hierarchy

A two-tier hierarchical strategy is designed to manage the data in HBase. At the bottom tier, we create a Path Table to store and index all the routes of taxis serving passengers. The Path Table field is a string that contains the GPS position and the timestamp of all the data points belonging to this path. At the upper tier, we create a Point Table to store all the GPS points, the path index a GPS point belongs to, and the position of this point on the corresponding path. The schemas of both tables are shown in Fig. 3.

This strategy can make possible customized query functionality, while previous works like Refs. [1, 7] only support limited numbers of locations (landmarks) as start/end points. Our system can extract routes that pass through any two arbitrary positions selected by the user. With two locations given, we search them as candidates of start/end points from the Point Table, based on their geographic locations. Then, two sets of path indexes are extracted for each point. We then compute the intersections of these two sets and acquire the indexes of paths that include these two points. As a result, all the candidate routes can be fetched from the Path Table.

The storage strategy of city road networks (Fig. 4) is set similar to Fig. 3. We utilize the same indexing method for the road networks data in order to provide efficient and effective range queries, suitable for map-matching and traffic estimation in Map-Reduce architecture. It means that nodes within a short distance of GPS points will be stored in the same partitions. This strategy enables quick range query for nodes and roads near a specific GPS point.

### 4 Route Clustering

All the historical taxi-routes passing two given positions can be retrieved based on the storage strategy discussed in Section 3. In this section, we perform clustering on these routes without traditional road map matching. It can merge raw GPS points into limited candidate routes and makes route analysis more accurate.

#### 4.1 Regularization

Different routes may consist of different numbers of GPS points (i.e., different dimensions). We should regularize route data at first in order to project all routes into a unified space with fixed dimensions.

A GPS position can be considered as a point defined in 2-dimensional space, and a route can be regarded as a discrete set consisting of a multiple number of GPS points. Suppose the raw data of a route $R$ having $n$ GPS...
points:
\[ R = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \]  
(1)

where
\[ n \in \mathbb{Z}^+ \] and \( x_i \in [-90^\circ, 90^\circ], y_i \in [-180^\circ, 180^\circ] \)

(2)

\( x_i \) is longitude and \( y_i \) is latitude, and the starting position is \((x_1, y_1)\), the terminal position is \((x_n, y_n)\).

Represent \( R \) to be a function set \( y = F_R(x) \).

\[ y = F_R(x) = \begin{cases} 
  a_1x + b_1, & \text{if } x \in [x_1, x_2); \\
  a_2x + b_2, & \text{if } x \in [x_2, x_3); \\
  \vdots \\
  a_nx + b_n, & \text{if } x \in [x_{n-1}, x_n] 
\end{cases} 
\]

(3)

where
\[ \begin{align*}
  a_i &= (y_{i+1} - y_i)/(x_{i+1} - x_i), \text{ if } x_{i+1} \neq x_i; \\
  b_i &= y_i - a_ix_i, \\
  a_i, b_i &\in \mathbb{R}, \\
  x_i \in [-90^\circ, 90^\circ], y_i \in [-180^\circ, 180^\circ]; \\
  n \in \mathbb{Z}^+, \\
  i \in [1, n] 
\end{align*} \]

(4)

After transforming route data by Eq. (3), all routes can be projected into a feature space with fixed dimensionality by regularization. Denote \( S \) as the Plane Rectangular Coordination Systems (PRCS) in which the GPS position is defined. \( S^\alpha \) is the result of rotating \( S \) counter-clockwise with angle \( \alpha \).

Suppose there is a set consisting of routes \( \{R_1, R_2, \ldots, R_m\} \), all of which have the same starting point and terminal point:

1. Construct PRCS \( S^\alpha \) and transfer \( F_R(x) \) to be \( F_R^\alpha(x) \).
2. Get \( k \) sample points \([x^\alpha_{i,1}, x^\alpha_{i,2}, \ldots, x^\alpha_{i,k}] \) from \( x \)-axis of \( S^\alpha \), and for each \( x^\alpha_{i,j} \), calculate \( y^\alpha_{i,j} = F_R^\alpha(x^\alpha_{i,j}) \) for each \( R_j \) \((j \in \mathbb{Z}^+ \) and \( 1 \leq j \leq m \)). At the same time, get \( k \) sample points \([y^\alpha_{1,1}, y^\alpha_{1,2}, \ldots, y^\alpha_{1,k}] \) from \( y \)-axis of \( S^\alpha \), and for each \( y^\alpha_{1,j} \), calculate \( x^\alpha_{1,j} = (F_R^\alpha)^{-1}(y^\alpha_{1,j}) \) for each \( R_j \) \((j \in \mathbb{Z}^+ \) and \( 1 \leq j \leq m \)) where \((F_R^\alpha)^{-1}\) is the inverse function of \( F_R^\alpha \).
3. The regularization of \( R_j \) \((j \in \mathbb{Z}^+ \) and \( 1 \leq j \leq n \)) is
\[ \left\{ (x^\alpha_{1,j}, y^\alpha_{1,j}), (x^\alpha_{2,j}, y^\alpha_{2,j}), \ldots, (x^\alpha_{k,j}, y^\alpha_{k,j}) \right\} \]

(5)

After regularization, the representation of route \( R \) is changed from Eq. (1) to Formula (5). The regularization is shown in Fig. 5.

There is a parameter set \{\( \alpha, k \)\}. A route can be seen as a curve which is in a 2-dimensional space, and \( \alpha \) is a parameter which can control the viewing angle of the route. \( k \) is a parameter to control the resolution of interpreting a route. More accurate and detailed characteristics of this route can be captured when \( k \) is larger. If \( k \) is too small, its shape feature may be lost.

4.2 Clustering algorithm

After regularization, all routes can be seen as being projected into a space with fixed-dimensions \( 2k \). Based on the representations in Formula (5), the computation of distance is defined as calculating the difference between any two routes. Suppose function \( f \) describes the distance between any two points in the space of any dimension. Given the regularization \( R_a = \{(x^\alpha_{1,a}, y^\alpha_{1,a}), \ldots, (x^\alpha_{k,a}, y^\alpha_{k,a})\} \), and \( R_b = \{(x^\alpha_{1,b}, y^\alpha_{1,b}), \ldots, (x^\alpha_{k,b}, y^\alpha_{k,b})\} \), we define the difference of any two routes \( R_a \) and \( R_b \) as

\[ D(R_a, R_b) = \sum_{i=1}^{k} f((x^\alpha_{i,a}, y^\alpha_{i,a}),(x^\alpha_{i,b}, y^\alpha_{i,b})) \]

(6)

For calculating the distance between any two points of any dimension, we adopt three types of distance (i.e., Euclidean Distance, Manhattan Distance, and Chebyshev Distance). Then, Eq. (6) can be simplified to be

\[ D(R_a, R_b) = \sum_{i=1}^{k} f((x^\alpha_{i,a}, x^\alpha_{i,b}),(x^\alpha_{i,a}, y^\alpha_{i,b})) \]

(7)

Equation (7) demonstrates that the difference between \( R_a \) and \( R_b \) can be calculated based on two
variables \( \{y_{1,a}^α, \ldots, y_{n,a}^α, x_{1,a}^α, \ldots, x_{k,a}^α\} \) and \( \{x_{1,b}^α, \ldots, x_{k,b}^α, y_{1,b}^α, \ldots, y_{k,b}^α\} \). Therefore, \( R_i \) (\( i \in \mathbb{Z}^+ \) and \( 1 \leq i \leq n \)) can be represented by
\[
\{x_{1,i}^α, \ldots, x_{k,i}^α, y_{1,i}^α, \ldots, y_{k,i}^α\}
\]

Based on Formula (8), all routes can be seen to be projected into a space with dimensions \( 2k \). Then we can enable general analysis methods on the regularized data. We utilize Principal Component Analysis (PCA) to reduce the dimensionality \( 2k \) to 3. In the real world, most normal trajectories repeat in several routes that people usually take, while abnormal routes are the result of error or abnormal GPS signals. So in Fig. 6 it shows that there are mainly four-point groups, or four popular routes, and other dispersed points which can be considered noises.

Next, we perform clustering by a classic clustering algorithm \( K \)-means\(^8\), which is computationally faster than hierarchical clustering when \( K \) is small. The performance of \( K \)-means greatly depends on the value of \( K \), which is a parameter determining the number of clusters in data. In this paper, we use the Davies-Bouldin Index (DBI)\(^9\) to measure clustering results and determine the value of \( K \). DBI can be calculated using Eq. (9):
\[
\text{DBI} = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)
\]
where \( n \) is the number of clusters, \( c_i \) is the center of cluster \( i \), \( \sigma_i \) is the average distance of all elements of cluster \( i \) to the centroid \( c_i \), and \( d(c_i, c_j) \) is the distance between centers \( c_i \) and \( c_j \). The clustering result which has the smallest DBI is considered to have the best performance, and \( K \) can be determined by that.

4.3 Time-complexity and parallelism

Suppose there are \( m \) routes in data, where route \( i \) has \( n_i \) GPS points and \( k \) sample points for regularizing each route. Then the computation complexity is \( O(n_1 + \cdots + n_m) \) for transforming all data into Eq. (3). For route \( i \), computing the projection of sample points needs at worst \( O(n_i) \) for each sample point, and therefore in total \( O(k(n_1 + \cdots + n_m)) \) for regularizing all route data. When \( k \) is a constant, it is \( O(n_1 + \cdots + n_m) \). In the clustering phase of \( K \)-means, every route is represented by a point in \( k \)-dimensional space. Suppose there are at most \( c \) iterations; then the complexity is \( O(mKc) \). When \( c \) and \( K \) are integers that only change in a small range, the complexity of \( K \)-means can be considered as \( O(m) \).

The time complexity of whole processing is generally \( O(n_1 + \cdots + n_m) \) in real application. Suppose every route has the same number of GPS point \( n \). Then it is \( O(mn) \) for a complete process. It is a good choice to speed up this processing by parallel computing. Most of the processing time is consumed by regularizing data, which is \( O(mn) \), while it can be performed off-line as a pre-processing step by leveraging Map-Reduce\(^4\). In the Map-Reduce mechanism, the Map process can convert original route data into a representation of alignments, and output a single trajectory with the key indicating a unique pair of starting point and terminal point. Then a set of routes with the same pair of endpoints will be partitioned into a single pair of keys and a list of values through the Reduce process. The regularization of these routes will be performed in each Reducer task. This data preparation process can be calculated offline, while the clustering algorithm \( K \)-means may be executed online, because its computation time increases linearly.

5 Dynamic Recommendation

The clustered routes are matched to the roads by map matching algorithm in Ref. [10]. Then a small number of frequently traveled routes are presented to users. Specifically, these routes may be different types (e.g., the shortest distance, the fastest driving path, and the most traveled path). We utilize traveling distance and frequency of use as measurements to identify the shortest route and the most-traveled route, respectively. When selecting the fastest driving route, we need to predict traffic conditions of each route for the specific time in the query. Therefore, the recommendation of

Fig. 6 The distribution of routes when we reduce the dimensionality to three through PCA. Different clusters indicate different shapes of routes.
our system is dynamic in terms of the query time, and also customized for each user.

5.1 Problem definition

From the analysis of the average speeds of sampling roads, shown in Fig. 7, we find that the average speeds of the roads probably share a periodic characteristic along the temporal dimension. That means the average speed of a specific road may vary periodically in a day and also a week.

To utilize the time-series patterns in average speed of each roads in traffic estimation, we introduce time-series analyzing tools to model the problem. We define the historical average speeds of a specific road as a vector \( V = (v_1, v_2, \ldots, v_T) \), where \( v_i \) is denoted as the average speed in the \( i \)-th time-slot (each time-slot is a 15-minute interval in our system). It should be noticed that the length of \( V \) is at least larger than one period (672 time slots in a weekly period). \( V \) may also be a sparse vector for some roads\(^{[11]}\). We also denote the average speed in the current time slot \( v_t \). If the \( v_t \) of a road inside our recommended routes is missing, we estimate it, denoted as \( \hat{v}_{t+1} \), with the known elements in \( V \).

5.2 Estimating methods

We adopt four methods mentioned by Ref.\(^{[12]}\) to estimate traffic conditions for the dynamic recommendation.

- Random Walk Estimating. Normally, traffic conditions change gradually and smoothly during a certain period. The variation in speeds of a specific road in successive time slots is slight. The logic of Random Walk is to recover the missing values \( \hat{v}_{t+1} \) with their nearest neighbors \( v_c \), which are the latest non-empty element in \( V \), by \( \hat{v}_{t+1} = v_c \).

- Historical Average Estimating. The daily or weekly homogeneity in traffic condition provides information to infer the missing values from historical data. If the data in the same time slot from the previous periods are available, we can simply calculate the average speed in the current period by this equation:

\[
\hat{v}_{t+1} = \omega v_{t+1-T} + (1 - \omega) \hat{v}_{t+1-T}
\]

where \( T \) is the number of time slots one period holds and \( \omega \) is the smoothing parameter derived from experience. Notice that each time slot can have its own smoothing parameter and it lies between 0 and 1. It is suggested to set it smaller than 0.3 (in Ref.\(^{[12]}\)). It responds well to roads whose average speeds increase or decrease smoothly and routinely.

- Deviation from Historical Average Estimating. We label the latest known value to be \( v_c \). This estimation for \( \hat{v}_{t+1} \) is a combination of both historical average \( \hat{h}_c \) and \( \hat{h}_{t+1} \) in a proportional equation:

\[
\hat{v}_{t+1} = \frac{v_c}{\hat{h}_c} \times \hat{h}_{t+1}
\]

- ARIMA Model. An approximation of the missing value \( v_{t+1} \) is calculated by a fitted seasonal ARIMA\((1,0,1) \times (0,0,1)\) model\(^{[12]}\) in Eq. (12).

\[
\hat{v}_{t+1} = v_{t-T} + \phi_1 (v_t - v_{t-T}) - \theta_1 (v_t - \hat{v}_t) - \theta_1 (v_{t-T} - \hat{v}_{t-T}) + \theta_1 \Theta_1 (v_{t-T} - \hat{v}_{t-T})
\]

This equation is recursive and can be intuitively divided into five components. \( v_{t-T} \) denotes the information from the last period. \( \phi_1 (v_t - v_{t-T}) \) holds a parameter \( \phi_1 \) denoting the weighting parameter of the periodical trends between days and weeks of the traffic condition. It is worthy to mention that \( \theta_1 \) and \( \Theta_1 \) are both weighting parameters, and \( \theta_1 (v_t - \hat{v}_t) \), \( \theta_1 (v_{t-T} - \hat{v}_{t-T}) \), \( \theta_1 \Theta_1 (v_{t-T} - \hat{v}_{t-T}) \) are recursive and represent estimating deviations, respectively.

5.3 Parallelism

Statistics on historical trajectory data can be easily implemented into a Map-Reduce framework. This involves two main procedures: map matching and historical average speed calculation. With multiple tables inputs ("Point Table" and "Path Table") into the Mapper process, each Mapper task quickly searches for candidate road segments from "Node Table" and "Road Table" and then map each GPS point to

---

Fig. 7 Weekly patterns for traffic condition (Jan. 2, 2012 – Jan. 9, 2012).
corresponding road segments. Indexes of roads and speed measurements of each GPS record will then be the key and value as output of each Mapper tasks to a Reducer task to complete the statistic. Parameters can be set before the Mapper process to determine the time window, and in the Reducer process to determine the time granularity. After the whole process of Map-Reduce, a matrix of historical traffic conditions will be stored in the database until a periodic update.

6 Evaluation

6.1 Performance of searching

The most common queries are specifying two arbitrary points as the departure and the destination on the map and finding all taxi-routes with passengers that pass through the neighborhood of these two points.

The evaluation is performed on 2 months of historical data (about 250 million data points in total) generated by approximately 20 000 taxis in Shenzhen. These records are reported every 30 to 60 s. The size of these raw data files is over 100 GB. For each taxi, the GPS records are stored in a file named by its ID, with five fields including time-stamp, longitude, latitude, instantaneous speed reading, and a Boolean value indicating whether the taxi is serving a client or not. The last Boolean value can be used to identify when and where a taxi is taking on or dropping a passenger. Different from Ref. [1], this work only uses the records collected from paid trips for analysis. The GPS positions in the records are assumed to have an accuracy of $\pm 10$ m. We employ territory range-check and taxi-derived-speed-check to repair the erroneous records.

We implement our system by utilizing two architectures as follows.

- **HBase cluster:** The HBase version is implemented in a Hadoop cluster deployed on 6 virtual machines (on two physical servers). Each virtual machine has two 2.40 GHz CPU cores with 2 GB memory. The storage strategy of the HBase server was introduced in Section 3.

- **MySQL database:** The MySQL database runs on a single server that has 16 CPU cores running at 2.40 GHz with 16 GB memory. We store taxi trajectories in two tables, which is the same as that of HBase version (see Fig. 3). But we divide the Point Table into 5000 sub-tables to speed the response time.

First, the sequential response experiment is evaluated by sending 1000 queries serially to search different taxi trajectories. The result is shown in Fig. 8. It shows that the response time of MySQL is stable and much shorter than that of HBase. However, it is noted that we entered more than 5000 sub-tables by hand in MySQL. Such a process is hard to maintain while remaining scalable. For practical usage, taxi trajectory data will increase day by day. The advantage of HBase can help us for massive data cases.

Second, we simulate concurrent queries, which could be common in real world use. The result is shown in Fig. 9. It indicates that HBase performs better than MySQL when simultaneous queries are common. The main advantage of the HBase solution is it can distribute large queries to different virtual machines.

6.2 Performance of route clustering

When two random points are chosen in the map, the system will response with lots of routes after searching. In the case of Fig. 10a, 121 routes are found and could be divided into 4 clusters by our clustering algorithm.

![Fig. 8 Response time of searching.](image)

![Fig. 9 Response time of intercurrent queries.](image)
Fig. 10  An example of route clustering. Our clustering algorithm is efficient to divide original 121 routes into 4 clusters. These clusters contain different types of routes (e.g., the shortest distance, the fastest driving path, and the most traveled path).

as Fig. 10 shows. In this subsection, we will evaluate both the correctness and the response time of route clustering.

6.2.1 Correctness
We extract those 121 routes from real trajectories, with additional Gaussian-distributed noises with variance in the range from $\pm 10$ m to $\pm 100$ m as our clustering test dataset. The result in Fig. 11 demonstrates that our algorithm has high precision and is robust even though noise increases. The correctness is higher than 90% while the noises increase to $\pm 50$ m and the correctness decreases slowly with the increase of noise. Because the GPS error in Shenzhen, China is generally around 10 m\[^{[13]}\], our clustering algorithm is accurate in practice.

6.2.2 Response time
The computation time of the clustering algorithm should be short enough to make our system applicable. We perform an experiment on the response time of the clustering algorithm by increasing the number of routes to be clustered. The results are illustrated in Fig. 12. The response time of our clustering algorithm increases from around 0.1 s to no more than 4.5 s with the number of routes increasing from $10^3$ to $2 \times 10^4$. In practice, the traveled routes are fewer than $10^4$ from recent data in months for one query. This proves that our system is suitable for online usage, since the clustering costs less than 1 s.

In addition, the results in Fig. 12 verify our analysis.
of the time complexity of the clustering algorithm in Section 4.3. The time cost of regularization increases quadratically, and dominates the time cost of the whole clustering process, while computation time of $K$-means increases linearly. It reveals that our clustering algorithm can be easily speeded up when the process of route regularization is calculated in parallel and offline.

6.3 Prediction and recommendation

This experimental data are the most traveled routes from December 20, 2011 to January 14, 2012, 15 min per time slot. All routes have the same length at 2976 time slots. We calculate the estimation at the last period (the last 672 time slots) for each route and measure the results with Root Mean Square Error (RMSE), Mean Absolute Deviation (MAD), and Mean Absolute Percentage Error (MAPE). A critical process in the implementation of Historical Average and ARIMA is to estimate all four parameters. We derive these parameters empirically from the training data: $\omega = 0.2, \phi_1 = 0.6, \theta_1 = 0.2, \theta_1 = 0.6$. From the result in Table 1, we can see that ARIMA performs better than other methods, with lower RMSE, MAD, and MAPE.

In addition, sparseness is ubiquitous in our dataset. Hence, we do a simulation experiment by dropping the latest values of each road and re-running Random Walk and ARIMA tests. The parameters in ARIMA are adjusted to: $\phi_1 = 0.3, \theta_1 = 0.2, \theta_1 = 0.6$ through training. Figure 13 presents the change of the precision for Random Walk and ARIMA when the recent real values are unavailable. In this case, the lack of updated data greatly restricts the performance of Random Walk, but its precision does not decrease monotonically because of the periodic nature of traffic conditions. In contrast, ARIMA maintains stable and good performance even though recent days’ data are missing. The robustness of ARIMA derives from its integration of nearby and periodic data to estimate traffic conditions. It reveals that our system can accurately answer queries on upcoming traffic conditions for route recommendation.

We also note the runtime of each of these four algorithms. Computation time is about 2.896 ms for Random Walk, 3.783 ms for Historical Average, 6.036 ms for Deviation from Historical Average, and 3.889 ms for ARIMA. Therefore it is appropriate to estimate traffic condition online with ARIMA. Furthermore, calculations for different routes can be performed in parallel, since we do not utilize spatial correlation between routes.

7 Related Work

Many previous research work have focused on the knowledge discovery from large numbers of taxi trajectories, such as trajectory storage[14], route clustering[13, 15, 16], and traffic prediction[2, 3, 12]. Specifically, integrated systems, which are typically built for route recommendations, are discussed in Refs. [1, 7].

Yuan et al.[1] presented a system work focusing on designing a smart route service for drivers based on historical taxi trajectories. They proposed a Top-k landmark graph method to estimate traffic conditions and model four traffic conditions with Markov chains. This method restricts the usability in queries from trivial locations and ignores the stochastic nature of real traffic conditions. Besides, its computation is very costly that they perform their traffic estimation and route recommendation after matching all the trajectories on the map. Different from their solution, our system provides route recommendations online and performs route clustering without the assistance of GIS information.

Another system, in Ref. [7], only supports route recommendation between fixed zones on a large scale.

| Table 1 Measurement on different estimation methods. |
|-------------------------------|
| RMSE  | MAD  | MAPE (%) |
|------------------|---|---------|
| Random Walk      | 65.1800 | 5.9727 | 15.99 |
| Historical Average | 92.1420 | 6.2150 | 17.17 |
| Deviation from H.A.  | 83.5278 | 6.5398 | 17.17 |
| ARIMA(1, 0, 1) × (0, 0, 1) | 7.3028 | 5.4963 | 15.47 |
and calculates average travel time of all trajectories between these zones as the estimated time cost. The method does not distinguish travel time from different routes. By comparison, we store all raw trajectories in two tables of a cloud platform and design a novel indexing and storage strategy for fast searching and data localization. Hence, our system is capable of searching individual trajectories efficiently in real use.

Additionally, neither Yuan et al. nor Balan et al. provided responses with multiple suitable routes. Multiple routes recommendation is important for the dispatching problems discussed in Refs. [17, 18]. In contrast to their elaborate methods with high workload, we simply enlarge our recommendation to at least three types of routes after our clustering algorithm (i.e., fastest routes, shortest routes, and most traveled routes). To the best of our knowledge, comparing to above systems, xGo is the first system of its kind, using distributed framework to retrieve information from a large scale of massive data. Our system provides a highly scalable framework that can respond to real-time queries on monthly trajectory datasets of over 100 GB, while only requiring a single commodity server.

8 Conclusion

A cloud-based route recommendation system, xGo, is presented in this paper to suggest suitable routes for smartphone users. An efficient and scalable storage strategy is designed on managing massive spatio-temporal trajectory data in order to provide real-time responses. Our system improves indexing and searching performance over those of existing approaches. A clustering algorithm is proposed to distinguish different routes without the assistance of GIS information. It gives best-fitting paths according to customized requirements and real-time traffic conditions. In our approach, near-future traffic conditions around these routes can also be accurately predicted, to support recommendations. It can be concluded from the results of our experiments that our system can accurately estimate traffic conditions and discover appropriate routes customized for users in real time.

References

[1] J. Yuan, Y. Zheng, X. Xie, and G. Sun, Driving with knowledge from the physical world, in Proc. 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Diego, CA, USA, 2011, pp. 316–324.
[2] R. Herring, A. Hofleitner, P. Abbeel, and A. Bayen, Estimating arterial traffic conditions using sparse probe data, in Proc. 13th IEEE Int. Conf. on Intelligent Transportation Systems (ITSC), Funchal, Portugal, 2010, pp. 929–926.
[3] T. Hunter, R. Herring, P. Abbeel, and A. Bayen, Path and travel time inference from gps probe vehicle data, presented at Workshop on Analyzing Networks and Learning with Graphs in the 23th Annu. Conf. on Neural Information Processing Systems (NIPS), Whistler, Canada, 2009.
[4] T. White, Hadoop: The Definitive Guide, 3rd ed. O’Reilly Media, Inc., 2012.
[5] L. George, HBase: The Definitive Guide. O’Reilly Media, Inc., 2011.
[6] Geohash, http://geohash.org/, 2016.
[7] R. K. Balan, K. X. Nguyen, and L. Jiang, Realtime trip information service for a large taxi fleet, in Proc. 9th ACM MobiSys Int. Conf. on Mobile Systems, Applications, and Services, Washington DC, USA, 2011, pp. 99–112.
[8] S. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129–137, 1982.
[9] D. L. Davies and D. W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 2, pp. 224–227, 1979.
[10] P. Newson and J. Krumm, Hidden Markov map matching through noise and sparseness, in Proc. 17th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems, Seattle, WA, USA, 2009, pp. 336–343.
[11] Z. Li, Y. Zhu, H. Zhu, and M. Li, Compressive sensing approach to urban traffic sensing, in Proc. 31th IEEE Int. Conf. on Distributed Computing Systems (ICDCS), Minneapolis, MN, USA, 2011, pp. 889–898.
[12] B. M. Williams and L. A. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results, Journal of Transportation Engineering, vol. 129, no. 6, pp. 664–672, 2003.
[13] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li, Towards mobility-based clustering, in Proc. 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Washington DC, USA, 2010, pp. 919–928.
[14] L. Wang, Y. Zheng, X. Xie, and W. Y. Ma, A flexible spatio-temporal indexing scheme for large-scale gps track retrieval, in Proc. 9th Int. Conf. on Mobile Data Management (MDM), Beijing, China, 2008, pp. 1–8.
[15] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, Optics: Ordering points to identify the clustering structure, in Proc. 1999 ACM SIGMOD Int. Conf. on Management of Data, Philadelphia, PA, USA, 1999, pp. 49–60.
[16] M. Nanni and D. Pedreschi, Time-focused clustering of trajectories of moving objects, Journal of Intelligent Information Systems, vol. 27, no. 3, pp. 267–289, 2006.
[17] S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman, and C. Ratti, Taxi-aware map: Identifying and predicting vacant taxis in the city, in Proc. 2010 1st Int. Joint Conf.
on Ambient Intelligence, Malaga, Spain, 2010.

[18] K. Yamamoto, K. Uesugi, and T. Watanabe, Adaptive routing of cruising taxis by mutual exchange of pathways, in Proc. 12th Int. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems (KES), Zagreb, Croatia, 2008.

Yaobin He is a postdoctoral fellow in Harbin Institute of Technology Shenzhen Graduate School and a fellow in the Smart City Research Institute of China Electronics Technology Group Corporation. He received the BSc degree from Sun Yat-sen University in 2003 and PhD degree in computer science from Chinese Academy of Sciences in 2013. His main research interests include urban computing, big data, and parallel computing.

Fan Zhang is an associate professor in the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He received the PhD in communication and information system from Huazhong University of Science and Technology in 2007. He was a postdoctoral fellow at University of New Mexico and University of Nebraska-Lincoln from 2009 to 2011. His research topics include big data processing, data privacy and network security, and wireless networks.

Chengzhong Xu received the PhD degree from the University of Hong Kong in 1993. He is currently a professor of the Department of Electrical and Computer Engineering of Wayne State University, USA. He also holds an adjunct appointment with the Shenzhen Institute of Advanced Technology of Chinese Academy of Science as the Director of the Institute of Advanced Computing and Data Engineering. His research interest is in parallel and distributed systems and cloud computing. He serves on a number of journal editorial boards, including IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Cloud Computing, Journal of Parallel and Distributed Computing, and CHINA SCIENCE Information Sciences.

Ye Li is a PhD candidate at the Department of Computer and Information Science, University of Macau, under the supervision of Dr. Leong Hou U. He received the BS degree from South China Normal University in 2009 and MS degree from Shenzhen Institutes of Advanced Technology in 2013. His current research focuses on spatial database and data mining.

Ling Yin received the PhD degree from the University of Tennessee at Knoxville in 2011. She is currently an associate professor at the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. Her research interests include spatiotemporal data analysis, spatiotemporal process simulation, and their applications in transportation and health areas.

Jun Huang received the BS and MS degrees from Sun Yat-Sen University in 2009 and 2013, respectively. He is currently an assistant researcher in the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His research interests include large scale data mining, large scale social network analysis, and recommender system.