IEICE TRANS. INF. & SYST., VOL.E99–D, NO.6 JUNE 2016

PAPER Special Section on Human Cognition and Behavioral Science and Technology

# Sparse Trajectory Prediction Method Based on Entropy Estimation

**Lei ZHANG**[†a)], ***Member***, **Leijun LIU**[†b)], ***and*** **Wen LI**[†], ***Nonmembers***

**SUMMARY** Most of the existing algorithms cannot effectively solve the data sparse problem of trajectory prediction. This paper proposes a novel sparse trajectory prediction method based on L-Z entropy estimation. Firstly, the moving region of trajectories is divided into a two-dimensional plane grid graph, and then the original trajectories are mapped to the grid graph so that each trajectory can be represented as a grid sequence. Secondly, an L-Z entropy estimator is used to calculate the entropy value of each grid sequence, and then the trajectory which has a comparatively low entropy value is segmented into several sub-trajectories. The new trajectory space is synthesised by these sub-trajectories based on trajectory entropy. The trajectory synthesis can not only resolve the sparse problem of trajectory data, but also make the new trajectory space more credible. In addition, the trajectory scale is limited in a certain range. Finally, under the new trajectory space, Markov model and Bayesian Inference is applied to trajectory prediction with data sparsity. The experiments based on the taxi trajectory dataset of Microsoft Research Asia show the proposed method can make an effective prediction for the sparse trajectory. Compared with the existing methods, our method needs a smaller trajectory space and provides much wider predicting range, faster predicting speed and better predicting accuracy.

***key words:*** *trajectory prediction, data sparsity, L-Z entropy estimation, sub-trajectory synthesis*

## 1. Introduction

As the usage of Global Positioning System (GPS) and Smart Mobile Device (SMD) becomes a part of our daily lives, the trajectory data is showing the explosive growth [1]. The landing location of hurricane and other natural disasters can be forecasted by predicting their moving routes, so we may prevent them. A number of Location Based Services (LBSs) require destination prediction of moving objects' trajectories. Patterson et al. [2] presented a method to learn a Bayesian model of a traveler moving through an urban environment. Anna Monreale et al. [3] extracted movement patterns named trajectory patterns from historical trajectories of moving objects, which are a concise representation of behaviors of moving objects as sequences of regions frequently visited with a typical travel time. A decision tree, named T-pattern tree, is learnt from the trajectory patterns that hold a certain area and it may be used as a predictor of the next location for a new trajectory by finding the best matching path in the tree. William Groves et al. [4] proposed a framework to predict trajectories by using global and local information based on four prediction algorithms: a frequency-based algorithm (FreqCount), a correlation-based algorithm (EigenStrat), a spectral clustering-based algorithm (LapStrat), and a Markov Chain-based algorithm (MCStrat). Brownian Bridge Model was improved for high resolution location predictions by Mao Lin [5]. PENG et al. [6] proposed a trajectory prediction based on markov chains, which builds directed graph of all trajectories and computes one-step to k-steps transition probability matrices. Guo Limin et al. [7] proposed an uncertain trajectory pattern mining algorithm to mine trajectory patterns, which are indexed by a novel access method for efficient query processing. However, the above methods suffer from the "data sparsity problem": there is no historical trajectory that can match the query trajectory.

To address the data sparsity problem of trajectory prediction, Zheng Yu et al. [8] proposed a novel method based on the Sub-Trajectory Synthesis (SubSyn) algorithm. SubSyn algorithm first decomposes historical trajectories into sub-trajectories comprising of two adjacent locations, and then connects the sub-trajectories into "synthesized" trajectories. However, this method has some drawbacks. The trajectory space is big so that the time taken by sub-trajectory synthesis is very long; the prediction accuracy would be reduced because of some abnormal trajectories which affect the reliability of "synthesized" trajectories in the trajectory space. In order to address these drawbacks, this paper improves the method that Zheng Yu proposed and proposes a sparse trajectory prediction method based on entropy estimation. Firstly, we use trajectory entropy to evaluate trajectory regularity.We implement the L-Z entropy estimation to compute the entropy of a trajectory sequence. Secondly, we do trajectory synthesis based on trajectory entropy and put synthesized trajectories into trajectory space. The trajectory synthesis can not only resolve the sparse problem of trajectory data, but also make the new trajectory space smaller and more credible. Finally, under the new trajectory space, we utilize Markov model and Bayesian Inference for destination prediction.

The remainder of this paper is organized as follows. In Sect. 2, we introduce the date sparsity problem. In Sect. 3, the trajectory synthesis based on entropy estimation is introduced. In Sect. 4, we provide an introduction of trajectory prediction method with data sparsity based on entropy estimation. In Sect. 5, we show the experiments and results to

Manuscript received September 28, 2015.
Manuscript revised March 4, 2016.
Manuscript publicized April 1, 2016.

[†]The authors are with School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, 221116, China.
a) E-mail: zhanglei@cumt.edu.cn
b) E-mail: ljliu@cumt.edu.cn

DOI: 10.1587/transinf.2015CBP0001

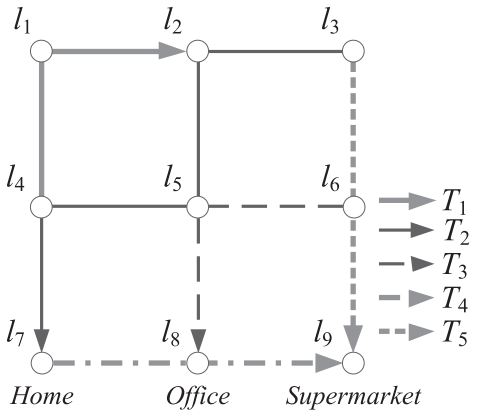


**Fig. 1** An example of trajectory prediction

demonstrate the effectiveness of the algorithm. Section 6 is the conclusion.

## 2. Data Sparsity Problem

A common approach to destination prediction is to make use of historical spatial trajectories of the public, available from trajectory sharing websites, or large sets of taxi trajectories. If an ongoing trip matches part of a popular route derived from historical trajectories, the destination of the popular route is very likely to be the destination of the ongoing trip (we refer to the ongoing trip as the query trajectory). As shown in Fig. 1, there are five historical trajectories: $T_1 = \{l_4, l_1, l_2\}$, $T_2 = \{l_3, l_2, l_5\ l_4\ l_7\}$, $T_3 = \{l_6, l_5, l_8\}$, $T_4 = \{l_8, l_7, l_9\}$ and $T_5 = \{l_3, l_6, l_9\}$. Each trajectory is represented by a type of line. For instance, a trip is taken from $l_3$ to $l_6$, and this query trajectory $\{l_3, l_6\}$ matches part of the historical trajectory $T_5$. Therefore, the destination of $T_5$ (i.e., $l_9$) is the predicted destination of the query trajectory.

However, the above strategy has a significant drawback. A location $l$ can be predicted as a destination only when the query trajectory matches a historical trajectory and the destination of the historical trajectory is $l$. In practice, $l_7$ and $l_8$ are also very likely to be the destination of the query trajectory, but will not be recommended to the user due to the limitation of the historical dataset. Moreover, if the query trajectory continues to be $l_5$, the above strategy will not be able to predict any destination since no historical trajectories contain the route $\{l_3, l_6, l_5\}$. We refer to this phenomenon as the data sparsity problem.

## 3. Trajectory Synthesis Based on Entropy Estimation

The main idea of trajectory synthesis based on entropy estimation is to serialize the trajectories and then use trajectory entropy to evaluate trajectory regularity. The sub-trajectories with low trajectory entropy are used to synthesize the new trajectory space with stronger regularity. Figure 2 shows the framework of trajectory synthesis based on entropy estimation.

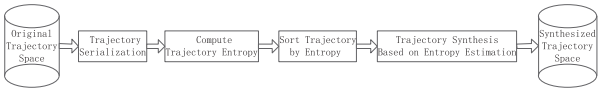


**Fig. 2** The framework of trajectory synthesis based on entropy estimation

### 3.1 Trajectory Description

For an original trajectory, it can be presented by n coordinate points with timestamp. Formally,

$$tra = \{(t_i, lon_i, lat_i) | t_i < t_{i+1}\}_{i=1}^{n} \tag{1}$$

where $t_i$, $lon_i$, $lat_i$ denote the point's time, longitude and latitude of trajectory. The map including all coordinate points is constructed as a two-dimensional grid which consists of $n \times n$ square cells. The granularity of this representation is a cell, i.e., all the locations within a single cell are considered to be the same object. Each cell has the side length of 1 and adjacent cells have the distance of 1.The whole grid is modelled as a graph where each cell corresponds to a node in the graph. All coordinate points are chronologically mapped to the grid graph so that a trajectory can be represented as a sequence of nodes according to the sequence of locations of the trajectory. Formally,

$$tra = (x_1, y_1) \rightarrow (x_1, y_1) \rightarrow \cdots \rightarrow (x_n, y_n) \tag{2}$$

where $x_n$, $y_n$ denote the row and column in the grid graph at time $n$ respectively. For the continuous time $t_i$ and $t_j$, if $(x_i, y_i) = (x_j, y_j)$, one can combine $(x_i, y_i)$ and $(x_j, y_j)$ into a node; by this analogy, one can combine all the neighbouring and same nodes of trajectory sequence.

$$tra = \{(x_i, y_i) | (x_i, y_i) \neq (x_{i+1}, y_{i+1})\}_{i=1}^{m} (m < n) \tag{3}$$

where $(x_i, y_i)$ is the node in the grid graph of trajectory sequence at time $i$.

### 3.2 Trajectory Entropy

In this paper, we use trajectory entropy to evaluate trajectory's regularity. We implement the L-Z entropy estimation on the basis of Lempel-Ziv complexity [9] and use it to compute the entropy of a trajectory sequence. For a trajectory sequence $tra = \{(x_i,y_i)\}_{i=1}^{m}$, the entropy can be computed by (4):

$$\begin{aligned} E(tra) &= \left(\frac{1}{len(tra)} \sum_{k=2}^{len(tra)} \frac{\Lambda_k}{\log_2(k)}\right)^{-1} \\ &= \left(\frac{1}{m} \sum_{k=2}^{m} \frac{\Lambda_k}{\log_2(k)}\right)^{-1} \end{aligned} \tag{4}$$

where $m$ is the number of nodes of trajectory $tra$, and $\Lambda_k$ is defined as the length of the shortest sub-trajectory starting at position $k$ that did not occur in the trajectory $\{(x_i, y_i)\}_{i=1}^{k-1}$ previously. It has been proven that $E(tra)$ converges to the actual entropy when $m$ approaches infinity [10], [11]. The

smaller entropy is, the stronger trajectory's regularity is, and the larger entropy is, the lower trajectory's regularity is (may be outlier trajectory).

### 3.3 Trajectory Synthesis Based on Entropy Estimation

It is obvious that there are some abnormal trajectories which affect prediction accuracy in the trajectory space. To enhance the regularity of the trajectory space, we do trajectory synthesis based on trajectory entropy and put synthesized trajectories into a trajectory space. Firstly, the map is constructed as a finer grid to make less overlap between the trajectories. For each trajectory $tra_i$ of the trajectory space, the entropy $e_i$ of $tra_i$ is computed. So the trajectory space can be obtained and it is sorted by entropy value $\{(tra_i,e_i)|e_i \le e_{i+1}\}_{i=1}^{n}$. Then $m$ (trajectory selection parameter we can set) trajectories which have comparatively low entropy value are chosen (higher regularity) as the new trajectory space. For every trajectory of the new trajectory space, if there are cross-nodes with other trajectories, one divides them into sub-trajectories by these cross-nodes and then compute the sub-trajectories' entropy by L-Z entropy estimation. The sub-trajectories are sorted by the sequence of nodes of the trajectory that is going to be synthesized. Keeping the sub-trajectories has lower entropy if there is overlapping among them (correspond to sub-trajectories of the trajectory is going to be synthesized). Finally, the remainder sub-trajectories with lower entropy is synthesized. The trajectory synthesis algorithm is shown as follows:

Algorithm 1 Trajectory Synthesis Algorithm Based on Entropy Estimation (TS-EE)

Input: historical trajectory space $Tra = \{tra_i\}_{i=1}^{n}$, trajectory selection parameter $m$.

Output: synthesized trajectory space *SynTra*.

1. $Sub\_Tra = \varnothing$ //store *sub_trajectories*
2. foreach $tra_i$ *in Tra*
3. $e_i = E(tra_i)$ as $(tra_i,e_i)$
4. arrange $\{(tra_i,e_i)\}_{i=1}^{n}$ by entropy as $\{(tra_i,e_i)|e_i \le e_{i+1}\}_{i=1}^{n}$
5. choose the minimum entropy of $m$ *trajectories* in *Tra* as $\{(SynTra_i,e_i)|e_i \le e_{i+1}\}_{i=1}^{m}$
6. foreach $tra_i$ in *SynTra*
7. foreach $tra_k \ne tra_i$ in *SynTra*
8. find all *cross-nodes* between $tra_k$ and $tra_i$
9. divide $tra_k$ and $tra_i$ into *sub_trajectories* by *cross-nodes*
10. store all *sub_trajectories* in *Sub_Tra*
11. foreach $sub\_tra_i$ in *Sub_Tra*
12. $E(sub\_tra_i)$
13. if *sub_trajectories* in *Sub_Tra* correspond to *sub_trajectories* of $tra_i$ have overlap then
14. keep *sub_tra* with minimum entropy
15. remove others from *Sub_Tra*
16. $syn\_tra_i$ = replace *sub_tras* of $tra_i$ with *sub_tras* in *Sub_Tra*
17. add $syn\_tra_i$ into *Syn_Tra*
18. return *Syn_Tra*

## 4. Trajectory Prediction Method with Data Sparsity Based on Entropy Estimation

The trajectory prediction method with data sparsity based on Entropy Estimation (TPDS-EE) uses the Markov model and Bayesian inference to do sparse trajectory prediction on the basis of the new trajectory space generated by TS-EE. It is proved that the method obtains good prediction efficiency.

### 4.1 Bayesian Inference Framework for Trajectory Prediction

The Bayesian inference framework for trajectory prediction [12], [13] contains two phases: training phase where the historical trajectories are mined offline and prediction phase to calculate the destinations online. Specifically, the probability of a node $n_j$ being the destination can be computed as the probability that $n_j$ is the destination location $n_d$, conditioning on the query trajectory $T^q$. Formally, the probability is computed using Bayes rule as

$$P(n_d = n_j|T^q) = \frac{P(T^q|n_d = n_j) \cdot P(n_d = n_j)}{\sum_{1\le k\le n^2}^{k}[P(T^q|n_d = n_j) \cdot P(n_d = n_k)]} \tag{5}$$

The prior probability $P(n_d = n_j)$ can be easily computed as the number of trajectories terminating at $n_j$ divided by the number of trajectories in the dataset. Formally,

$$P(n_d = n_j) = \frac{|T_{n_d=n_j}|}{|D|} \tag{6}$$

where $|D|$ is the cardinality of the historical trajectory dataset $D$, and $|T_{n_d=n_j}|$ is the number of trajectories in $D$ that terminate at location $n_j$. Therefore, the crux of Eq. (5) lies in computing the posterior probability $P(T^q|n_d = n_j)$. In order to solve this issue, we first count the number of trajectories satisfying two conditions: (i) it is partially matched by the query trajectory $T^q$; (ii) it terminates at location $n_j$. The count is then divided by the number of trajectories that terminate at location $n_j$ to serve as the likelihood function. Formally,

$$P(T^q|n_d = n_j) = \frac{|T_{n_d=n_j}|T^q \subset T_{n_d=n_j}|}{|T_{n_d=n_j}|} \tag{7}$$

where $|T_{n_d=n_j}|T^q \subset T_{n_d=n_j}|$ denotes the number of trajectories that satisfy both the aforementioned conditions and $|T_{n_d=n_j}|$ denotes the number of trajectories that terminate at a location in $n_j$.

The above method, which uses trajectory matching as the posterior probability, will be used as the baseline prediction algorithm. As discussed in Sect. 2, this method suffers from the data sparsity problem, i.e., if the query trajectory $T^q$ cannot be partially matched by any trajectory in $|D|$, then the numerator $|T_{n_d=n_j}|T^q \subset T_{n_d=n_j}|$ equals 0 and the probability of any node being the destination is 0. Consequently,

no destination can be predicted. It should be made clear that the baseline algorithm is not directly borrowed from the existing work, but rather an adapted version that utilizes the same approach. An adapted version of the baseline algorithm has been implemented such that the current node $n_c$ of the query trajectory is used as a predicted destination in the case where insufficient predicted destinations are generated by baseline algorithm.

### 4.2 Trajectory Prediction by Trajectory Synthesis Based on Entropy Estimation

SubSyn algorithm decomposes historical trajectories into sub-trajectories. The sub-trajectories comprise of only two adjacent locations. Then, the algorithm connects the sub-trajectories into synthesized trajectories. It produces so many trajectories that the training time is very long. Therefore, we propose trajectory prediction by trajectory synthesis based on entropy estimation. Our method generates smaller and more credible trajectory space by TS-EE. Under the new trajectory space, similar to SubSyn algorithm, we uses Markov model [14], [15] to train transition probability offline.

To fully leverage the information of historical trajectories, a Markov model is constructed by associating a state to each node $n_i$ in the $n\times n$ grid graph. Two directed transitions of states corresponding to nodes $n_i$ and $n_j$ are established, i.e., $n_i$ to $n_j$ and $n_j$ to $n_i$. Then an $n^2\times n^2$ transition matrix comprises the probability $p_{ij}$ that a user moves $n_i$ to $n_j$.

The transition probability of travelling from a location $n_i$ to a location $n_j$ is denoted by $p_{ij}$. These transition probabilities are conditional probabilities and can be computed as the number of sub-trajectories $ST_{n_s=n_i,n_d=n_j}$(starting at $n_i$ and finishing at $n_j$, $len(ST_{i,\cdots,j}) > 2$) divided by the cardinality of the sub-trajectory space $D_{sub-tra}$. Formally,

$$P_{ij} = \frac{|ST_{n_s=n_i,n_d=n_j}|}{|D_{sub-tra}|} \tag{8}$$

For each pair of nodes in the grid graph, we compute the transition probabilities offline using (8). These probabilities are stored as entries of a two-dimensional $n^2\times n^2$ matrix where one dimension corresponds to the node of current state and the other dimension corresponds to the next state. $M$(9) is the transition matrix of the example presented in the 3×3 grid graph.

$$M = \begin{pmatrix} 0 & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & p_{17} & p_{18} & p_{19} \\ p_{21} & 0 & p_{23} & p_{24} & p_{25} & p_{26} & p_{27} & p_{28} & p_{29} \\ p_{31} & p_{32} & 0 & p_{34} & p_{35} & p_{36} & p_{37} & p_{38} & p_{39} \\ p_{41} & p_{42} & p_{43} & 0 & p_{45} & p_{46} & p_{47} & p_{48} & p_{49} \\ p_{51} & p_{52} & p_{53} & p_{54} & 0 & p_{56} & p_{57} & p_{58} & p_{59} \\ p_{61} & p_{62} & p_{63} & p_{64} & p_{65} & 0 & p_{67} & p_{68} & p_{69} \\ p_{71} & p_{72} & p_{73} & p_{74} & p_{75} & p_{76} & 0 & p_{78} & p_{79} \\ p_{81} & p_{82} & p_{83} & p_{84} & p_{85} & p_{86} & p_{87} & 0 & p_{89} \\ p_{91} & p_{92} & p_{93} & p_{94} & p_{95} & p_{96} & p_{97} & p_{98} & 0 \end{pmatrix} \tag{9}$$

The total transition probability of travelling from one node $n_i$ to another node $n_k$, denoted by $p_{i\to k}$, is the sum of the $k$-steps transition probabilities of all possible paths between $n_i$ and $n_k$. Formally:

$$\begin{aligned} P_{i\to k} &= \sum_{r=2}^{SubSyn_{max}} M_{ik}^r \\ &= M_{ik}^2 + M_{ik}^3 + \cdots + M_{ik}^{SubSyn_{max}} \end{aligned} \tag{10}$$

In Eq. (10), $M^r(2\leq r\leq SubSyn_{max}$, $SubSyn_{max}$ is the maximum number of sub-trajectory synthesis) can be obtained by using C-K equation and $M$. $M^r$ holds the probabilities of transition from one node to another in exactly $k$-steps sub-trajectory synthesis. In an $n\times n$ grid graph, $M$ is an $n^2\times n^2$ matrix, and each matrix multiplication requires $O(n^6)$. The longest distance between two cells in the grid is $2(n$ - $1)$ so that $SubSyn_{max} \leq (n$ - $1)$. Therefore, the computation complexity of sub-trajectory synthesis is $O((n$ - $1) \times n^6)$.

In general, given any query trajectory $T^q$, the definition of path probability $P(T^q)$ is:

$$P(T^q) = P(T^k_{1,2,\cdots,k}) = \prod_{i=1}^{k} p_{i(i+1)} \tag{11}$$

After defining the total transition probability in (10) and the path probability in (11), given a query trajectory $T^q$, we calculate the posterior probability of a user travelling from the staring node $n_s$ to the current node $n_c$ via $T^q$ conditioned on the destination being in node $n_j$ by (12):

$$P(T^q|n_d = n_j) = \frac{P(T^q)\cdot p_{c\to j}}{p_{s\to j}} \tag{12}$$

where $P(T^q)$ is the path probability of the given query trajectory $T^q$; $p_{c\to j}$ is the total transition probability of people going from the current node of $T^q$, $n_c$, to a predicted destination $n_j$; and $p_{s\to j}$ is the total transition probability of travelling from the starting node of $T^q$, $n_s$, to a predicted destination $n_j$.

Lastly, the posterior probability is used when a user issues a query to compute destination probabilities by Bayesian inference. Formally:

$$\begin{aligned} P(n_d = n_j|T^q) &= \frac{P(T^q)\cdot \frac{p_{c\to j}}{p_{s\to j}}\cdot P(n_d = n_j)}{\sum_{1\leq k\leq n^2}^{k}[P(T^q)\cdot \frac{p_{c\to k}}{p_{s\to k}}\cdot P(n_d = n_k)]} \\ &\sim \frac{p_{c\to j}}{p_{s\to j}}\cdot P(n_d = n_j) \end{aligned} \tag{13}$$

### 4.3 The TPDS-EE Algorithm

TPDS-EE uses L-Z entropy estimation and TS-EE to reduce the size of the trajectory space and enhance the regularity of the trajectory space. For the new trajectory space, in order to overcome the data sparsity problem, TPDS-EE uses a Markov model and sub-trajectory synthesis to offline train the total transition probabilities needed to efficiently compute the posterior probability for any given query trajectory

online. On these bases, the destination probabilities of all nodes in the grid graph can be computed by Bayes rule.

Algorithm 2 Trajectory Prediction method with Data Sparsity based on Entropy Estimation

Input: historical trajectory space $Tra = \{tra_i\}_{i=1}^{n}$, grid granularity $n$, query trajectory $T^q$.

Output: *nodes* with *top-k* destination probabilities.

1. do trajectory synthesis based on L-Z entropy estimation for the historical trajectory space *Tra*, and then get the synthesized trajectory space *SynTra*
2. $SubSyn_{max} = n - 1$ // maximum number of sub-trajectory synthesis
3. decompose all the trajectories in *SynTra* into sub-trajectories by trajectory intersections and store the sub-trajectories in the space $D_{sub-tra}$
4. foreach pair of nodes $n_i$ and $n_i$ *in grid graph* do
5. $M \leftarrow p_{ij}$ // first order transition matrix
6. $M^T \leftarrow \sum_{r=2}^{SubSyn_{max}} M^r$ // total transition matrix
7. *result* $\leftarrow \varnothing$ // a set to store the destination probabilities
8. construct query path probability $P(T^q)$
9. foreach $n_j$ *in grid graph* do
10. retrieve $p_{c \rightarrow j}$ and $p_{s \rightarrow j}$ from $M^T$
11. compute $P(T^q \mid n_d = n_j)$, and then $P(n_d = n_j \mid T^q)$
12. store $P(n_d = n_j \mid T^q)$ in *result*
13. sort *result*
14. return *nodes* with *top-k* destination probabilities in *result*

## 5. Experiments Study and Analysis

In this section, we conduct an extensive experimental study to evaluate the performance of our TPDS-EE algorithm. It is worth mentioning that all of the experiments were run on a commodity computer with Intel Core i5 CPU (2.3GHz) and 4GB RAM. We use a real-world large scale taxi trajectory dataset from the T-drive project in our experiments [16]. It contains a total of 580,000 taxi trajectories in the city of Beijing, 15 million GPS data points from February 2, 2008 to February 8, 2008.

### 5.1 The Result of Trajectory Entropy

To evaluate trajectory regularity, we divide every day into twelve periods, and then compute the average trajectory entropy of each period of time on weekend and weekday respectively.

The result presented in Fig. 3 clearly shows that the trajectory entropies of twelve periods of time conform to the taxi traveling path, i.e., it is the go-to-work hours between 6:00 and 8:00, and the taxi traveling path is always regular from home to company, so the average entropy is the smallest. Consequently, trajectory entropy can be used to evaluate trajectory regularity.

### 5.2 Compared with Baseline Algorithm

As discussed in Sect. 4.1, the only available algorithm that can perform generic destination prediction is the baseline algorithm.To certify our TPDS-EE algorithm can effectively solve the data sparsity problem, we make the comparison between the baseline algorithm and TPDS-EE from three aspects: Prediction Accuracy, Prediction Time and Coverage.The prediction accuracy is computed as the ratio between the number of correctly predicted trajectories and the total number of trajectories. Prediction time is the time used to predict destination for one query trajectory online. And the coverage counts the number of query trajectories for which some destinations are provided. We use this property to demonstrate the difference in robustness between the baseline algorithm and our TPDS-EE algorithm.

Figure 4 and Fig. 5 show the trend in both prediction accuracy and prediction time with respect to grid granularity. The prediction accuracy of TPDS-EE is about 5% higher than the baseline algorithm in all grid granularities. For the baseline algorithm, the number of query trajectories which have sufficient destinations drops slightly as the grid

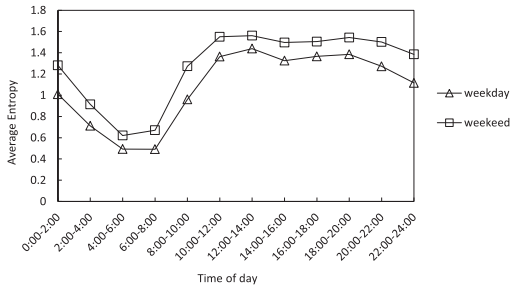


**Fig. 3** Trajectory entropy of different times of the day

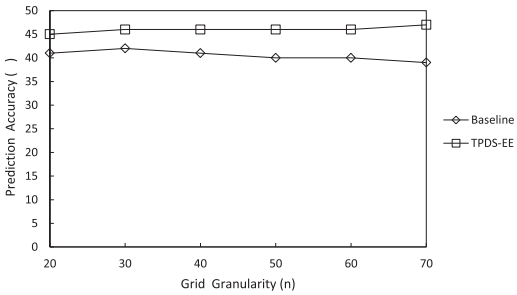


**Fig. 4** Prediction accuracy of different grid granularity obtained by Baseline and TPDS-EE

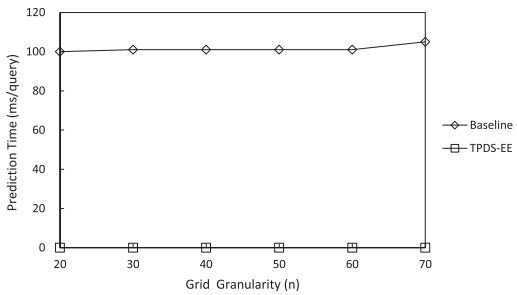


**Fig. 5** Prediction time of different grid granularity obtained by Baseline and TPDS-EE

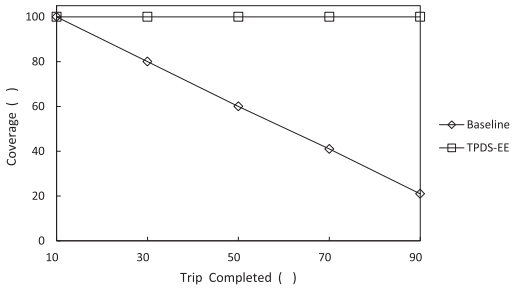


**Fig. 6** The coverage versus the percentage of trip completed obtained by Baseline and TPDS-EE

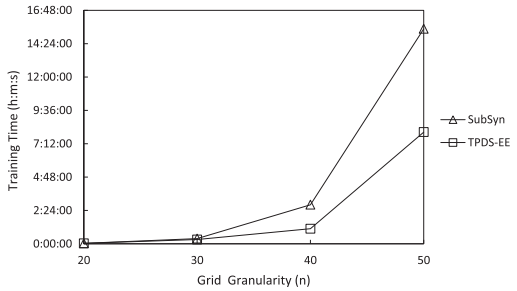


**Fig. 7** Training time of different grid granularity

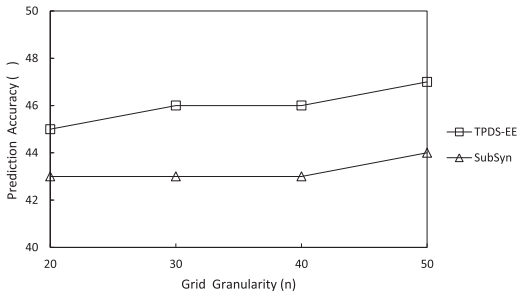


**Fig. 8** Prediction accuracy of different grid granularity obtained by TPDS-EE and SubSyn

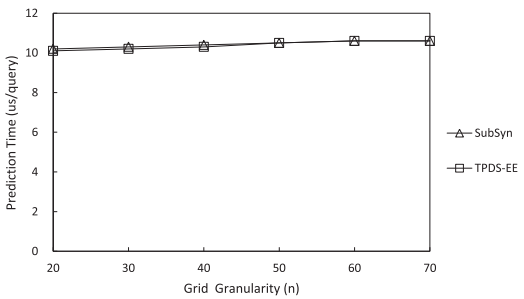


**Fig. 9** Prediction time of different grid granularity obtained by TPDS-EE and SubSyn

granularity increases due to more trajectories in the training dataset falling into different nodes. Hence query trajectories are less likely to have a partial match in the trajectory space. Meanwhile, the prediction accuracy of TPDS-EE is stable (46%). We compare the runtime performance of our TPDS-EE algorithm with the baseline algorithm in terms of online query prediction time. Since the information is stored during the offline training stage, TPDS-EE only requires little extra computation when answering a user's query ($10\mu s$), whereas the baseline algorithm requires too much time ($100ms$) to predict. Our TPDS-EE algorithm is at least four orders of magnitude better constantly. The reason is that the baseline algorithm is forced to make a full sequential scan of the entire trajectory space to compute the posterior probability, whereas TPDS-EE can fetch most transition probability values from the stored matrices $M^r$ directly. It is noted that grid granularity has little influence on the prediction time of the three algorithms.

Apart from the huge advantage of TPDS-EE in prediction accuracy and prediction time, its coverage is comparable with that of the baseline algorithm. Figure 6 shows the coverage versus the percentage of trip completed. For the baseline algorithm, the amount of query trajectories for which sufficient predicted destinations are provided decreases as the trip completed increases due to the fact that longer query trajectories (i.e., higher trip completed percentage) are less likely to have a partial match in the training dataset. Specifically, when trip completed percentage increases towards 90%, the coverage of the baseline algorithm decreases to almost 20%. Our TPDS-EE algorithm successfully copes with it as it is expected with only an unnoticeable drop in coverage and it constantly answers almost 100% of query trajectories. It proves that the baseline algorithm cannot handle long trajectories because the chances of finding a matching trajectory decrease when the length of a query trajectory grows.

### 5.3 The Comparison of TPDS-EE Algorithm with SubSyn Algorithm

To evaluate the performance of our TPDS-EE Algorithm and SubSyn Algorithm, we use Training Time, Prediction Accuracy, Prediction Time and Coverage as measurement. The first is the time used to train the total transition probability matrix. As discussed in Sect. 4.2, the time of both TPDS-EE and SubSyn Algorithm is mainly the time to compute total transition probabilities.

Figure 7 shows that the training time of both our TPDS-EE algorithm and SubSyn algorithm rises rapidly due to a larger matrix and more matrix multiplication in a fine grid, and with the increase of grid granularity n, the training time of TPDS-EE algorithm is less than the time of SubSyn. In a typical where $n = 50$, the training time of TPDS-EE is only half of SubSyn. The reason is that SubSyn algorithm decomposes historical trajectories into sub-trajectories just comprising two adjacent nodes, but TPDS-EE decomposes historical trajectories into sub-trajectories which include at least two nodes (the finer grid graph, the longer sub-trajectories) by trajectory crosses, so the time of matrix multiplication of SubSyn is more than the time of matrix multiplication of TPDS-EE. Therefore, TPDS-EE can obtain better runtime efficiency than SubSyn. Accord-

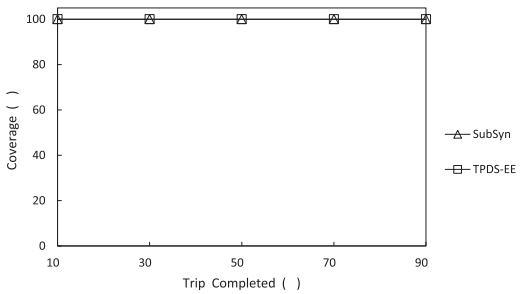


**Fig. 10** The coverage versus the percentage of trip completed obtained by TPDS-EE and SubSyn

ing to Fig. 8, the prediction accuracy of TPDS-EE is a little higher than SubSyn algorithm (about 3%). Where the bad effect of abnormal trajectories have been removed from the trajectory space by TS-EE so that the new trajectory space is more credible for trajectory prediction.

In Fig. 9 and Fig. 10, the prediction time ($10\mu s$) and coverage (100%) of both our TPDS-EE and SubSyn algorithm is same. Because our TPDS-EE and SubSyn algorithm adopt similar learning way in prediction. It makes their prediction time and coverage consistent.

## 6. Conclusions

In this paper, we propose an improved TPDS-EE algorithm to address the data sparsity problem. TPDS-EE algorithm uses L-Z entropy estimation to compute a trajectory's entropy and does trajectory synthesis based on a trajectory's entropy which can make the trajectory space smaller and more credible. And TPDS-EE uses Markov model and Bayesian inference to predict destination based on the new trajectory space that generated by TS-EE. Experiments based on the real dataset have shown that our TPDS-EE algorithm can predict destinations for almost all query trajectories, so it has successfully addressed the data sparsity problem. Comparing with SubSyn algorithm, TPDS-EE needs less training time. At the same time, our TPDS-EE algorithm also requires less time to predict and runs over four orders of magnitude faster than the baseline algorithm.

## Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities (2014XT04).

### References

[1] World Telecommunication/ICT development report, 2010, International Telecommunication Union, available from http://www.itu.Int/ITU-D/ict/publications/wtdr_10/index.html

[2] D.J. Patterson, L. Liao, D. Fox, and H. Kautz, "Inferring high-level behavior from low-level sensors," Proc. 5th Int. Conf. on Ubiquitous Computing (UbiComp), pp.73–89, Seattle, USA, Oct. 2003.

[3] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "Wherenext: a location predictor on trajectory pattern mining," Proc. 15th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining, pp.637–646, Paris, Franch, June 2009.

[4] W. Groves, E. Nunes, and M. Gini, "A framework for predicting trajectories using global and local information," Proc. 11th ACM Conf. on Computing Frontiers, no.37, Cagliari, Italy, May 2014.

[5] M. Lin and W.-J. Hsu, Brownian Bridge Model for High Resolution Location Predictions, vol.8444, pp.210–221, Springer International Publishing, 2014.

[6] Q. Peng, Z. Ding, and L. Guo, "Prediction of trajectory based on Markov chains," J. Computer Science, vol.37, no.8, pp.189–193, 2010.

[7] L. Guo, Z. Ding, Z. Hu, and C. Chen, "Uncertain path prediction of moving objects on road networks," J. Computer Research and Development, vol.44, no.1, pp.104–112, 2010.

[8] A.Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," Proc. 2013 IEEE International Conf. on Data Engineering (ICDE), pp.254–265, Brisbane, Australia, April 2013.

[9] Y. Gao, I. Kontoyiannis, and E. Bienenstock, "Estimating the entropy of binary time series: methodology, some theory and a simulation study," J. Entropy, vol.10, no.2, pp.71–99, 2008.

[10] C. Song, Z. Qu, N. Blumm, and A. Barabási, "Limits of predictability in human mobility," J. Science, vol.327, no.5968, pp.1018–1021, 2010.

[11] J. McInerney, S. Stein, A. Rogers, and N.R. Jennings, "Exploring Periods of Low Predictability in Daily Life Mobility," Proc. 10th Intl. Conf. on Pervasive Computing (Pervasive'12), Newcastle, UK, June 2012.

[12] J. Krumm and E. Horvitz, "Predestination: Inferring destinations from partial trajectories," Proc. 8th Int. Conf. on Ubiquitous Computing (UbiComp), Orange Country, CA, pp.243–260, Sept. 2006.

[13] L.-Y. Wei, Y. Zheng, and W.-C. Peng, "Constructing popular routes from uncertain trajectories," Proc. 18th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining, pp.195–203, BeiJing, China, Aug. 2012.

[14] D. Ashbrook and T. Starner, "Using GPS to learn significant locations and predict movement across multiple users," J. Personal Ubiquitous Computing, vol.7, no.5, pp.275–286, 2003.

[15] J.A. Alvarez-Garcia, J.A. Ortega, L. Gonzalez-Abril, and F. Velasco, "Trip destination prediction based on past GPS log using a hidden markov model," J. Expert Systems with Applications, vol.37, no.12, pp.8166–8171, 2010.

[16] T-drive trajectory data sample, Aug. 2011, M. Research, available from http://research.microsoft.com/apps/pubs/?id=152883

**Lei Zhang** was born in 1977 and received the Ph.D. degree in Computer Application Technology from Nanjing University of Aeronautics and Astronautics in 2006. He is now an associate professor in the School of Computer Science and Technology, China University of Mining Technology. He has published more than 20 papers in international conferences and journals. His research interests include trajectory data analysis and mining.

**Leijun Liu** received the B.E. degree in Computer Science and Technology from China University of Mining Technology in 2010. He is currently working as a M.E. student in the School of Computer Science and Technology, China University of Mining Technology. His main research interests include trajectory data analysis and mining.

**Wen Li** is currently a PhD candidate in the School of Computer Science and Technology, China University of Mining and Technology. Her main areas of interests are trajectory data mining and pattern recognition.