Fast Map-Matching Based on Hidden Markov Model

Shenglong Yan, Juan Yu(✉), and Houpan Zhou

Smart City Research Center, Hangzhou Dianzi University, Hangzhou, China
yujuan@hdu.edu.cn

Abstract. Map matching is the processing of recognizing the true driving route in the road network according to discrete GPS sampling datas. It is a necessary processing step for many relevant applications such as GPS trajectory data analysis and position analysis. The current map-matching algorithms based on HMM (Hidden Markov model) focus only on the accuracy of the matching rather than efficiency. In this paper, we propose a original method: Instead of focusing on a point-by-point, we consider the trajectory compression method to find the key points in the discrete trajectory, and then search for optimal path through the key points. The experiments are implemented on two sets of real dataset and display that our method significantly improve the efficiency compared with HMM algorithm, while keeping matching accuracy.

Keywords: Map matching · Efficiency · Trajectory compression · Key points

1 Introduction

With the popularity of electronic mobile devices with built-in GPS sensors, a large amount of driving tracks which contain rich traffic information and user behavior are generated every day. However, with limited satellite visibility and high-rise buildings, the satellite signal is blocked and refracted, and the positioning data we obtain through the GPS sensor is can be inaccurate and noisy, which means that GPS trajectories can not accurately reflect the location of moving objects. Therefore, map-matching is proposed to identify the actual road segment where the user (or vehicle) is/was driving. This process is useful in applications such as vehicle navigation [1], path planning and recommendation [2], traffic forecasting and management [3] and many other LBS (Location-Based Service).

Over the last decades, hundreds of map-matching algorithms were proposed by abundant researchers.

According to the information involved in the input data, Quddus et al. [4] divided the existing map-matching algorithms into four categories: geometric [5], topological [6], probabilistic and advanced. Geometric algorithm only considers the geometric information for identifying the real paths, such as distance, angle and shape, this type of algorithm can provide good accuracy and fast matching efficiency in the case of high sampling rate and accurate positioning, but it is not suitable for trajectory data with low sampling rate and large positioning error. Based on the geometric algorithm, topological algorithm considers the connectivity between road segments, so that the

Y. Yin et al. (Eds.): MobiCASE 2019, LNICST 290, pp. 85–95, 2019.
https://doi.org/10.1007/978-3-030-28468-8_7

accuracy can be improved to a certain extent. However, they are still susceptible to the influence of noise collection and sparse data, and cannot completely solve the complex urban road problems. In fact, we can classify probability algrthim as advanced algorithm which tends to incorporate comprehensive information and use more refined concepts, such as kalman filter [7] fuzzy logic model [8] hidden markov model [9–12] and so on, these advanced algorithms have generally higher accuracy, and the HMM matching algorithm has the highest accuracy. Although the advanced algorithm is better than geometry and topology in matching accuracy, it has poor matching efficiency.

QMM [11] (Quick Map Matching) is the first matching algorithm that emphasizes running time. The algorithm is designed to run on multi-core cpus, because the processing of road segments can be separated from each other during indexing, and each sample trajectory point is independent of each other during matching. The application of multithreading technology greatly reduces the running time of the algorithm. Furthermore, efficiency-based algorithms [13, 14] both consider parallel processing to speed up map matching computation. The above three methods are all based on the idea of processing multiple trajectories at the same time.

Different from the three efficiency-based algorithms, we propose a algorithm called FHMM(Fast map matching based on HMM) to speed up single track matching efficiency. Firstly, our algorithm employs the idea of trajectory compression to find a set of key points. Subsequently, riginated from HMM, we employs measurement probabilities and transition probabilities to measure the relationship between consecutive candidate points of key points in map-matching. Finally, to solve the HMM problem, we make use of dynamic programming Viterbi algorithm to search for optimal travel route.

This paper is organized as follows. Section 2 states the problem of map matching algorithm. The detail of the FHMM is introduced in Sect. 3. The experiment results are presented and analyzed in Sect. 4. Finally we summarize the paper in Sect. 5.

2 Preliminary

2.1 Map Matching Problem

To facilitate the description, we first define some basic concepts and symbols, and then formalize the map matching problem.

- **Definition 1 (GPS Trajectory)**: A GPS track $T : p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$ is an ordered sequence composed of a series of GPS coordinate points, in which each GPS point $p_i = (t, lat, lon, heading, speed)$ contains information such as sampling time t, latitude and longitude coordinates, GPS real-time direction heading and speed.
- **Definition 2 (Road Network)**: road network is composed of a series of intersection and connection crossroads sections, can be represented as a directed graph G (V, E). V is a vertex set, which contains all the intersection in the road or the road end point. E is directed edge set, which represents sections of road network.

- **Definition 3 (Path)**: A Path is a road segment sequence P: $r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_n$, where r_{i-1}.e = r_i.s, ($1 \leq i \leq n$). r.s and r.e represent the start point and end point of the road segment r respectively.

2.2 Candidate Points Selection

The preparation process of candidate sets is divided into two sub-steps. Step 1: to establish the r-tree index of road network data, which is mainly aimed at the road section to facilitate the quick search of candidate road sections. Step 2: based on the r-tree index of road segment, fast query all possible candidate road segments of each sampling point in the road network on track T, and then calculate the corresponding candidate points.

Specifically, for each sampling point $p_i(1 \leq i \leq n)$ in GPS track $T : p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, all sections within the search radius r of the road network with GPS point $(p_i.lon, p_i.lat)$ as the center of the circle are taken as candidate sections, denoted as $R_i = \{r_i^k | k = 1, 2, \cdots\}$, r_i^k represents the candidate section k of GPS point p_i, and the point closest to the sampling point on the candidate section is called the candidate point, denoted as c_i^k. Figure 1 gives an example of candidate points selection. After calculation, GPS point p_1 obtains four candidate sections, and the corresponding candidate points are c_1^1,c_1^2 , c_1^3 and c_4^1.

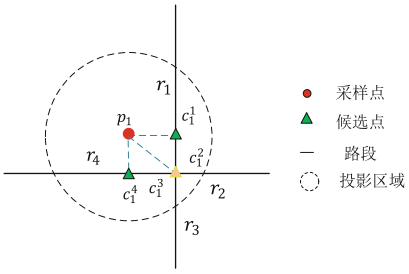

Fig. 1. Illustration of candidate sections and candidate points.

3 FHMM Algorithm

The proposed FHMM algorithm consists of four phases: trajectory preprocess, candidate preparation, HMM training, and result matching, of which candidate preparation has been introduced in Sect. 2. Figure 2 shows a framework of the FHMM algorithm.

3.1 Trajectory Preprocess

Generally speaking, the number of driving vehicle records with GPS acquisition devices is greater than the amount of data required by existing algorithms, such as HMM matching algorithm. The influences of intersections and signal controls mean that vehicles travelling on normal arterials often display a 'stop-and-go'pattern. In other

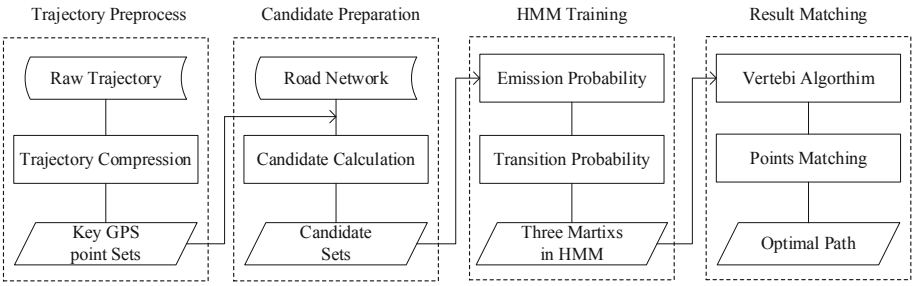

Fig. 2. Framework of the FHMM algorithm.

words, there will be lots of stop points in the raw trajectory. In addition, the vehicles tends to drive on the same road for a period of time, which means the driving route is close to a straight line. As a result, there is also a lot of GPS redundancy between the two ends of the line.

Our insight that compress a trajectory can remove stop points and redundant points that are between the key points. An example is illustrated in Fig. 3, obviously, the blue sampling points that can increase the computational cost in map matching are redundant. In the actual matching process, we only need the key sampling points (such as red sampling points) in the original trajectory to infer the entire driving route.

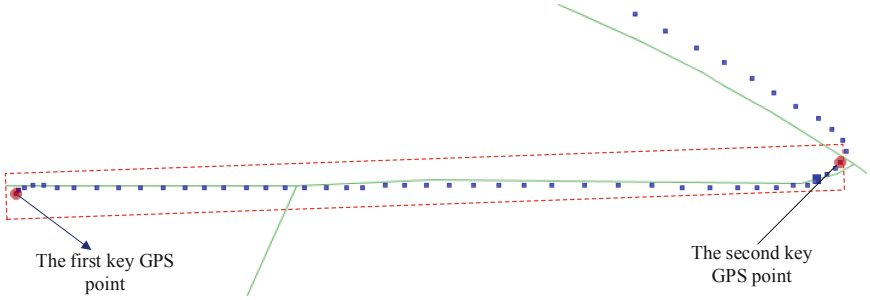

Fig. 3. An example to illustrate key points and the redundant. (Color figure online)

In this paper, we propose a method to compress trajectory by sliding window, which can be used for both online and offline compression. Our approach aims to describe the original trajectory with fewer key points by using the distance threshold which can be changed according to the required compression. We apply an example to describe the main ideas of our method. As Fig. 4 shows, there is a raw GPS trajectory $T : p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_6$.we firstly put p_1, p_2 into sliding window, then as a new point arrives in the sliding window, it uses the new point and the first point to calculates the PED (Perpendicular Euclidean Distance) ($PED_{p_m|(p_i,p_j)}$) by Eq. 1 for all the points in the sliding window. If there is $PED_{p_m|(p_i,p_j)} > PED_{\text{threshold}}$(i < m<j), the p_{j-1} is called key point. In Fig. 4, because $PED_{p_3|(p_1,p_4)} > PED_{\text{threshold}}$, we take p_3 as the key point

and p_3,p_4 as the new starting point of the sliding window. $PED_{p_4|(p_3,p_6)} > PED_{\text{threshold}}$, so p_5 also is a key point. p_6 is the end point of trajectory, we also add p_6 to the key points sequence.

$$PED_{p_m|(p_i,p_j)} = \frac{\left|(y_j - y_i)x_m - (x_j - x_e)y_m + x_j y_i - x_i y_j\right|}{\sqrt{(y_j - y_i)^2 + ((x_j - x_e))^2}} \tag{1}$$

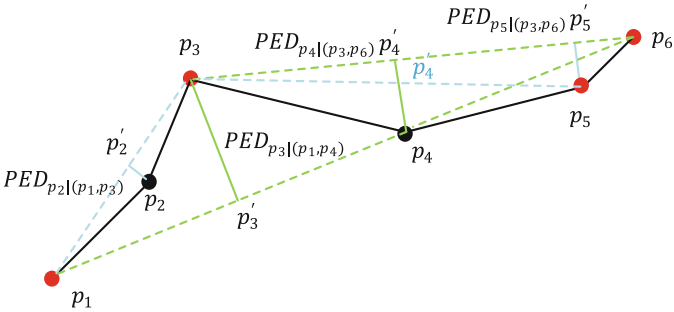

Fig. 4. An example to illustrate proposed method.

3.2 HMM Training

We give a brief description of the matching algorithm based on HMM [9]. For each GPS point, a group of candidate sections is determined first. Each candidate section is represented as a hidden state (vertex) in the markov chain, and has the probability of observation state, which is the feasibility of observing whether the GPS point matches the candidate section. If the GPS point is found to be very close to a section, assign a high probability value to that section. Then, the weight of each pair of adjacent vertices connected in the markov chain is calculated, that is, the state transition probability. Finally, the maximum likelihood path with the highest observed state probability and state transition probability is found on the markov chain. Fig. 5 illustrates a hidden markov chain of map matching.

Observation probability. Although it is simple to calculate the observation probability based on the gaussian distribution model, it has been proved to be effective in the previous work of [9–12] map matching. So the observation probability is as:

$$N(c_i^j) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{\frac{(x_i^j - \mu_1)^2}{2\sigma_1^2}} \tag{2}$$

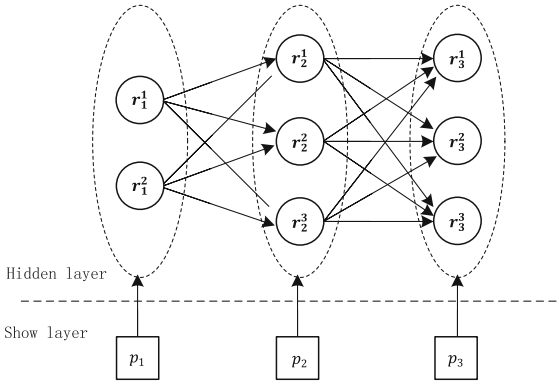

Fig. 5. An example to illustrate a hidden markov chain of map matching

Where c_i^j is a candidate point of sampling point p_i, and X_i^j is the Euclid distance from c_i^j to p_i.

Transition probability. Compared to circuitous paths, the true driving distance tends to be close to the euclidean distance between the adjacent GPS points, especially after trajectory compression. Based on the above insight, We adopt the transition probability method proposed by Lou et al. [10], which further improves the robustness of the algorithm based on HMM [9].

$$V\left(c_{i-1}^s \rightarrow c_i^t\right) = \frac{\text{Euclid}(i-1 \rightarrow i)}{route\left(c_{i-1}^s \rightarrow c_i^t\right)} \tag{3}$$

Where $E(i-1 \rightarrow i)$ is Euclid distance from p_{i-1} to p_i, and $route\left(c_{i-1}^s \rightarrow c_i^t\right)$ represents the driving path distance from c_{i-1}^s to c_i^t.

3.3 Result Matching

The most likely sequence of hidden states in a HMM is commonly found apply a DP (dynamic programming) algorithm known as the Viterbi algorithm, which can quickly find the optimal path in the road network to maximize the product of observation probability and transition probability.

Figure 6 shows an example of finding the optimal path by Viterbi algorithm. We consider candidate c_2^1, the weight of path$(c_1^1 \rightarrow c_2^1)$ is $1.36 = 0.8 + 0.8 \times 0.7$, and the path's $(c_2^1 \rightarrow c_2^1)$ weight is 0.61. Therefore candidate $c_2^{1'}s$ weight is 1.36, and its parent is c_1^1. We repeat the above process for all c_i^j. After completing the calculation we find that c_3^1 has the highest score, and its parent is c_2^1. the Viterbi algorithm finally output the matching path is $c_1^1 \rightarrow c_2^1 \rightarrow c_3^1$.

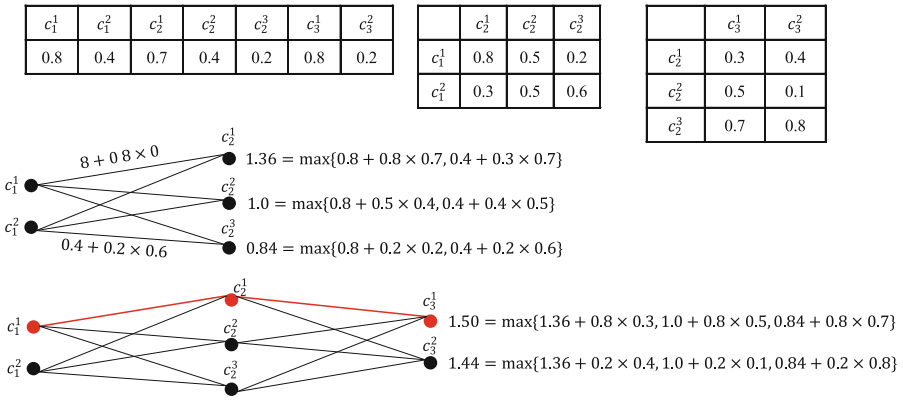

Fig. 6. An example of finding the optimal path by Viterbi algorithm.

4 Experiment

In this section, we use real-world trajectory data to evaluate our FHMM algorithm. We first describe the setting of experiment and then report the experiment results.

4.1 Parameter Selection

In our experiment, we set k = 6 as the maximum number of candidates for each sampling point. and the query radius is r = 100 m. In the trajectory compression stage, in order to ensure that the travel path between two adjacent GPS points after compression is close to a straight line, we set PED threshold as 20 m. For observation probability calculation, we use a normal distribution with $\mu = 0$ and $\sigma = 4.61$ estimated by Eq. 4 [9]. In addition, the algorithms are implemented in python 2.7, on an Intel i5-7200u PC with memory 8 GB on windows10 operating system.

$$\sigma = 1.4826 \operatorname{median}\left(\| p_i - c_i^j \|_{Euclid}\right) \tag{4}$$

4.2 Experiment Datasets

We evaluate the performance of our proposed algorithm using two real-world trajectory datasets, and the details of the dataset are as follows:

- **Dataset 1**: The dataset [9] was collected by Krumm et al. to test HMM matching algorithm and its sampleing rate, number of sampling points et al. are showed in Table 1. Moreover, the dataset provides the road network and ground truth data for comparsion with matching result.
- **Dataset 2**: The dataset [15] was used for map matching in the ACM SIGSPATIAL CUP 2012, and the details are also shown in Table 1. Besides, the dataset also provides the ground true.

Table 1. Description of experimental datasets

Dataset	Trips	Sampling	Size	Time	Length
Dataset 1	1 trip	1 s	7531	~2 h	~80 km
Dataset 2	10 trips	1 s	14436	~4 h	~228 km

4.3 Evaluation Approach

The FHMM is an improved algorithm based on the HMM [9], so we compared the matching accuracy and efficiency of the FHMM with the HMM. The accuracy is the correctness of matching the road, and the efficiency is to compare the running time of the algorithm on the same platform. We use matching precision (MP) to evaluate the accuracy of the algorithm.

$$MP = \frac{|correct\ matched\ road\ segments|}{|road\ segments\ to\ be\ matched|} \times 100\% \tag{5}$$

4.4 Result and Analysis

Figure 7 present the visualized compressing result using the GPS trajectory of dataset 1. In this picture, the blue dots represent raw sampling GPS points, and the red dot represents the GPS sampling points in the compressed trajectory. Figure 7(a), (b) and (c) represent global compression results and local compression results respectively. Obviously, the trajectory obtained by our method of compression can completely describe the original trajectory to the maximum extent. The original trajectory has 7,351 trajectory points. After compression, there are 167 key GPS points left, with a compression rate of 97.3%. A large number of redundant GPS points are removed, which greatly reduce the calculation cost in the subsequent matching process. The dataset 2 has similar compression results as dataset 1.

Tables 2 and 3 shows the matching accuracy and running time of HMM and FHMM algorithms. We compare the performance of the algorithms on the same platform. We can see that the FHMM algorithm is about faster 2 to 3 times than the HMM algorithm, while keeping matching accuracy.

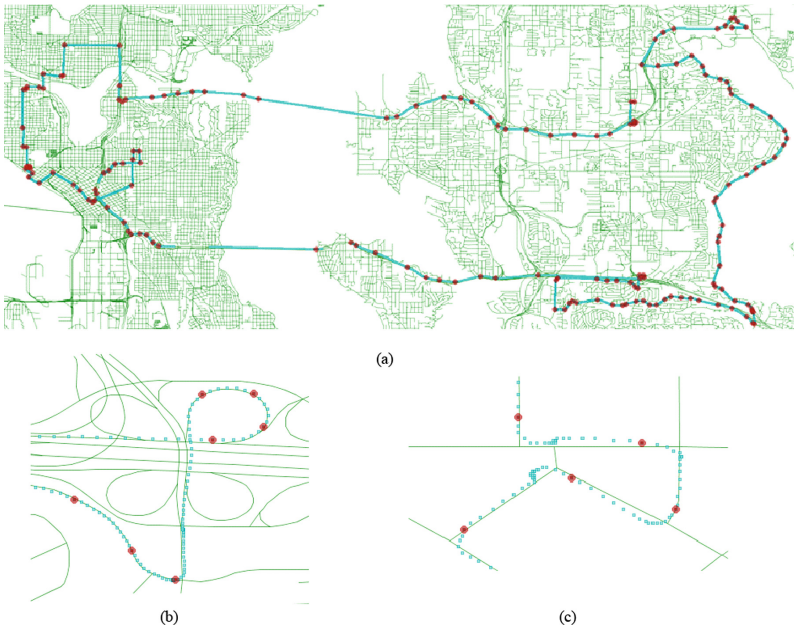

Fig. 7. Result of trajectory compression of Dataset 1. (Color figure online)

Table 2. Comparsion of two algorithms on accruracy

Dataset	Method	
	HMM	FHMM
Dataset 1	99.30%	99.30%
Dataset 2	99.45%	98.61%

Table 3. Comparsion of two algorithms on efficiency

Dataset	Track	HMM		FHMM	
		Size (raw)	Running time/s	Size (compressed)	Running time/s
Dataset 1	Track1	7351	131.6	167	46.1
Dataset 2	Track1	2356	29.4	47	14.7
	Track2	1070	13.6	39	6.6
	Track3	1566	17.4	52	5.8
	Track4	1177	21.5	50	10.0
	Track5	885	7.4	19	3.7
	Track6	1017	9.1	35	4.2
	Track7	2368	26.5	54	7.9
	Track8	1135	13.2	53	5.9
	Track9	1543	21.4	60	9.7
	Track10	1320	9.7	40	4.3

5 Conclusion

In this paper, we present an efficient algorithm for matching nosiy vehicle trajectories onto road network. Experiments show that our algorithm achieves excellent efficiency, while keep reasonable matching accuracy. In fact, our compression method can be applied to the track acquisition stage, which can greatly reduce the difficulty of subsequent matching. In the future, we plan to develop algorithms which provide higher mapping accuracy with moderate computation costs for more noisy datasets.

Acknowledgment. This study was supported by the national natural science foundation of China(61702148). We thank the judges and thank you for your support.

References

1. Joshi, R.R.: A new approach to map matching for in-vehicle navigation systems: the rotational variation metric. In: 2001 Proceedings Intelligent Transportation Systems, pp. 33–38 (2001)
2. Jing Yuan, Y., Zheng, X.X., Sun, G.: T-Drive: enhancing driving directions with taxi drivers' intelligence. IEEE Trans. Knowl. Data Eng. **25**(1), 220–232 (2013)
3. Pang, L.X., Chawla, S., Liu, W., Zheng, Y.: On detection of emerging anomalous traffic patterns using GPS data. Data Knowl. Eng. **87**(9), 357–373 (2013)
4. Quddus, M.A., Ochieng, W.Y., Noland, R.B.: Current map-matching algorithms for transport applications: state-of-the art and future research directions. Transp. Res. Part C: Emerg. Technol. **15**(5), 312–328 (2007)
5. White, C.E., Bernstein, D., Kornhauser, A.L.: Some map matching algorithms for personal navigation assistants. Transp. Res. Part C: Emerg. Technol. **8**(1), 91–108 (2000)
6. Velaga, N.R., Quddus, M.A., Bristow, A.L.: Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. Transp. Res. Part C **17**(6), 672–683 (2009)
7. Obradovic, D., Lenz, H., Schupfner, M.: Fusion of sensor data in siemens car navigation system. IEEE Trans. Veh. Technol. **56**(1), 43–50 (2007)
8. Quddus, M.A., Noland, R.B., Ochieng, W.Y.: A high accuracy fuzzy logic based map matching algorithm for road transport. J. Intell. Transp. Syst. **10**(3), 103–115 (2006)
9. Newson, P., Krumm, J.: Hidden markov map matching through noise and sparseness. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,, pp. 336–343. New York, NY, USA (2009)
10. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate GPS Trajectories. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 352–361. New York, NY, USA (2009)
11. Song, R., Lu, W., Sun, W., Huang, W., Chen, C.: Quick map matching using multi-core CPUs. In: Proceedings of ACM-GIS, pp. 605–608 (2012)
12. Ozdemir, E., Topcu, A.E., Ozdemir, M.K.: A hybrid HMM model for travel path inference with sparse GPS samples. Transportation **45**(1), 233–246 (2018)
13. Tiwari, V.S., Arya, A., Chaturvedi, S.: Framework for horizontal scaling of map matching: using map-reduce. In: 2014 International Conference on Information Technology, Bhubaneswar, pp. 30–34. India (2014)

14. Huang, J., Qiao, S., Yu, H., Qie, J., Liu, C.: Parallel map matching on massive vehicle GPS data using MapReduce. In: 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, Zhangjiajie, pp. 1498–1503. China (2013)
15. Ali, M., et al.: ACM SIGSPATIAL GIS cup 2012. In: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2012, Redondo Beach, pp. 597–600. California. New York: ACM (2012)