Truncating Shortest Path Search for Efficient Map-Matching

Takashi Imamichi
IBM Research – Brazil
Av. Pasteur 138/146, Botafogo,
Rio de Janeiro, 22290-240, Brazil
tima@br.ibm.com

Takayuki Osogami and **Rudy Raymond**
IBM Research – Tokyo
19-21 Nihonbashi Hakozaki-cho,
Chuo-ku, Tokyo, 103-8510, Japan
{osogami,rudyhar}@jp.ibm.com

Abstract

We study the problem of map-matching, or finding the route on a road network from a trace of noisy and sparse observed points, particularly when a huge number of points are given. The algorithms based on Hidden Markov Models (HMMs) are known to achieve high accuracy for noisy and sparse data but suffer from high computational cost. We find that the bottleneck of the HMM-based map-matching is in the shortest path search for calculating transition probabilities. We propose a technique to truncate the shortest path search before finding all the shortest paths in the HMM-based map-matching without losing accuracy. We run the one-to-many shortest path search on the reversed network and terminate the search based on the log likelihood of the Viterbi algorithm. Computational experiments show that the proposed approaches can reduce the computational cost by a factor of at least 5.4.

1 Introduction

The Global Positioning System (GPS) is heavily used in devices such as cell phones and car navigation systems, and a huge amount of GPS traces, or sequences of GPS points, are being recorded every day. For example, OpenStreetMap provides public GPS traces[1], and some city governments such as Dublin[2] and Rio de Janeiro[3] provide GPS traces of buses to public. Moreover, as *Internet of Things* emerges, more devices are to come with GPS sensors and be connected to the Internet. This gives us the opportunities to leverage the knowledge that can be extracted from such a huge amount of GPS traces. For example, one might analyze the traffic conditions such as congestion from the GPS traces.

However, the GPS traces often involve noise and are sparse (interval between consecutive points are long and irregular), and it is nontrivial even to extract such basic information as the route that the GPS device has traversed. This task of determining the route corresponding to a given GPS trace on a road network is known as map-matching and has been studied extensively in the literature. Figure 1 shows examples of GPS traces from OpenStreetMap in Rio de Janeiro, Brazil. Here, we can see that the lines connecting the GPS traces (red line segments) are not always on the road in the map. The left panel in Figure 2 illustrates the discrepancy between a road network (black line segments) and a GPS trace (red points), where blue dashed line segments connect consecutive points in the GPS trace. The right panel in Figure 2 illustrates the sequence of road segments (green line segments) that actually corresponds to the GPS trace in the left. Observe that, connecting road segments closest to the GPS points, we would obtain an unrealistic winding route, which is drastically different from the actual route shown in the right panel.

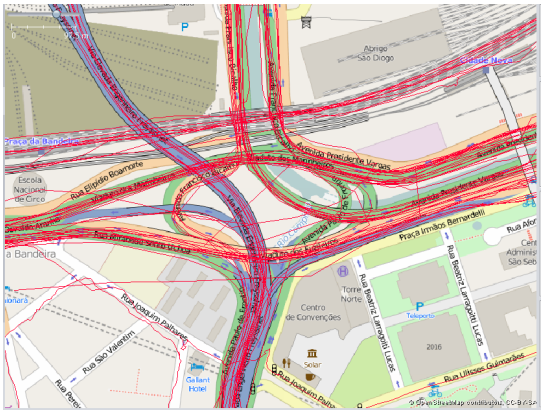


Figure 1: GPS traces in Rio de Janeiro, Brazil (red lines segments); © OpenStreetMap contributors.

Recently, some popular algorithms for map-matching are based on Hidden Markov Models (HMMs) [Newson and Krumm, 2009; Osogami and Raymond, 2013]. Here, a hidden state corresponds to (a point on) a road segment, and an observed variable corresponds to a GPS point. Map-matching with HMMs is known to achieve high accuracy and be robust against the noise and sparsity in the GPS traces. Unfortunately, these algorithms are computationally expensive and not suitable for dealing with a large amount of GPS traces.

We observe, from preliminary experiments, that the bottleneck of existing algorithms for map-matching with HMMs is in the calculation of transition probabilities. Specifically, the

[1] http://www.openstreetmap.org/traces
[2] http://www.dublinked.ie/
[3] http://data.rio/

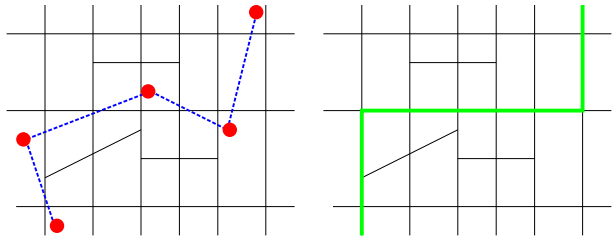


Figure 2: Left: A road network (black) and a GPS trace (red). Right: The corresponding route (green).

transition probability from a hidden state to another is calculated based on the length of the shortest path between the two states on a road network. Notice that the length of a path in the model can take into account various measures such as the number of turns and the expected travel time in addition to the length of traversed roads. If we consider k hidden states for each GPS point, the existing algorithms invoke shortest path search for k^2 times at every step of the Viterbi algorithm. Our goal is to speed up the map-matching with HMMs in order to enable highly accurate map-matching with HMMs for a huge amount of GPS traces. To this end, we have two contributions.

First, we propose a technique to reduce the computational time needed for the shortest path search in HMM-based map-matching. Our technique is to run one-to-many shortest path search on a reversed network, where the direction of every directed arc is reversed, and truncate this shortest path search early without exploring all the shortest paths. We will see that this truncation produces no loss of accuracy.

Our second contributions are in numerical experiments, which show that the proposed approach can drastically reduce computational time without losing accuracy. Specifically, our proposed approach reduces the computational time by a factor of at least 5.4 for 24 GPS traces (1.18 million GPS points) in the *T-drive* data set [Yuan *et al.*, 2010; 2011].

1.1 Related Work

Existing methods of map-matching can be grouped into four categories [Quddus *et al.*, 2007; Zheng, 2015]: *geometrical*, *topological*, *probabilistic*, and advanced methods that incorporate the advantages from multiple categories, for example, an HMM-based approach [Lamb and Thiébaux, 1999; Hummel, 2006; Newson and Krumm, 2009; Osogami and Raymond, 2013]. The geometrical approach focuses only on the shapes of the routes. The topological approach considers the topological aspects such as the connectivity of the road network. The probabilistic approach defines the confidence region around the GPS points and looks for the best possible route. An HMM-based approach takes advantage of a probabilistic model of measurement error, similar to Kalman filters [Kim *et al.*, 2000] and Particle filters [Pink and Hummel, 2008]. It thus has the advantages of probabilistic methods, which have been successfully applied for recovering lanes of roads [Sehestedt *et al.*, 2007] and for localizing autonomous vehicles [Thrun *et al.*, 2001]. Unlike purely probabilistic methods, an HMM-based approach also takes into account the topological information of the road network to achieve high accuracy, similar to topological methods [Brakatsoulas *et al.*, 2005]. HMM-based approaches can also run in real-time [Goh *et al.*, 2012] with guaranteed accuracy using an online decoding algorithm [Wang and Zimmermann, 2014], unlike geometrical methods [White *et al.*, 2000; Yanagisawa, 2010], which often suffer from large computational cost.

Our work builds upon the recent work on an HMM-based method [Osogami and Raymond, 2013], which extends the method by Newson and Krumm [2009] by taking into account drivers' preferences about routes. Although the running time of the HMM-based methods is usually not an issue for a single GPS trace, it becomes unacceptable for many GPS traces. The bottleneck is in the computational cost for calculating the shortest paths between many hidden states.

Despite the long history of research on the shortest path problem, only the standard algorithm of one-to-one shortest path has been used in the existing HMM-based map-matching. We use one-to-many shortest path search and investigate when we can truncate the search particularly in the context of HMM-based map-matching. One might further improve the efficiency of HMM-based map-matching by exploiting sophisticated techniques of shortest path search, including Floyd-Warshall [Floyd, 1962; Warshall, 1962] for multi-source and multi-destination as well as advanced heuristics such as A* [Hart *et al.*, 1968].

Another approach to accelerate shortest path search is preprocessing [Goldberg and Harrelson, 2005; Geisberger *et al.*, 2008; Sturtevant *et al.*, 2009]. Preprocessing produces the auxiliary data to accelerate the shortest path search. Many of preprocessing approaches rely on a particular *fixed* metric of the shortest path, or can only deal with small changes to the metric. Those preprocessing approaches need to recalculate the auxiliary data whenever we change the metric. To avoid recalculation, one can apply an advanced preprocessing approach by Delling *et al.* [2015] that essentially produces auxiliary data supporting multiple metrics. This preprocess partitions the road network into connected cells, generates an *overlay graph* connecting the cells, and runs shortest path search on the overlay graph.

However, we do not adopt the preprocessing approaches in our map-matching because of the following reasons. Firstly, we use the weighted sum of the travel distance and the turn cost, where the weight can vary drastically for each GPS trace, depending on who traveled along that trace. Secondly, consecutive GPS points on a trajectory of map matching that we consider are likely to be located closely because their intervals are usually short (up to several hundreds seconds). In this case, because the source and destination points are often close, the preprocessing by Delling *et al.* [2015] is likely to perform a shortest path search in the same cell, or at most between a pair of neighboring cells. Thus, the shortest path search with preprocessing is essentially the same as a shortest path search on the original road network. Our proposed method exploits techniques for fast shortest path computations in the HMM context similar to that of Felzenszwalb *et al.* [2004]. Our approach can also accelerate HMMs for high-

volume data processing beyond map matching applications, as long as the transition probabilities are determined by some distance metrics on a directed graph whose nodes represent hidden states.

2 Map-matching Algorithm

We first introduce the HMM for map-matching briefly and then explain the details of the novel technique to reduce the computational time for calculating transition probabilities. We suppose that a set of GPS traces is given as input. The points in each GPS trace are ordered according to the time they are observed, but the exact time is not necessarily recorded. Our goal is to output a set of sequences of road segments corresponding to the GPS traces.

2.1 Map-matching with a Hidden Markov Model

An HMM consists of the observable variables, the hidden states, the initial state probability, the emission probability, and the transition probability. For map-matching, an observable variable is the GPS point, i.e., the longitude and latitude. In the following, we describe the other components of the HMM for map-matching, except the initial probability, which is given analogously to the emission probability. The approach described in this section closely follows Osogami and Raymond [2013].

A hidden state corresponds to a road segment of a road network. Here, we define the location of a hidden state by the midpoint of the corresponding road segment where we assume that road segments are straight line segments. We adopt a simpler form of a hidden state than the model of Osogami and Raymond [2013], where an arbitrary point on a road segment can become a hidden state. A GPS point, z_t at time t, is associated with k hidden states, where k is a parameter to be fixed. The k hidden states correspond to the k nearest road segments to z_t.

The emission probability of a GPS point follows from a model of measurement error with the GPS. Let z be the location of the GPS point and s be the location of the hidden state, namely the midpoint of the road segment corresponding to the hidden state, and σ be the standard deviation of the measurement error. Then the emission probability follows the normal distribution whose probability density function is given by

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|s-z\|^2}{2\sigma^2}\right), \tag{1}$$

where $\|\cdot\|$ denotes a distance measure such as the great circle distance or the Euclidean distance.

The transition probability from a hidden state s to another s' is given by the probability density function of an exponential distribution. Let $d^*(s,s')$ be the minimum value of a weighted travel cost from s to s', namely, the *shortest path* from s to s'. Following Osogami and Raymond [2013], we use the weighted sum of the length of traversed road segments and the cost of turns for $d^*(s,s')$. To calculate the travel cost, we construct a network $G=(V,E)$ whose nodes V correspond to road segments, and edges E connect adjacent road segments. Each edge is weighted by the corresponding travel cost, as is defined in the following. Let v and v' be adjacent road segments (v' follows v) and d_v and $d_{v'}$ be the length of the road segments. The cost u of turn is defined as follows.

$$u_{v,v'} = \begin{cases} 0 & \text{if } |\theta_{v,v'}| < \pi/4, \\ 1 & \text{if } \pi/4 \le |\theta_{v,v'}| < 3\pi/4, \\ 2 & \text{if } 3\pi/4 \le |\theta_{v,v'}| < \pi, \\ 10 & \text{if } |\theta_{v,v'}| = \pi, \end{cases}$$

where $\theta_{v,v'}$ is the angle between the road segments v and v'. We add an edge $e \in E$ connecting v and v' on the network G and define the cost c_e of e by

$$c_e \equiv \frac{d_v + d_{v'}}{2} + w_{\text{turn}} u_{v,v'}, \tag{2}$$

where w_{turn} is a parameter. Note that (2) contains costs with multiple metrics. Osogami and Raymond [2013] used Inverse Reinforcement Learning [Ziebart *et al.*, 2008] to estimate the weight w_{turn}. The optimal path that minimizes the weighted cost between a source node and a destination node can be found by applying a shortest path algorithm such as the Dijkstra algorithm [Dijkstra, 1959].

The transition probability from s to s' is given, with a parameter β, by

$$\frac{1}{\beta} \exp\left(-\frac{d^*(s,s') - \|s-s'\|}{\beta}\right), \tag{3}$$

where $\|s-s'\|$ denotes the distance between the midpoints of the road segments that correspond to the hidden states s and s'. Note that the following transition probability is used by Osogami and Raymond [2013]:

$$\frac{1}{\beta} \exp\left(-\frac{|d^*(s,s') - \|s-s'\|\,|}{\beta}\right).$$

We omit the absolute value, because the shortest path distance $d^*(s,s')$ is longer than the distance $\|s-s'\|$ in most cases, and our condition for truncating shortest path search, to be explained later, is suitable for the form (3).

Having all HMM elements, we compute the most likely sequence of hidden states by the Viterbi algorithm [Viterbi, 1967]. Let $z_1,\ldots,z_n$ be the sequence of GPS points and $S_t = \{s_{t,1},\ldots,s_{t,k}\}$ be the set of hidden states at time t, where k is the parameter denoting the number of the hidden states. The Viterbi algorithm recursively computes the maximum log likelihood $\mathrm{LL}^*(s)$ for each hidden state, s:

$$\mathrm{LL}^*(s_{1,i}) = -\frac{\|s_{1,i}-z_t\|^2}{2\sigma^2}, \tag{4}$$

$$\mathrm{LL}^*(s_{t,i}) = \max_{1\le j\le k}\left\{\mathrm{LL}^*(s_{t-1,j}) - \frac{d^*(s_{t-1,j},s_{t,i}) - \|s_{t-1,j}-s_{t,i}\|}{\beta}\right\} - \frac{\|s_{t,i}-z_t\|^2}{2\sigma^2},\ t>1. \tag{5}$$

2.2 Truncating Shortest Path Search

We observe, from preliminary experiments, that the bottleneck of the existing algorithms for map-matching with

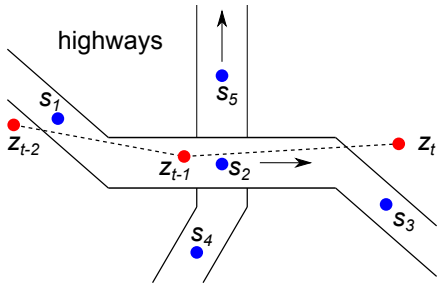


Figure 3: A time consuming case of shortest path search. Red dots are GPS points and blue dots are hidden states. The hidden states are on two disconnected highways.

HMMs is in the calculation of transition probabilities whose values depend on $d^*(s, s')$, the shortest paths between two consecutive hidden states (road segments). Computation of these shortest paths is time consuming for long and irregular interval of hidden states. The existing algorithms invoke shortest path search for k^2 times, one for each pair of hidden states, at every step of the Viterbi algorithm.

Instead of many runs of one-to-one shortest path search, we consolidate k^2 runs of one-to-one shortest path search from all $s \in S_{t-1}$ to all $s' \in S_t$ into k runs of one-to-many shortest path search from all $s' \in S_t$ to all $s \in S_{t-1}$ on a *reversed network*, where the travel cost from a node s' to another s is the travel cost from s to s' on the original network. The duplicated one-to-one traversal is avoided by the one-to-many traversal, and the number of the shortest path searches at every step in Viterbi algorithm is reduced from $O(k^2)$ to $O(k)$. We call it *baseline algorithm* and compare it with our technique. Note the *reversed network* is necessary for our technique.

We propose a technique to truncate the one-to-many shortest path search in map-matching without loss of accuracy of the Viterbi algorithm. Consider a case where there is a pair of a source node and a destination node that are not reachable or very far. The shortest path search takes long time in such a case, because it traverses all the nodes on the road network. Note that the transition probability (3) for a long distant pair of nodes is so small that we can safely truncate its shortest path search. For example, see Figure 3, where there are two highways, (s_1, s_2, s_3) and (s_4, s_5), and three GPS points, (z_{t-2}, z_{t-1}, z_t). The first highway lies above the second one, but they are not connected on the map under consideration. Let s_1 be a hidden state of z_{t-2}, and s_2, s_4, s_5 be the hidden states of z_{t-1}. In the computation of the transition probability, the shortest path search would attempt to compute $d^*(s_1, s_4)$ and $d^*(s_1, s_5)$ by traversing all nodes on the road network, even though the s_4 and s_5 are not (directly) reachable from s_1, and the corresponding transition probabilities are negligible.

In order to truncate such redundant shortest path search, we carefully study the log likelihood (5) of the Viterbi algorithm. At one stage of (5), the Viterbi algorithm has found $\mathrm{LL}^*(s_{t-1,j})$ for all j and seeks to find $\mathrm{LL}^*(s_{t,i})$ for an i. To do so, we compute the shortest path from each of $s_{t-1,j}$ for $1 \le j \le k$ to $s_{t,i}$. This many-to-one shortest paths can be computed by finding one-to-many shortest paths on the reversed network.

We then show how to exactly, i.e., without losing accuracy, truncate the search for one-to-many shortest paths. This exact truncation is made possible by exploiting the property of the Viterbi algorithm on the reversed network. Let

$$G_{t,i}(s) \equiv \beta \mathrm{LL}^*(s) + \|s - s_{t,i}\| \tag{6}$$

for $s \in S_{t-1}$. Plugging (6) into (5), we obtain

$$\mathrm{LL}^*(s_{t,i}) = \frac{1}{\beta} \max_{1 \le j \le k} \{G_{t,i}(s_{t-1,j}) - d^*(s_{t-1,j}, s_{t,i})\} - \frac{\|s_{t,i} - z_t\|^2}{2\sigma^2}. \tag{7}$$

We have already computed $\mathrm{LL}^*(s_{t-1,j})$ for every j in the $t-1$-st iteration of the Viterbi algorithm. In the beginning of the t-th iteration of the Viterbi algorithm, we compute and store the value of $G_{t,i}(s_{t-1,j})$ for all j. In (7), a j cannot become the maximizer if there exists an ℓ such that

$$G_{t,i}(s_{t-1,j}) - d^*(s_{t-1,j}, s_{t,i}) < G_{t,i}(s_{t-1,\ell}) - d^*(s_{t-1,\ell}, s_{t,i}).$$

Thus, we can stop searching for the shortest path from $s_{t,i}$ to $s_{t-1,j}$ on the reversed network if the following inequality holds for an ℓ that has been searched in previous steps:

$$d^*(s_{t-1,j}, s_{t,i}) > G_{t,i}(s_{t-1,j}) - G_{t,i}(s_{t-1,\ell}) + d^*(s_{t-1,\ell}, s_{t,i}). \tag{8}$$

The procedure for the exact truncation to compute (7) is as follows. It invokes shortest path search from $s_{t,i}$ to $S_{t-1} = \{s_{t-1,i} \mid 1 \le i \le k\}$ on the reversed network by the Dijkstra algorithm. Starting from the source node $s_{t,i}$, the algorithm extends the search gradually and determines the shortest path distance. Let W be a subset of all nodes V whose shortest path distances from $s_{t,i}$ have been determined and, let $F \equiv S_{t-1} \cap W$. Every time the algorithm determines the shortest path distance to a new node u, which is not necessarily in S_{t-1}, it updates W and F and checks the following inequality (9) to see if we can truncate the shortest path search:

$$d^*(u, s_{t,i}) > \max\{G_{t,i}(s) \mid s \in S_{t-1} \setminus F\} - \max\{G_{t,i}(s) - d^*(s, s_{t,i}) \mid s \in F\}. \tag{9}$$

The inequality (8) holds if inequality (9) holds because the undetermined nodes $S_{t-1} \setminus F$ are further than u. Note that $\ell = \arg\max\{G_{t,i}(s) - d^*(s, s_{t,i}) \mid s \in F\}$ in (8). If the inequality (9) holds, the procedure truncates the Dijkstra algorithm and computes (7) using ℓ as the maximizer.

3 Experimental Results

We conduct two types of experiments. The first is to evaluate the quality of the results of our map-matching, while the second is to compare the computational time against the baseline algorithm. We implemented our algorithm in Python as a single threaded program and run it on PyPy runtime version 2.6.1 on a PC with 3.3 GHz Intel Xeon E5-2643 CPU.

3.1 Comparison of Accuracy

We apply our map-matching algorithm to a benchmark set that is used and made publicly available by Newson and Krumm [2009]. It contains GPS points for a route of approximately 80 km in Seattle and the ground truth of the traversed route, or simply, the *true route*. We compare the accuracy of the outcomes by our algorithm against that by Osogami and Raymond [2013][4]. We change the sampling interval of GPS trace data from 30 seconds to 600 seconds. Throughout we fix $\sigma = 10.0$ meters and $\beta = 0.1$ meters in (1) and (3) and we use $k = 5$ hidden states at every time step in accordance with Osogami and Raymond [2013].

In order to measure the accuracy of results, we use *the route mismatched fraction* (error) introduced by Newson and Krumm [2009]. Roughly speaking, the error of a route is defined by the ratio of the total length of the erroneously added or subtracted road segments and the length of the correct route. We also calculate the *precision* and *recall* of the resulting route. Let R be a set of road segments of a resulting route and T be a set of road segments of the true route. The precision and recall of the route are defined by $|R \cap T|/|R|, |R \cap T|/|T|$, respectively.

We divide the data set into 12 data sets in accordance with Osogami and Raymond [2013] and use the w_{turn} inferred by Osogami and Raymond [2013], which ranges from 117.1 to 2,257. We plot the average values of the metrics of the error, precision and recall in Figure 4 among the 12 experiments, where the horizontal axis denotes the sampling interval. The lines labeled with 'Ours' and 'OR' represent the average values of the corresponding metrics by our algorithm and Osogami and Raymond [2013], respectively.

Overall, the accuracy of our algorithm is comparable to that of Osogami and Raymond [2013]. This first experiment confirms the soundness of the implementation of our algorithm. Recall that the only essential difference between our algorithm and that of Osogami and Raymond [2013] is in the way transition probabilities (or shortest paths) are calculated, and our truncation of shortest path search does not lose accuracy.

We also apply our map-matching without the truncation of shortest path search and observe that the outputs with and without the truncation are equivalent. It demonstrates that the truncation does not lose accuracy of the Viterbi algorithm as we discussed in Section 2.2 .

3.2 Comparison of Speed

We now apply our algorithm and the baseline algorithm to a big data set to compare their computational time. Recall that the baseline performs one-to-many shortest path search, without truncation, to calculate the transition probabilities.

We use the first 24 GPS traces from the *T-drive* data set [Yuan *et al.*, 2010; 2011], which contains a large amount of GPS trace data in Beijing but not the true routes. In our experiment, we remove the GPS points with the following properties. First, we delete the GPS points with the exact same coordinates as the previous time step. Second, we eliminate the GPS points which are located too far (more than 100 km) from ones at the previous time step. Third, we delete the GPS points whose estimated speed from ones at the previous time step is too fast (more than 200 km/h). The total number of GPS points in the resulting data is approximately 1.18 million. We extract the road network data of Beijing from OpenStreetMap, which contains approximately 301,000 nodes and 461,000 edges.

[4]We also performed the comparison against the results of Newson and Krumm [2009] and obtained the results similar to those reported in Osogami and Raymond [2013].

Table 1: Comparison of the total computation time for 24 GPS traces in the T-drive data set. Note that the baseline algorithm exceeded the time limit 10,000 seconds for 4 GPS traces with $k = 10$.

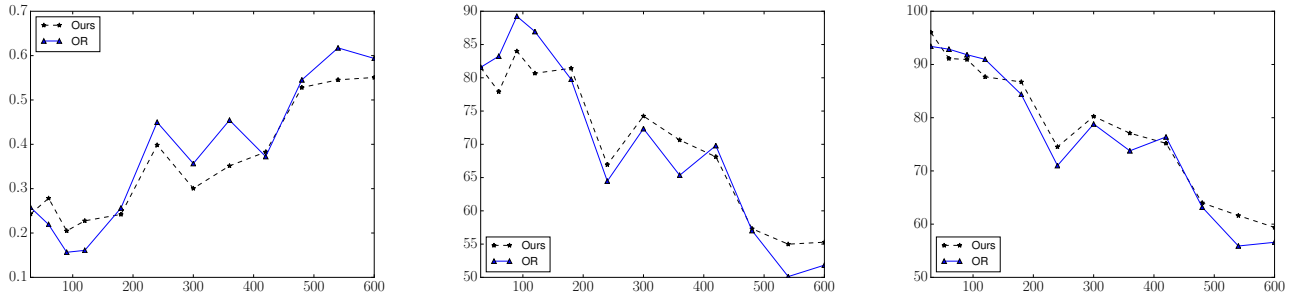
Algorithm	Time (seconds)	
	$k = 5$	$k = 10$
baseline	50915.2	131316.2
ours	9356.6	20281.5
baseline + LRU	9744.2	18833.5
ours + LRU	2480.8	3727.4

We apply our algorithm and the baseline algorithm to the data set and compare the computation time. Note that we do not compare the precision as in the previous subsection, because the data set does not include the ground truth data. We fix $\sigma = 10.0$ meters, $\beta = 0.1$ meters and $w_{\text{turn}} = 1000.0$ in (1), (3) and (2), respectively.

Table 1 shows the total computational time of the baseline algorithm and our algorithm with $k = 5$ and 10. We set the time limit for each GPS trace by 10,000 seconds. There are 4 GPS traces (among 24 traces) in which the baseline with $k = 10$ exceeded the time limit. Our algorithm finished the computation within the time limit for all 24 GPS traces in the two cases. We observe that our algorithm is 5.4 times faster than the baseline with $k = 5$ and *at least* 6.4 times faster with $k = 10$. See Figure 5 for the computation time for each GPS trace. Our algorithm runs faster than the baseline by a factor of 1.8 to 20.5 (or even faster), depending on the trace and k.

The advantages of the proposed algorithm over the baseline can increase with a larger value of k. While $k = 5$ appears to be sufficient for the data set of Newson and Krumm [2009], a larger k can improve the accuracy for other data sets. Our algorithm indeed finds different routes with a larger k for the T-drive data set, although the changes in the accuracy cannot be evaluated without true routes.

We additionally test an approach to reuse the intermediate data structure of the shortest path search such as the shortest path tree and the priority queue of nodes. Because we deal with geospatial temporal GPS data sets, we expect the locality property to hold: namely, recently accessed hidden states and its neighbors are likely to be accessed soon. Therefore, storing the intermediate data structure in a Least Recently Used (LRU) cache, which is fast and easy to implement, we can avoid duplicate traversal in the shortest path search. "baseline + LRU" and "ours + LRU" denote the computation time of the baseline algorithm with an LRU cache and our algorithm with an LRU cache, respectively. We observe that the LRU cache accelerated our algorithm and the baseline algorithm by a factor of 3.7 and 5.2, respectively, with $k = 5$. Our algorithm with the cache runs 3.9 times faster than the



(a) Route mismatched fraction (b) Precision (%) (c) Recall (%)

Figure 4: Comparison of the accuracy, where x-axis indicates the sampling intervals of the GPS data and y-axis indicates the route mismatched fraction, precision, and recall.

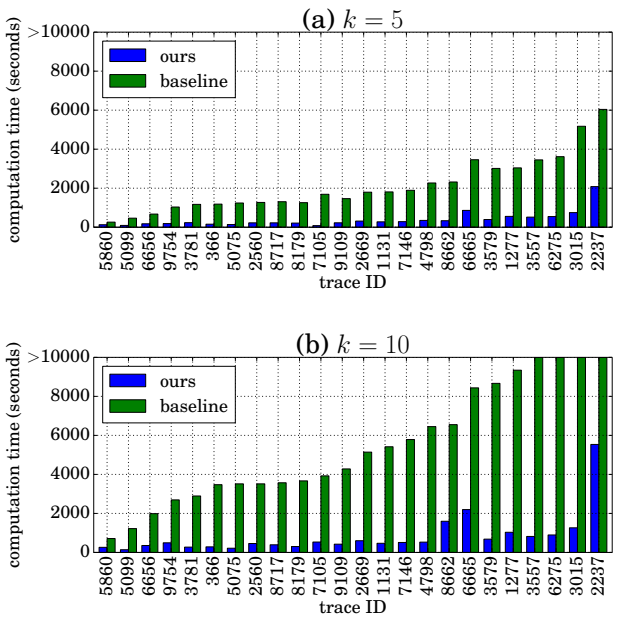


Figure 5: Computation time of the T-drive data set for each GPS trace: (a) $k = 5$, (b) $k = 10$. If a bar reaches the top, the map-matching exceeded the time limit 10,000 seconds.

baseline algorithm with the cache with $k = 5$.

Finally, we compare the outputs of "baseline + LRU" and "ours" with $k = 5$ and 10 in Figure 5 and observe that they are equivalent for all traces. The difference between the algorithms is the LRU and the truncation of shortest path search, and it is clear that the LRU does not affect outputs of map-matching. Thus, the observation confirms that the truncation does not lose accuracy of the Viterbi algorithm. Note that we use "baseline + LRU" instead of "baseline" because "baseline" exceeded the time limit 10,000 seconds for trace ID 2237, 3105, 3557, and 6275 with $k = 10$.

4 Conclusion

We propose a new technique to speed up HMM-based map-matching algorithms. Our technique computes the transition probabilities in the HMM by applying one-to-many shortest path search on the reversed network and truncate the shortest path search early without exploring all the shortest paths by taking into account the log likelihood of the Viterbi algorithm. It was shown effective to significantly reduce the computational time without loss of accuracy. We observe that the proposed method runs 5.4 times faster than the baseline algorithm through the computational experiments with $k = 5$ and even faster with larger k.

We used the proposed method in the framework of Osogami and Raymond [2013], but the proposed method can also be applied to a larger set of HMM-based map-matching algorithms. For example, Osogami and Raymond [2013] define the cost of the path only from the length of the traversed road segments and the number of turns, but one could also take into account other distance measures such as the travel time, type of turns (as some drivers might prefer less number of left turns than right ones), distance to major landmarks, and so on. This is important in modeling the real-world driving, especially in the context of *imitation learning*, where the utility function of the drivers to be recovered often depends on many features or distance measures. In future work, we plan to evaluate the proposed model with such rich features against large data sets that come with the ground truth data, such as, bus trajectories that are usually fixed.

Acknowledgments

A part of this research was supported by CREST, JST.

References

[Brakatsoulas *et al.*, 2005] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. of 31st Int. Conf. Very Large Data Bases*, pages 853–864, 2005.

[Delling *et al.*, 2015] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable route planning in road networks. *Transportation Science*, to appear, 2015.

[Dijkstra, 1959] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[Felzenszwalb *et al.*, 2004] P. F. Felzenszwalb, D. P. Huttenlocher, and J. M. Kleinberg. Fast algorithms for large-state-space HMMs with applications to Web usage analysis. In *Advances in Neural Information Processing Systems 16*, pages 409–416. MIT Press, 2004.

[Floyd, 1962] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.

[Geisberger *et al.*, 2008] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Experimental Algorithms*, volume 5038 of *LNCS*, chapter 24, pages 319–333. Springer, 2008.

[Goh *et al.*, 2012] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet. Online map-matching based on hidden Markov model for real-time traffic sensing applications. In *Proc. of 15th Int. IEEE Conf. Intelligent Transportation Systems*, pages 776–781, 2012.

[Goldberg and Harrelson, 2005] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA-05, pages 156–165, 2005.

[Hart *et al.*, 1968] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Hummel, 2006] B. Hummel. Map matching for vehicle guidance. In *Dynamic and Mobile GIS: Investigating Changes in Space and Time*, pages 175–186. CRC Press, 2006.

[Kim *et al.*, 2000] W. Kim, G.-I. Jee, and J. Lee. Efficient use of digital road map in various positioning for ITS. In *Proc. of 2010 IEEE/ION Position Location and Navigation Symposium*, pages 170–176, 2000.

[Lamb and Thiébaux, 1999] P. Lamb and S. Thiébaux. Avoiding explicit map-matching in vehicle location. In *Proc. of 6th ITS World Congress on Intelligent Transport Systems*, 1999.

[Newson and Krumm, 2009] P. Newson and J. Krumm. Hidden Markov map matching through noise and sparseness. In *Proc. of 17th ACM SIGSPATIAL Int. Conf. Adv. in Geographic Information Systems*, pages 336–343, 2009.

[Osogami and Raymond, 2013] T. Osogami and R. Raymond. Map matching with inverse reinforcement learning. In *Proc. of 23rd Int. Joint Conf. Artificial Intelligence*, IJCAI-13, pages 2547–2553, 2013.

[Pink and Hummel, 2008] O. Pink and B. Hummel. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In *Proc. of 11th Int. IEEE Conf. Intelligent Transportation Systems*, pages 862–867, 2008.

[Quddus *et al.*, 2007] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15:312–328, 2007.

[Sehestedt *et al.*, 2007] S. Sehestedt, S. Kodagoda, A. Alempijevic, and G. Dissanayake. Efficient lane detection and tracking in urban environments. In *Proc. of 3rd European Conf. Mobile Robots*, 2007.

[Sturtevant *et al.*, 2009] N. R. Sturtevant, A. Felner, M. Barrer, J. Schaeffer, and N. Burch. Memory-based heuristics for explicit state spaces. In *Proc. of 21st Int. Joint Conf. Artificial Intelligence*, IJCAI-09, pages 609–614, 2009.

[Thrun *et al.*, 2001] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artif. Intell.*, 128(1-2):99–141, 2001.

[Viterbi, 1967] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Information Theory*, 13(2):260–269, 1967.

[Wang and Zimmermann, 2014] G. Wang and R. Zimmermann. Eddy: An error-bounded delay-bounded real-time map matching algorithm using HMM and online Viterbi decoder. In *Proc. of 22nd ACM SIGSPATIAL Int. Conf. on Adv. in Geographic Information Systems*, pages 33–42, 2014.

[Warshall, 1962] S. Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962.

[White *et al.*, 2000] C. E. White, D. Bernstein, and A. L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, 8(1-6):91–108, 2000.

[Yanagisawa, 2010] H. Yanagisawa. An offline map matching via integer programming. In *Proc. of 20st Int. Conf. Pattern Recognition*, pages 4206–4209, 2010.

[Yuan *et al.*, 2010] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: Driving directions based on taxi trajectories. In *Proc. of 18th SIGSPATIAL Int. Conf. Adv. in Geographic Information Systems*, pages 99–108, 2010.

[Yuan *et al.*, 2011] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proc. of 17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, KDD '11, pages 316–324, 2011.

[Zheng, 2015] Y. Zheng. Trajectory data mining: An overview. *ACM Trans. Intelligent Systems and Technology*, 6(3):Article No. 29, 2015.

[Ziebart *et al.*, 2008] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence*, AAAI-08, pages 1433–1438, 2008.