Heuristic optimizations for high-speed low-latency online map matching with probabilistic sequence models

George R. Jagadeesh and Thambipillai Srikanthan

Abstract— **The computation speed and output latency of map matching are important considerations when processing location data, especially smartphone-generated noisy and sparse data, from a large number of users for real-time transportation applications. In this paper, we examine the factors affecting the efficiency of online map matching algorithms that are based on probabilistic sequence models such as Hidden Markov Models (HMM) and present several heuristic optimizations to improve their speed and latency. As shortest path computations account for most of the running time of probabilistic map matching algorithms, we propose a method for reducing the total number of such computations by pruning unlikely states in the probabilistic sequence model. Furthermore, we speed up the one-to-many shortest path computations by limiting the search space to an elliptical area that encompasses all the targeted destinations. We present a technique for reducing the latency of the Viterbi algorithm used to find the most likely state sequence in a HMM or a similar model. This technique enables the early output of partial state sequences based on an estimate of the probability of a state being part of the eventual most likely sequence. Experiments using real-world location data show that the heuristic optimizations significantly reduce the running time and output latency with negligible loss of accuracy.**

I. INTRODUCTION

Map matching is the problem of associating a sequence of inaccurate location measurements, usually corresponding to the actual locations of a moving vehicle, with a sequence of road segments. It is the first step in utilizing location data sourced from travelers for applications such as road speed estimation and automatic traffic incident detection. In recent years, there has been an abundance of smartphone-generated location data that are typically characterized by high levels of measurement noise and temporal sparsity. Their use in real-time transportation applications is conditional upon the availability of accurate and efficient map matching algorithms. The focus of this paper is on improving the efficiency of online map matching in terms of processing speed and output latency, which are important considerations in real-time applications handling large-scale streaming location data.

Map matching is widely viewed as a sequence labelling problem that is best solved by probabilistic sequence models such as Hidden Markov Models (HMM) [1], [2], [3], and Conditional Random Fields (CRF) [4]. This paper conforms to the HMM framework, but the proposed techniques are equally applicable to CRF-based map matching. In the HMM-based approach, for each location measurement, a number of candidate road segments are represented as states (nodes) in a HMM trellis graph such as the simple one shown in Figure 1, where the edges represent the transitions between states. The most likely sequence of states is found using the Viterbi algorithm [5], a well-known dynamic programming algorithm. While this approach provides the best results in terms of accuracy, it typically involves a large amount of computation and high latency due to the reasons described below.

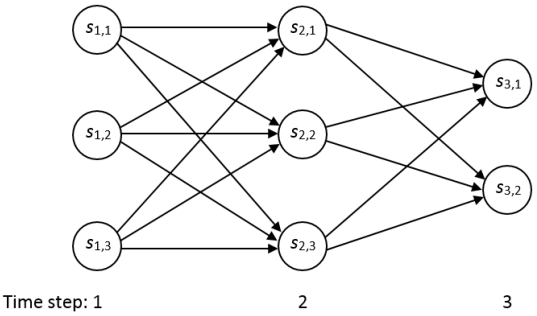

Figure 1. A simple example of a HMM trellis graph. In the map matching context, the nodes correspond to road segments and the edges represent transitions between them.

The Viterbi algorithm requires the probabilities of all the transitions in the trellis graph to be calculated. The transition probability between two states (i.e. road segments) depends on the shortest path between them. The shortest path calculations are the most computationally expensive part of the map matching process [6], [7]. Assuming that there are N states at time step $t-1$ and M states at time step t, a total of NM shortest path computations need to be performed to calculate the probabilities of all possible transitions between the two time steps. The number of states per time step becomes large when the location measurements are provided by noisy positioning technologies such as Wi-Fi positioning and cellular network positioning, which are preferred in smartphones due to their energy efficiency. In such a situation, an immense number of shortest path computations are needed, resulting in unacceptably long running times. In its basic form, the Viterbi algorithm outputs the most likely state sequence from the first time step to the last time step in the trellis graph after all the observations (i.e. location measurements) are received. This results in a high latency, which is not suitable for applications with timeliness constraints.

A number of techniques have been used in the literature to improve the computation speed of map-matching algorithms that are based on HMM or similar models. Some researchers limit the number of shortest path computations by not evaluating the probabilities of transitions emanating from some states at each time step. The simplest way of doing this is to select and retain a fixed number of states at each time step for further computation on the basis of a score that indicates the probability of the most likely sequence ending at each state [6], [8]. However, the minimum number of states that should

be retained varies considerably at each time step depending on the road network topology and the location measurement error. Hence, retaining a fixed number of states at all the time steps may either result in poor accuracy or may incur more computation than necessary. An optimization that reduces the number of state transition probability calculations without compromising optimality was recently proposed by Koller et al. [9]. Their optimization involves replacing the Viterbi algorithm with the bidirectional Dijkstra algorithm that finds the optimal state sequence in the trellis graph without exploring all the states in it. However, it is applicable only for offline map matching and not for the online case.

In some works [7], [8], the processing speed is significantly improved by replacing the large number of one-to-one shortest path computations with a smaller number of one-to-many shortest path computations. Furthermore, the shortest path search space is limited by the maximum possible distance that can be travelled from the source within the time elapsed between successive location samples. However, such a search bound, which accounts for vehicles travelling at the maximum possible speed throughout, causes the search space to be much larger than actually required in practice. Eisner et al [10] adopted a hierarchical routing approach, which involves preprocessing the road network, for speeding up the shortest path computations needed for map matching.

Several published map matching algorithms include techniques for reducing the latency of the Viterbi algorithm for online operation. A simple latency-reduction method is to subdivide the location sequence into several chunks using a fixed-sized sliding window and to apply the Viterbi algorithm to each chunk separately. However, in order to achieve good accuracy, larger windows are needed, resulting in higher output latencies [6]. Another approach known as lagged smoothing [4] involves processing a fixed-number, say k, of subsequent observations before outputting the state corresponding to a given observation. Lagged smoothing is performed using the forward-backward algorithm, which computes the probability of a state at time step t, given the observations up to time step $t + k$. However, a fixed lag may not suit all situations, as in some cases a higher than usual number of subsequent observations may be needed to resolve the uncertainty at a time step. While the abovementioned latency-reduction techniques do not guarantee an optimal solution, it is possible to reduce the latency of the Viterbi algorithm without compromising its accuracy. As explained in Section II, the Viterbi algorithm can be made to output partial state sequences each time a convergence point is detected in the trellis graph such that the partial state sequence up to that point cannot be altered by future observations [3]. However, even with this improvement, the Viterbi algorithm typically incurs a latency of several time steps, especially for noisy and sparse location data.

In this paper, we propose and evaluate three heuristic optimizations to reduce the computation time and latency of map matching algorithms that perform Viterbi inference on a probabilistic sequence model. First, we propose a simple yet effective method for reducing the number of states considered at each time step. Instead of retaining a fixed number of states at each time step for further computation, we heuristically retain a variable number of states by using the probability of the most likely state as a reference value. Second, we present a geometric pruning technique that substantially reduces the search space of the one-to-many shortest path computations by confining the search to an elliptical region that encompasses the source and all the destinations. Our third heuristic optimization seeks to reduce the latency of the Viterbi inference process by outputting partial state sequences even before a convergence point indicating the irreversibility of the partial state sequence is detected. Such early generation of the partial sequence up to a time step is made when the probability of a state at that time step being part of the eventual most likely state sequence exceeds a threshold. The proposed heuristic optimizations have been evaluated with smartphone-generated location sequence data and compared with a baseline algorithm that represents the state-of-the-art for online map matching of noisy and sparse data.

The next section describes the online map matching algorithm that is used as the baseline. Section III presents the proposed heuristic optimizations. Section IV presents the experimental setup and the evaluation results. Section V summarizes and concludes the paper.

II. The Baseline Algorithm

Let o_t denote a location observation, consisting of the measured latitude and longitude of a moving vehicle, at time step t. Let $G = (V, E)$ be a road network where V is a set of nodes (road intersections) and E is a set of edges (road segments). Each road segment in the road network has a number of attributes such as length and free-flow travel time. The shape of the road segment is represented as a piecewise linear curve using a sequence of latitude-longitude pairs.

Formally, the objective of map matching is to match a sequence of T location observations $o_1, \dots, o_T$ to a sequence of T road segments $e_1, \dots, e_T$. However, these road segments may not form a connected sequence, especially if the time interval between the location observations is large. For the output to be of practical use, the gaps between each pair of road segments in the sequence should be interpolated with the road segments that make up the shortest path between them. We consider this interpolated sequence of road segments, which form the complete reconstructed path of the moving vehicle, as the output of map matching.

The baseline map matching algorithm used as a reference in this paper is based on the HMM approach and incorporates optimality-preserving techniques from the literature for reducing the runtime and latency. A brief description of the algorithm is provided below.

For each location observation, the closest point on each road segment that lies within a certain error range is considered as a state in the HMM. We denote the k^{th} state at time step t as $s_{t,k}$. The observation probability $P(o_t|s_{t,k})$ represents the probability of a vehicle on the road segment corresponding to state $s_{t,k}$ generating the observation o_t. The observation probability is defined as a Gaussian function of the distance between the observed location and the state.

$$P(o_t|s_{t,k}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{d_g(o_t, s_{t,k})^2}{2\sigma^2}} \quad (1)$$

where $d_g(o_t, s_{t,k})$ is the great-circle distance between o_t and $s_{t,k}$ and σ is the estimated standard deviation of the location measurement error.

We calculate the transition probability $P(s_{t,k}|s_{t-1,j})$ between states $s_{t-1,j}$ and $s_{t,k}$ based on the model proposed in our previous work on map matching noisy and sparse location data [11]. The probability of transitioning from one state to another depends on the shortest path between them. In our work, the shortest path is computed based on the travel time metric. The shortest paths from a state to all the states in the next time step is computed through a single shortest path tree computation using Dijkstra's algorithm [12]. As proposed in [7], the search space of Dijkstra's algorithm is bounded by the distance that can be travelled at the maximum possible speed of 50 m/s (180 km/h) within the given time interval. It is worth clarifying that although the shortest path is computed based on the travel time metric, the distance from the source is also stored in order to limit the search space.

For the shortest path between states $s_{t-1,j}$ and $s_{t,k}$, we quantify its circuitousness and temporal implausibility using the functions $y(s_{t-1,j}, s_{t,k})$ and $z(s_{t-1,j}, s_{t,k})$, respectively. These functions are defined as follows:

$$y(s_{t-1,j}, s_{t,k}) = \frac{d_d(s_{t-1,j}, s_{t,k}) - d_g(s_{t-1,j}, s_{t,k})}{\Delta T} \quad (2)$$

$$z(s_{t-1,j}, s_{t,k}) = \frac{max\left(\left(f(s_{t-1,j}, s_{t,k}) - \Delta T\right), 0\right)}{\Delta T} \quad (3)$$

In the above equations, $d_d(s_{t-1,j}, s_{t,k})$ is the driving distance, $d_g(s_{t-1,j}, s_{t,k})$ is the great-circle distance and $f(s_{t-1,j}, s_{t,k})$ is the free-flow travel time, respectively, between states $s_{t-1,j}$ and $s_{t,k}$. ΔT is the elapsed time, in minutes, between time steps $t-1$ and t. The measure of temporal implausibility in (3) indicates how unlikely it is for a vehicle to drive through the path within the time interval ΔT. Based on analysis of data from real drives, which suggests that circuitous and temporally implausible paths are less likely to be taken by vehicles, the transition probability is defined as

$$P(s_{t,k}|s_{t-1,j}) = \lambda_y e^{-\lambda_y y(s_{t-1,j}, s_{t,k})}\, \lambda_z e^{-\lambda_z z(s_{t-1,j}, s_{t,k})} \quad (4)$$

where λ_y and λ_z are parameters to be estimated.

In a HMM, the joint probability between a state sequence $\boldsymbol{s}'_{1:T} = \{s'_1, \dots, s'_T\}$ and an observation sequence $\boldsymbol{o}_{1:T} = \{o_1, \dots, o_T\}$ is given by

$$P(\boldsymbol{s}'_{1:T}, \boldsymbol{o}_{1:T}) = P(o_1|s'_1) \prod_{t=2}^{T} P(o_t|s'_t) P(s'_t|s'_{t-1}) \quad (5)$$

Among the many possible state sequences in the HMM trellis graph, the most-likely one with the maximum joint probability is efficiently found using the Viterbi algorithm. For each state in the trellis graph, the Viterbi algorithm stores a back pointer pointing to the state in the previous time step that is part of the most-likely sequence ending at the former state. Starting with the final time step, the most-likely sequence is retrieved in the reverse order by following the chain of back pointers. However, waiting until the final time step before generating the output results in a high latency.

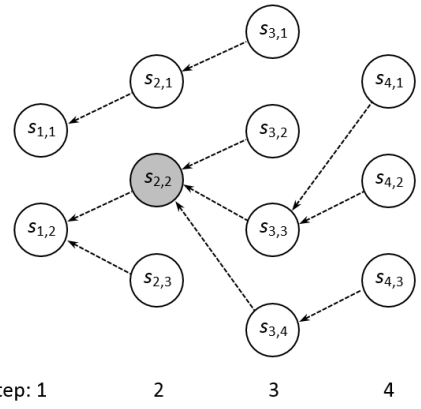

Figure 2. An example of the convergence of back pointer chains in the Viterbi algorithm. The state $s_{2,2}$ in this figure is a convergence state.

To reduce the latency of the Viterbi algorithm without any loss of accuracy, the baseline algorithm uses an optimization based on the concept of reachability between states through back pointers [13], which was applied for map matching in [3]. This optimization involves checking if following the chain of back pointers from each state at the current time step leads to the same state at any past time step. Such a state, where the chains of back pointers converge, is known as a convergence state. For example, in Figure 2, tracing back along the back pointers from any of the states in time step 4 leads to the convergence state $s_{2,2}$ in time step 2. As all the states at time step 4 point to the state $s_{2,2}$ as their ancestor in the most likely sequence, the sequence up to $s_{2,2}$ cannot be altered by events occurring after time step 4. It therefore follows that the partial sequence up to the convergence state can be output without compromising optimality. This optimization enables the baseline map matching algorithm to generate a partial output each time a convergence state is detected and thus helps to keep the output latency low.

III. Heuristic Optimizations

To further reduce the computation time and output latency of online map matching, we propose the following heuristic optimizations.

A. Pruning of Low-Probability States in the HMM

In the HMM, the number of states per time step can be high, especially when the error range of the location measurements is large. However, many of the states typically have very low probability of being the true state given the history of location observations up to that time step. Such states can be pruned from the HMM Trellis graph without incurring any significant loss of accuracy. By not evaluating the transitions from such states, the total number of shortest-path tree computations can be reduced. Besides reducing the computation time, pruning of low-probability states also helps to reduce the output latency by discarding unlikely state

sequences and thereby causing the convergence of the back pointer chains in the Viterbi algorithm to occur earlier.

The joint probability of the state $s_{t,k}$ at time step t and the observation sequence $\boldsymbol{o}_{1:t} = \{o_1, \ldots, o_t\}$ is denoted as the forward probability $\alpha_t(s_{t,k})$, which can be recursively calculated as follows.

$$\alpha_t(s_{t,k}) = P(o_t|s_{t,k}) \sum_{j=1}^{N_{t-1}} P(s_{t,k}|s_{t-1,j})\, \alpha_{t-1}(s_{t-1,j}) \quad (6)$$

where N_{t-1} is the number of states at time step $t-1$.The forward probability for the first time step is initialized as $\alpha_1(s_{1,i}) = P(o_1|s_{1,i})$, for all i.

We aim to prune the low-probability states and exclude them from further computation. The states to be pruned at each time step are identified by comparing their forward probabilities with that of the state with the highest forward probability at that step. A state $s_{t,k}$ is pruned if the following inequality is satisfied: $\frac{max_i(\alpha_t(s_{t,i}))}{\alpha_t(s_{t,k})} > \theta$, where θ is a parameter to be empirically set. When pruning is performed based on the above criterion, the number of states retained for further computation at each time step is obviously not fixed, but varies depending on the situation.

B. Geometric Pruning of the Shortest Path Search Space

The baseline map matching algorithm computes a shortest path tree, bounded by a distance based on the maximum possible travel speed, starting from each candidate road segment at each time step in order to determine the respective shortest paths to all the candidate road segments at the next time step. In most practical scenarios, the shortest path search space can be reduced substantially by adopting a geometric pruning approach. For computing the shortest path between a single source and a single destination, confining the search to an elliptical region is known to be the best choice, even when the link costs (e.g. travel time) vary dynamically [14]. In this work, we extend the elliptical pruning technique to the single-source multiple-destination case.

In the map matching context, all the destinations targeted by a one-to-many shortest path tree computation lie within an error radius of an observed location measurement. We limit the shortest path search to nodes (i.e. end points of road segments) that lie within an ellipse, whose focal points lie at the source of the shortest path tree computation and the observed location measurement, which is the center of a circular error region containing the destinations. The ellipse is constructed such that it encompasses all the destinations, including those that lie closer to the circumference of the error region, as well as the shortest paths to them from the source.

Let the state $s_{t,k}$, which represents an on-road location, be the source from which a shortest path tree computation needs to be carried out. Let o_{t+1} be the location observation corresponding to the next time step. Let *R* be the radius of the error region with o_{t+1} as its center. Based on the above, the focal points of the ellipse are $s_{t,k}$ and o_{t+1}. During the shortest path search based on Dijkstra's algorithm, a node v is explored only if the sum of the great-circle distances from it to the two focal points of the ellipse is equal to or less than a certain value $Q_{t,k}$ (i.e. $d_g(v, s_{t,k}) + d_g(v, o_{t+1}) \leq Q_{t,k}$). We define $Q_{t,k}$ as

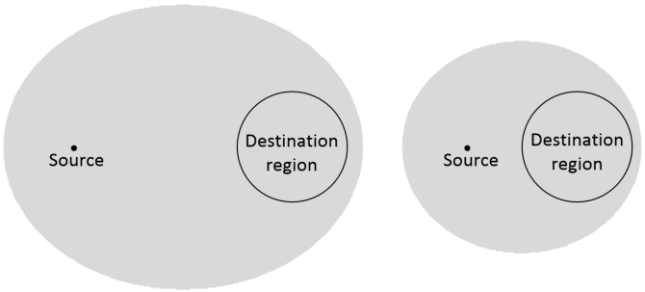

Figure 3. Geometric pruning of the shortest path tree search space. The eccentricity of the ellipse varies with the distance between the source and the center of the circular region containing the destinations. In the ellipses shown above, the parameter γ is set to 1.1.

$$Q_{t,k} = \gamma.\left(R + d_g(s_{t,k}, o_{t+1})\right) + R \quad (7)$$

where γ is a parameter to be empirically set. The ellipse thus formed always encompasses all the targeted destinations irrespective of the distance between the focal points. It needs to be noted that the eccentricity of the ellipse used for pruning the search space is not fixed, but depends on the distance between the source and the center of the error region containing the destinations as illustrated in Figure 3.

C. Heuristic Early Generation of Partial State Sequences

The baseline algorithm outputs an optimal partial state sequence when it detects a condition in which the chains of back pointers from all the states at the current time step converge at a state corresponding to a past time step. However, in many situations, the partial state sequences can be generated with reasonable confidence even before the convergence condition is detected, thus reducing the output latency. To enable this, we use an adaptation of the approach proposed by Narasimhan et al [13], who applied it to the problem of spotting contact information in large documents.

We calculate the normalized forward probabilities associated with each state at time step t as

$$\bar{\alpha}_t(s_{t,k}) = \frac{\alpha_t(s_{t,k})}{\sum_{i=1}^{N_t} \alpha_t(s_{t,i})} \quad (8)$$

where $\alpha_t(s_{t,k})$ is the forward probability of the state $s_{t,k}$ given by (6) and N_t is the number of states at time step t. We define a reachability function $\bar{\delta}$ between the state $s_{t,k}$ at time step t and the state $s_{t-m,j}$ at a past time step $t-m, m \geq 1$, to indicate if the latter state is reachable from the former through a chain of back pointers.

$$\bar{\delta}(s_{t,k}, s_{t-m,j}) = \begin{cases} 1 & \text{if } s_{t-m,j} \text{ is reachable from } s_{t,k} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Based on the location observations $\boldsymbol{o}_{1:t}$ received up to time step t, the probability that state $s_{t-m,j}$ at time step $t-m$ is on the eventual most likely state sequence is estimated using the score $h(s_{t-m,j}, \boldsymbol{o}_{1:t})$ given below.

$$h(s_{t-m,j}, \boldsymbol{o}_{1:t}) = \sum_{k=1}^{N_t} \bar{\alpha}_t(s_{t,k})\, \bar{\delta}(s_{t,k}, s_{t-m,j}) \quad (10)$$

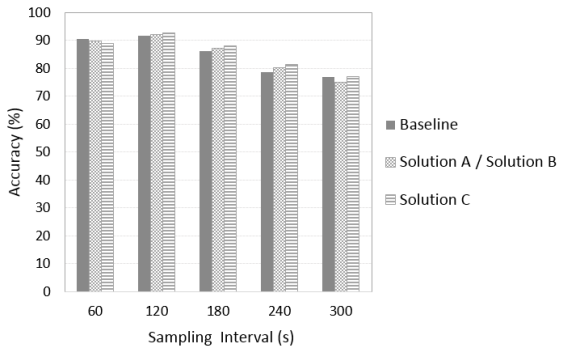

Figure 4. Evaluation of the map matching accuracy. The proposed heuristic optimizations do not incur any significant loss of accuracy compared to the baseline algorithm.

The score $h(s_{t-m,j}, \boldsymbol{o}_{1:t})$ of state $s_{t-m,j}$ will be equal to 1 (and the scores of all the other states at time step $t-m$ will be equal to 0) if $s_{t-m,j}$ is a convergence state (i.e. all the states at time step t point back to $s_{t-m,j}$). However, we heuristically generate partial state sequences before a convergence state is found. This is based on the idea that if the probability of a state being on the eventual most likely sequence is much higher than that of all the other states for any time step, then the partial state sequence up to that time step can be generated with little chance of it being invalidated by future events. In our proposed solution for online map matching, we generate the partial output up to the state $s_{t-m,j}$ (corresponding to a past time step $t-m$) if its score $h(s_{t-m,j}, \boldsymbol{o}_{1:t})$ is greater than an empirically determined threshold τ.

IV. Evaluation

A. Experimental Setup

For our experiments, we use noisy location sequence data collected on Singapore roads through 20 taxi rides covering a distance of 421 km. An Android smartphone-based application that determines the location based on cellular network positioning was used for recording the data at the rate of 1 sample per second. The corresponding ground truth locations and the travelled paths were determined using relatively-accurate Global Positioning System (GPS) data that was separately recorded during the taxi rides.

The parameters of the baseline map matching algorithm are set based on the analysis of ground truth data. The standard deviation σ of the location measurement error is estimated to be 382 m. The error range of the location measurements R is set to be 4 times the standard deviation. The parameters λ_y and λ_z in the transition probability model are set to 0.01 and 5, respectively. In our experiments, the parameters in the heuristic optimization methods are set conservatively in order to minimize the loss of accuracy. The parameter θ used for pruning the low-probability states in the HMM is set to 100. The parameter γ that determines the size of the elliptical search area is set to 1.1. The threshold τ used for the early generation of partial state sequences is set to 0.9.

We evaluate the effectiveness of the proposed heuristic optimizations by comparing the following three solutions with the HMM-based baseline algorithm described in Section II.

1. Solution A, which is the baseline algorithm plus the pruning of low-probability states in the HMM.

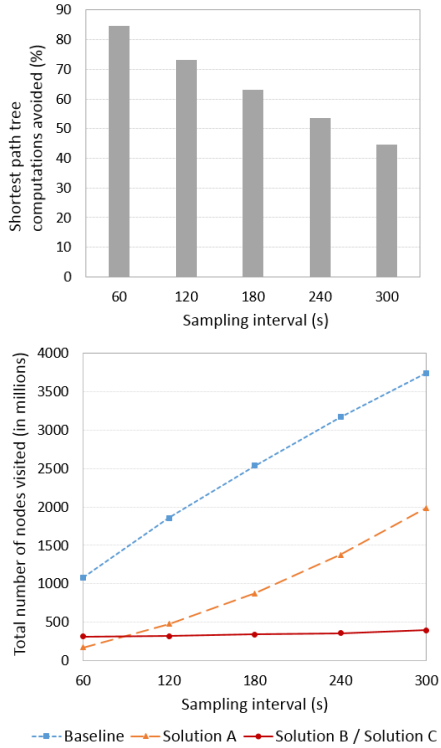

Figure 5. Evaluation of the shortest path computation effort, which determines the speed of map matching. The pruning of the low-probability states in the HMM substantially reduces the total number of shortest path tree computations needed (top). The geometric pruning technique (present in Solutions B and C) further reduces the total number of nodes visited during the shortest path search (bottom).

2. Solution B, which is Solution A plus the geometric pruning of the shortest path search space.
3. Solution C, which is Solution B plus the heuristic early generation of partial state sequences.

For evaluating the map matching accuracy, we use the F-measure, which is the harmonic mean of precision (the proportion of the map matched output path that is correct) and recall (the proportion of the true path that is map matched). As the computation times of the map matching algorithms are mainly determined by the amount of shortest path computation involved, we evaluate the latter. The running times of shortest path algorithms, such as Dijkstra's algorithm that we use, are a function of the number of nodes visited during the shortest path search. Therefore, we express the computational effort in terms of the total number of nodes visited instead of the absolute running times, which are implementation-dependent. The output latency of the map matching algorithms is expressed in terms of the number of time steps.

B. Results

A comparison of the map matching accuracy for sampling intervals ranging from 1 to 5 minutes is shown in Figure 4. The solutions that include the proposed optimizations (with the parameters set as stated above) achieve almost similar accuracies compared to the baseline algorithm. In fact, in some cases, the heuristic optimizations produce a slight increase in accuracy. This is because, due to the differences in the output

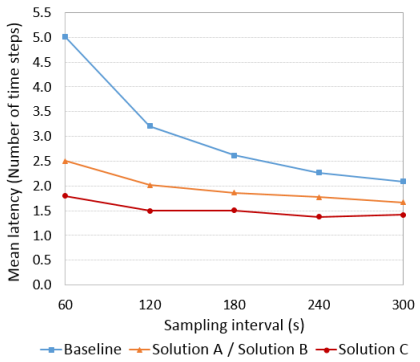

Figure 6. The effect of the heuristic optimizations on the mean output latency.

latencies, the last portion of each travelled path that is left unmatched varies for each solution. This causes the lengths of the map matched paths to be different, thus resulting in small variations in the accuracy. Solution B has the same accuracy as Solution A in all the cases, which shows that the geometric pruning technique used in the former does not cause any loss of optimality.

Figure 5 shows the effect of the heuristic optimizations on the amount of shortest path computations. The pruning of low-probability states in the HMM (present in Solutions A, B and C) eliminates up to 85% of the shortest path tree computations. The total number of nodes visited during the shortest path searches increases with the sampling interval for the baseline algorithm and Solution A, both of which limit the search space based on the maximum possible distance that can be travelled within the sampling interval. On the other hand, Solutions B and C, which include the geometric pruning method, visit a smaller and almost constant number of nodes. It is worth clarifying that while longer sampling intervals generally involve the computation of longer shortest paths and hence require a larger number of nodes to be visited, they also result in a fewer number of location samples and thus fewer shortest path tree computations. It is evident from the results that the geometric pruning method, which limits the search space to an elliptical area just enough to cover the source and the destinations, avoids wasteful exploration of irrelevant nodes.

As shown in Figure 6, the heuristic optimizations achieve significantly lower mean latencies for all the sampling intervals compared to the baseline algorithm. Solution C, which includes the heuristic early generation of partial state sequences, can be made to achieve even lower mean latencies, at the cost of some accuracy, by setting the threshold τ lower. Figure 7 shows the overall accuracy loss and mean latency values observed for different values of τ with the results for all the sampling intervals combined.

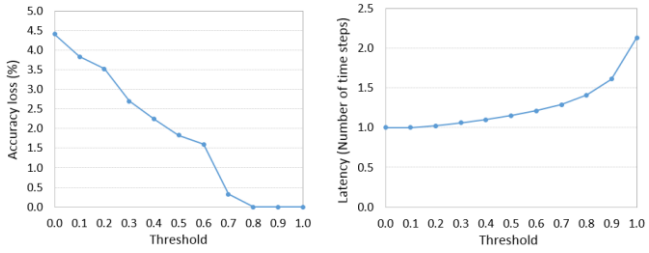

Figure 7. The overall loss of accuracy (left) and mean latency (right) for a range of values of the threshold τ used for the heuristic early generation of partial state sequences.

V. Conclusions

We have presented several heuristic techniques for improving the efficiency of probabilistic map matching algorithms that involve applying the Viterbi algorithm on a HMM or similar trellis graph. Tests conducted with real-world cellular network positioning data show that a substantial reduction in the computation time can be achieved by heuristically pruning unlikely states in the probabilistic model and irrelevant nodes in the shortest path computations. The results also suggest that the output latency of online map matching algorithms can be maintained at low levels by generating partial outputs, based on a probabilistic estimate, before they are finalized by the Viterbi algorithm. The findings are especially important for map matching noisy and sparse location data, which typically involve evaluating a large number of highly improbable hypotheses. As part of future work, we intend to explore the possibility of further improving the efficiency of the proposed optimizations by dynamically varying the parameters in a situation-aware manner.

References

[1] P. Newson and J. Krumm, "Hidden Markov map matching through noise and sparseness," in *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 336-343, 2009.

[2] A. Thiagarajan, L. S. Ravindranath, K. LaCurts, S. Toledo, J. Eriksson, S. Madden, and H. Balakrishnan, "VTrack: Accurate, energy-aware traffic delay estimation using mobile phones," in *7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 85-98, 2009.

[3] 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 *15th IEEE Intelligent Transportation Systems Conference*, pp. 776-781, 2012.

[4] T. Hunter, P. Abbeel, and A. M. Bayen, "The path inference filter: model-based low-latency map matching of probe vehicle data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15(2), pp. 507-529, April 2014.

[5] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13(2), pp. 260-269, 1967.

[6] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," in *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 352–361, 2009.

[7] H. Wei, Y. Wang, G. Forman, Y. Zhu, and H. Guan, "Fast Viterbi map matching with tunable weight functions," in *20th International Conference on Advances in Geographic Information Systems*, pp. 613-616, ACM, 2012.

[8] S. Fang and R. Zimmermann, "EnAcq: energy-efficient GPS trajectory data acquisition based on improved map matching," in *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 221-230, 2011.

[9] H. Koller, P. Widhalm, M. Dragaschnig, and A. Graser, "Fast hidden Markov model map-matching for sparse and noisy trajectories," in *18th IEEE Intelligent Transportation Systems Conference*, Las Palmas, Spain, pp. 2557-2561, 2015.

[10] J. Eisner, S. Funke, A. Herbst, A. Spillner, and S. Storandt, "Algorithms for matching and predicting trajectories," in *13th Workshop on Algorithm Engineering and Experiments*, pp. 84–95, SIAM, 2011.

[11] G. R. Jagadeesh and T. Srikanthan, "Probabilistic map matching of sparse and noisy smartphone location data," in *18th IEEE Intelligent Transportation Systems Conference*, pp. 812-817, 2015.

[12] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[13] M. Narasimhan, P. Viola, and M. Shilman, "Online decoding of Markov models under latency constraints," in *23rd International Conference on Machine learning*, pp. 657-664, ACM, 2006.

[14] L. Han, H. Wang, W. Mackey, "Finding shortest paths under time-bandwidth constraints by using elliptical minimal search area," *Transportation Research Record: Journal of the Transportation Research Board*, 1977, pp. 225-233, 2006.