EAR-MM: An Efficient Adaptive and Robust Algorithm for Streaming Map Matching

Yangyang Sun, Tianyu Huang, Zijian Zhang, Ruiyuan Li, Chao Chen, and Yu Zheng *Fellow, IEEE*

***Abstract*—Streaming map matching is essential for real-time location-based services, such as navigation systems and traffic monitoring, but maintaining accuracy under dynamic conditions with noisy GPS data presents significant challenges. We propose EAR-MM, an efficient, adaptive and robust streaming map matching algorithm designed to address these issues. To mitigate the impact of GPS noise, EAR-MM incorporates a candidate reusing mechanism that ensures stable candidate selection, enhancing robustness against noisy data. Additionally, a rollback mechanism is employed to leverage new data to correct previous matching errors, incrementally improving accuracy. EAR-MM also features an adaptive parameter tuning component, which dynamically adjusts parameters based on changing environmental conditions, ensuring consistent performance across diverse contexts. To meet real-time processing requirements, EAR-MM accelerates shortest-path calculations using a bidirectional Dijkstra algorithm with step-level caching, balancing memory consumption and computational efficiency effectively. Experimental evaluations on two real-world trajectory datasets from Chengdu and Wuxi demonstrate that EAR-MM significantly outperforms existing methods in terms of accuracy and efficiency, and adapts effectively to diverse geographic conditions. The source code is publicly available[1].**

***Index Terms*—map matching, trajectory data mining, streaming processing.**

I. Introduction

THE rapid growth of location-based services has led to an enormous volume of GPS (Global Positioning System) trajectory data [1], [2], which often contains inherent inaccuracies and noise, necessitating effective methods for mapping GPS points to corresponding road segments—a process known as map matching [3]. Streaming map matching, which involves continuously updating the matched path as new GPS points are received, is particularly important for real-time applications such as navigation systems [4], [5], traffic monitoring [6], and intelligent transportation systems [7]–[9].

This paper is supported by the National Natural Science Foundation of China (62572086, 62322601, 62172066), the Fundamental Research Funds for the Central Universities (2024IAIS-QN017), Major Basic Research Project of Shandong Provincial Natural Science Foundation (ZR2024ZD03) and the Independent Research Project of State Key Laboratory of Mechanical Transmission for Advanced Equipment (SKLMT-ZZKT-2024R07).

Yangyang Sun, Tianyu Huang, Zijian Zhang and Ruiyuan Li are with College of Computer Science, Chongqing University, China and Start Lab, Chongqing University, China (e-mail: sunyangyang@cqu.edu.cn; t1anyu@stu.cqu.edu.cn; zijian@stu.cqu.edu.cn; ruiyuan.li@cqu.edu.cn). Chao Chen is with the State Key Laboratory of Mechanical Transmission for Advanced Equipments, Chongqing University, China (e-mail: cschaochen@cqu.edu.cn). Yu Zheng is with JD Intelligent Cities Research, China, JD iCity, JD Technology, China, Xidian University, China and Beijing Key Laboratory of Traffic Data Mining and Embodied Intelligence, China (e-mail:msyuzheng@outlook.com).

Yangyang Sun and Tianyu Huang contributed equally to this work.

Ruiyuan Li is the corresponding author of this paper.

[1]https://github.com/Spatio-Temporal-Lab/StreamingTrajectoryMapMatching

Streaming map matching presents unique challenges due to the complexities of real-time processing and the dynamic nature of GPS data. Firstly, unlike batched methods that utilize complete trajectories to improve accuracy, streaming map matching must incrementally process GPS points without accessing to future data, which imposes strict constraints on computational efficiency and algorithmic robustness. Secondly, GPS data are often noisy, prone to signal loss, and irregularly sampled [10], which complicates the accurate alignment of a trajectory with the underlying road network. Thirdly, streaming environments are characterized by fluctuating road conditions, including sudden changes in road density and frequent path deviations, necessitating adaptive strategies to maintain consistent accuracy and efficiency. Therefore, an effective streaming map matching approach must be both adaptive and highly efficient. Balancing these competing demands—accuracy, efficiency, and adaptability—in real time constitutes the core challenges of streaming map matching.

Map matching methodologies can be broadly classified into batched and streaming approaches. Batched methods operate on complete trajectory datasets, leveraging global information to achieve high accuracy [11]–[14]. However, they are unsuitable for streaming scenarios due to their reliance on complete trajectories and inability to adapt in real time. In contrast, streaming methods can be divided into real-time output and delayed output approaches. real-time output methods [15]–[17], provide immediate results as GPS points are received, enabling real-time processing. However, these methods often struggle with robustness and accuracy, especially in the presence of noisy GPS signals or irregular sampling intervals. On the other hand, delayed output methods [18]–[20] improve accuracy by processing data within a defined time window before generating the matched path, but this introduces latency, which is undesirable for real-time applications. Moreover, most streaming methods use fixed parameters or rely on models trained specifically for certain urban environments, which makes them unable to adapt dynamically to changing real-time conditions, thereby affecting accuracy.

To address these challenges, we propose EAR-MM, an efficient, adaptive and robust algorithm specifically designed for the complexities of streaming map matching, delivering real-time output that seamlessly integrates with dynamic data sources. EAR-MM achieves these goals through a combination of innovative techniques that balance real-time processing demands, ensure robustness against noisy GPS data, and adapt to diverse environmental conditions. The primary contributions of this work are summarized as follows:

(1) To enhance robustness, EAR-MM combines Candidate Searching and Candidate Reusing to improve candidate selec-

tion and mitigate noisy GPS data.

(2) To ensure adaptiveness, we propose an automatic parameter adjustment mechanism with gradient descent to optimize observation and transition probabilities based on changing conditions.

(3) To improve efficiency, we propose a Streaming Inference method that ensures real-time map matching. It accelerates pathfinding using bidirectional Dijkstra and reduces redundant computations with step-level caching.

(4) Experimental results show that EAR-MM outperforms existing methods in robustness, adaptiveness and efficiency, delivering faster and more accurate real-time map matching.

The remainder of this paper is organized as follows: Section II reviews related works, and Section III provides necessary preliminaries. Section IV outlines the framework of EAR-MM, while Section V and Section VI present the detailed algorithmic design and experimental results, respectively. Section VII concludes with a summary of findings.

II. Related Works

Various techniques are employed in the development of map matching algorithms [3], which are generally categorized into batched and streaming methods. The streaming methods, in turn, can be further subdivided into delayed output methods and real-time output methods.

A. Batched Methods

Batched methods utilize the complete GPS trajectory to find a globally optimal matched path. Li et al. [12] propose a distributed HMM (Hidden Markov Model)-based map matching framework with a many-to-many shortest path query algorithm, which leverages road network hierarchical contraction to achieve fast map matching on large-scale trajectory data. Several studies [20]–[23] have introduced improvements to HMM-based methods. To address low-quality trajectory data, Jiang et al. [13] present a deep learning-based approach that enhances trajectory representation by incorporating historical patterns. Wang et al. [10] propose a probabilistic interpolation method that dynamically generates probability values for interpolated points. Feng et al. [24] introduce DeepMM, which combines deep learning and data augmentation to improve matching performance under data sparsity and noise. Building on this, Liu et al. [25] develop GraphMM, a graph-based method that exploits correlations between trajectories and road networks. Since batched methods require global data, they are not suitable for streaming scenarios.

B. Streaming Methods

Streaming methods incrementally infer matched paths based on streaming GPS observations. Goh et al. [18] implement an HMM-based approach using an online Viterbi algorithm with a variable sliding window, allowing for partial matching results when convergence points are detected. However, in the presence of noisy data or a large sampling interval, the matching accuracy of HMM decreases significantly. In addition to HMM-based approach, Chen et al. [19] present a high-quality and computationally efficient algorithm that segments trajectories and makes full use of vehicle heading information. The aforementioned methods typically require the collection of a specified number of points prior to matching, which improves accuracy but results in delayed output.

To achieve real-time output while ensuring accuracy, Taguchi et al. [26] introduce a method that employs path prediction and Bayesian filtering to identify the most probable future candidate paths. Qi et al. [17] present a novel framework that leverages Spark Streaming and GPU-based heterogeneous computing to process large-scale, real-time trajectory data. Lv et al. [27] propose a self-adjusting online map matching method based on the Hidden Markov Model, addressing the need for real-time, low-latency trajectory matching. This approach defines calculation formulas for observation and transition probabilities in HMM and incorporates three adjustment strategies to handle noise points, dense points and offset points, thereby enhancing algorithm stability and accuracy. Similarly, He et al. [15] introduce an enhanced weight-based real-time map matching algorithm tailored for complex and dense urban road networks. This algorithm dynamically adjusts weights for key criteria—such as distance, heading difference, direction difference and segment connectivity—and introduces a novel confidence level calculation to improve reliability. In streaming scenarios, where road conditions change dynamically, these methods often fall short due to their reliance on fixed parameters, limiting their adaptability to changing conditions. To address this, Hu et al. [16] propose an adaptive algorithm that incorporates a collaborative evaluation model for filtering out low-quality measurements and a self-tuning mechanism for dynamically adjusting feature weights. While this enables streaming map matching, these methods still encounter challenges such as accuracy degradation and computational delays under varying conditions.

III. Preliminaries

Definition 1. ***GPS Point.*** *A GPS point represents the geographical location of an object at a specific time, defined by its latitude, longitude, and timestamp:* $p_i = (lat_i, lon_i, t_i)$*, where* (lat_i, lon_i) *are the coordinates at time* t_i*.*

Definition 2. ***Road Segment.*** *A road segment is a directed edge connecting two intersections in a road network, defined as* $e_i = (v_j, v_k)$*, where* v_j *and* v_k *are intersections. Each segment has an associated distance* $d(e_i)$*.*

Definition 3. ***Directed Road Network.*** *A directed road network consists of intersections (nodes) and road segments (edges), where edges may be one-way or two-way. Formally,* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, where* $\mathcal{V}$ *is the set of intersections and* $\mathcal{E}$ *the set of road segments. Two-way edges have two road segments in opposite directions.*

Definition 4. ***Path.*** *A path in a directed road network is a sequence of connected road segments from a starting node to a destination node. Formally, a path* $\mathcal{P}$ *from node* v_s *to node* v_d *is* $\mathcal{P} = \{e_1, e_2, \ldots, e_k\}$*, with each* $e_i = (v_j, v_k)$ *connecting nodes* v_j *and* v_k*. The total length of the path is* $Length(\mathcal{P}) = \sum_{i=1}^{k} d(e_i)$*.*

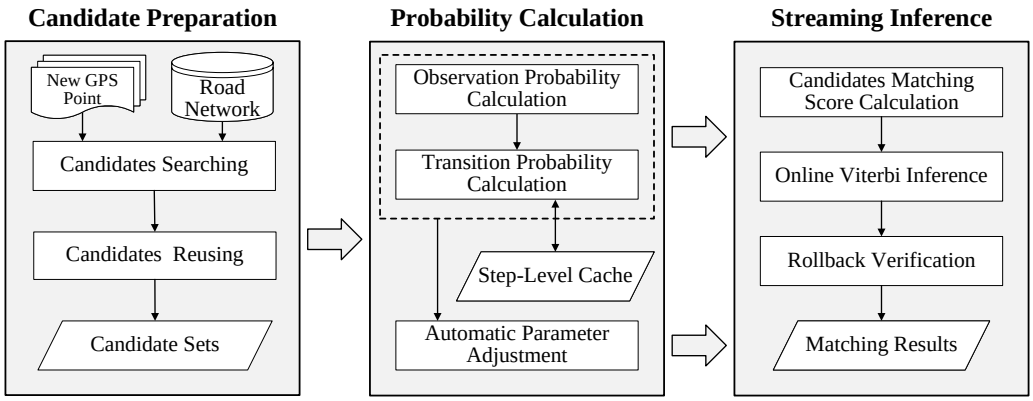


Fig. 1. Framework of EAR-MM.

Definition 5. ***Trajectory Stream.*** *A trajectory stream is a sequence of GPS points:* $\mathcal{T} = \{p_1, p_2, \ldots, p_n, \ldots\}$. *Real-world trajectories may have noise and irregular sampling intervals.*

Problem Definition. Streaming Map Matching is a process of incrementally updating a path $\mathcal{P}$ on a road network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as new GPS points from a trajectory stream $\mathcal{T} = \{p_1, p_2, \ldots\}$ are received.

Fig. 2 illustrates the process of streaming map matching, green hollow points represent the original GPS locations, which typically deviate from the road due to noise. Red solid points mark the GPS locations matched to the road network. The blue solid line represents the current matched path, and the orange dashed line shows the previous matched path before the GPS point, p_3, was updated. In this example, the matched path **changes** between points p_2 and p_3, meaning that the original path is no longer suitable for the current GPS location, and a new matched path is computed. At the same time, the path is extended between points p_3 and p_5, reflecting that as new GPS points are received, the path is further **extended** along the road network.

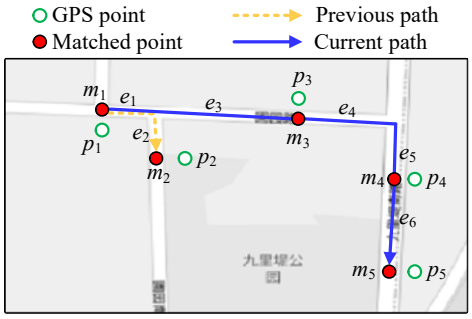


Fig. 2. Example of Streaming Map Matching.

IV. Framework

The framework of EAR-MM is illustrated in Fig. 1, comprising three primary modules: *Candidate Preparation*, *Probability Calculation*, and *Streaming Inference*.

Candidate Preparation. This phase involves identifying potential candidates based on the new GPS point and road network data. To reduce GPS noise, candidates identified previously are reused, enhancing the stability of the candidate sets. These refined sets are then passed to the next module.

Probability Calculation. This phase calculates key probabilities to evaluate candidate transitions. The observation probability assesses how accurately each candidate matches the GPS point, while the transition probability measures the likelihood of moving between candidates. To optimize efficiency, a step-level cache limits stored data, balancing memory usage with computational speed. An automatic parameter adjustment module is also proposed to improve matching accuracy based on environmental conditions and data quality.

Streaming Inference. This phase focuses on real-time path inference. The system calculates a matching score for candidates and uses online Viterbi inference to determine the most probable path. A rollback verification mechanism allows the system to revisit previous inferences with new GPS points, improving the overall precision of matching results.

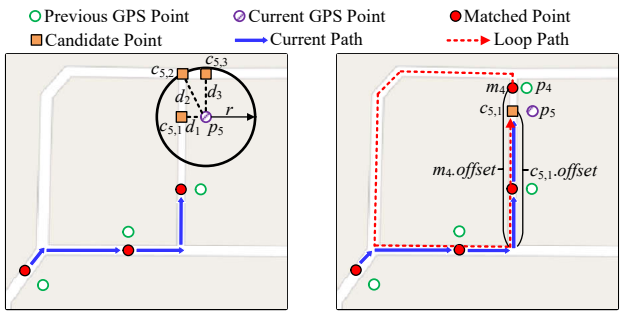
V. Design of EAR-MM

A. Candidate Preparation

This section introduces the *Candidate Preparation* module of the EAR-MM algorithm. The main goal of this module is to generate candidate locations on the road network for each incoming GPS point in real-time. We propose a candidate selection strategy designed to enhance the robustness of the candidate set. Accurate candidate generation is critical, as it directly influences subsequent calculations and, ultimately, the overall accuracy of the map matching process.

Candidate Searching. The first step in *Candidate Preparation* is candidate searching, where potential candidate points on the road network are identified for a given GPS point. Due to GPS inaccuracies, such as signal interference or multipath effects. It is essential to project the GPS point onto nearby road segments for identifying candidate locations.

As illustrated in Fig. 3(a), a search radius r (defaulting to 50m) is established around the GPS point p_5. For each road segment within this radius, the closest position to p_5 is identified as a candidate point, i.e., $c_{5,1}, c_{5,2}, c_{5,3}$. To expedite the search process, an R-tree structure is employed to efficiently locate these candidate points within the radius. Only these closest points from each segment are retained for further



(a) Candidate Searching (b) Candidate Reusing

Fig. 3. Illustration of Candidate Searching and Candidate Reusing.

Algorithm 1: Candidate Reusing Algorithm

Input: Previous matched point m_{pre}, current GPS point p_{cur}, current candidate set C_{cur}, speed threshold v
Output: Updated candidate set C_{cur} for p_{cur}

1 **foreach** *candidate point* $c_i \in C_{cur}$ **do**
2 **if** $c_i.eId = m_{pre}.eId$ ***and*** $c_i.offset < m_{pre}.offset$ **then**
3 **if** $d_{path}(c_i, m_{pre})/(c_i.t - m_{pre}.t) > v$ **then**
4 Add m_{pre} to C_{cur}; **break**;
5 **return** C_{cur}

evaluation, ensuring that only the most relevant candidate points are selected for subsequent processing.

Candidate Reusing. This step ensures robust candidate selection, particularly in noisy environments where GPS data may suggest unrealistic movements, such as backward loops.

As depicted in Fig. 3(b), the current GPS point p_5 suggests a backward movement compared to the previously matched point m_4. In specific, a new candidate point $c_{5,1}$ may indicate a loop path, which is unrealistic in this context. Instead of adopting such a candidate, the algorithm incorporates the previously matched point m_4 into the candidate set, thereby ensuring path continuity and avoiding mapping errors. By incorporating m_4 and comparing it with $c_{5,1}$ through probabilistic calculations, the algorithm effectively prevents the formation of a backward loop, thereby generating a more reliable matched path that accurately reflects the true trajectory.

Algorithm 1 starts by iterating over each candidate point c_i in the current candidate set C_{cur}. For each candidate, it checks whether the candidate lies on the same road segment as the previously matched point m_{pre} by comparing their road IDs (Line 2). If the candidate is on the same segment but its offset is smaller than the offset of the previously matched point, as shown in the Fig. 3(b), indicating a potential backward movement, the algorithm calculates the speed between the candidate c_i and the current GPS point p_{cur} (Line 3). This speed is determined by dividing the path distance traveled by the time elapsed since the previous point. If the speed exceeds a threshold v (three times the maximum speed limit of the road segment), the backward movement is implausible. In this case, the algorithm adds the previously matched candidate point m_{pre} to the current candidate set and calculates probabilities together with the current unrealistic candidate, thereby enhancing the robustness of the algorithm (Lines 4-5).

B. Probability Calculation

This section first reviews the traditional method of HMM probability calculation and discusses its limitations, and then proposes two optimizations to address these limitations.

Traditional HMM Probability Calculation. The classical HMM probability calculation method has demonstrated good accuracy, particularly with low sampling interval data. Therefore, this paper is based on the traditional approach, which consists of two main components: observation probability calculation and transition probability calculation.

(1) Observation Probability Calculation. This step quantifies the likelihood that the observed position data aligns with potential positions on the map. The observation probability $P(c_{t,i}|p_t)$, where p_t represents the observed location at time t and $c_{t,i}$ is the candidate map-matched position, measures the likelihood that the observed data corresponds to a particular map position. This calculation is essential for mitigating noisy GPS signals and improving the accuracy of matching observed data to the actual map path, as has been proven effective in prior research [18]. The observation probability is defined as:

$$P(c_{t,i}|p_t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{d_E(p_t, c_{t,i})}{2\sigma^2}\right) \tag{1}$$

where σ represents the standard deviation of the observation noise, with a default value of 5 based on the previous research [28]. $d_E(p_t, c_{t_i})$ represents the Euclidean distance between p_t and $c_{t,i}$.

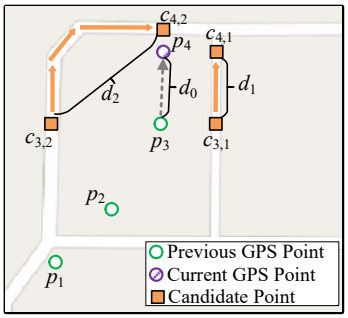


Fig. 4. Illustration of Transition Probability Calculation for a New GPS Point.

(2) Transition Probability Calculation. This step estimates the likelihood of transitioning between consecutive candidate points. Denoted as $P(c_t|c_{t-1})$, it represents the likelihood of moving from candidate point c_{t-1} at one timestamp to candidate point c_t at the next. This step ensures that the inferred trajectory adheres to physical constraints and realistic travel patterns. The transition probability is expressed as:

$$P(c_{t,i}|c_{t-1,j}) = \frac{1}{\beta} e^{-d_t/\beta} \tag{2}$$

where β denotes the scale parameter, set to 5 based on the prior research [28]. d_t represents the absolute difference between the Euclidean distance of the observed points and the corresponding path distance on the map, defined as follows:

$$d_t = |d_E(p_t, p_{t-1}) - d_{path}(c_{t,i}, c_{t-1,j})| \tag{3}$$

where $d_{path}(c_{t,i}, c_{t-1,j})$ represents the distance on the road. A smaller value of d_t results in a higher transition probability,

indicating that the model prefers transitions that closely align with both the observed distances and path-inferred distances, thereby ensuring realistic trajectory alignment. As illustrated in Fig. 4, d_1 and d_2 are two path distances of two pairs of candidates respectively, and d_0 is the Euclidean distance of the observed points. Since $|d_0 - d_1| < |d_0 - d_2|$, $c_{3,1}$ and $c_{4,1}$ show higher transition probability than $c_{3,2}$ and $c_{4,2}$.

Limitations of Traditional Methods. Although traditional HMM methods have demonstrated effectiveness in many scenarios, they encounter significant challenges, particularly in real-time applications involving streaming data. These challenges primarily stem from computational inefficiency and a lack of adaptability to dynamic environmental changes.

(1) Computational Inefficiency in Transition Probability Calculation. The traditional approach to calculating transition probabilities relies heavily on the single-source Dijkstra algorithm, which is computationally expensive. In streaming data scenarios, recalculating the shortest path for every consecutive pair of candidates leads to redundancy, especially when the points are temporally and spatially close. This redundancy results in unnecessary computational overhead.

(2) Inability to Adapt to Dynamic Environmental Changes. Traditional methods typically use fixed weights for observation and transition probabilities, which do not account for the dynamic nature of real-world environments. For instance, when a vehicle transitions from an open area to an urban environment with tall buildings, GPS signal quality may degrade significantly, leading to data errors. In such cases, the observation probability should be adjusted to reflect the reduced reliability of the GPS data. However, traditional methods fail to accommodate these variations, resulting in suboptimal performance under changing conditions.

Optimization Strategies. To address the limitations outlined above, we propose two optimization strategies: an efficient method for calculating transition probabilities using a bidirectional Dijkstra algorithm with a step-level shortest path cache, and a dynamic parameter adjustment mechanism that adapts to changing environmental conditions.

(1) Transition Probability Calculation with Step-Level Cache. In Equ. (3), the path distance d_{path} requires frequent shortest path queries between candidate sets of adjacent time steps. However, due to the spatial and temporal continuity of trajectories, GPS points at nearby time steps often yield overlapping or closely located candidates. This results in repeated queries for the same or similar candidate pairs across multiple steps, causing redundant computations.

To mitigate this inefficiency, we propose a step-level cache that stores shortest path results for candidate pairs computed in recent steps. When a previously encountered pair reappears, the cached result is directly reused, avoiding recomputation. The cache maintains shortest path results within an appropriate number of recent time steps, determined by a trade-off between computational efficiency and memory overhead. The default cache step for caching shortest path is 5. Additionally, we employ a bidirectional Dijkstra algorithm [29] to accelerate shortest path queries by reducing the search space from both ends, which is particularly effective for computing shortest paths between candidate sets in continuous matching, significantly improving efficiency while maintaining accuracy.

(2) Dynamic Parameter Adjustment. Traditional map matching methods often use fixed weights to combine observation and transition probabilities, i.e.,

$$S_c(c_{t,i}) = w_{ob} \log P(c_{t,i}|p_t) + w_{tr} \log P(c_{t,i}|c_{t-1,j}) \quad (4)$$

where $w_{ob}, w_{tr} \in [0, 1]$ are fixed weights satisfying $w_{ob} + w_{tr} = 1$, used to balance the the observation probability and the transition probability. However, using static weights across all time steps fails to reflect their varying importance under different conditions. However, such weights cannot adapt to environmental variability, such as GPS noise in urban canyons or irregular driving patterns.

To address this, we propose to dynamically adjust the relative importance of observation and transition terms. At each time step t, we define the loss function as:

$$L = -\sum_n \left(w_{ob}^t \log P(c_t|p_t) + w_{tr}^t \log P(c_t|c_{t-1}) \right) \quad (5)$$

where n is the number of candidate points at time step t, and w_{ob}^t and w_{tr}^t are respectively the weights of observation and transition probabilities at time step t (we set $w_{ob}^0 = w_{tr}^0 = 0.5$). This loss represents the negative total score of all candidates under the current weights. Minimizing it means maximizing the overall candidate scores, which encourages the model to favor configurations that better highlight the true matched points.

The weights w_{ob}^t and w_{tr}^t are updated via gradient descent and then normalized, where $\alpha = 0.01$ is the learning rate:

$$w_{ob}^t = w_{ob}^{t-1} + \alpha \sum_n \log P(c_t|p_t) \quad (6)$$

$$w_{tr}^t = w_{tr}^{t-1} + \alpha \sum_n \log P(c_t|c_{t-1}) \quad (7)$$

$$w_{ob}^t = w_{ob}^t/(w_{ob}^t + w_{tr}^t), w_{tr}^t = w_{tr}^t/(w_{ob}^t + w_{tr}^t) \quad (8)$$

The loss function L is linear with respect to w_{ob} and w_{tr}, implying convexity. Therefore, with a fixed and sufficiently small learning rate α, the gradient descent process is guaranteed to converge to a global minimum. Since the update only involves simple accumulation of log-probabilities at each step, the adjustment process has a constant time complexity of $O(1)$ and is well-suited for streaming scenarios.

C. Streaming Inference

The *Streaming Inference* module is responsible for processing incoming GPS data in real-time and generating the final matched path. Although our proposed method is also based on the online Viterbi algorithm, it differs from existing approaches that use a sliding window for backtracking calculations and output results only when convergence conditions are met. In contrast, our method calculates the probability for each new point in real-time and immediately outputs the corresponding matched point.

Candidates Matching Score Calculation. The first step in streaming inference is to calculate matching scores for

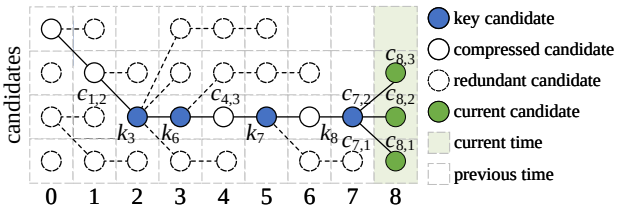


Fig. 5. An Example of the State Tree Used for Rollback Verification.

candidate points. For each incoming GPS point, candidate points are evaluated based on the observation and transition probabilities. These scores determine how well each candidate aligns with the GPS data. A composite matching score is then assigned to each candidate point using Equ. (4) but with dynamic weights.

Online Viterbi Inference. The second step involves online Viterbi inference [30], utilizing a state tree as illustrated in Fig. 5 to identify the most probable candidate point at the current time step. The online Viterbi algorithm is particularly effective for real-time streaming data, continuously updating the sequence of states to maximize the overall matching probability as new GPS data arrives. By maintaining this state tree, the algorithm efficiently identifies the candidate point with the highest matching probability at each step, based on the previous matching scores. Thus, the most probable candidate point for time step t is determined, allowing the algorithm to generate matched point in real time without the need for a complete history of all GPS data. The max path score $S(\mathcal{P}_{c_{t,i}})$ of the candidate point $c_{t,i}$ is defined as:

$$S(\mathcal{P}_{c_{t,i}}) = max_{j \in C_{t-1}}(S_c(c_{t,i}) + S(\mathcal{P}_j)) \tag{9}$$

We select the candidate point with the highest path score from the current candidate set C_t as the matching point m_t.

As illustrated in Fig. 5, we compute the score for each candidate point at time step t by utilizing the candidate points from the previous time step $t-1$, following Equ. (9). For each candidate at time t, the optimal connection is established by selecting the candidate from $t-1$ that yields the maximum path score. For instance, to determine the probability of $c_{8,1}$, we compare it with $c_{7,1}$ and $c_{7,2}$, selecting the one that maximizes the resulting path score as its parent. Consequently, $c_{8,1}$ is linked as the child node of the optimal candidate point $c_{7,2}$ from t_7. Finally, among $c_{8,1}$, $c_{8,2}$, and $c_{8,3}$, we identify $c_{8,1}$ as the matched point m_8 by selecting the one with the highest cumulative path score.

Rollback Verification. This step is to refine short-term matching accuracy and compensate for the myopic nature of online inference. This technique leverages the state tree structure preserved by online Viterbi inference and enables retrospective corrections within a bounded sliding window of size L.

(1) Key Candidate Points. Rollback is founded on the concept of key candidate points, defined as the nearest common ancestor (NCA) among all current candidate nodes in the state tree [30]. These key candidate points act as intermediate points that segment the final matching sequence into reliable subpaths. As shown in Fig. 5, candidates $c_{7,1}$ and $c_{7,2}$ at time step t_7 share a common ancestor k_7, which is marked as the key candidate. The matched subpath between two consecutive key candidate points is a part of the final matching sequence, thereby enabling facilitating consistent backtracking and efficient compression.

Algorithm 2: Sliding Window Backtracking

Input: State tree T, window size L, current time step t, original matched path $\mathcal{P}$, current matched path $\mathcal{P}'$
Output: Updated matched path $\mathcal{P}_{\text{new}}$

```
verifyKeyCandidatePoint(T.k_pre, T.k_cur);
foreach candidate point c' in P' do
    if t − L ≤ c'.t ≤ t then
        Update c in P with c' in P';
return P as P_new;
```

(2) State Tree Compression. The state tree maintains multiple candidate points at each time step to enable dynamic programming during Viterbi inference. However, only a single candidate from each time step will eventually be included in the final matching sequence. All other candidates are considered redundant once inference at that step is complete. To eliminate this unnecessary overhead, we introduce two compression mechanisms that significantly reduce the size of the state tree while preserving its rollback functionality: 1) Prune Redundant Leaf Nodes. Candidates with no child nodes are regarded as non-contributing to any possible future path and are deleted directly from the tree. 2) Bypass Unary Nodes. Candidates with only one child node are considered compressible. Their parent and child nodes are connected directly, bypassing the unary node. The bypassed node can be removed to further reduce memory usage.

For example, as shown in Fig. 5, dashed circles represent deleted candidates. At time step t_8, only 4 candidate points remain in the state tree: $c_{7,2}, c_{8,1}, c_{8,2}, c_{8,3}$. All other nodes have been removed via compression. Among the retained nodes, those with exactly one child (e.g., $c_{4,3}$) are labeled as compressed candidates, indicating they have been structurally bypassed by their children during tree simplification. Compression is executed immediately after each round of Viterbi inference. Because this process is applied to a small and localized portion of the state tree, it incurs negligible overhead. Moreover, by maintaining a compact tree structure, compression accelerates subsequent steps, including the identification of key candidate points and the rollback verification process.

(3) Backtrack with Sliding Window. Backtrack operates continuously at each time step. When a new key candidate k_{cur} is identified, the algorithm locates the previous key candidate k_{prev}. The earlier path is repalced by optimal path segment between k_{cur} and k_{prev}. To prevent unbounded delay and memory growth, we restrict rollback to a sliding window of length L. If k_{prev} is no longer within the window, we backtrack from k_{cur} to the earliest retained node. Otherwise, we rollback directly between k_{cur} and k_{prev}. The complete process is formalized in Algorithm 2.

Generating Matching Results. The final step in the *Streaming Inference* module is generating the matching results. Once rollback verification is complete and the optimal matching sequence is determined, the algorithm generates the final

matching results. If no rollback occurs, the result is classified as **extending** (i.e., the path continues without significant corrections). If rollback is performed, the result is classified as **changing** (i.e., the path undergoes significant revisions).

Regardless of the result type, EAR-MM ensures consistent per-point processing complexity. In the *candidate preparation* stage, spatial indexing enables candidate search in $O(\log k)$ time, where k is the number of road segments. The *probability computation* stage, which involves observation and transition calculations over candidate sets of size m and n, has a time complexity of $O(\log mn)$. This is the main computational bottleneck, but is mitigated by a caching mechanism that reuses previously computed shortest paths to reduce overhead.

In the *streaming inference* stage, EAR-MM applies an online Viterbi algorithm with a compact state tree to maintain optimal paths. Only partial results are emitted at merge points, achieving $O(n)$ time per step and significantly lower memory usage than the standard $O(Tn)$. The backtracking mechanism, triggered only when necessary, adds minimal latency.

Overall, the per-point cost is mainly influenced by the candidate set size across two steps, making EAR-MM efficient and scalable for real-time scenarios.

VI. Experiments

A. Datasets and Experimental Settings

Datasets. To evaluate the proposed method, we use two taxi trajectory datasets and two road network datasets. The key statistics of trajectory datasets are summarized in Table I.

TABLE I
Summary of Datasets

Property	CD-Taxi	WX-Taxi
# Objects	2940	653
# Trajectories	48,223	3,673
Sampling Interval	3.13s	2.45s
Spatial Coverage	(30.65°N, 104.04°E) – (30.73°N, 104.13°E)	(12.15°N, 40.68°E) – (76.78°N, 129.76°E)
Time Span	2018/10/01 – 2018/10/10	2020/07/18 – 2020/07/24
Disk Size	18.2GB	1.56GB

• **Chengdu Taxi Dataset (CD-Taxi)**: This dataset contains 48,223 trajectories collected over 10 days from October 1 to October 10, 2018, by 2940 taxis in Chengdu, China. The spatial coverage ranges from (30.65°N, 104.04°E) to (30.73°N, 104.13°E), with an average sampling interval of 3.13 seconds.

• **Wuxi Taxi Dataset (WX-Taxi)**: This dataset includes 3,673 trajectories recorded by 653 taxis in Wuxi, China, over a week-long period from July 18 to July 24, 2020. The average sampling interval is 2.45 seconds. This dataset is available[2].

• **Road Network Dataset**: We extract the corresponding road networks from OpenStreetMap, comprising 245,577 road segments in Chengdu and 91,270 road segments in Wuxi. This dataset is available[3].

[2]https://www.nbsdc.cn/general/dataSetHome
[3]https://www.openstreetmap.org/

We randomly select 2,000 trajectories from each dataset for the experiments. Trajectories are split if the time gap between consecutive points exceeds one hour. To simulate different sampling intervals, we down sample the GPS points by selecting them at regular intervals. For example, to simulate a 10-second interval from a dataset with a 2-second interval, we select every fifth point.

To evaluate the EAR-MM method, we generate ground truth paths using the approach from [13], [31], [32]. Specifically, we align high-frequency trajectories to the road network using the HMM map-matching algorithm [28], treating the matched results as the ground truth paths.

Parameters. We evaluate the model stability by varying the sampling interval I from 2 to 60 seconds, with a default value of 6 seconds. For backtracking analysis, the window size L is increased from 5 to 25 in 5 steps, with a default value of 20. Additionally, the weight parameter w is evaluated using five values: 0.3, 0.4, 0.5, 0.6, and 0.7, with a default value of 0.5, and compared with our automatic parameter tuning.

Metrics. The metrics include the average match time (AMT) and the accuracy (ACC):

$$AMT = \frac{\sum T_{match}}{N_{traj}} \tag{10}$$

$$ACC = \frac{\sum N_{correct}}{N_{total}} \tag{11}$$

where T_{match} is the match time for each trajectory, N_{traj} is the number of trajectories, $N_{correct}$ is the number of correctly matched points, and N_{total} is the total number of points.

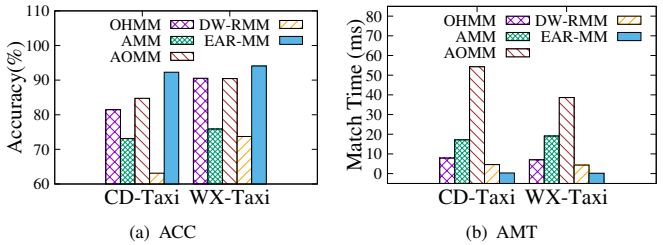
Baselines. We compare EAR-MM against four representative HMM-based map matching algorithms: OHMM [18], AMM [16], AOMM [27], and DW-RMM [15]. For a fair comparison in streaming scenarios, we implement streaming-compatible versions of all baselines and carefully tune their parameters for optimal performance. We do not include deep learning-based methods like [25] and [24], as they typically require full trajectories and offline training, making them unsuitable for real-time streaming scenarios.

• **OHMM** [18] is a classic HMM-based approach that introduces a variable sliding window and implements the online Viterbi algorithm. This allows the method to output partial matching results when a convergence point appears along the path.

• **AMM** [16] establishes a collaborative evaluation model for GPS points and candidate road segments. It filters low-quality GPS points, assigns different feature weights to adapt to various traffic environments, and introduces a backtracking correction mechanism to enhance matching accuracy.

• **AOMM** [27] is an HMM-based method that provides three adjustment strategies to address issues such as trajectory noise, dense trajectory points, and offset trajectory points.

• **DW-RMM** [15] uses dynamic weights to select the best matching road segment for each GPS point based on four criteria: distance, heading difference, direction difference, and road connectivity. It then calculates the confidence of the candidate segment. If the confidence is below a threshold, the road segment matched in the previous time step is retained.



(a) ACC (b) AMT

Fig. 6. Overall Comparison with Baselines.

Settings. All experiments are conducted on a server running Ubuntu 22.04, equipped with a TS80X E-2224G CPU, 128GB RAM, and dual 2TB hard disks. We use Kafka to simulate streaming data input for real-time processing. All methods are implemented in Java using JDK 1.8.

B. Overall Performance

Fig. 6 presents the comparative performance of our method against four alternative methods across two datasets, CD-Taxi and WX-Taxi. Our method, EAR-MM, consistently achieves superior performance in both accuracy and match time. Specifically, compared to AOMM on the two datasets, EAR-MM requires only 0.61% and 0.56% of the processing time of AOMM while achieving accuracy improvements of 8.87% and 4.03%, respectively.

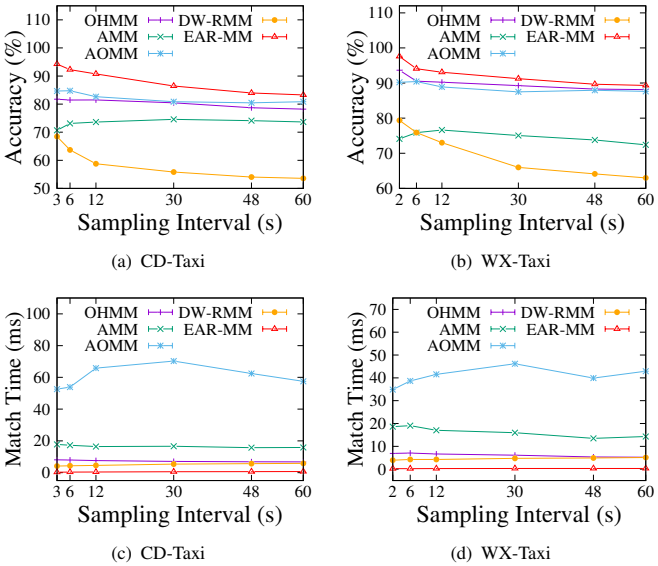
In terms of accuracy, EAR-MM outperforms AOMM and AMM with significant improvements. The accuracy gain is particularly evident on the CD-Taxi dataset, where EAR-MM achieves the highest accuracy compared to all baselines. This performance boost is largely attributed to the candidate reuse and rollback verification mechanisms. Regarding match time, EAR-MM shows remarkable efficiency, requiring significantly less time than AOMM and AMM. The reduced match time is primarily due to the use of the bidirectional Dijkstra algorithm and cache calculation techniques, which speed up the process without sacrificing accuracy.

Additionally, EAR-MM maintains stable performance across both datasets, demonstrating its robustness. This stability can be attributed to the automatic parameter adjustment mechanism, which ensures that the method performs optimally across diverse operational contexts.

C. Parameter Analysis in Performance

Sampling Interval I**.** The sampling interval plays a crucial role in the performance of map matching. As shown in Fig. 7, increasing the time interval between GPS points leads to lower accuracy and typically shorter match times. As the sampling interval increases, accuracy drops due to larger gaps between trajectory points, but EAR-MM consistently achieves the highest accuracy with the smallest decline.

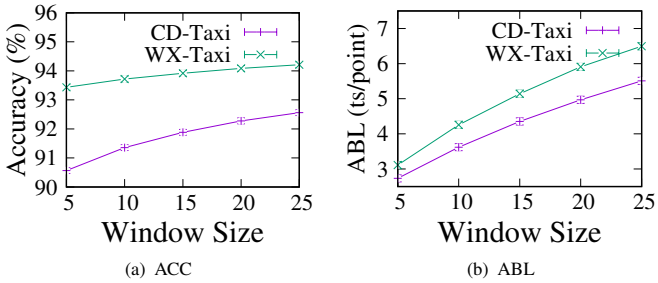
In terms of match time, while most methods see a slight reduction in time as the sampling interval increases, AOMM shows fluctuations, with match time initially increasing and then decreasing. This suggests that AOMM faces computational bottlenecks at specific sampling intervals. In contrast, EAR-MM maintains a stable, low match time without such



(a) CD-Taxi (b) WX-Taxi

(c) CD-Taxi (d) WX-Taxi

Fig. 7. Performance with Different Sampling Intervals.



(a) ACC (b) ABL

Fig. 8. Performance with Different Window Sizes.

fluctuations on both datasets, which demonstrates its robustness across different sampling intervals.

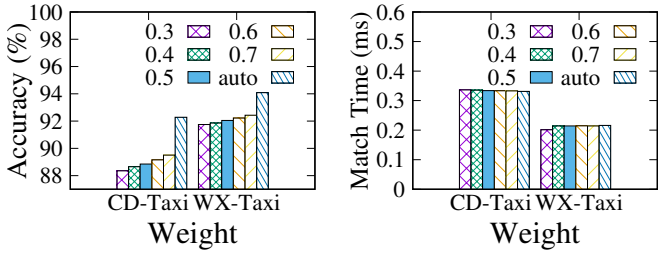
Window Size L**.** The window size determines the temporal scope for backtracking, focusing on recent points to improve the efficiency. To assess how the window size affects backtracking performance, we introduce the *Average Backtracking Latency* (ABL), which quantifies the average time difference between the correct backtracking point and the current matching point, normalized by the sampling interval I:

$$ABL = \frac{\sum(TS_{corrected} - TS_{match})}{I \times N_{corrected}} \tag{12}$$

Here, $TS_{corrected}$ represents the timestamp of the corrected candidate points, TS_{match} denotes the timestamp of the current matching point, and $N_{corrected}$ is the total number of corrected candidate points.

As illustrated in Fig. 8(a)-(b), an increase in the window size improves matching accuracy, but also increases ABL. This occurs because larger windows incorporate more candidate points, thereby providing a broader set of potential backtracking points and increasing the time required for backtracking.

Weight Parameter w**.** As shown in Fig. 9(c), our method's automatic parameter adjustment mechanism (auto) significantly improves accuracy over fixed weights (e.g., 0.3, 0.4, 0.5) across both CD-Taxi and WX-Taxi datasets. This mechanism



(a) ACC (b) AMT

Fig. 9. Performance with Different Different Weights.

is particularly beneficial in complex traffic environments, where it allows the model to adapt in real time, improving accuracy even under changing conditions.

Furthermore, as shown in Fig. 9(d), the automatic adjustment has little effect on match time. Both on CD-Taxi and WX-Taxi datasets, match time remains stable, meaning the adjustment improves accuracy without adding significant computational cost. Thus, the use of automatically adjusted weight parameters is ideal for real-time and high-precision applications.

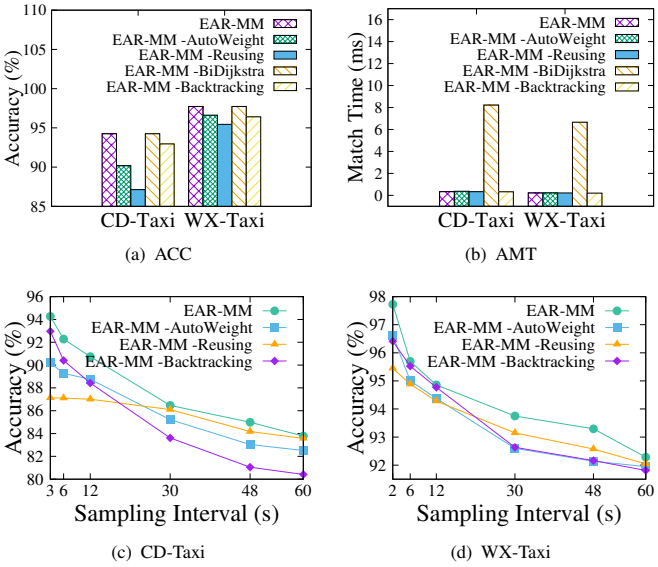
D. Ablation Study on Model Components

Fig. 10(a) and Fig. 10(b) present the ablation study results, illustrating the contributions of each EAR-MM component to accuracy and match time. The cached bidirectional Dijkstra algorithm primarily enhances efficiency by accelerating shortest-path computations. In terms of accuracy, candidate reusing is particularly effective under high GPS noise (e.g., in the CD-Taxi dataset) by filtering loop paths and producing more stable matched paths. Dynamic parameter adjustment further improves accuracy by adaptively tuning the balance between observation and transition probabilities in complex environments. In addition, as shown in Table II, the backtracking mechanism enhances robustness by enabling efficient rollbacks to real match result when key candidate is found. Leveraging the state tree structure, it supports multiple backtracking operations with minimal overhead. Specifically, the backtracking strategy improve the accuracy of 1.29% and 1.32% but with only match time increment of 0.01s and 0.02s per trajectory on CD-Taxi and WX-Taxi, respectively.

Moreover, we analyze how each component responds to varying sampling intervals. As shown in Fig. 10(c)-(d), all variants experience accuracy declines as the sampling interval increases, with sharper declines from 3s to 6s, indicating that component effectiveness relies more on low sample-interval trajectories, making them well suited for streaming scenarios. The dominant component also varies: at 3s, candidate reusing is the most impactful in CD-Taxi, while dynamic parameter adjustment leads in WX-Taxi. Within a single dataset, their influence shifts with sampling intervals. For instance, in CD-Taxi, candidate reusing is more effective at lower intervals, whereas backtracking helps more at higher intervals. These results demonstrate that the components complement each other, enabling EAR-MM to adapt across diverse conditions.

TABLE II
STATISTICS OF BACKTRACKING

Statistic Results	CD-Taxi	WX-Taxi
Times of Backtracking	4,929	653
No. of Backtracking Points	10,736	3,673
Match Time Increment	0.01s	0.02s
Accuracy Improvement	2.42%	0.98%



(a) ACC (b) AMT

(c) CD-Taxi (d) WX-Taxi

Fig. 10. Ablation Performance.

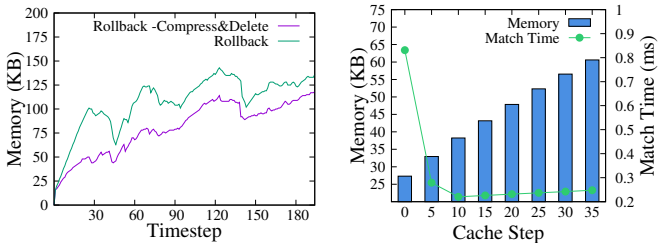
E. Memory Analysis

We conduct memory analysis experiments using WX-Taxi as an example. Fig. 11(a) illustrates the effect of compression and deletion operations on memory usage during rollback verification. The results demonstrate that both compression and deletion significantly reduce memory consumption throughout the entire matching process. Specifically, compared to traditional rollback verification methods, applying the compression and deletion strategies results in a much slower increase in memory usage, particularly during long-running operations. By deleting unnecessary candidate point data and compressing historical states, memory consumption is effectively controlled, thus reducing pressure on system resources.

Fig. 11(b) examines the impact of cache step size in the BiDijkstra algorithm on memory usage and matching time. A cache step of 0 indicates no caching of shortest paths between candidate points. As the cache step increases from 0 to 35, the memory usage grows linearly from 27.31KB to 60.59KB due to the expanded cache. Matching time initially decreases (cache step $\leq$ 10) as redundant shortest path computations are avoided, but increases slightly beyond that point (cache step $>$ 10). This is because shortest paths between temporally distant candidates are less likely to be reused, while the cache maintenance overhead increases.

F. Case Study

Fig. 12 and Fig. 13 compare the path matching performance of EAR-MM with several other algorithms. In intersection



(a) Memory (b) BiDijkstra

Fig. 11. Memory Performance.

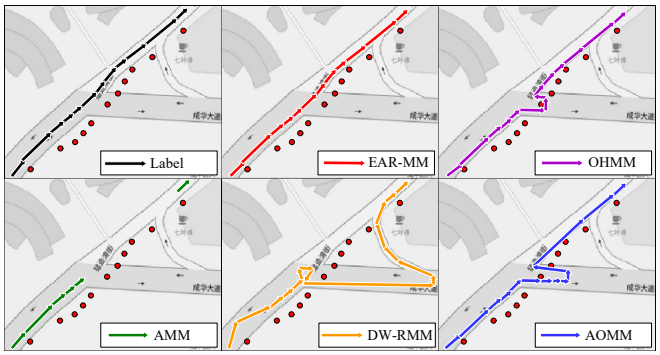


Fig. 12. The Case of the Intersection.

scenarios (Fig. 12), frequent turns and closely spaced branches often cause GPS points near junctions to be misaligned. Traditional methods like OHMM, DW-RMM, and AOMM tend to incorrectly match to a nearby side road instead of the correct through path, especially when the deviation seems locally optimal. AMM alleviates this issue to some extent by filtering out low-confidence points, but still lacks robustness in such situations. In contrast, EAR-MM addresses these challenges through adaptive parameter optimization and backtracking, minimizing the impact of offset points and avoiding unnecessary detours for better accuracy and robustness. In cases of traffic jam on parallel roads (Fig. 13), prolonged stops or slow movements often introduce GPS drift, which can resemble potential backward movements. This misleads algorithms into generating loop matches or switching to adjacent parallel roads. OHMM, DW-RMM, AOMM, and AMM frequently fall into such errors due to their lack of temporal consistency checks. EAR-MM, however, maintains high reliability by reusing previous candidates, allowing it to suppress false backward transitions and avoid forming incorrect loops.

VII. Conclusion

This work introduces EAR-MM, an efficient, adaptive and robust streaming map matching algorithm that uses an online Viterbi algorithm for real-time matched point generation. EAR-MM enhances robustness through candidate reuse and rollback verification, while dynamic parameter adjustment ensures adaptability to changing road conditions. With bidirectional Dijkstra and step-level caching, EAR-MM significantly reduces processing time. Extensive experiments show that EAR-MM achieves only 0.61% and 0.56% of AOMM's

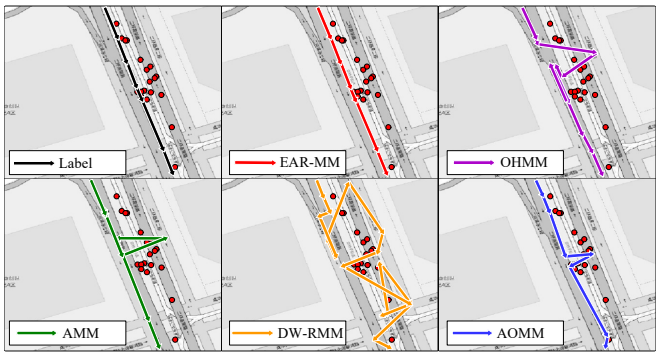


Fig. 13. The Case of Parallel Road Congestion.

processing time on two datasets, with accuracy gains of 8.87% and 4.03%, respectively. Future work will focus on enhancing scalability through parallelization.

References

[1] R. Li, H. He, R. Wang, S. Ruan, T. He, J. Bao, J. Zhang, L. Hong, and Y. Zheng, "Trajmesa: A distributed nosql-based trajectory data management system," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 1013–1027, 2023.

[2] R. Li, Z. Li, Y. Wu, C. Chen, and Y. Zheng, "Elf: Erasing-based lossless floating-point compression," *Proceedings of the VLDB Endowment*, vol. 16, no. 7, pp. 1763–1776, 2023.

[3] W. Gao, G. Li, and TAna, "Survey of map matching algorithms," *Journal of Software*, vol. 29, no. 2, pp. 225–250, 2017.

[4] K. Nagai, M. Spenko, R. Henderson, and B. Pervan, "Fault-free integrity of urban driverless vehicle navigation with multi-sensor integration: A case study in downtown chicago," *NAVIGATION: Journal of the Institute of Navigation*, vol. 71, no. 1, 2024.

[5] Y. Sun, F. Meng, R. Li, Y. Tang, C. Chen, and J. Zhong, "Streaming trajectory segmentation based on stay-point detection," in *International Conference on Database Systems for Advanced Applications*. Springer, 2024, pp. 203–213.

[6] L. Zhou, J. Shi, and D. Yang, "Vehicle positioning monitoring system based on gps/bds dual-mode positioning technology," in *2024 IEEE 4th International Conference on Electronic Technology, Communication and Information (ICETCI)*. IEEE, 2024, pp. 704–710.

[7] H. Manjunath and R. Mulangi, "Spatio-temporal analysis of public transit gps data: application to traffic congestion evaluation." *Advances in Transportation Studies*, vol. 62, 2024.

[8] L. Peng, X. Liao, T. Li, X. Guo, and X. Wang, "An overview based on the overall architecture of traffic forecasting," *Data Science and Engineering*, vol. 9, no. 3, pp. 341–359, 2024.

[9] A. R. M. Forkan, Y.-B. Kang, F. Marti, A. Banerjee, C. McCarthy, H. Ghaderi, B. Costa, A. Dawod, D. Georgakopolous, and P. P. Jayaraman, "Aiot-citysense: Ai and iot-driven city-scale sensing for roadside infrastructure maintenance," *Data Science and Engineering*, vol. 9, no. 1, pp. 26–40, 2024.

[10] W. Wang, Q. Yu, R. Duan, Q. Jin, X. Deng, and C. Chen, "Low-frequency trajectory map-matching method based on probability interpolation," *TIG*, 2024.

[11] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, 2018.

[12] R. Li, H. Zhu, R. Wang, C. Chen, and Y. Zheng, "Fast and distributed map-matching based on contraction hierarchies," *Journal of Computer Research and Development*, vol. 59, no. 2, pp. 342–361, 2022.

[13] L. Jiang, C.-X. Chen, and C. Chen, "L2mm: learning to map matching with deep models for low-quality gps trajectory data," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 3, pp. 1–25, 2023.

[14] R. Li, H. He, R. Wang, Y. Huang, J. Liu, S. Ruan, T. He, J. Bao, and Y. Zheng, "Just: Jd urban spatio-temporal data engine," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1558–1569.

[15] M. He, L. Zheng, W. Cao, J. Huang, X. Liu, and W. Liu, "An enhanced weight-based real-time map matching algorithm for complex urban networks," *Physica A: Statistical Mechanics and its Applications*, vol. 534, p. 122318, 2019.
[16] H. Hu, S. Qian, J. Ouyang, J. Cao, H. Han, J. Wang, and Y. Chen, "Amm: an adaptive online map matching algorithm," *TITS*, vol. 24, no. 5, pp. 5039–5051, 2023.
[17] H. Qi, Z. Huang, Y. Chen, Y. Zhang, and Y. Gao, "Streamlining trajectory map-matching: a framework leveraging spark and gpu-based stream processing," *International Journal of Geographical Information Science*, vol. 38, no. 6, pp. 1158–1178, 2024.
[18] 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 *2012 15th ITSC*. IEEE, 2012, pp. 776–781.
[19] C. Chen, Y. Ding, X. Xie, and S. Zhang, "A three-stage online map-matching algorithm by fully using vehicle heading direction," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 5, pp. 1623–1633, 2018.
[20] Z. Liu, Y. Zhou, X. Liu, H. Zhang, Y. Dong, D. Lu, and K. Wu, "Learning road network index structure for efficient map matching," *TKDE*, no. 01, pp. 1–15, 2024.
[21] G. Wang and R. Zimmermann, "Eddy: An error-bounded delay-bounded real-time map matching algorithm using hmm and online viterbi decoder," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2014, pp. 33–42.
[22] G. R. Jagadeesh and T. Srikanthan, "Online map-matching of noisy and sparse location data with hidden markov and route choice models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2423–2434, 2017.
[23] S. Ma and H. Lee, "A practical hmm-based map-matching method for pedestrian navigation," in *2023 International Conference on Information Networking (ICOIN)*. IEEE, 2023, pp. 806–811.
[24] J. Feng, Y. Li, K. Zhao, Z. Xu, T. Xia, J. Zhang, and D. Jin, "Deepmm: Deep learning based map matching with data augmentation," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2372–2384, 2022.
[25] Y. Liu, Q. Ge, W. Luo, Q. Huang, L. Zou, H. Wang, X. Li, and C. Liu, "Graphmm: Graph-based vehicular map matching by leveraging trajectory and road correlations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 1, pp. 184–198, 2024.
[26] S. Taguchi, S. Koide, and T. Yoshimura, "Online map matching with route prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 338–347, 2018.
[27] W. Lv, Y. Chen, P. Shang, F. Zheng, and J. Wang, "A self-adjusting online map matching method," in *Journal of Physics: Conference Series*, vol. 2006, no. 1. IOP Publishing, 2021, p. 012035.
[28] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2009, pp. 336–343.
[29] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *Experimental Algorithms: 7th International Workshop, WEA 2008 Provincetown, MA, USA, May 30-June 1, 2008 Proceedings 7*. Springer, 2008, pp. 319–333.
[30] R. Šrámek, B. Brejová, and T. Vinař, "On-line viterbi algorithm for analysis of long biological sequences," in *Algorithms in Bioinformatics: 7th International Workshop, WABI 2007, Philadelphia, PA, USA, September 8-9, 2007. Proceedings 7*. Springer, 2007, pp. 240–251.
[31] C. Chen, Q. Liu, X. Wang, C. Liao, and D. Zhang, "semi-traj2graph identifying fine-grained driving style with gps trajectory data via multi-task learning," *IEEE Transactions on Big Data*, vol. 8, no. 6, pp. 1550–1565, 2021.
[32] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *2012 ICDE*. IEEE, 2012, pp. 1144–1155.

Yangyang Sun is currently a master student at the School of Computer Science, Chongqing University. His research direction is streaming spatiotemporal data mining and management.

Tianyu Huang is currently a master student at the School of Computer Science, Chongqing University. His research direction is integrated streaming spatiotemporal data management.

Zijian Zhang is currently an undergraduate student at the School of Computer Science, Chongqing University. His research direction is streaming spatiotemporal data mining.

Ruiyuan Li is an associate professor with Chongqing University, China. He is the director of Start Lab (Spatio-Temporal Art Lab). He received the B.E. and M.S. degrees from Wuhan University, China in 2013 and 2016, respectively, and the Ph.D. degree from Xidian University, China in 2020. He was the Head of Spatio-Temporal Data Group in JD Intelligent Cities Research, leading the research and development of JUST (JD Urban Spatio-Temporal data engine). Before joining JD, he had interned in Microsoft Research Asia from 2014 to 2017. His research focuses on Spatio-temporal Data Management and Urban Computing.

Chao Chen received the B.E. and M.S. degrees from Northwestern Polytechnical University, Xi'an, China, in 2007 and 2010, respectively, and the Ph.D. degree from Pierre and Marie Curie University and the Institut Mines-Télécom/Télécom SudParis, France, in 2014. In 2009, he was a research assistant with the Hong Kong Polytechnic University. He is currently a full professor with the College of Computer Science, Chongqing University, China. His research interests include pervasive computing, mobile computing, urban logistics, data mining from large-scale GPS trajectory data, and big data analytics for smart cities.

Yu Zheng is the Vice President of JD.COM and president of JD Intelligent Cities Research. Before Joining JD.COM, he was a senior research manager at Microsoft Research. He is also a chair professor at Shanghai Jiao Tong University. He was the Editor-in-Chief of ACM Transactions on Intelligent Systems and Technology and has served as the program co-chair of ICDE 2014 and CIKM 2017. He is a keynote speaker of AAAI 2019, KDD 2019 Plenary Keynote Panel and IJCAI 2019 Industrial Days. He received SIGKDD Test-of-Time Award twice (in 2023 and 2024). He was named one of the Top Innovators under 35 by MIT Technology Review (TR35), an ACM Distinguished Scientist and an IEEE Fellow, for his contributions to spatio-temporal data mining and urban computing.