Hindawi
Scientific Programming
Volume 2022, Article ID 3107779, 15 pages
https://doi.org/10.1155/2022/3107779

Research Article

Map-Matching on Low Sampling Rate Trajectories through Frequent Pattern Mining

Lei Yu,[1] **Zhiqiang Zhang**,[2] **and Rongtao Ding**[3]

[1]*School of Business Management, HangZhou Polytechnic, Hangzhou 314200, China*
[2]*School of Software Engineering, Tongji University, Shanghai 200092, China*
[3]*School of E-commerce, Zhejiang Business College, Hangzhou 310053, China*

Correspondence should be addressed to Lei Yu; yulei@mail.hzpt.edu.cn

Received 7 December 2021; Accepted 26 February 2022; Published 21 March 2022

Academic Editor: Sheng Bin

Map-matching, an important preprocessing task in many location-based services (LBS), projects each point of the global positioning system (GPS) within a trajectory dataset onto a digital map. The state-of-the-art map-matching algorithms typically employ Hidden Markov model (HMM) via shortest path computation. But the computation of the shortest path might not work well on low-sampling-rate trajectory data (e.g., one GPS point every 1–5 min), leading to low matching precision and high running time. To solve the problem, this paper firstly identifies frequent patterns (FPs) in historical trajectories to capture meaningful mobility behaviors, and then extracts mobile behavior criterion (MBC) of mobile users. Such a criterion generally represents the route choice of mobile users on road networks. Moreover, the temporal information within trajectory data was employed to estimate the speed of mobile users on road segments. The identified FPs, coupled with MBC and moving speed, help to improve the map-matching precision of low-sampling-rate trajectories. In addition, an FP-forest structure was proposed to index the identified FPs. The structure could greatly speed up the lookup of frequent paths for shorter running time. Furthermore, the FP-forest structure was pruned to reduce redundancy with smaller space cost. Finally, experiments were carried out on real-world datasets. The results confirm that our FP-matching method outperforms state-of-the-art in terms of effectiveness and efficiency.

1. Introduction

Recent years bear witness to the proliferation of location-based services (LBSs) on mobile devices, such as Uber provides users with taxi service and Google Map provides navigation services. Relying on global positioning system (GPS) sensors, these LBSs record life trajectories of humans, and generate massive trajectory data. The data have been utilized in many applications to understand human mobility patterns, namely, activity recognition, hot route finder, geographical social network, and urban planning.

Nevertheless, GPS trajectories often contain lots of noisy GPS coordinates, owing to the inevitable errors of GPS positioning [1–7]. These coordinates deviate from the true positions of mobile devices, calling for the important preprocessing task of map-matching [8–15]. Map-matching aims to project every recorded GPS point within the trajectory data onto a digital map. It has applications in satellites navigation, GPS tracking of freight, and transportation engineering. In this way, the location point recorded by GPS sensor can be corrected onto the road network, even if their coordinates deviate from the true values.

The frequency of the GPS points of mobile users recorded by mobile devices or third-party applications is known as the sampling rate of GPS positions. Low-sampling-rate trajectories with sparse GPS positions are ubiquitous. So we need to study the map matching technology when the trajectory is low sampling rate. For example, when mobile users often switch off GPS sensors to save energy or preserve privacy, their GPS trajectories will contain very sparse GPS points with low sampling rate. LBS applications like Foursquare maintain sparse check-in data of mobile users. In addition, telecom operators like Telco/Cellular could extract sparse GPS coordinates within Web logs, when mobile users are using data services [16, 17]. All these sparse positions will reduce the sampling rate of the trajectories.

It is a challenging task to design an accurate map-matching algorithm for low-sampling-rate trajectories with sparse positions [18–20]. The state-of-the-arts essentially adopt the hidden Markov model (HMM) and its variants [11, 12] to perform map-matching for low-sampling-rate GPS trajectories, under the assumption that the path between two GPS points is the shortest path of a certain metric. But the assumption does not necessarily hold in real life, especially in the face of low-sampling-rate trajectories. Low sampling rate only makes the position points in the trajectory sparse, but it does not mean that the positioning accuracy decreases. When the location points are sparse, the distance between two continuous points is too large. At this time, using the shortest path to represent its real path cannot represent the real situation.

Take Figure 1 for example. There is a trajectory of six GPS points, where P1 and P6 are the source and destination, respectively. If map-matching is performed for the trajectory based on the shortest path, the correct matching result is the blue route $P_1 \longrightarrow P_2 \longrightarrow \cdots \longrightarrow P_5 \longrightarrow P_6$. However, when the trajectory is so sparse as to contain only two points $P1$ and $P6$, the shortest-path-based map-matching will converge to an incorrect result: the red route $P_1 \longrightarrow P_6$.

The above example shows the difficulty in choosing between the multiple optional routes between the sparse GPS points on a trajectory. Each route could indicate the preference of some mobile users, and could be the matching result of the trajectory. Therefore, it is impossible to estimate the mobile user's long-term use habits, using the shortest path solely based on one metric.

To tackle the issue above, this paper proposes a frequent pattern (FP)-based map-matching technique for low-sampling-rate trajectories. Firstly, the FPs were identified in third-party data on historical trajectories. Then, the FPs that best match the input low-sampling-rate trajectory were identified, and linked as the map-matching result of that trajectory. These FPs clearly reflect the mobility behaviors of most mobile users among historical trajectories: each of them is with the most familiar route, the fastest route, or the shortest route, rather than with the shortest route alone. By linking the multiple FPs, the map-matching result of the input trajectory mixes various preferences, pushing up the precision of map-matching.

Nowadays, almost all smartphones and vehicles are equipped with GPS sensors to record trajectories. As a result, there are many publicly available third-party trajectories, such as those recorded by moving taxis and Open Street Map. Some of these trajectories contain high-sampling-rate GPS points. If FPs are mined from these trajectories, it would be highly possible to design an accurate map-matching algorithm for low-sampling-rate trajectories. For instance, Zhu et al. [17] lowered the positioning errors with the help of the third-party data on historical GPS trajectories of taxis, and then realized more precise map-matching for the low-sampling-rate trajectories of sparse locations within the big data of Telco/Cellular.

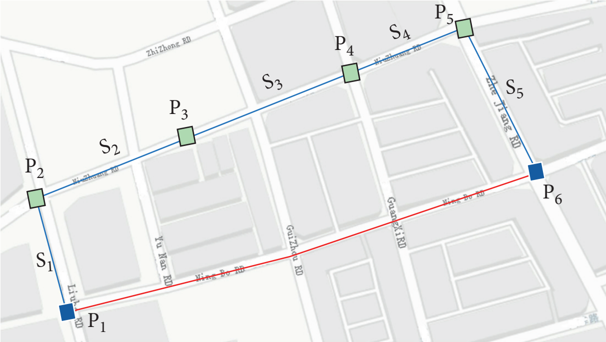


Figure 1: An example of the failure of shortest-path-based map-matching.

Apart from the mined FPs, several other meaningful factors were leveraged in this research. One of the most important factors remains the selection of short and straight routes. Firstly, two meaningful route features (length and transition) were chosen to define the general mobile behavior criterion (MBC). Next, the temporal information within trajectory data was employed to estimate the moving speed on road segments, for people tend to stay close to the speed limit of each segment. Based on the massive trajectories moving on every segment, the overall speed distribution was captured for different road segments. For each input estimated speed, it is possible to infer the probability of the speed truly moving on each segment. Finally, the probability, together with the two route features (length and transition), was adopted to derive the best route of connected road segments, laying the basis for better matching precision.

Besides the effectiveness of map-matching, the mined FPs make map-matching more computationally efficient. Specifically, an FP-forest structure was designed to index the identified FPs. The structure offers fast query response to look up a certain pattern used by map-matching, allowing the proposed FP-matching algorithm to converge quickly to the matching result. To reduce the space cost of FP-forest, the redundant road segments were pruned from the mined FPs.

The main contributions of this research are as follows:

(1) FP-matching, a novel map-matching algorithm, was designed to capture the mobility behaviors of most people. However, the state-of-the-arts use only one metric, FP-matching effectively maps low-sampling-rate trajectories onto a digital map, using FPs mixed with multiple metrics, general MBC, and temporal information.

(2) An FP-forest indexing structure was developed to respond quickly to the lookup for a certain FP, turning the raw GPS sequence to a road segment sequence. In addition to matching effectiveness, our FP-matching algorithm can thereby achieve high computational efficiency, as evidenced by short running time and low space overhead.

(3) The effectiveness and efficiency of FP-matching were compared with those of the state-of-the-arts. The

comparison shows that our method outperforms the contrastive methods in both matching accuracy and running time.

The rest of the paper is organized as follows: Section 2 reviews the related work; Section 3 defines the problem and overviews the solution; Section 4 describes the FP-forest structure; Section 5 presents the FP-matching algorithm; Section 6 verifies the proposed algorithm; Section 7 puts forward the conclusions.

2. Literature Review

This section mainly reviews the related work on map-matching and FPs.

2.1. Map-Matching. Map-matching intends to locate a sequence of GPS points, consisting of timestamp, longitude, and latitude onto a digital map. Over the past decades, extensive research has been conducted on map-matching. The relevant works can be broadly classified into three categories: incremental approach, global approach, and geometrical approach [21].

2.1.1. Incremental Approaches. Quddus et al. [13] predicted the road segment of a GPS point based on the previous prediction of that segment, and decided which road segment should the current GPS point be mapped to, taking into account of two factors: the projection distance from the current GPS point to its candidate segments; the difference between the GPS heading and direction of the candidates. White et al. [14] evaluated the performance of four incremental algorithms, and found that all of them have a low accuracy. The reason is that the incremental approach selects the best candidate for each sample at the current time stamp, based on a small range of recent samples. However, the selected candidate is not the global optimal candidate, and only works well online.

2.1.2. Global Approach. Like the incremental approach, the global approach searches for the candidates of each GPS sample, and then computes a weight for each candidate. Considering the entire trajectory, the global approach calculates the aggregated weight of the candidate sequence, and looks for the candidate with the largest weight. With the aid of the HMM, Lou and Newson et al. [11, 12] calculated the emission probability under the normal distribution of projection distances between GPS points and segment candidates. Lou et al. [11] also integrated spatial function with temporal function to compute the transition probability, while Newson and Krumm [12] adopted an exponential function of route distance and straight-line distance to deduce that probability. However, the transition probability was calculated with a long running time, due to the necessity to compute the shortest distance between candidates.

2.1.3. Geometrical Approach. The geometrical approach frequently searches for the optimal path on a digital map by geometric similarity measures, e.g., the Fréchet distance [22], eliminating the need to find a candidate set for each GPS point. For example, Alt and Brakatsoulas et al. [8, 9] looked for the path with the minimum Fréchet distance to the GPS sequence.

FP-matching, our map-matching algorithm, is a global approach. This paper compares our algorithm with the HMM algorithms proposed by Lou et al. [11], Newson and Krumm [12], and Zheng et al. [15], which were proved to outshine other map-matching methods on low-sampling-rate trajectories. Our algorithm differs from these HMM algorithms in the following aspects:

First, our algorithm leverages the FPs mined from the historical trajectories moving on at least two continuous road segments to replace the transition probability of these segments, and maps the GPS points onto a road map through dynamic programming. Second, the FP-forest structure was utilized to find the most frequent routes between two mapped segments, and all the routes were connected to complement the entire trajectory. FP-matching greatly reduces the running time by eliminating the computation of the shortest distance.

Zheng et al. [15] presented a history-based route inference system (HRIS) for map-matching based on reference trajectories. Different from HRIS, our approach is grounded on the mined FPs, which reflect the mobility patterns of those historical trajectories successfully map-matched onto road networks. The accuracy and efficiency of our approach are both better than those of HRIS, for the mined FPs are with map-matching result, while HRIS is still with raw trajectories.

2.2. FP Mining. FPs play an important role in many data mining tasks, and contribute immensely to the solution of practical problems [23]. Traditionally, FPs are mined in two ways: sequence pattern mining, and association rules mining. The sequence pattern mining, represented by example PrefixSpan [24] and generalized sequential pattern (GSP) [25, 26], takes basis on divide-and-conquer algorithm, and draws the merits of database projection pattern and correlation algorithms. The association rules mining, e.g., Apriori [1], FP-growth [27] and Eclat [28], is underpinned by candidate set generation and test. For instance, sequential pattern discovery using equivalent class (SPADE) [29] extends the Apriori algorithm into sequential FP problem. Lin et al. [30] proposed a hybrid multilevel search algorithm to mine long FPs. Prabamanieswari [31] invented the fuzzy-based frequent itemset mining to reduce the number of scanning database, and demonstrated that the method is far superior to fuzzy Apriori [32].

While Apriori [33] faces high time and space costs, FP-growth [27] builds a FP-tree structure from the database, and recursively looks for frequent item sets by traversing the FP-tree without explicitly generating redundant candidates. In our FP-matching algorithm, FP-forest is adopted to index the FPs. Despite some similarities, our FP-forest has some

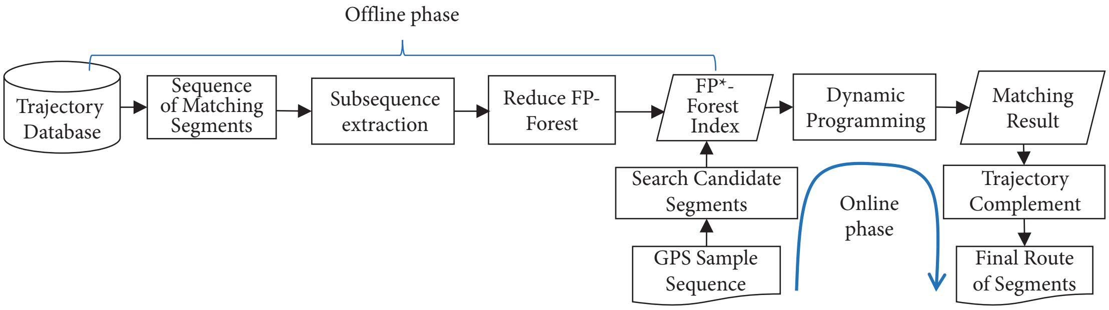


Figure 2: Overview of our solution.

significant differences from FP-tree. First, a long sequence of road segments, i.e., the map-matching result of the input high-sampling-rate trajectory, was split into multiple subsequences, and an FP-forest was established on the multiple FP-trees for these subsequences. The FP-forest speeds up the lookup of a certain sub-trajectory from a pair of source and sink inputs (road segments). But the fast speed is realized at the cost of many redundant FP-trees. Thus, the redundant segment identities (IDs) were pruned to save the space cost. Second, more other items were introduced to the FP-forest, including the speed histogram on each indexed segment, trajectory count, and mobility behavior weight W_m, to improve the effectiveness of map-matching.

3. Problem Definition and Solution Overview

This section firstly defines the research problem and then highlights our solution to the problem.

Let $M = (V, S)$ be a digital map containing a set of intersections (vertices) V and a set of segments (edges) S; T be a GPS trajectory containing a sequence of GPS samples (p_i, t_i), $1 \leq i \leq |T|$, each of which has a GPS position p_i and a timestamp t_i. Position p_i is a pair of GPS longitude and latitude. Map-matching aims to project the GPS points within the input trajectory T onto a sequence of consecutive segments $s_i \in S$ in M. In the example of Figure 1, a route of connected segments $s_1 \longrightarrow \cdots \longrightarrow s_5$ can be generated through map-matching for the raw trajectory containing six GPS points.

At a very low sampling rate, the trajectory T only contains very sparse GPS points. In this case, it is highly possible to project these points onto disconnected road segments, i.e., the source and destination segments are not connected. The disconnections should be complemented with a path in the digital map M, such as to recover the entire route of connected segments.

To match a sparse trajectory T, our general idea is to exploit a third-party database containing historical high-sampling-rate trajectories. Such a database is widely available nowadays. Any mobile device, namely, taxis and smartphones, with GPS sensors can generate high-sampling-rate trajectories. Hence, this paper relies on the said database to optimize the map-matching of the input trajectory T.

Problem 1. For a historical trajectory database D containing $|D|$ high-sampling-rate trajectories $T_1 \ldots T_{|D|}$, it is necessary to identify a set of meaningful trajectory patterns from the database, and derive from these patterns the most likely route for a very sparse input trajectory T onto map M.

Figure 2 shows our solution to the above problem: a two-phase map-matching framework. In the offline phase, an FP-forest structure was maintained based on the historical trajectory database. On the high-sampling-rate trajectories in the database, a classic map-matching algorithm was applied to find the sequence of matching segments, onto which GPS points in those trajectories are projected. Next, subsequences were extracted from the sequences of matching segments. Then, all the sequences and subsequences were indexed by the FP-forest.

In the online phase, the digital map was traversed to find the nearest candidate segments for the input trajectory of very sparse GPS points. After that, the FPs containing these candidates were found under the FP-forest structure, and the path that best matches the input trajectory was identified through dynamic programming. The purpose of dynamic programming is to select the best candidate for each GPS sample according to the FPs obtained via the FP-forest. Finally, the disconnected segments in the best matching path, if necessary, were complemented to obtain the entire route.

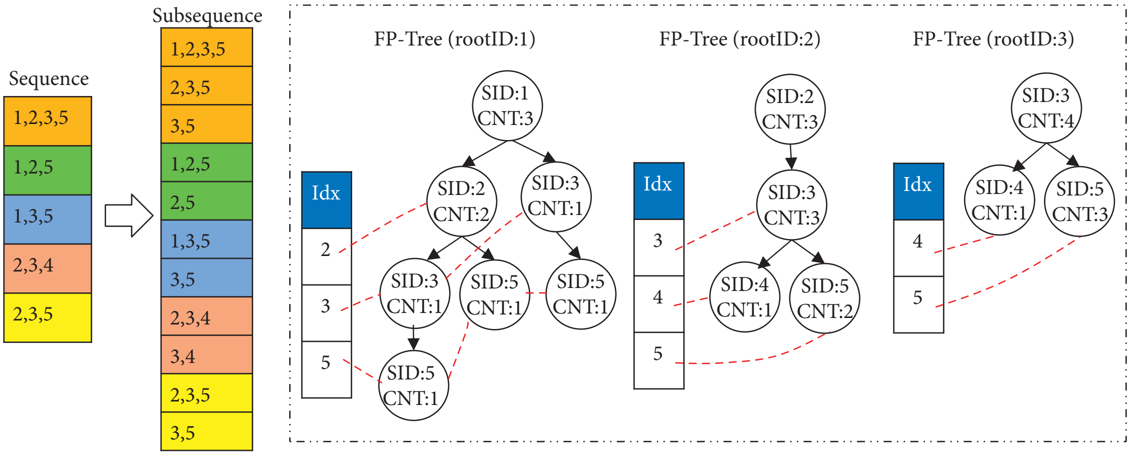
Overall, the offline phase attempts to maintain the FP-forest based on the historical database D, and the online phase seeks to evaluate the input trajectory T against the FP-forest to generate the most likely route of connected road segments.

In addition, a new FP-forest structure was developed to improve the accuracy of map-matching. To reduce the space cost, the FP-forest was downsized by an efficient method. Our algorithm applies to the new FP-forest structure, because it is similar to the original FP-forest structure.

4. FP-Forest

This section introduces the structure and generation of the FP-forest, details the lookup operation, presents a simple and efficient method to reduce the FP-forest, and finally reconstructs the FP-forest.

4.1. FP-Forest. In the FP-forest, there are multiple FP-trees rooted at the associated segment IDs. Each FP-tree is similar to the tree in the famous FP-growth structure, whose root is



(a)

(b)

Figure 3: An example of FP-forest.

null. Every node inside an FP-tree contains two items: a road segment ID (SID) and a counter (CNT) of trajectories, which records the number of trajectories matching the current segment. Besides, each FP-tree has an associated dictionary, in which each key is an SID pointing to a list of tree nodes, whose SIDs are equal to the key.

The FP-forest in Figure 3 contains three FP-trees, rooted at SIDs 1, 2 and 3, respectively. For the FP-tree rooted at SID 1, the dictionary maintains three keys 2, 3 and 5. Each key points to a list of internal nodes in the FP-tree. For example, key 2 points to the list of only one node with the pair <2, 2>, key 3 points to the list of two nodes with the pair <3, 1>, and key 5 points to the list of three nodes with the pair <5, 1>.

4.2. Generation of FP-Forest.

Under the given FP-forest structure, this subsection demonstrates how to generate an FP-forest from historical database D. The first step is to identify the high-sampling-rate trajectories T_i in the database D. After matching trajectories T_i onto map M through classic map-matching [11, 12], the sequences of matching segments are available: once a GPS point in T_i is mapped onto a certain segment $s_j \in M$, we have a sequence of distinct segments s_j with respect to (w.r.t) T_j. After that, the FP-forest structure in Figure 3 can be called to index all segment sequences with respect to the high-sampling-rate trajectories T_i.

The generation of the FP-forest can be realized in two steps: (1) extracting subsequences from each segment sequence after map-matching each high-sampling-rate trajectory $T_i \in D$; (2) inserting the subsequences into the FP-forest structure to enable fast lookup of the forest.

Step 1. Subsequence extraction:

The following $(k-1)$ subsequences: $\{s_2 \longrightarrow s_3 \longrightarrow \cdots \longrightarrow s_k\}$, $\{s_3 \longrightarrow \cdots \longrightarrow s_k\}$, …, $\{s_{k-1} \longrightarrow s_k\}$ are extracted from the given sequence of segments $S = \{s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_k\}$. This step generates a total of k subsequences, including the extracted subsequences plus the original sequence S. The 11 subsequences extracted from 5 sequences of segments are presented in Figure 3.

Step 2. Subsequence insertion:

A new subsequence $S = s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_k$ is inserted into the FP-forest structure in the following manner. First, it is necessary to check if the FP-forest contains a tree whose root equals SID s_1. If not, a new FP-tree is built with s_1 as the root ID, with the root counter set to 1. Otherwise, the counter of this root is increased by 1.

After that, the rest of sequence S, e.g., $\{s_2 \longrightarrow \cdots \longrightarrow s_k\}$, is processed by one of the two operations. One of the operations is to update the counter of an existing node in this tree, and the other to create a new node as a child of an existing node. Take consider a subsequence $\{s_i \longrightarrow s_{i+1}\}$ for example. It is assumed that an existing node with SID s_i has been visited just now. Then, it is necessary to visit the segment node s_{i+1}. If node s_i contains a child with SID s_{i+1}, then the counter of the child s_{i+1} is increased by 1. Otherwise, a new child node is created with the SID, and its counter is increased by 1.

The pseudocode of the above steps is given as Algorithm 1. Lines 1–14 provide the details on how to generate an FP-forest. In Algorithm 1, each trajectory of the input database D is with a sequence $S \in D$ of connected road segments $S = \{s_1 \ldots s_{|s|}\}$, which are obtained through classic map-matching [11, 12]. Lines 15–18 search for a specific FP-tree in FP-forest, and Lines 19–22 look for the child of a given node.

The complexity of Algorithm 1 can be evaluated as follows: suppose each trajectory sequence $S \in D$ contains at most k segments. Thus, the number of generated subsequences is no greater than k, i.e., the total number of operations amounts to O(k) (i.e., findTree). Thus, As shown in Figure 4 the running time of Algorithm 1 is $O(|D| \times k)$, where $|D|$ is the number of trajectories in D.

The Conclusions section should clearly explain the main findings and implications of the work, highlighting its importance and relevance.

```
Algorithm 1: CreateForest(Trajectory Database $D$)
Output: FP-model F
$F = \Phi$;
foreach $S \in D$ do
    TREE $t \leftarrow$ FindTree($F$,$S.s_1$); NODE $r$ = null;
    if $t$! = $null$ then {$r = t$.root; $r$.counter++ };
    else $r$.SID = $S.s_1$; $r$.counter = 1; $t$.root = $r$; add $t$ to $F$;
    NODE $curNode = r$;
    for $i = 2$; $i \le |S|$; $i + +$ do
        NODE $n \leftarrow$ FindChild($curNode$, $s_i$);
        if $n$! = $null$ then {$curNode = n$; $n$.counter++};
        else
            $n$.counter = 1;
            $n$.parent = $curNode$; $curNode$.addChild($n$);
            $t$.addIndex($n$); $curNode = n$;
return $F$;
FindTree (FOREST $F$, SID $s$)
    if $F$ contains a tree whose root has the segment ID $s$ then
        return the tree in $F$ whose root is the SID $s$;
    else return NULL;
FindChild (NODE $curNode$, SID $s$)
    if $curNode$ has a child whose segmentId is $s$ then
        return the child of $curNode$;
    else return NULL;
GetFreq (FOREST $F$, SID $sid$, SID $eid$)
    $f = 0$;Tree $t \leftarrow$ findTree($F$, $sid$); PATH $p$ = [];
    for NODE $n$ in $t.index(eid)$ do $f \leftarrow f + n$.counter;
    return $f$;
```

Figure 4: Algorithm 1.

4.3. FP-Forest Lookup. This subsection introduces two lookup operations on FP-forest: frequency lookup and trajectory complementation.

4.3.1. Frequency Lookup. The goal is to find the counter (frequency) of trajectories passing from one segment sid to another eid To this end, the returned frequency is initiated as zero. Then, the FP-tree is looked for, whose root with the SID equal to sid. After that, a list of tree nodes with the SIDs equal to eid is found with the help of the dictionary. For each tree node with the SID eid, if there exists a path from the root (with SID sid) to the tree node (with SID eid), the counter of such a tree node is added to the returned frequency.

Here is an example for the frequency lookup from sid = 1 to eid = 5. Firstly, the FP-tree with the root ID 1 is looked up. Next, it is assured that the associated dictionary contains key 5 in the FP-tree. Key *t* points to the list of three internal nodes with the pair <5, 1,>. For each node in the list, it is necessary to judge if there exists a path from the root 1 to the node with ID 5. After identifies all such paths, the counters of these nodes are added up, and frequency 3 is returned.

Apparently, the frequency lookup is very fast, because the operation only needs to check the connectivity of the path from the root sid to each internal node with SID eid, without any further pruning. Note that the fast lookup is enabled by the above-mentioned subsequence extraction: each extracted subsequence starting from sid is inserted to the FP-tree rooted at sid. That is why the lookup only needs to check the connectivity from root sid to the node eid. The time complexity of the frequency lookup is $O(|\text{Tree}|)$, where |Tree| is the number of nodes in an FP-Tree.

4.3.2. Trajectory Complementation. If two segments are disconnected, it is necessary to complement a route between them, such as to linking up all the segments in the entire route. The main idea is to select the most suitable route, in the light of FPs, general MBC, and speed limits. For this purpose, the first step is to find the FP-tree whose root has the SID sid. Then, the dictionary is looked up to find the list of tree nodes whose SIDs are eid. Based on the found tree nodes, the associated paths from the root to the tree nodes are searched for. Then, the paths are sorted by counters and weights, and the most suitable path is returned.

```
Algorithm 2: Complement (Forest F, SID sid, SID eid)
Output: BestPath
$f = 0$; Tree $t \leftarrow$ findTree($F$, $sid$);
$pa$ = [];$path$ = {};
for Node $n$ in $t.index(eid)$ do
    while $n \neq NULL$ do
        $pa.append(n.SID)$;
        $n \leftarrow n.parent()$;
    $pa.reverse()$;
    compute $w(pa)$ by Equation(2);
    $path[w] = pa$;
return $max(path)$;
```

Figure 5: Algorithm 2.

As shown in Figure 5, the trajectory complementation is detailed in Algorithm 2, whose time complexity is $O(|\text{Tree}| * d)$, where d is the depth of an FP tree.

4.4. Redundancy Reduction. Recall that the FP-forest is created based on historical trajectory data. All the subsequences of trajectories are indexed by an FP-forest index, which could cause road segment IDs redundancy. To solve this problem, we propose a set of techniques to optimize the space cost of FP-forest. Our basic idea is to first extract no branch paths from FP-forest, which the tree nodes only contains one child node. Next, we mine the meaningful patterns on the extracted sequences, and then encode a long trajectory by the patterns in order to reduce the redundant road segment IDs, leading to smaller space cost.

As mentioned before, the FP-forest is generated from the data on historical trajectories. All the subsequences of trajectories are indexed by an FP-forest index, which could cause SIDs redundancy. To solve the problem, a set of techniques were proposed to optimize the space cost of FP-forest. Firstly, no branch paths are extracted from FP-forest, where the tree nodes only contain one child node each. Next, meaningful patterns are mined from the extracted sequences, and used to encode a long trajectory. The redundant SIDs are thus reduced, resulting in a decline in space cost.

As shown in Figure 6, five no branch paths $T_1 \ldots T_5$ can be extracted from the FP forest in subgraph 4(a). The paths can be viewed as map trajectories. In subgraph 4(b), the middle column presents the two patterns Pa and Pb mined from the trajectories. Each of them, namely $p_a = \{3, 4\}$, represents 2 consecutively connected edges, that is, a path, in the road network $3 \longrightarrow 4$. Thus, the two edge IDs can be represented by a single virtual SID Pa, thereby reducing the space cost. The rightmost column depicts the encoded trajectories as patterns. For each trajectory, namely, T_2, the redundant edge IDs $\{3, 4\}$ can be replaced by the segment P_a. Next is an introduction to the search for meaningful patterns.

The FPs of subtrajectories indicates the redundancy of partially connected segments. Thus, it is natural to leverage them to reduce redundancy. Inspired by Han et al. [34], a simple and efficient approach was proposed to mine FPs under the constraint of the road network. Considering the adjacent edges in the network, the redundant candidate patterns, i.e., those containing disjoint edges, are removed to improve operating efficiency. Let $M = (V, S)$ be the road network; $\tau = \{T_1, T_2 \ldots T_N\}$ be the set of N trajectories from the mobile objects traveling on M. The following concepts must be defined before finding the FPs.

Definition 1. Let $P = \{T_I/T_i(s) \in \tau\}$ be the base pattern, with s being the segment. Then, the trajectory containing the base pattern can be denoted as $T_i(s)$.

Definition 2. The FP, denoted by $\text{FP} = \{P_1, P_2 \ldots P_n\}$, is an ordered list of base patterns.

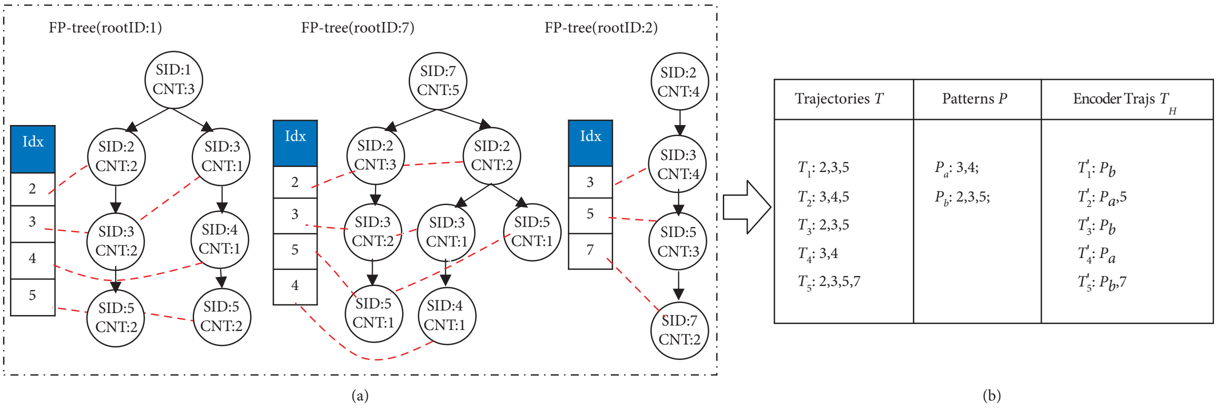
Definition 3. The support $\sup(P)$ of pattern P is the number of the trajectories containing pattern P.

Then, all the frequent subtrajectories can be found through the following steps. Firstly, the segments are organized into base patterns from the map trajectories, and the support of these base patterns (Definition 3) is calculated. Then, the base patterns are sorted in descending order of support. Thus, the generation phase of base pattern outputs an ordered list of base patterns, which serve as the building blocks of FPs in the next phase. Secondly, the base pattern with the maximum support in the said list is selected iteratively, and merged into FPs whose support is above the given lower bound. (If the base patterns are picked randomly, the merged FPs might be filtered out, due to their small supports). After that, the selected base patterns are connected into the FPs based on the road network, and all the frequency subtrajectories are returned.

4.5. Reconstruction of FP-Forest. This subsection improves the effectiveness of FP-forest structure, creating a new structure called FP-forest*. The improvement takes account of the temporal information within trajectories, and the general mobility behavior of users. The FPs intuitively depict the spatial patterns within the trajectories. Therefore, the FP-forest* can represent the temporal information within trajectories and the general mobility behavior. The temporal information, i.e., the timestamps within the trajectories, is taken to estimate the moving speed of the users. Meanwhile, length and transition features of each route are leveraged to infer the general mobility behavior of the users.

As shown in Figure 7, FP-forest* has only two differences from FP-tree: (1) each node in FP-forest* maintains an additional attribute, i.e., the mobility behavior weight Wm about the importance of the path from the current node to the root node; (2) The dictionary record the speed probability distribution of each referred segment. For the FP-tree* rooted at SID 1, the node with SID 2 has a mobility behavior weight of 0.94, and the item with segment SID 2 in the dictionary exists as an equal-width histogram of moving speeds on the associated segment. In this histogram, each equal-width bucket is with a rate of those trajectories that move on the associated segment within the speed interval of the bucket. For simplicity, the same number of buckets (e.g., 8 in Figure 7) is adopted for all histograms in FP-forest*.

The speed histogram is maintained through the following procedure. For each segment in the third-party



Trajectories T	Patterns P	Encoder Trajs T_H
T_1: 2,3,5 T_2: 3,4,5 T_3: 2,3,5 T_4: 3,4 T_5: 2,3,5,7	P_a: 3,4; P_b: 2,3,5;	T_1': P_b T_2': P_a,5 T_3': P_b T_4': P_a T_5': P_b,7

Figure 6: An example of redundancy reduction.

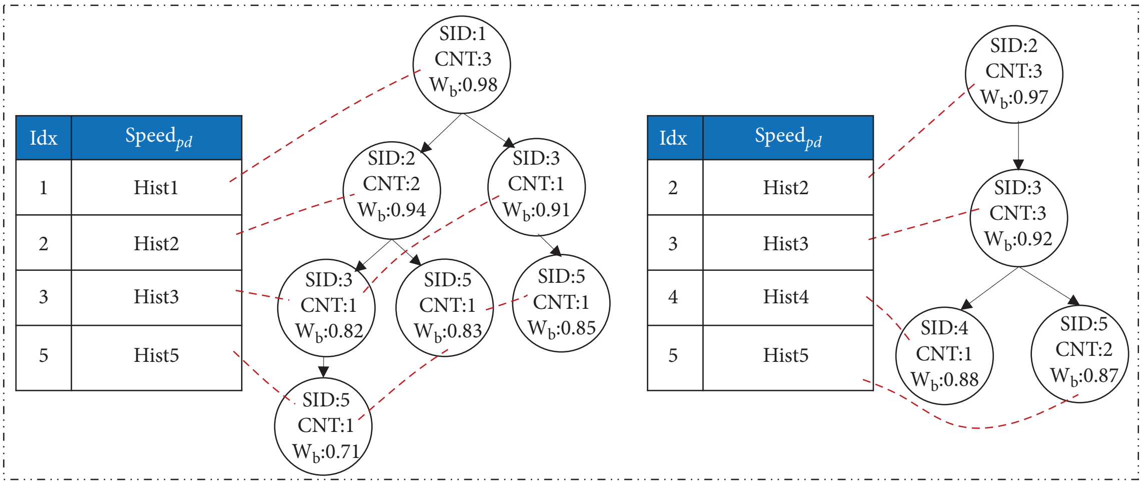


Figure 7: An example of FP-forest*.

database D, there is a set of high-sampling-rate historical trajectories that can be projected onto that segment. Then, it is necessary to compute the mean moving speed of each trajectory passing through the segment. On this basis, a speed histogram can be comfortably maintained based on the moving speeds of all the trajectories passing through that segment.

Once the speed histogram is available for a segment, the probability for an input trajectory to match the segment can be derived from the histogram. In the example of Figure 7, an input trajectory is projected to the segment SID 2, and the mean speed of the trajectory moving on the segment is estimated as 6 m/s. Referring to the speed histogram of SID 2, it can be observed that 6 m/s falls in the bucket interval [4, 8), and the probability 0.2 of the input trajectory matches this segment.

Next is to estimate the weight of mobility behavior. The mobility behavior weight of each node in the FP-forest* can be computed intuitively. Since people generally prefer short and straight routes, the mobility behavior should be weighted according to the distance and transition angle of the route. Let $P = \{s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_k\}$ be the route from a root node to a tree node in the FP-forest*. Then, the mobility behavior weight Wm of node sk can be calculated by:

$$W_m = \frac{(W_{\text{len}} + W_{\text{turn}})}{2},$$
$$W_{\text{len}} = e^{-\left(\sum_{u=1}^{k} s_u.\text{len}/l_{\max}\right)}, \qquad (1)$$
$$W_{\text{turn}} = e^{-\left(\sum_{u=1}^{k-1} \theta_u/\theta_{\max}\right)}.$$

As shown in formula (1), W_m can be computed by taking the average of the weight W_{len} of route length and the weight W_{turn} of the transition angle. The weight W_{len} of route length is the sum of the length su · len of all segments s_u within route. The weight W_{turn} of the transition angle is the sum of the transition angles θ_u from the preceding segments to the current segment. Min-max normalization is performed to unify the vastly different scales of the two sums. Here, $l_{\max}$ (resp · $\theta_{\max}$) indicates the maximal route length (resp. transition angles) from a root node to a tree node within the FP-Forest*. Following the route weights defined above, a

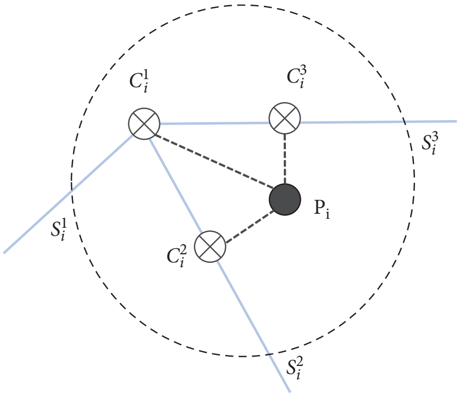


Figure 8: Candidate search.

short and straight route can be selected by reducing W_{len} and W_{turn}, i.e., increasing W_m.

5. FP-Matching Algorithm

In this section, we will present the detail of the proposed FP-matching algorithm with two matching steps: (1) candidate search, and (2) finding a best route.

5.1. Candidate Search. For a given sequence of GPS points $P = \{p_1 \longrightarrow p_2 \longrightarrow \cdots \longrightarrow p_k\}$, a set of candidate segments is searched for each point pi by a radius r $(1 \le n \le i)$. As shown in Figure 8, GPS point p_i is three segments $S1/i$, $S2/i$, $S3/i$ within the search radius r. Then, the distance from p_i to each candidate $Sj/i\,(1 \le j \le 3)$ can be calculated by $\text{dist}\,(p_i, S_i^j) = \min_{\forall c \in S_i^j} \text{dist}\,(p_i, c)$, where c_i^j is the nearest point inside S_i^j to p_i.

To find candidates efficiently, an R-tree index is built for the whole digital map. Then, the top-k nearest segment to p_i can be found in two steps: the first is to search for a set of candidate minimal boundary rectangles (MBR) nearest to p_i. Next is to scan the segments inside (or intersected by) the MBRs. By exploring more MBRs and segments, it is possible to select the top-k nearest segments as candidates.

5.2. Route Optimization. After finding the candidate segments with respect to a sequence of GPS points, it is necessary to derive a route of candidates that best match the entire sequence of GPS points.

Let S_i^j denote the j-th candidate segment of a GPS point p_i; p_i and p_{i+1} be two neighboring GPS points; S_i^j and $S_{i+1}^{j'}$ be the candidate segments associated with the two neighboring GPS points. Then, the two candidate segments S_i^j and $S_{i+1}^{j'}$ might be directly connected or disconnected on the road network. If they are disconnected, the two segments need to be complemented by candidate patterns on FP-forest* (Subsection 4.3). If no FP is available for the complementation, the two segments need to be completed with the best route, under the following weight. Up to now, all candidate segments are linked into an entire route, such as to maximize the cumulative weight of the segments connected in that route. The route weight between two candidate segments, S_i^j and $S_{i+1}^{j'}$, depends on four factors:

(1) $W_{\text{d}}\,(S_i^j, S_{i+1}^{j'})$, the weight of the distance between p_i and candidate segment S_i^j (resp. the distance from S_i^j to $S_{i+1}^{j'}$)
(2) $W_t\,(S_i^j \longrightarrow S_{i+1}^{j'})$, the weight of the moving speed on the route between S_i^j and $S_{i+1}^{j'}$
(3) $W_f\,(S_i^j \longrightarrow S_{i+1}^{j'})$, the weight of the frequency between S_i^j and $S_{i+1}^{j'}$, which can be determined through the lookup operation
(4) $W_m\,(S_i^j \longrightarrow S_{i+1}^{j'})$, the weight of the mobility behavior on the route between S_i^j and $S_{i+1}^{j'}$ which can be obtained in the FP-forest*

However, the previous approaches [11, 12] only consider weight $W_d\,(S_i^j, S_{i+1}^{j'})$, this paper compute the overall route weight $W\,(S_i^j \longrightarrow S_i^{j'})$ between two candidates S_i^j and $S_{i'}^{j'}$, which combines all the four weights above:

$$W\left(S_i^j \longrightarrow S_{i+1}^j\right) = W_f\left(S_i^j \longrightarrow S_{i+1}^j\right) * W_d\left(S_i^j, S_{i+1}^j\right) * W_r\left(S_i^j \longrightarrow S_{i+1}^j\right) * \mathrm{Wm}\left(S_i^j \longrightarrow S_{i+1}^j\right). \tag{2}$$

Following Li et al.'s approach [11], the distance weight $W_d\,(S_i^j, S_{i+1}^{j'})$ is computed under the assumption that the distance between a GPS point and its real position obeys the normal distribution $N = (\mu, \sigma^2)$. Thus, the probability of candidate S_i^j being the correct map match can be estimated by:

$$P\left(S_i^j\right) = \frac{1}{\sqrt{2\pi\sigma}} e^{-[\text{Dist}\,(i,j) - \mu]^2/2\sigma^2}, \tag{3}$$

where $\text{Dist}\,(i,j) = \text{Dist}\,(p_i, S_i^j)$ is the nearest distance between pi and its candidate segment S_i^j. The route $S_i^j \longrightarrow S_{i+1}^{j'}$ involves two candidate segments: S_i^j and $S_{i+1}^{j'}$. Thus, a parameter $\beta \ge 0$ is arranged to mediate the factors with respect to the two candidates S_i^j and $S_{i+1}^{j'}$, and the following weight is defined:

$$W_d\left(S_i^j, S_{i+1}^{j'}\right) = \frac{(1+\beta) * P\left(S_i^j\right) * P\left(S_{i+1}^{j'}\right)}{\beta * P\left(S_i^j\right) + P\left(S_{i+1}^{j'}\right)}, \tag{4}$$

where β is the relative importance of S_i^j over $S_{i+1}^{j'} \cdot \beta = 1.0$ means the two candidates S_i^j and $S_{i+1}^{j'}$ are equally important on the route between them; $\beta < 1.0\,(\text{resp.}\beta > 1.0)$ means S_i^j is $\text{less}\,(\text{resp} \cdot \text{more})$ important than $S_{i+1}^{j'}$ on the route.

To compute $W_t\,(S_i^j \longrightarrow S_{i+1}^j)$, it is necessary to estimate the mean moving speed on the route between S_i^j and S_{i+1}^j. Here, two candidate segments S_i^j and $S_{i+1}^{j'}$ are given for two neighboring GPS sampling points p_i and p_{i+1}, respectively. With the aid of the FP-forest*, it is possible to obtain the route $P * \,(S_i^j, S_{i+1}^{j'})$ from S_i^j to $S_{i+1}^{j'}$. On this basis, the mean speed $\overline{V}_{(i,j) \longrightarrow (i+1,j')}$ of $P * \,(S_i^j, S_{i+1}^{j'})$ can be calculated by

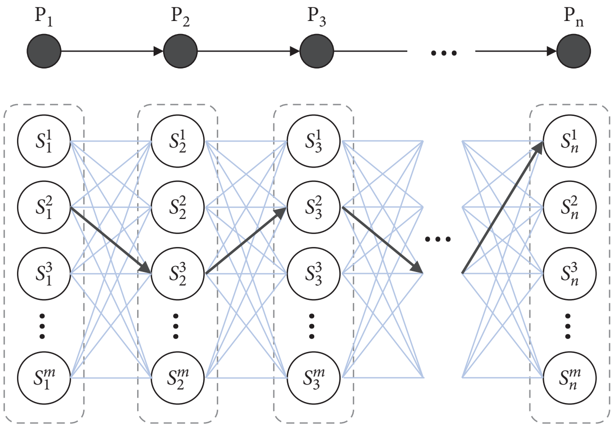


Figure 9: Route optimization. Note: the solid black circles are raw GPS points $p_1 \ldots p_4$; the hollow black circles under the solid black circles are candidate segments.

$$\overline{V}_{(i,j)\longrightarrow(i+1,j_{\prime})} = \frac{w_{(i,j)\longrightarrow(i+1,j')}}{\Delta t_{i,i+1}}, \quad (5)$$

where $w_{(i,j)\longrightarrow(i+1,j')}$ is the traversed distance along route $P*(S_i^j, S_{i+1}^{j'})$; $\Delta t_{i,i+1} = p_i \cdot t - p_{i+1} \cdot t$ is the time interval between p_i and p_{i+1}. Based on the mean speed $\overline{v}_{(i,j)\longrightarrow(i+1,j')}$, the probability $P_{S_i^j}(\overline{v}_{(i,j)\longrightarrow(i+1,j')})\text{resp} \cdot P_{S_i^{j'}}(\overline{v}_{(i,j)\longrightarrow(i+1,j')})$ of mobile devices moving on $S_i^j(\text{resp} \cdot S_{i+1}^{j'})$ with the speed $\overline{v}_{(i,j)\longrightarrow(i+1,j')}$ can be inferred from the speed histogram with respect to the candidate segments $S_i^j(\text{resp} \cdot S_{i+1}^{j'})$. After that, the speed weight can be calculated by

$$W_t\left(S_i^j \longrightarrow S_{i+1}^{j'}\right) = P_{S_i^j}\left(\overline{v}_{(i,j)\longrightarrow(i+1,j')}\right) * P_{S_{i+1}^{j'}}\left(\overline{v}_{(i,j)\longrightarrow(i+1,j')}\right). \quad (6)$$

Furthermore, the frequency weight $W_t(S_i^j \longrightarrow S_{i+1}^{j'})$ can be calculated through min-max normalization, such that the frequency weight is linearly proportional to the frequency of historical routes from S_i^j to $S_{i+1}^{j'}$, i.e., $\text{Freq}(S_i^j \longrightarrow S_{i+1}^{j'})$:

$$W_f\left(S_i^j \longrightarrow S_{i+1}^{j'}\right) = \frac{\text{Freq}\left(S_i^j \longrightarrow S_{i+1}^{j'}\right) - \text{Freq}_{\min}}{\text{Freq}_{\max} - \text{Freq}_{\min}}, \quad (7)$$

where $\text{Freq}_{\max}(\text{resp} \cdot \text{Freq}_{\min})$ is the maximum (resp. minimum) frequency from a root to a tree node within FP-forest*.

The calculation of mobility behavior weight $W_m(S_i^j \longrightarrow S_{i+1}^{j'})$ is already detailed in the preceding subsections.

Until now, the FP-matching problem is to find a route with the highest overall route weight (e.g., the black line in Figure 9), such that the sum of all edge weights is maximized among all optional routes. Formally, the best matching route P of segments can be defined as

$$P = \arg\max \sum_{i=1}^{n-1} W\left(S_i^{\text{best}_i} \longrightarrow S_{i+1}^{\text{best}_{i+1}}\right). \quad (8)$$

Our purpose is to find one route with the highest overall weight in a directed acyclic graph (DAG). As shown in Figure 10 Algorithm 3 shows how to find the best route through dynamic programming. After the algorithm outputs the sequence of matching segments, it is necessary to judge whether these segments are disconnected within the sequence. If some of them are disconnected, the full route must be recovered through the complementation operation in Algorithm 2.

```
Algorithm 3: FindBestRoute (DAG: $(S_1^1 \ldots S_1^k) \rightarrow \ldots \rightarrow (S_n^1 \ldots S_n^{k'})$, Forest $F$)

Output: Best Route $P{:}S_1^{best_1} \rightarrow \ldots \rightarrow S_n^{best_n}$
Let f[] denote the route weight computed so far;
Let pre[] denote the previous of current candidate;
for $i = 1; i \le k; i++$ do $f[S_1^i] = 0$;
for $i = 2; i \le n; i++$ do
    for $j = 1; j \le k; j++$ do
        $max = -1$;
        for $l = 1; l \le k; l++$ do
            $temp = f[S_{i-1}^l] + W(S_{i-1}^l \rightarrow S_i^j)$;
            if $max \le temp$ then $max = temp$; pre$[S_i^j] = S_{i-1}^l$;
            $f[S_{i-1}^j] = max$;
$p = []$; $cur = argmax_{s_n^x} f[s_n^x]$;
for $i = n; i \ge 2; i--$do{ $P$.append($cur$); $cur$ = pre[$cur$]};
$P$.append($cur$); $P$.reverse();
return $P$;
```

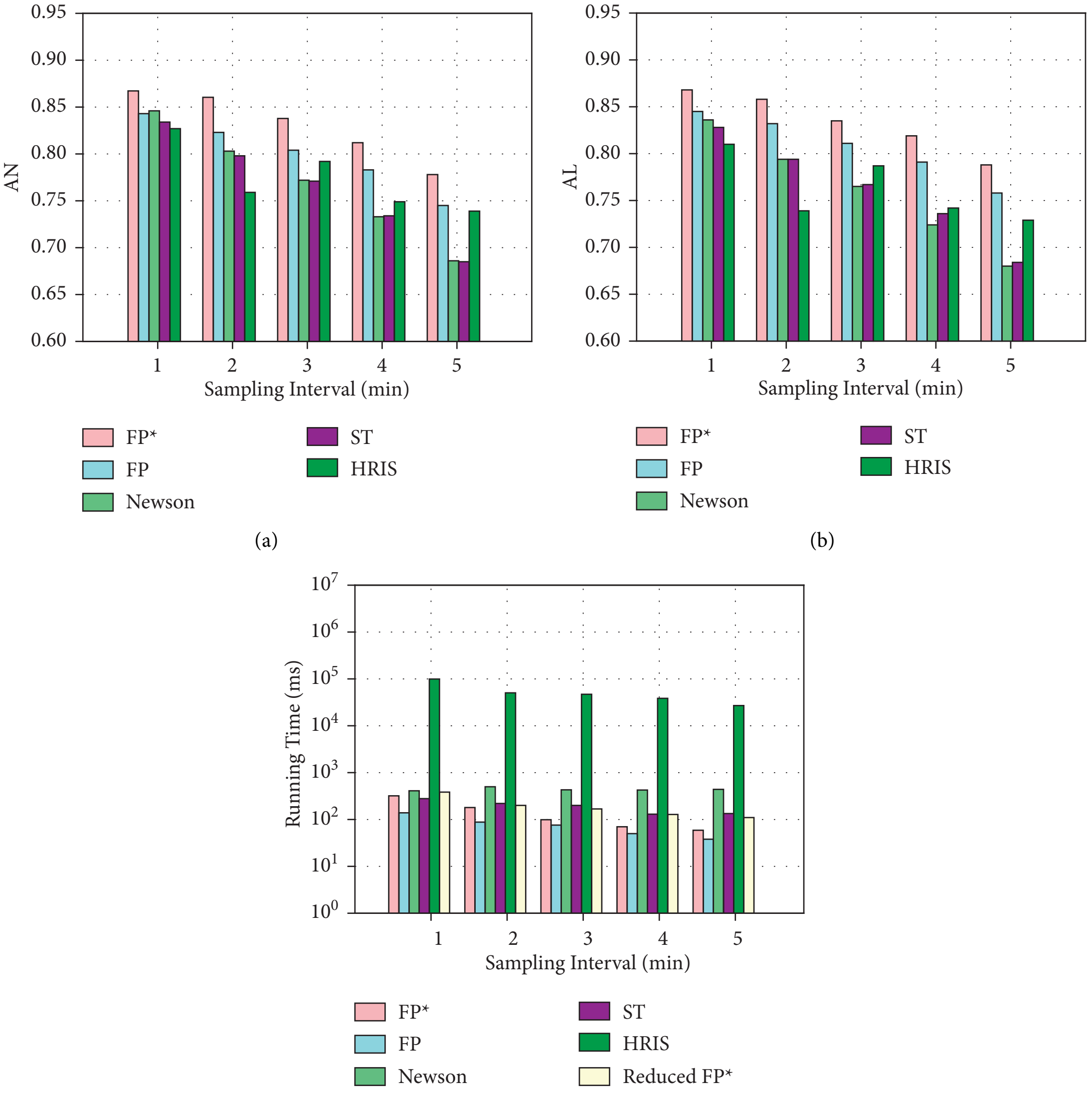
Figure 10: Algorithm 3.

Depending on the specific input DAG, the time complexity of Algorithm 3 is $O(n*m^2)$, where n is the number of GPS points within the input trajectory, and m is the mean number of candidate segments in DAG.

6. Evaluation

6.1. Datasets, Contrastive Algorithms, and Metrics. Our experiments were carried out on two datasets. The first dataset is the digital map of Shanghai road network, extracted from OpenStreetMap. The dataset contains 146,804 vertices and 95,950 edges. The second dataset, known as TaxiData, covers 92,602 taxi trajectories in one day. Each GPS sample contains a timestamp, occupied/empty state, GPS longitude and latitude, speed, and direction. For the taxi trajectories, the GPS sampling rate is as high as 1 point per 10 seconds. The candidate segments were searched for mainly based on GPS longitude and latitude. With help of the timestamp, GPS points were sampled from the trajectories to simulate various sampling rates.

The ground truth of the experiments, i.e., the routes with high accuracy, was obtained by applying classic map-matching algorithms to the original high-sampling-rate trajectories. Based on the occupied/empty state, each long trajectory of a taxi were divided into multiple short subtrajectories, each of which represents one route of a passenger. The division makes sense, because the subtrajectories with passengers directly indicate the travel route from sources to destinations. Then, a FP-forest was established based on the subtrajectories. A total of 155,725 trajectories were randomly selected to build the forest, and 337 were chosen to evaluate the map-matching approaches. Note that historical trajectories involve GPS points of high sampling rate; with help of



(a)

(b)

(c)

Figure 11: Effect of sampling intervals: AN, AL and running time (from left to right).

the timestamp, this paper samples GPS points within testing trajectories to simulate various sampling intervals from 1 to 5 mins.

For comparison, our FP-matching algorithm was contrasted against three approaches: HMM-matching [12], ST-Matching [11], and HRIS [15]. The performance of each map-matching approach was evaluated against the following metrics about matching accuracy and running time.

6.1.1. *Accuracy by Number A_N.*

The accuracy by number is the ratio of the number of correctly matched segments to the total number of segments within the trajectories.

6.1.2. *Accuracy by Length A_L.*

The accuracy by length is the ratio of the length of matched segments to the total length of all segments within the trajectories.

6.1.3. *Route Mismatch Fraction (RMF).*

Let d_+ and d be the total length of incorrectly matched segments added to that of the correct route d_0 and the total length of incorrectly matched segments subtracted from d_0, respectively. Then, the summation between d_+ and d is the total mismatched distance. The RMF is defined as the fraction of $(d_+ + d_-)/d_0$, which quantifies the map-matching error, i.e., RMF = $(d_+ + d_-)/d_0$.

6.1.4. *Minimum Fréchet Distance (MFD).*

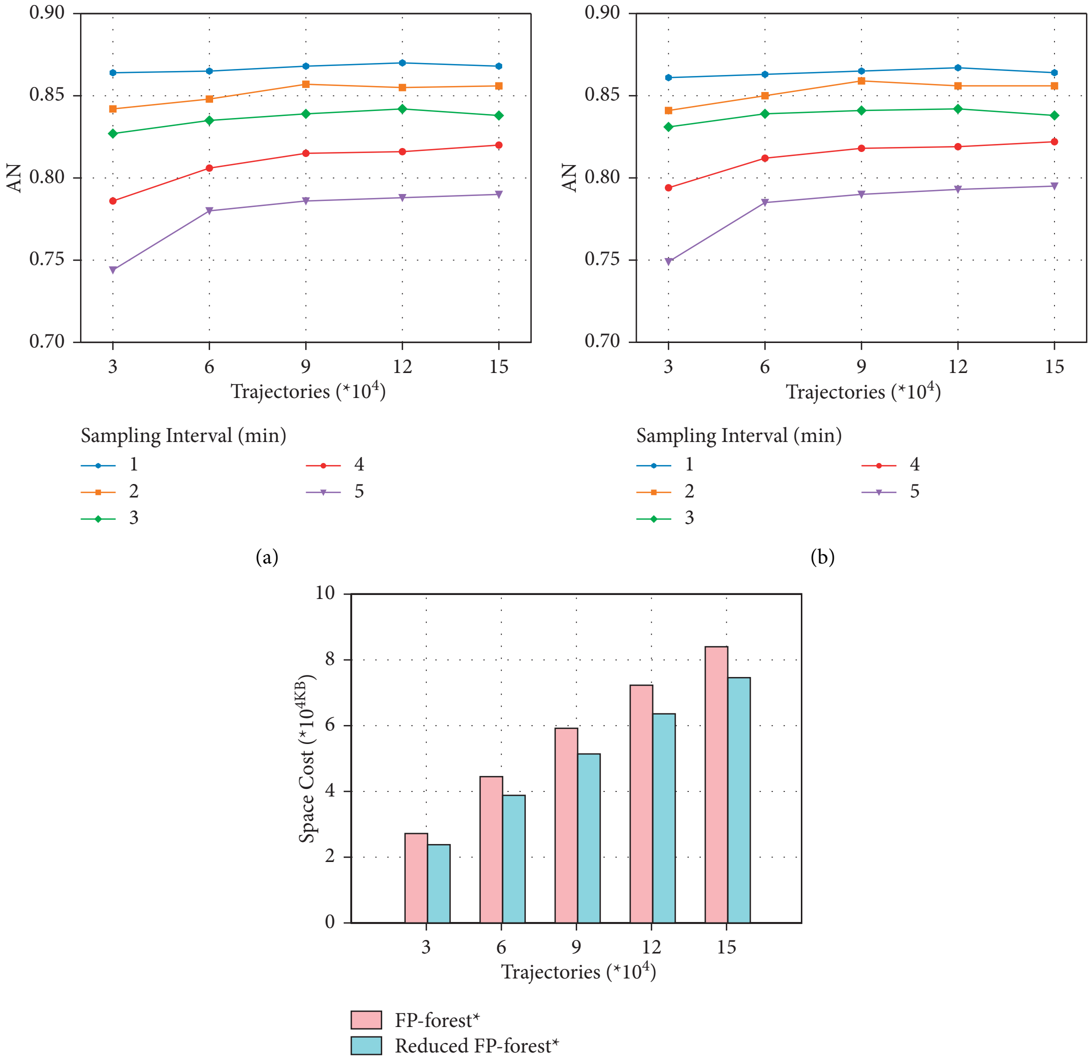
To evaluate the correctness of the matched path, the MFD was computed between the true path and the matched path. The smaller the MFD, the greater the similarity between the two paths.

6.2. *Baseline Experiment.*

The baseline experiment aims to evaluate the accuracy of five approaches, namely, FP*-matching, FP-matching, ST-matching [11], HMM-matching

Table 1: Effect of sampling intervals from 1, 2, 3, 4 to 5 (mins).

	RMF					MFD (m)				
	1	2	3	4	5	1	2	3	4	5
HMM [20]	0.21	0.29	0.35	0.43	0.5	354.03	383.37	436.35	495.09	569.61
ST [19]	0.16	0.28	0.36	0.44	0.49	314.91	398.03	477.64	517.34	586.18
HRIS [31]	0.25	0.32	0.31	0.39	0.45	379.42	427.48	411.32	458.46	534.42
FP	0.18	0.26	0.33	0.38	0.42	296.21	368.34	390.97	451.66	513.67
FP*	0.16	0.25	0.29	0.36	0.40	285.16	349.77	375.13	416.40	473.53



(a)

(b)

(c)

Figure 12: Effect of historical trajectories and sampling rate on map-matching accuracy and space cost (from left to right).

[12], and HRIS [15] at various sampling rates. Note that FP*-matching (resp. FP-matching) indicates the map-matching algorithm using FP*-forest (resp. FP-forest).

As shown in Figure 11 (left and middle), the accuracy of every algorithm declined with the expansion of the sampling interval, i.e., the reduction of sampling rate. Among the five approaches, FP*-matching and FP-matching achieved the best performance, while ST-matching and HMM-matching exhibited similar trends. After the interval of 3 mins, ST-matching and HMM-matching witnessed drastic degradation of accuracy, while FP*-matching and FP-matching still performed the best. This means FP*-matching works excellently on very sparse GPS points. In addition, HRIS operated

Table 2: Reliability of mobility behavior weight.

Distance (m)	200	400	600	800	1000	1200
Proportion	0.999	0.975	0.952	0.924	0.899	0.852

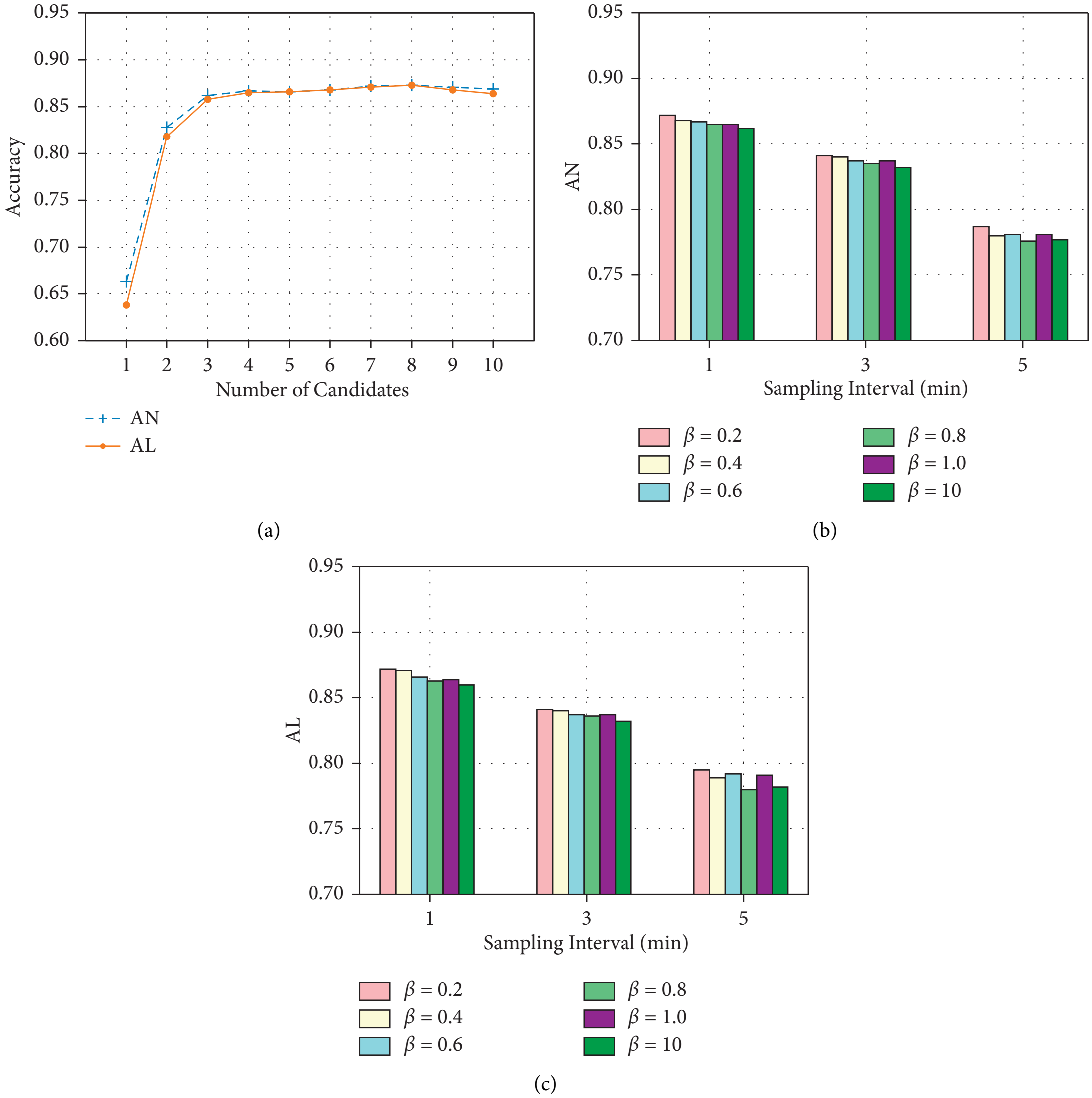


Figure 13: Effect of candidate segments, AN and AL at various sampling rates and β (from left to right).

well on sparse trajectories, when the sampling interval was greater than 3 mins. This is because HRIS can exploit historical trajectories to improve the map-matching precision for the input trajectories, even if the sampling rate is very low.

Next, the five map-matching approaches were compared in terms of running time. As shown in Figure 11 (right), all approaches realized a shorter running time, with the expansion of the sampling interval, i.e., the increase of sparse GPS samples. Among them, HRIS had the longest running time at any sampling interval. The result is consistent with the findings of Zheng et al. [15]. FP*-matching generally outperformed HMM-matching [12] and ST-matching [11], for the latter two consume too much time in computing the shortest path. Besides, FP*-matching consumed a slightly longer running time than FP-matching, for a few time is required for the redundancy reduction in FP-forest*.

Table 1 compare the map-matching effectiveness of the five approaches in terms of RMF and MFD. With the increase of sampling interval, i.e., the growing number of sparse GPS samples, the RMF and MFD rose for all five approaches. Among them, FP*-matching and FP-matching consistently outperformed the contrastive approaches.

6.3. *Sensitivity Analysis.*

This subsection analyzes the sensitivity of FP-matching to several key parameters.

6.3.1. *Effect of Historical Data.*

The size of the historical trajectory dataset D was varied from $3 * 10^4$ to $15 * 10^4$ to test the effect of D on map-matching performance. As shown in Figure 12, when the sampling rate was fixed, the map-matching accuracy, as measured by A_N and A_L, increased with the number of historical trajectories; when the number of trajectories was fixed, better accuracy was achieved with the reduction of sampling interval, i.e., the growing density of GPS samples.

The authors are also interested in the space of FP-forest and the one by redundancy reduction. As shown in Figure 12 (right), when the size D grew, the space cost of FP-forest* was clearly smaller than that of the original FP-forest.

6.3.2. Reliability of Mobility Behavior Weight. In Subsection 4.5, the mobility behavior weight Wm is designed for a tree node in FP-forest*, based on people's preference for short, straight paths within a certain distance. To verify the reliability of this weight, 10,000 source-destination pairs were randomly sampled from the GPS trajectory database at different distances. For each sampled pair, the authors calculated the trajectories that traverse the shortest and straightest path as a proportion of all trajectories passing through these two points. The shortest path was measured by Euclidean distance. As shown in Table 2, the said proportion was greater than 0.92, when the two points were less than 800 m apart. The proportion was still as high as 0.852, when the distance between the two points grew to 200 m. Hence, the proposed mobility behavior weight is rather reliable.

6.3.3. Effect of Candidate Segments. In subsection 5.1, the candidate segments are searched for to perform map-matching. Here, the number of candidate segments is varied to test the matching accuracy of FP*-matching. As shown in Figure 13, more candidates led to a higher accuracy for both A_N and A_L. As the number of candidates increased from 1 to 8, the map-matching accuracy surged up to the peak at 3 candidates, and remained stable thereafter.

6.3.4. Effect of Parameter β. In formula (6), the road distance weight W_d employs a parameter β to tune the importance of two neighboring candidate segments. As shown in Figure 13 (middle and right), when the β was fixed, the greater the sampling interval (the more the sparse GPS samples), the lower the map-matching accuracy; when the sampling interval was fixed, the map-matching accuracy increased with the decrease of β. These trends can be explained as follows:

For example, taxi drivers decide the route mainly based on the current position. That is, they choose the next segments leading to the final destination, in view of the traffic situation around the current position. As a result, it makes sense for FP-matching to perform better at a smaller β, which leads to a higher weight for the source segment in the route $S_i^j \longrightarrow S_{i'}^{j'}$.

7. Conclusions

To solve the problem of map-matching for low-sampling-rate trajectories, this paper mines the FPs out of a huge amount of historical taxi routes, extracts the MBC, and proposes a novel matching algorithm. The proposed algorithm has a high map-matching accuracy, as it incorporates the weights of the distance from GPS points to road segments, moving speeds, trajectory frequency and mobility behaviors on the segments. To speed up the map-matching, the identified FPs were indexed by the FP-forest structure. Then, redundant paths were pruned within the indexing structure to save space. Comparative experiments demonstrate that our algorithm outperformed existing HMM-based and historical data-based algorithms in terms of matching accuracy and running time. The future work plans to extend our algorithm online, and develop a map-matching approach by the popular deep neural networks (DNNs).

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

A shorter conference version of this paper appeared in 19th IEEE International Conference on Mobile Data Management Conference. The map matching algorithm in initial conference paper is relatively simple, so there is still a lot of room for improvement in the research method. Later, their new team continued to further study the algorithm in the paper and finally proposed a more advanced map matching algorithm.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors thank the authors of the conference paper and General Scientific Research Project of Zhejiang Provincial Department of Education in 2020 (Y202043149).

References

[1] A. M. Pinto, A. P. Moreira, and P. G. Costa, "A localization method based on map-matching and particle swarm optimization," *Journal of Intelligent and Robotic Systems*, vol. 77, no. 2, pp. 313–326, 2015.

[2] X. B. Chen, L. Zhao, Y. Hao, L. H. Yu, and C. C. Lv, "An evaluation algorithm for the interoperability of global navigation satellite systems," *Traitement du Signal*, vol. 37, no. 1, pp. 137–144, 2019.

[3] J. Guo, X. Li, Z. Zhang, and J. Zhang, "Traffic flow fluctuation analysis based on Beijing taxi GPS data," *Knowledge Science, Engineering and Management*, in *Proceedings of the Knowledge Science, Engineering and Management - 11th International Conference*, pp. 452–464, KSEM 2018, Changchun, China, August 2018.

[4] X. Li, J. Han, J. Lee, and H. Gonzalez, "Traffic density-based discovery of hot routes in road networks," in *Proceedings of the Advances in Spatial and Temporal Databases, 10th International Symposium, SSTD 2007*, pp. 441–459, Springer, Boston, MA, USA, July 2007.

[5] L. Liao, D. Fox, and H. A. Kautz, "Hierarchical conditional random fields for gps-based activity recognition," in *Proceedings of the Robotics Research: Results of the 12th International Symposium*, pp. 487–506, ISRR 2005, San Francisco, CA, USA, October-2005.

[6] K. Zhao, M. Musolesi, P. Hui, W. Rao, and S. Tarkoma, “Explaining the power-law distribution of human mobility through transportationmodality decomposition,” *Scientific Reports*, vol. 5, no. 1, p. 9136, 2015.

[7] A. Ali, M. Hegaze, and A. Elrodesly, “In-flight correction of the satellite orientation parameter during target mode,” *Mathematical Modelling of Engineering Problems*, vol. 6, no. 2, pp. 249–262, 2019.

[8] H. Alt, A. Efrat, G. Rote, and C. Wenk, “Matching planar maps,” in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 589–598, ACM, Baltimore, MD, USA, 12 January 2003.

[9] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, “On map-matching vehicle tracking data,” in *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 853–864, ACM, Trondheim, Norway, August 2005.

[10] Y. Huang, W. Rao, and Z. Zhang, “Frequent pattern-based map-matching on low sampling rate trajectories,” in *Proceedings of the 19th IEEE International Conference on Mobile Data Management*, pp. 266–273, IEEE, Aalborg, Denmark, June 2018.

[11] Y. Lou, C. Zhang, and Y. Zheng, “Map-matching for low-sampling-rate GPS trajectories,” in *Proceedings of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, pp. 352–361, ACM-GIS 2009, Seattle, WA, USA, November 2009.

[12] P. Newson and J. Krumm, “Hidden Markov map matching through noise and sparseness,” in *Proceedings of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, pp. 336–343, ACM-GIS 2009, Seattle, WA, USA, November 2009.

[13] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland, “A general map matching algorithm for transport telematics applications,” *GPS Solutions*, vol. 7, no. 3, pp. 157–167, 2003.

[14] C. E. White, D. Bernstein, and A. L. Kornhauser, “Some map matching algorithms for personal navigation assistants,” *Transportation Research Part C: Emerging Technologies*, vol. 8, no. 1–6, pp. 91–108, 2000.

[15] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, “Reducing uncertainty of low-sampling-rate trajectories,” in *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE 2012)*, pp. 1144–1155, Washington, DC, USA, April, 2012.

[16] F. Zhu, C. Luo, and M. Yuan, “City-scale localization with telco big data,” in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pp. 439–448, ACM, Indianapolis, USA, October 2016.

[17] F. Zhu, M. Yuan, and X. Xie, “A data-driven sequential localization framework for big telco data,” *IEEE Transactions on Knowledge and Data Engineering*, p. 1, 2019.

[18] Z. Li, K. Liu, Y. Zhao, and Y. Ma, “MaPIT: an enhanced pending interest table for NDN with mapping bloom filter,” *IEEE Communications Letters*, vol. 18, no. 11, pp. 1915–1918, 2014.

[19] Z. Li, L. Song, and H. Shi, “Approaching the capacity of k-user MIMO interference channel with interference counteraction scheme,” *Ad Hoc Networks*, vol. 58, pp. 286–291, 2017.

[20] Y. Liu and Z. Li, “A novel algorithm of low sampling rate GPS trajectories on map-matching,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, pp. 1–5, 2017.

[21] H. Wei, Y. Wang, G. Forman, Y. Zhu, and H. Guan, “Fast viterbi map matching with tunable weight functions,” in *Proceedings of the SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS)*, pp. 613–616, ACM, Redondo Beach, CA, USA, November 2012.

[22] T. Eiter and H. Mannila, “Computing Discrete Fréchet Distance,” Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Vienna, Austria, 1994.

[23] J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent pattern mining: current status and future directions,” *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp. 55–86, 2007.

[24] J. Pei, J. Han, and B. Mortazavi-Asl, “Prefifixspan: mining sequential patterns by prefix-projected growth,” in *Proceedings of the 17th International Conference on Data Engineering*, pp. 215–224, IEEE, Heidelberg, Germany, 2 April 2001.

[25] F. Huang and N. Zheng, “A novel frequent pattern mining algorithm for real-time radar data stream,” *Traitement du Signal*, vol. 36, no. 1, pp. 23–30, 2019.

[26] R. Srikant and R. Agrawal, “Mining sequential patterns: generalizations and performance improvements,” in *Proceedings of the Advances in Database Technology - EDBT’96, 5th International Conference on Extending Database Technology*, pp. 3–17, Avignon, France, March 1996.

[27] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, “New algorithms for fast discovery of association rules,” in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pp. 283–286, Newport Beach, CA, USA, August 1997.

[28] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” *ACM SIGMOD Record*, in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, vol. 29, no. 2, pp. 1–12, ACM, Dallas, Texas, USA, May 2000.

[29] M. J. Zaki, “SPADE: an efficient algorithm for mining frequent sequences,” *Machine Learning*, vol. 42, no. 1-2, pp. 31–60, 2001.

[30] S. Lin, Y. Chen, D. Yang, and J. Wu, “Discovering long maximal frequent pattern,” in *Proceedings of the Eighth International Conference on Advanced Computational Intelligence, ICACI 2016*, pp. 136–142, IEEE, Chiang Mai, Thailand, February 2016.

[31] R. Prabamanieswari, “A combined approach for mining fuzzy frequent itemset,” *International Journal of Computer Applications*, vol. 975, p. 8887, 2013.

[32] C. M. Kuok, A. Fu, and M. H. Wong, “Mining fuzzy association rules in databases,” *ACM SIGMOD Record*, vol. 27, no. 1, pp. 41–46, 1998.

[33] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules in large databases,” in *Proceedings of the VLDB’94, Proceedings of 20th International Conference on Very Large Data Bases*, pp. 487–499, Morgan Kaufmann Publishers Inc., Santiago de Chile, Chile, September 1994.

[34] B. Han, L. Liu, and E. Omiecinski, “Road-network aware trajectory clustering: integrating locality, flow, and density,” *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 416–429, 2015.