Network-Matched Trajectory-Based Moving-Object Database: Models and Applications

Zhiming Ding, Bin Yang, Ralf Hartmut Güting, and Yaguang Li

***Abstract*—Tracking and managing the locations of moving objects are essential in modern intelligent transportation systems (ITSs). However, a number of limitations in existing methods make them unsuitable for real-world ITS applications. In particular, Euclidean-based methods are not accurate enough in representing locations and in analyzing traffic, unless the locations are frequently updated. Network-based methods require either digital maps to be installed in moving objects or transmission of prediction policies, which inevitably increase the cost. To solve these problems, we propose a *network-matched trajectory-based moving-object database* (NMTMOD) mechanism and a traffic flow analysis method using the NMTMOD. In the NMTMOD, the locations of moving objects are tracked through a dense sampling and batch uploading strategy, and a novel edge-centric network-matching method, which is running at the server side, is adopted to efficiently match the densely sampled GPS points to the network. In addition, a deviation-based trajectory optimization method is provided to minimize the trajectory size. Empirical studies with large real trajectory data set offer insight into the design properties of the proposed NMTMOD and suggest that the NMTMOD significantly outperforms other mobile-map free-moving-object database models in terms of precision of both location tracking and network-based traffic flow analysis.**

***Index Terms*—Moving-object database, network-matched trajectory, spatiotemporal database, traffic flow analysis.**

I. Introduction

ADVANCES in wireless communication and positioning technologies make tracking and managing the dynamic locations of moving objects in databases [also known as moving-object databases (MODs)] a key research issue. To describe the time-dependent locations and spatial extensions of moving objects, e.g., vehicles, ships, hurricanes, oil spills, MODs employ nonconventional data types including moving points, moving lines, and moving regions [1]. Among these, moving points, e.g., representing vehicles, are the most commonly used in real-world applications. Moving points are also known as "trajectories," which are sampled and generated through location updates.

Managing trajectories are crucial in modern intelligent transportation systems (ITS). Given the trajectory of a vehicle, its complete moving procedure, i.e., its location at any given time instant, is available. In addition, given the trajectories of massive vehicles, complicated functionalities such as traffic flow analysis and driving pattern analysis can be supported [2].

Trajectories can be classified into three categories: Euclidean-based trajectories [1], [3]–[6], network-based trajectories [7]–[11], and network-matched trajectories. A Euclidean-based trajectory is a sequence of sampled GPS points. The locations of the moving object at any time can be derived through Euclidean-based interpolations. However, the Euclidean-based interpolations may cause errors because the underlying transportation networks are ignored. Fig. 1(a) shows an example that a moving object moves along a curvilinear road. The interpolated position from two sampled locations considerably deviates from the actual position of the moving object. To avoid this, it requires more frequent location updates from moving objects. However, this can dramatically increase the communication cost.

A network-based trajectory is a sequence of timestamped network motion vectors. The network motion vectors are derived by matching the sampled GPS points to the transportation network through network-based location update mechanisms [7], [9], and [10]. The locations of other than the sampling time instants can be derived through network-based interpolations. In Fig. 1(b), there are two possible paths between two sampled GPS points sampling_1 and sampling_2. Normally, the shortest path, (e_1, e_2, e_5), is taken during network matching. However, it may introduce an error when the driver actually follows a different path (e_1, e_3, e_4, e_5), perhaps because the shortest path is blocked by a traffic accident.

The network-based trajectory model requires mobile maps and powerful mobile computing platform to perform network matching on the moving object side. In this paper, we call the maps installed on the moving object side and on the server side as "mobile map" and "server map," respectively. Moreover, all mobile maps should be updated whenever the server map changes due to the changes of the real-world transportation network. This significantly increases the cost and reduces the flexibility of the system considering the fact that, in many countries, e.g., China, most GPS-equipped vehicles do not have digital maps installed, and upgrading them is quite costly.

Manuscript received June 17, 2014; revised October 9, 2014 and December 9, 2014; accepted December 12, 2014. Date of publication January 21, 2015; date of current version July 31, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 91124001, by the National High Technology Research and Development Program of China (863 program) under Grant 2013AA01A603, and by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06010600. The Associate Editor for this paper was F.-Y. Wang.

Z. Ding is with the College of Computer Science, Beijing University of Technology, Beijing 100124, China (e-mail: zmding@bjut.edu.cn).

B. Yang is with the Department of Computer Science, Aalborg University, 9220 Aalborg, Denmark (e-mail: byang@cs.aau.dk).

R. H. Güting is with the Department of Mathematics and Computer Science, University of Hagen, 58084 Hagen, Germany (e-mail: rhg@fernuni-hagen.de).

Y. Li is with the Institute of Software, the Chinese Academy of Sciences, Beijing 100190, China (e-mail: yaguang@nfs.iscas.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2014.2383494

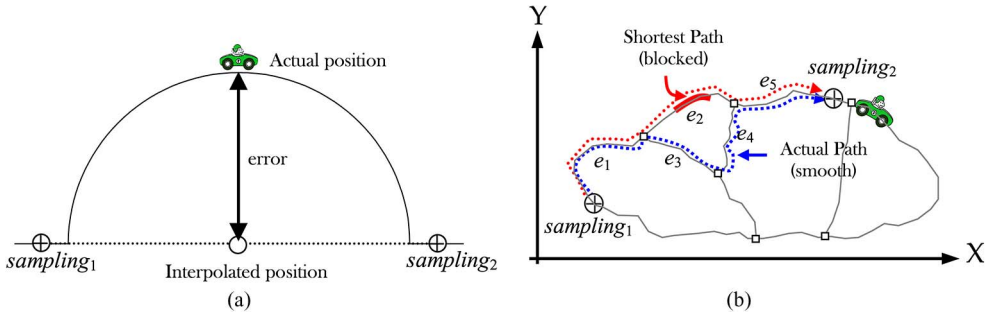


Fig. 1. Euclidean-based and network-based trajectories. (a) Euclidean interpolation errors. (b) Network-matching errors.

Solutions without a mobile map exist [10], [12]. Instead, the server sends either the current road information, e.g., the geometry of the current road represented by polyline, or the prediction policy to moving objects. Such solutions work fine when the roads on which the moving objects traverse do not change frequently, e.g., traveling on highways or in a sparse road network. However, these solutions are not effective when traveling in dense urban areas. The server needs to send either the road information or the prediction policy whenever the moving objects traverse to new roads. Consequently, the communication cost is large. Further, these solutions are vulnerable to communication failure. These methods rely on the predicted location information of moving objects, and the server assumes the moving objects traverse on the predicted routes as long as it does not receive a location update. However, in the case of communication failure which often happens when moving objects run across high buildings, moving objects' actual positions can be miles away from the predicted ones as the moving objects are unable to make location updates.

To solve the given problems, we propose a network-matched trajectory-based moving-object database (NMTMOD) and present a traffic flow analysis method using the NMTMOD. A network-matched trajectory is composed of a sequence of network motion vectors and a network path describing the route of the moving object. In the NMTMOD, the installation of mobile maps is not required. The locations of moving objects are tracked through a dense sampling and batch uploading (DSBU) method, and the network matching is conducted at the server side. To reduce the storage size of network-matched trajectory, the server discards unimportant samplings so that only key information is included in the trajectories.

This paper makes the following contributions. First, a mobile-map-free NMTMOD mechanism is proposed, which adopts a DSBU location update strategy and manages to obtain highly precise, storage-optimized, and network-matched trajectories. Second, an edge-centric network-matching algorithm is proposed, which is suitable for efficiently matching densely sampled GPS points to the network with high precision. Third, a network-matched trajectory-based traffic flow analysis method is presented, Taking advantage of network-matched trajectories, this method manages to achieve higher precision than existing GPS-based traffic flow analysis methods.

The remainder of this paper is organized as follows. Section II covers related work, and Section III formally defines the traffic-parameterized network and network-matched trajectories. The generation of network-matched trajectories is presented in Section IV. The traffic flow statistical analysis algorithms are described in Section V, and the performance evaluation results are provided in Section VI. Finally, conclusion and future work are summarized in Section VII.

II. Related Work

Moving-Object Databases: MODs can be classified into Euclidean-based MODs (EMODs) and network-based MODs (NMODs). Earlier work on MODs mainly focuses on Euclidean-based solutions. Güting *et al.* presented a data model for managing moving objects based on abstract data types and query operators [1]. Wolfson *et al.* propose a moving-object spatiotemporal (MOST) model, which is capable of tracking not only the current but also the near future positions of moving objects [3], [4]. A few studies [13]–[15] explored traffic analysis based on Euclidean-based trajectories. However, as discussed in Section I, Euclidean-based methods suffer from low precision in location tracking and traffic analysis.

Recently, increasing research interests have focused on network-based moving objects. In [7], Güting *et al.* proposed a traffic-network-based MOD model containing a rich set of predefined data types and operations. Speicys *et al.* described a computational data model for network-constrained moving objects [9]. The modeling for dynamic transportation networks and network-constrained moving objects is discussed in [8]. In addition, indexing methods for network-constrained moving objects are investigated in [10] and [11]. The location tracking strategies for network-constrained moving objects are explored in [6], [12], and [16]. Compared with Euclidean-based solutions, network-based solutions are more precise in location representation and more efficient in terms of storage, location updates, and indexing. The common drawback of network-based methods is that the tracking of network-constrained moving objects usually needs mobile maps installed at the moving object side, which can significantly increase the cost of the whole system. From the earlier analysis, a mobile-map-free NMOD model is desired.

Network-Matching Methods: Network-matching (also known as map matching) is a process of aligning a sequence of sampled positions with the underlying transportation network. It is an essential step to generate network-matched trajectory and network-matching methods can be categorized into three groups [17]: local methods [18], [19], global methods [17],

[20], [21], and statistical methods [22]. The local methods find local matches of geometries and perform matching based on the previous matching result of a point. The global methods aim to find global optimal matching paths for entire trajectories. The algorithm in [21] is based on Frechét distance and its variant. In [17], an algorithm was proposed, which takes temporal information into consideration and manages to get higher accuracy for low-sampling-rate GPS trajectories. Statistical method is also widely used, A method based on the Bayesian classifier is presented in [23]. In [22], an algorithm was proposed that takes advantage of the rich information extracted from the historical trajectories to infer the real routes. Two network-matching methods [24] and [25], which utilize the multihypothesis technique (MHT), are proposed. However, the MHT-based methods have very high computational complexity, The aforementioned studies are more focused on the accuracy than the efficiency, which makes them unsuitable for processing massive trajectory data. Moreover, these methods have to use extra information, e.g., heading, acceleration, and speed, which is not always available.

GPS-Based Traffic Analysis: We are witnessing increasing research interests on GPS-based traffic flow analysis methods. Probe cars are equipped with GPS devices and wireless communication interfaces, and they periodically report their locations, speeds, and directions to the central server where various traffic analysis can be conducted. In addition, location-based social networks facilitate people to use GPS-enabled mobile devices to generate geo-tagged social postings, which can also be used for traffic flow analysis using so-called social transportation techniques [26].

In [15], the architecture and the data sampling methods in floating-car systems were analyzed. In [27], several data analysis methods for floating-car systems were proposed. In [28], the data sampling frequency for floating cars were analyzed. In [8]–[10], how to derive traffic information from periodically collected GPS data of moving objects were discussed. A few recent studies has used GPS trajectories to estimate travel time and fuel consumption for different roads in various settings [29]–[33]. GPS trajectories also enable advanced vehicle navigation systems, e.g., personalized navigation [34], [35] and ecorouting [36], [37].

Compared with stationary-sensor-based and camera-based methods and airborne methods, GPS-based traffic-flow-analysis methods are less expensive and more flexible. However, these methods suffer from low precision of traffic flow analysis because considerable errors can be introduced in the network-matching process. The accuracy of GPS-based traffic analysis significantly improves using the proposed NMTMOD.

III. NMTMOD Data Modeling

To express traffic parameters in fine granularities, we propose a novel *traffic parameterized network*, where each edge is associated with a set of traffic flow parameters.

Definition 1 (Traffic Parameterized Network): A traffic parameterized network Net is defined as $\text{Net} = (\text{Nodes}, \text{Edges})$, where Nodes is a set of nodes, and Edges is a set of traffic parameterized edges.

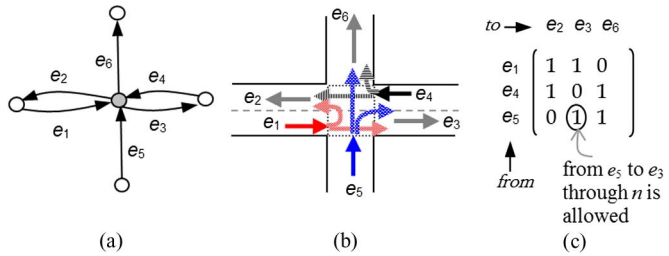


Fig. 2. Edges around a node and its connectivity matrix. (a) Road structure. (b) Traffic flows inside the node. (c) Connectivity matrix.

Definition 2 (Traffic Parameterized Edge): A traffic parameterized edge $e \in$ Edges is defined as $e = (\text{eid}, \text{geo}, \text{len}, \text{nid}_{\text{from}}, \text{nid}_{\text{to}}, \text{Para})$, where eid is the unique identifier of edge e; geo is the polyline representing edge e; len is the length of edge e; nid_{from} and nid_{to} are the identifiers of the starting node and the ending node of edge e, respectively; and Para is a *traffic parameter set* describing the traffic condition of edge e.

Definition 3 (Traffic Parameter Set): The traffic parameter set of an edge, which is denoted Para, is defined as $\text{Para} = \{(\text{paraName}_i, \text{paraType}_i, \text{paraValue}_i)\}_{i=1}^{m}$, where $(\text{paraName}_i, \text{paraType}_i, \text{paraValue}_i)$ is the ith traffic parameter of the edge with name paraName_i, data type paraType_i, and value paraValue_i.

For example, an edge is associated with a traffic parameter ("avgSpeed" "real," "57"). It indicates that the edge's average speed (represented as a real number) is 57 km/h.

Definition 4 (Node): A node $n \in$ Nodes is defined as $n = (\text{nid}, \text{loc}, \{\text{eid}_i\}_{i=1}^{m}, matrix)$, where nid is the unique identifier of node n, loc is the location of node n, $\{\text{eid}_i\}_{i=1}^{m}$ is the set of the identifiers of the edges that are connected by node n, and $matrix$ is the connectivity matrix of node n, which describes the node's traffic transferability between different edges.

Fig. 2(a) and (b) shows an example of the road structure and possible traffic flows around node n. The connectivity matrix of the node is shown in Fig. 2(c). The matrix describes possible traffic flows through the node, and each element (either 0 or 1) indicates whether moving objects can transfer from the "from" edge to the "to" edge through the node. For instance, the marked element value in Fig. 2(c) indicates that moving objects can transfer from e_5 to e_3 through the node.

Next, we introduce some important concepts used in the NMTMOD.

Definition 5 (Euclidean Motion Vector): A Euclidean motion vector of a moving object is defined as $\text{emv} = (t, (x, y), v, d)$, where t is the time when the vector is sampled, and (x, y), v, and d indicate the location, the speed, and the direction of the moving object at time t, respectively, A Euclidean motion vector can be derived from a GPS record.

Definition 6 (Network Motion Vector): A network motion vector of a moving object is defined as $\text{nmv} = (\text{emv}, E)$, where emv is a Euclidean motion vector, and E is a set of the identifiers of the edges that are network matched based on emv. Network motion vectors are classified into three categories according to the cardinality of E: 1) when $|E| = 0$, emv is outside the traffic network, and nmv is called "nonmatched"; 2) when $|E| = 1$, emv is matched to a specific edge, and nmv is

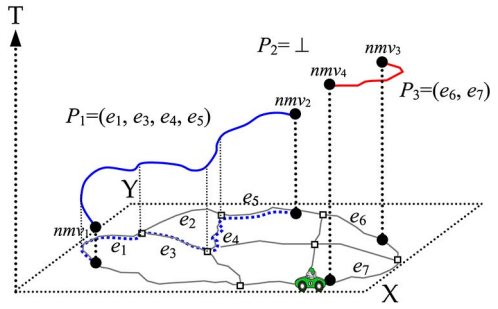


Fig. 3. Network-matched trajectory.

called "single matched"; and 3) when $|E| > 1$, emv is matched to multiple edges, and nmv is called "multiple matched."

Definition 7: The network-matched trajectory of a moving object, which is denoted nmtr, is defined as nmtr $= (\text{nmv}_1, P_1, \text{nmv}_2, P_2, \ldots, P_{n-1}, \text{nmv}_n)$, where nmv_i $(1 \leq i \leq n)$ indicates the ith network motion vector, and $P_i = \langle e_1, e_2, \ldots, e_k \rangle$, where $1 \leq i < n$ is a sequence of edge identifiers representing the network path of the moving object between nmv_i and nmv_{i+1}. P_i is empty in three cases: 1) one or both of nmv_i and nmv_{i+1} are nonmatched or multimatched; 2) nmv_i and nmv_{i+1} are in the same edge or in adjacent edges; 3) the path between nmv_i and nmv_{i+1} is unknown. P_i takes value $\perp$ to indicate the situation when the moving object is offline.

A network-matched trajectory can describe multiple continuous movements. For example, as shown in Fig. 3, the network-matched trajectory is $(\text{nmv}_1, (e_1, e_3, e_4, e_5), \text{nmv}_2, \perp, \text{nmv}_3, \text{nmv}_4)$. Note that P_3 is omitted since nmv_3 and nmv_4 correspond to two adjacent edges and the edges in P_3 can be directly inferred from nmv_3 and nmv_4. Network-matched trajectories are different from the trajectories of network-constrained moving objects [8], which do not have the path information between adjacent network motion vectors.

IV. Generating Network-Matched Trajectories

Fig. 4 provides an overview of the NMTMOD, which consists of three logical servers: network-matching server, trajectory database server, and traffic analysis server. Two or more logical servers can be deployed on a physical server.

A. Dense Sampling of Euclidean Motion Vectors

In the NMTMOD, each registered moving object (e.g., a vehicle) is uniquely identified and is equipped with a GPS receiver and a wireless interface. A moving object follows a *DSBU* strategy.

In every τ_s time, a moving object's GPS receiver samples a Euclidean motion vector and keeps it at its local storage. In addition, whenever its speed change or direction change exceeds predefined thresholds,[1] an additional local sampling is triggered. In every τ_u time $(\tau_u \gg \tau_s)$, the moving object *uploads* the sampled Euclidean motion vectors to the network-matching server as a location update message LUMsg $=$ (moid, $\{\text{emv}_i\}_{i=1}^{n}$), where moid is the identifier of the moving object, and emv_i is the ith Euclidean motion vector. In the DSBU mechanism, τ_s is set to a relatively small value, e.g., 10 s, whereas τ_u is set to a relatively large value, e.g., 2 min. Therefore, each location update message usually contains multiple Euclidean motion vectors. The DSBU mechanism is shown in the left side of Fig. 4, where each point indicates a sample and each car icon indicates a location update.

When the network-matching server receives a location update message, it transforms the Euclidean motion vectors into a network-matched trajectory (nmtr), which is stored in the trajectory database server. Finally, the trajectories are processed by the traffic analysis server to update the traffic flow parameters of the road segments.

B. Generating Network-Matched Trajectories

After receiving a location update message, the network-matching server matches the sequence of Euclidean motion vectors to the underlying road network and generates a network-matched trajectory.

In the following, we first define a basic network-matching algorithm in Section IV-B1; then, we describe the point-to-path matching algorithm to deal with multimatched points in Section IV-B2, and finally, we provide the complete edge-centric network-matching algorithm in Section IV-B3.

1) Distance-Direction-Connectivity-Based Matching Method: This method takes as input the transportation network Net and an Euclidean motion vector emv and returns an edge set E that consists of the edges to which emv is mapped while considering the distance, direction, and connectivity constraints.

First, the method retrieves the edges whose *distances* to emv is smaller than the GPS measurement error threshold as candidate edges. Next, it eliminates edges whose *directions* do not match the direction of the motion vector. Finally, it eliminates candidate edges according to the *connectivity* constraint. Suppose that e_{from} and e_{to} are the edges of emv's previous matched edge and next matched edge, respectively. For $\forall\, e \in E$, if e is not connected from e_{from} or is not connected to e_{to}, e is removed from E. Nodes' connectivity matrices enable testing the connectivity constraints.

In most cases, a single-matched network motion vector, as defined in Definition 6, is obtained. However, cases when the motion vector is nonmatched (e.g., when the moving object is running outside the network) or multimatched (e.g., when the moving object is running around intersections or in dense network) may occur. These motion vectors can be transformed to single-matched ones if we know the path information.

2) Transforming Multimatched Motion Vectors to Single Matched Based on Path Information: In the network-matching process, it often occurs that the previous and succeeding motion vectors are single matched, whereas the intermediate motion vectors are multimatched. In this case, we are able to infer the path information from the single-matched motion vectors and transform the multimatched motion vectors into single-matched ones.

As shown in Fig. 5, the path between the two single-matched motion vectors nmv_1 and nmv_3 can be derived if they are in the same edge [see Fig. 5(a)], in two adjacent edges [see Fig. 5(b)], or in two edges connected by another edge [see Fig. 5(c)]. Therefore, by further considering the path information, the

[1]After a careful analysis on the average speeds and the topological and geometrical properties of the Beijing road network, we set the thresholds as 5 km/h and 30°, respectively.

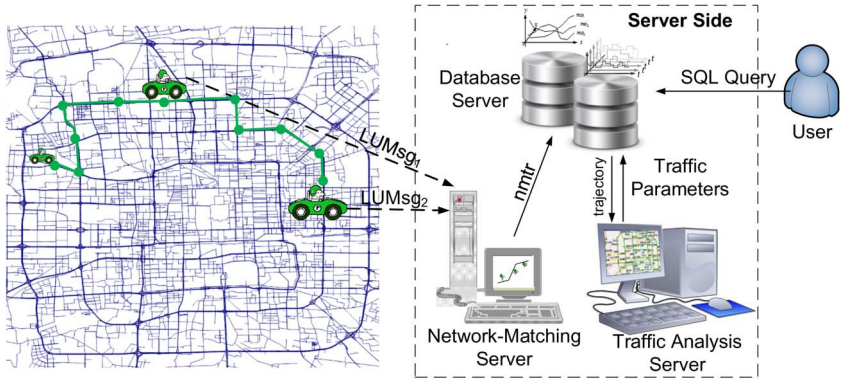


Fig. 4. Structure of the NMTMOD system.

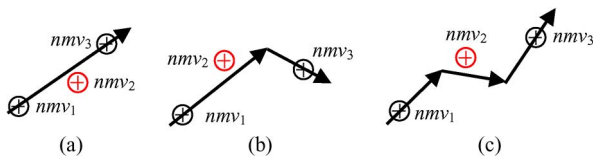


Fig. 5. Multimatched vectors to single-matched vectors. (a) Single-edge path. (b) Dual-edge path. (c) Triple-edge path.

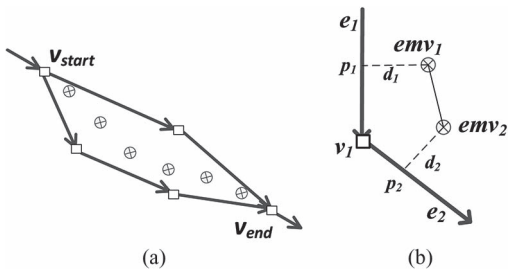


Fig. 6. Ambigous situation and path measurement. (a) Ambiguous situation. (b) Path measurement.

multimatched motion vector nmv_2 can be converted to single-matched one.

However, there exist several multimatched motion vectors and/or possible paths between two single-matched motion vectors. Fig. 6(a) gives such an example. In this case, we evaluate the probability of each candidate path and choose the one with the highest probability. Take Fig. 6(b) to illustrate, suppose the nearest edges in the path P for emv_1 and emv_2 are e_1 and e_2 respectively, the probability of the resulting nmtr that uses path P is

$$\mathbf{Pr}(P, \text{nmtr}) = \prod_{i=2}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{d_i}{2\sigma^2}\right) \cdot \frac{1}{\beta} \exp\left(\frac{l_i - \hat{l}_i}{\beta}\right)$$

where σ is the standard deviation of the GPS points as normal distribution; d_i is the distance between emv_i and its nearest edge in P, e.g., d_1, d_2; l_i is the distance between emv_i and emv_{i-1}; $\hat{l}_i$ is the length of the corresponding path fragment, e.g., p_1, v_1, p_2; and β is the scaling factor.

3) Edge-Centric Network-Matching Algorithm: We proceed to describe the edge-centric network-matching process. It takes as input two arguments, i.e., the traffic network Net and

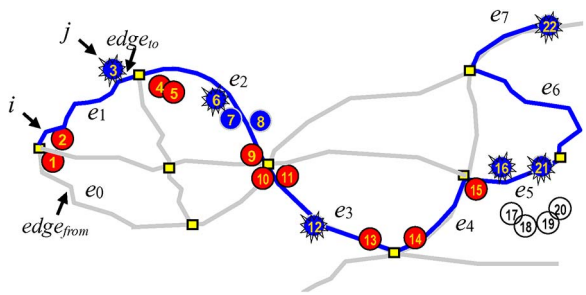


Fig. 7. Motion vectors to be matched to the network.

the location update message $\text{LUMsg} = (\text{moid}, ((t_i, (x_i, y_i), v_i, d_i))_{i=1}^{n})$, and returns a network-matched trajectory.

The algorithm processes the Euclidean motion vectors in LUMsg through multiple rounds. For each round, e_{from} represents the edge where the moving object is located at the end of the previous round, whereas i and j denote the starting and the ending motion vectors, respectively. Normally, nmv_j is the first single-matched motion vector in the current round whose matched edge is not e_{from} or the last motion vector. For instance, Fig. 7 shows a sequence of motion vectors in LUMsg, where the motion vectors with red color represent the multimatched cases, and blue ones denote the single-matched cases, whereas white ones indicate the none-matched cases. The motion vectors in Fig. 7 are processed in six rounds with the (i, j) pairs being (1, 3), (4, 6), (7, 12), (13, 16), (17, 21), and (22, 22), respectively.

In each round, the algorithm match the motion vectors $\text{nmv}_i \sim \text{nmv}_j$ to the network. If nmv_j is single matched to an edge e_j other than e_{from}, then e_j is assigned to e_{to}. After that, the algorithm generates all possible paths S_P, and then chooses the path with the highest probability. For the multimatched motion vectors in the current round, the algorithm uses the connectivity constraint to further eliminate infeasible edges.

For instance, in the first round of motion vectors shown in Fig. 7, $e_{\text{from}} = e_0, e_{\text{to}} = e_1$, and $P = (e_0, e_1)$. Therefore, the multimatched motion vectors $\text{nmv}_1, \text{nmv}_2$ can be single matched to e_0 and e_1 respectively. Similarly, the multimatched motion vectors in the second, third, and fourth rounds, can also be transformed to single-matched ones. In the fifth round, $e_{\text{from}} = e_5$ and $e_{\text{to}} = e_5$. Since $\text{nmv}_{17} \sim \text{nmv}_{20}$ are

nonmatched motion vectors, and nmv_{21} is single matched, they remain unchanged. In the sixth round, $e_{\text{from}} = e_5$, $e_{\text{to}} = e_7$, $P = (e_5, e_6, e_7)$, and consequently, $P_{21} = (e_5, e_6, e_7)$.

The matching result is nmtr = $((\text{emv}_1, e_0), (\text{emv}_2, e_1), (\text{emv}_3, e_1), (\text{emv}_4, e_2), (\text{emv}_5, e_2), (\text{emv}_6, e_2), (\text{emv}_7, e_2), (\text{emv}_8, e_2), (\text{emv}_9, e_2), (\text{emv}_{10}, e_3), (\text{emv}_{11}, e_3), (\text{emv}_{12}, e_3), (\text{emv}_{13}, e_3), (\text{emv}_{14}, e_4), (\text{emv}_{15}, e_5), (\text{emv}_{16}, e_5), (\text{emv}_{17}, \perp), (\text{emv}_{18}, \perp), (\text{emv}_{19}, \perp), (\text{emv}_{20}, \perp), (\text{emv}_{21}, e_5), (e_5, e_6, e_7), (\text{emv}_{22}, e_7))$.

The edge-centric network-matching algorithm is able to match most multimatched motion vectors to the network correctly with no shortest path calculation involved. This is one of the main advantages compared with "previous-point-dependent" network-matching methods.

C. Discarding Unimportant Motion Vectors

Since the Euclidean motion vectors are sampled densely, the size of the network-matched trajectory may be very large if all the sampled motion vectors are stored in the database. Therefore, the server needs to optimize the storage of the trajectories by discarding unimportant motion vectors so that only key motion vectors are kept in the database. Moreover, when a single-matched motion vector is discarded, its edge information should remain in the database.

The procedure of discarding unimportant motion vectors is based on the assumption that, in most cases, a moving object traverses on its route with a constant speed. A new motion vector, called a key motion vector, is needed only when the deviation between the current position of the moving object and the predicted position becomes larger than a threshold δ_ε. Given $\text{nmtr} = (\text{nmv}_1, P_1, \text{nmv}_2, P_2, \ldots, P_{n-1}, \text{nmv}_n)$, a motion vector $\text{nmv}_i = ((t_i, (x_i, y_i), v_i, d_i), E_i)$ can be discarded if the following constraint is satisfied:

$$\mathbf{distance}\left(\mathbf{pos}\left(\mathbf{predict} \times \left(path, \text{nmv}_{\text{start}}, \Delta t\right)\right), \mathbf{pos}(\text{nmv}_i)\right) \leq \delta_\varepsilon.$$

The function **predict** takes three arguments: the path P that mo traverse, and the starting motion vector $\text{nmv}_{\text{start}}$ that contains the starting position and the speed, and the elapsed time Δt. It returns the predicted motion vector based on the constant speed assumption. The function **distance** returns the distance between two positions.

Algorithm 1 takes two arguments, i.e., the traffic network Net and the network-matched trajectory $\text{nmtr} = (\text{nmv}_1, P_1, \text{nmv}_2, P_2, \ldots, P_{n-1}, \text{nmv}_n)$, and returns the network-matched trajectory, which consists of only key motion vectors. In Algorithm 1, function $\mathbf{len}(\text{nmtr})$ returns the number of motion vectors in nmtr, function $\mathbf{merge}(\text{path}, P_{j-1})$ returns the merge result of the two paths, function $\mathbf{distance}_{\text{Net}}(\text{nmv}_i, \text{nmv}_j, path)$ returns the distance between nmv_i and nmv_j on the path of the moving object, i.e., P. i is the index of the starting motion vector, j is the index of the end motion vector, and the algorithm tries to find the maximum j satisfying the following constraint for $\forall\, i \leq k \leq j$:

$$\mathbf{distance}\left(\mathbf{pos}\left(\mathbf{predict}(path, mv_i, \Delta t)\right), \mathbf{pos}(\text{nmv}_k)\right) \leq \delta_\varepsilon.$$

Algorithm 1 $\mathbf{Discarding}(\text{Net}, \text{nmtr})$

```
1: j ← 1; v_max = ∞; v_min = 0
2: while j ≤ len(nmtr) do
3:     i ← j
4:     j ← i + 1
5:     if |E_i| = 1 then
6:         P = {}                          ▷initialize P as an empty list
7:         while j ≤ len(nmtr) ∧ |E_j| = 1 do
8:             P ← merge(P, P_{j-1})
9:             l ← distance_Net(nmv_i, nmv_j, P)
10:            Δt ← nmv_j.t − nmv_i.t
11:            v ← l/Δt
12:            v'_min ← v − δ_ε/Δt; v'_max ← v + δ_ε/Δt
13:            if (v'_max ≥ v_min) ∧ (v'_min ≤ v_max) then
14:                v_max ← min(v_max, v'_max)
15:                v_min ← max(v_min, v'_min)
16:                nmv_i.v ← (v_min + v_max)/2
17:                remove(nmtr, nmv_j, P_j)
18:                j ← j + 1
19:            else
20:                break
21: return nmtr
```

This can also be considered the process of finding the constant speed v, so that it predicts as many motion vectors as possible.

The algorithm initializes a window of possible speed of the moving object $[0, \infty]$; then, it iterates through the trajectory to find the maximum j by gradually decreasing the window size: 1) it finds the first single matched motion vector nmv_i; 2) then it calculates the average speed from nmv_i to nmv_j as v (line 8–line 11), followed by deriving the speed window: $[v'_{\min}, v'_{\max}]$ (line 12); 3) if $[v'_{\min}, v'_{\max}]$ intersects with $[v_{\min}, v_{\max}]$ (line 13), which means the location nmv_j can be predicted, then the algorithm calculates the intersection, sets the speed of nmv_i, and discards nmv_j (line 14–line 17); 4) finally, it moves to the next motion vector nmv_{j+1} (line 18); and 5) otherwise, the algorithm will break and reinitialize (line 20).

At the server side, whenever it receives a new location update message, it first conducts network-matching and trajectory optimizing, and then it appends the newly generated network-matched trajectory to the database. Sometimes, the last several motion vectors of the location update message may remain multimatched if the moving object is around intersections. In this case, they are also stored to the database and are further processed when more information is available at the next location update.

V. Traffic Flow Analysis Based on Network-Matched Trajectories

Network-matched trajectories enable accurate traffic-network-based data analysis. We proceed to discuss traffic flow analysis using network-matched trajectories.

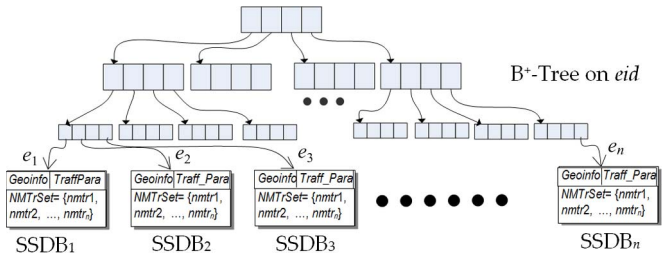


Fig. 8. Structure of the TSA Tree.

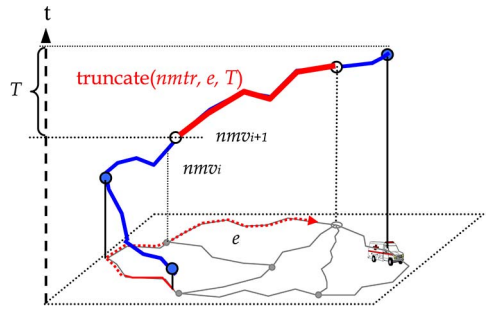


Fig. 9. $\mathbf{truncate}(\text{nmtr}, e, T)$ function.

In NMTMOD, we adopt an incremental traffic analysis method. Two thresholds are introduced to control the frequencies of statistical computations and database updates: 1) the least time gap Ψ_t, i.e., for any edge, the time interval between two consecutive statistical computations must exceed Ψ_t; and 2) the least value gap Ψ_v, i.e., for any traffic parameter on any edge, the absolute value difference between two consecutive traffic-parameter database writes must exceed Ψ_v. Smaller gaps cause high frequencies of database updates and thus more accurate traffic flow information.

For any edge, its parameters are calculated from the trajectories which passed through it during the last Δt_{stat} time, where Δt_{stat} is the size of the time window for choosing statistical analysis source data.

To speed up the statistical analysis, a traffic statistical analysis Tree (TSA-Tree) is built in NMTMOD, as shown in Fig. 8. The TSA-Tree adopts a B^+-Tree like structure. Each entry in a leaf node concerns an edge and associates a pointer to a statistical source data block (SSDB). The SSDB contains three parameters, Geo, TraffPara, and S_{nmtr}, where Geo $\in$ polyline is the geometry of the edge, TraffPara includes the traffic parameters of the edge (see Definition 3), and S_{nmtr} contains all the trajectory pieces that passing through the edge during the recent Δt_{stat} time. For a trajectory nmtr, the part of the trajectory corresponding to edge e during period T can be calculated through truncation function $\mathbf{truncate}(\text{nmtr}, e, T)$, as shown in Fig. 9.

Set S_{nmtr} is dynamically maintained whenever a new location update message is received. Suppose that the newly generated trajectory from the location update message is nmtr. For each edge e in nmtr, the server needs to get the trajectory piece through $\mathbf{truncate}(\text{nmtr}, e, [t_{\text{now}} - \Delta t_{\text{stat}}, t_{\text{now}}])$ and save it to the SSDB of the edge. Meanwhile, each existing trajectory piece in the SSDB needs to be checked, and the obsolete ones are eliminated.

The traffic parameters of the edge can be computed based on the trajectory pieces in SSDB. We assume that $S_{\text{nmtr}} = \{\text{nmtr}_1, \text{nmtr}_2, \ldots, \text{nmtr}_n\}$, $\mathbf{first}(\text{nmtr})$ and $\mathbf{final}(\text{nmtr})$ return the first and last motion vectors of the trajectory nmtr, respectively, and $\mathbf{time}(\text{nmv})$ and $\mathbf{pos}(\text{nmv})$ return the time and location of motion vector nmv, respectively.

Consider how to compute the average speed of an edge. For each nmtr $\in S_{\text{nmtr}}$, the speed through the edge can be computed as $v(\text{nmtr}) = \mathbf{len}(\text{nmtr})/(\mathbf{time}(\mathbf{final}(\text{nmtr})) - \mathbf{time}(\mathbf{first}(\text{nmtr})))$. Therefore, the average speed can be computed as $v_{\text{avg}} = \sum_{\text{nmtr} \in S_{\text{nmtr}}} v(\text{nmtr})/|S_{\text{nmtr}}|$.

The number of moving objects on edge e, which is denoted η_{mo}, can be computed as $\eta_{\text{mo}} = \sum_{\text{nmtr} \in S_{\text{nmtr}}} \mathbf{inside}(\mathbf{pos}(\mathbf{final}(\text{nmtr})), e)$, where the function $\mathbf{inside}(p, e)$ returns 1 if point p is on edge e, and returns 0 if not.

The flux of moving objects on edge e, which is denoted χ_{mo}, is the number of moving objects passing through e at each time unit, and can be computed as

$$\chi_{\text{mo}} = \frac{\sum_{\text{nmtr} \in S_{\text{nmtr}}} \left(1 - \mathbf{inside}\left(\mathbf{pos}\left(\mathbf{final}(\text{nmtr})\right), e\right)\right)}{\Delta t_{\text{stat}}}.$$

The traffic jams and their locations on edge e can also be derived from S_{nmtr}. For mtr $\in S_{\text{nmtr}}$ (suppose nmtr $= (\text{nmv}_1, P_1, \text{nmv}_2, P_2, \ldots, P_{n-1}, \text{nmv}_n)$), the speed between any two adjacent samplings, nmv_i and nmv_{i+1}, can be computed as $v(\text{nmv}_i, \text{nmv}_{i+1}) = (\mathbf{distance}_{\text{Net}}(\text{nmv}_i, \text{nmv}_{i+1}, P_i))/(\mathbf{time}(\text{nmv}_{i+1}) - \mathbf{time}(\text{nmv}_i))$. Based on the above, we can compute the sections of e where the moving object travels with speed slower than a certain threshold $\delta_{v_{\text{jam}}}$, which is denoted $\mathbf{slowSections}(\text{nmtr}, \delta_{v_{\text{jam}}})$. Then, the traffic jams of e, which is denoted α_{jam}, can be computed as $\alpha_{\text{jam}} = \bigcap_{\text{nmtr} \in S_{\text{nmtr}}} \mathbf{slowSections}(\text{nmtr}, \delta_{v_{\text{jam}}})$. If there are no traffic jams in e, then α_{jam} is $\emptyset$.

Note that more and advanced traffic parameters can be derived. For instance, we can infer the number of vehicles based on η_{mo} if we know the proportion of moving objects among all vehicles. Moreover, we can further infer the degree of saturation of e with additional road parameters, e.g., the number of lanes and the length of the edge.

The detailed procedure of traffic statistics data refreshment is provided in Algorithm 2. For each edge in the trajectory piece, the function $\mathbf{searchTSA}(e)$ is used to retrieve the SSDB of e by searching the TSA-Tree (line 2), and then function $\mathbf{insert}(\text{SSDB}, \text{nmtr})$ is used to insert the piece of trajectory nmtr into SSDB (line 4). After that, it calls function $\mathbf{update}(\text{SSDB}, T)$ to update the SSDB with obsolete trajectory pieces being removed (line 5). If the time interval is larger than Ψ_t, the algorithm will refresh traffic parameters of this edge (line 7–line 11). The function $\mathbf{getValue}(\text{SSDB}, \text{para})$ is used to get the value corresponding to para from SSDB, function $\mathbf{compute}(\text{Para}, SSDB)$ computes the para of the trajectory pieces in SSDB, and function $\mathbf{writeDB}(e, \text{para})$ is used to save the parameter para of e into the database.

Algorithm 2 $\mathbf{TrafficRefresh}(\text{nmtr})$

1: **for** $\forall e \in$ nmtr **do**
2: SSDB $= \mathbf{searchTSA}(e)$
3: $t' =$ SSDB.lastRefresh
4: $\mathbf{insert}(\text{SSDB}, \mathbf{truncate}(\text{nmtr}, e, [t_{\text{now}} - \Delta t_{\text{stat}}, t_{\text{now}}]))$
5: $\mathbf{update}(\text{SSDB}, [t_{\text{now}} - \Delta t_{\text{stat}}, t_{\text{now}}])$
6: **if** $t_{\text{now}} - t' > \Psi_t$ **then**
7: **for** $\forall para \in$ TraffPara **do**
8: $\text{val}_{\text{old}} \leftarrow \mathbf{getValue}(\text{SSDB}, \text{para})$
9: $\text{val}_{\text{new}} \leftarrow \mathbf{compute}(\text{para}, \text{SSDB})$
10: **if** $|\text{val}_{\text{new}} - \text{val}_{\text{old}}| > \Psi_v$ **then**
11: $\mathbf{writeDB}(e, \text{para})$

VI. Empirical Studies

We describe implementation issues and report on empirical studies of the accuracy and efficiency of the NMTMOD.

A. Implementation Issues

We implement the proposed NMTMOD in PostgreSQL 8.2.4 with PostGIS 3.2 extension for spatial support. The NMTMOD runs on servers where each is with Intel(R) Core i5 CPU 2.5 GHz and 2GB RAM under Linux. PostgreSQL is an object-relational database management systems (DBMSs), which allows user defined data types and operators, which facilities the NMTMOD model to be plugged in the PostgreSQL seamlessly. To speed up the query processing, we use an R-Tree to index the road network and an MOSTR-Tree [38] to index trajectories.

In the trajectory database server, we implement a set of new data types, including nmtr, matrix, traffpara, etc., so that the motion vectors and trajectories can be expressed and managed at the database kernel directly. Based on these data types, we created three relational tables at the database server: edges (eid: string, geo: polyline, len: real, nid_{from}: string, nid_{to}: string, $para$: traffpara); nodes (nid: string, loc: point, co-edges: struct-string, mat: matrix); and MObjs (moid: string, moDescript: string, traj: nmtr).

In addition, we define a set of operators so that the new data types can be processed using SQL. The operators are divided into five categories: 1) extraction operators, which extract detailed information from the edges or nodes of the transportation network; 2) truncation operators, which select part of the trajectory according to spatial and temporal ranges; 3) projection operators, which project trajectory to the latitude–longitude plane or to the time axis; 4) transform operators, which transform network-based points to Euclidean-based points and *vice versa*; 5) location update operators, which take location update messages and conduct location update procedures at the server side as discussed in Section IV.

B. Experimental Setup

We use the road network of Beijing with 75 268 edges and 56 201 nodes, which covers both urban and suburban roads, i.e., the roads within and outside the fourth ring road, respectively.

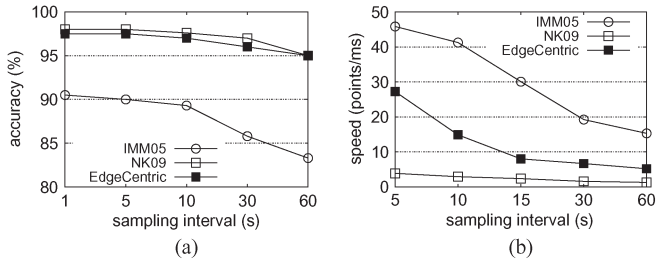


Fig. 10. Network matching. (a) Accuracy. (b) efficiency.

We use more than 10 million GPS records collected from 7100 vehicles that traveled during both peak and off-peak periods.

1) Settings for Network Matching: To evaluate the accuracy of the edge-centric matching algorithm, we compare it with an incremental algorithm *IMM05* [21], and a hidden-Markov-model-based algorithm *NK09* [20]. For both methods, we follow the original papers to choose parameters. In particular, for IMM05, we use $\mu_d = 10$, $\alpha = 0.17$, $n_d = 1.4$, $\mu_a = 10$, $n_\alpha = 4$; for NK09, we use $\sigma = 4.1$ and $\beta = 5$. Preprocessing and optimization suggested in [20] are made to NK09.

For each trajectory, we create a NMTMOD trajectory accordingly and compare it with the two trajectories produced by IMM05 and NK09 to check if the matching results are correct. The precision is then defined as the ratio of the correct matches to the total matches.

2) Settings for Moving-Object Database: To evaluate the performance of the NMTMOD, which adopts the DSBU strategy, we implement two other MODs: the EMOD, which uses the vector-based location update method, and the NMOD, which uses the segment-based location update method. The two methods for location updates are introduced in [12]. We set $\tau_u = 5$ min.

3) Ground-Truth Data: In the network-matching experiment, we use 14 436 manually labeled trajectory points collected from ten cars as the ground-truth data. In the traffic analysis experiment, however, manually labeling millions of points is infeasible. Thus, the ground-truth data are derived from the matching result of the NK09 algorithm on densely sampled trajectory points as the matching accuracy of the NK09 algorithm is expected to be higher than 98% when the sampling interval is within 10 s.

C. Evaluating Network-Matching Algorithm

Fig. 10(a) shows the relationship between the network-matching precision and the sampling time interval τ_s. As stated in Fig. 10(a), when τ_s is lower than 15 s, the accuracy of the edge-centric matching algorithm is close to 100%, and then gradually decreases to 95% when $\tau_s = 60$ s. Generally speaking, the network-matching precision increases when the sampling interval τ_s decreases, the reason for this result is that the network-matching precision is actually decided by the density of the sampling points. When τ_s decreases, the distance between any two neighboring sampling points becomes shorter, consequently the network-matching precision increases.

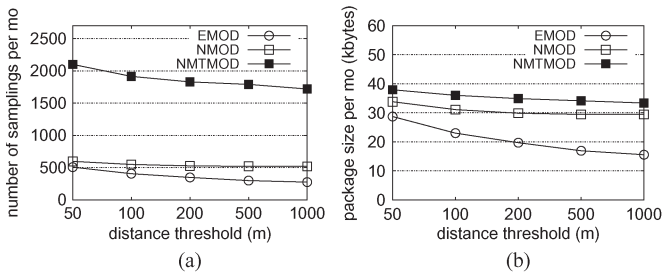


Fig. 11. Communication evaluation. (a) Number of samplings. (b) Communication cost.

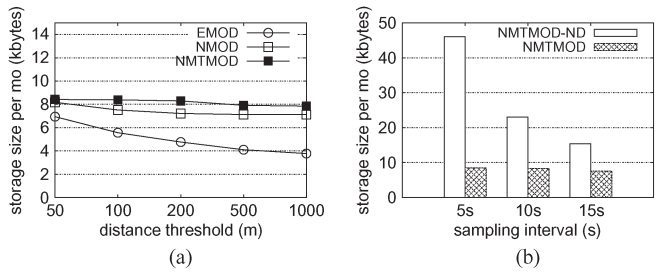


Fig. 12. Storage evaluation. (a) Storage consumption. (b) Effect of discarding.

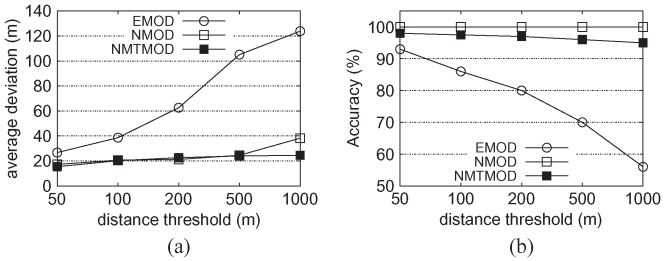


Fig. 13. Tracking analysis. (a) Average position deviation. (b) Edge accuracy.

Fig. 10(b) shows the efficiency of the network-matching algorithms. Compared with NK09, the proposed edge-centric matching algorithm achieves much higher speeds while maintaining roughly the same accuracy. This is due to the following reasons. First, the edge-centric matching algorithm is independent on previous points; thus, the errors in previous points do not result in further errors. Second, the algorithm considers distance, direction, and connectivity to eliminate infeasible edges, and adopts an efficient method dealing with the ambiguous situations. Finally, no shortest path calculation is required, which makes the algorithm much faster than NK09.

D. Analysis of the NMTOD

We compare the NMTMOD with the traditional EMOD and the NMOD in terms of communication cost, storage consumption, tracking accuracy, and traffic analysis accuracy.

1) Communication Cost and Storage Consumption: We assume that the moving objects use GPRS for wireless communication, and therefore, the communication cost is measured as the total size of the packages that the moving objects need to send to the servers. Further, we assume the Transmission Control Protocol/Internet Protocol (TCP/IP) is applied for transmitting the data, which bring an overhead of 40 B (20 B for IPv4 header and 20 B for TCP header). The location update message $\text{LUMsg} = (\text{moid}, ((t_i, (x_i, y_i), v_i, d_i))_{i=1}^{n})$, where moid and t_i are integers, x_i and y_i are floats, andv_i and d_i are represented by a single short integer; therefore, the size of the package is $40 + 4 + (4 + 4 + 4 + 2) * n = 44 + 14n$ B.

Fig. 11(a) shows the number of samplings, whereas Fig. 11(b) shows the communication cost. Although the number of samplings of NMTMOD is much higher than NMOD, their communication cost is roughly the same. This is because the location update messages are wrapped into IP packages, which contain many data dedicated for communication, e.g., routing information, checksum, and flags. When the location update message size is small, the package size is mainly determined by the communication data, i.e., five motion vectors per package and one motion vector per package do not make much difference in terms of package size. As a result, the communication cost is basically decided by the number of packages, which is determined by τ_u. Moreover, the number of samplings of NMOD is higher than that of EMOD. This is because the road segments in the underlying traffic network are relatively short and straight, and it frequently happens that several connected road segments can be roughly represented by a single vector.

Fig. 12(a) shows the storage consumption of the three MODs, whereas Fig. 12(b) demonstrates the relationship between storage consumption and the sampling interval τ_s and the effect of discarding. The storage consumption of NMTMOD is also roughly the same with NMOD and just slightly higher than EMOD. As shown in Fig. 12(b), the storage consumption of NMTMOD barely changes when the sampling interval τ_s decreases. The reason is that, after discarding unimportant motion vectors, the size of a network-matched trajectory is mainly decided by the route of the moving object, which is independent of individual sampling points. Since most unimportant sampling points are discarded, the size of the resulting trajectory becomes stable. On the other hand, as indicated in Fig. 12(b), the storage consumption without discarding optimization increases rapidly when τ_s decreases.

2) Tracking Accuracy: The tracking accuracy is evaluated in terms of both the position and corresponding edge of the moving object. In NMTMOD, the position of the moving object is calculated using the prediction method based on the constant speed assumption, and the Euclidean-based prediction method is also used if the moving object reaches the end of its path.

First, we randomly choose several timestamps and get the positions and edges of all the moving objects from the MODs at that moment. After that, we compute the average position deviation and edge accuracy of each moving object. Fig. 13(a) and (b) shows the average position deviation and the edge accuracy of the MODs, respectively. The tracking accuracy of NMTMOD is much higher than the EMOD, and also quite comparable to the NMOD. We can conclude that, with trajectory points being densely sampled and unimportant ones being discarded, the NMTMOD manages to achieve a higher tracking accuracy while maintaining low storage consumption.

3) Traffic Analysis: We choose two most commonly used traffic parameters in ITS applications, i.e., the average speed and the flux of the edge, to evaluate the traffic analysis accuracy.

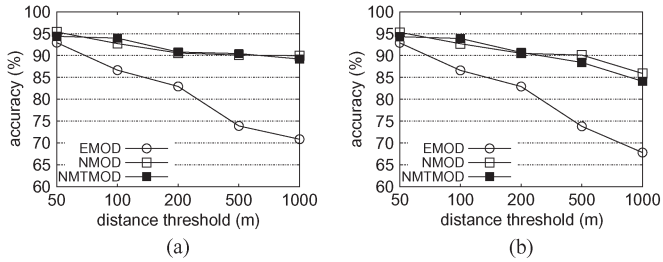


Fig. 14. Traffic analysis. (a) Average speed accuracy. (b) flux accuracy.

The accuracy of the statistics about speed is calculated according to $\xi_{\mathrm{tp}} = |\{e|e \in \mathrm{Edges} \wedge \mathbf{abs}(\widetilde{tp}_e - tp_e) \leq \delta_{\mathrm{tp}}\}|/|\mathrm{Edges}|$, where δ_{tp} is the threshold for the traffic parameter, e.g., 2 m/s for speed and 1 for flux, and an edge is considered as correct if the absolute difference between its corresponding parameter value tp_e and the real one $\widetilde{\mathrm{tp}}_e$ is lower than δ_{tp}.

Fig. 14(a) shows the accuracy of speed statistics, whereas Fig. 14(b) shows the accuracy of flux. For similar reasons of high tracking accuracy, the speed accuracy and flux accuracy of NMTMOD are also much higher than EMOD and are quite comparable to NMOD.

To sum up, compared with EMOD models, NMTMOD has higher precision in location tracking under similar communication cost and higher precision in TSA. Compared with other NMOD models, NMTMOD achieves similar location tracking accuracy and storage efficiency while being mobile-map free. In addition, the edge-centric network-matching algorithm has better performance in dealing with densely sampled motion vectors.

Discussion: It is nontrivial to choose an appropriate parameter τ_s when using the proposed DSBU strategy. If τ_s is too short, the communication costs increase substantially. On the other hand, if τ_s is too long, the tracking accuracy decreases significantly. A strategy is needed such that parameter τ_s can be automatically configured. Specifically, when vehicles travel in regions with dense road networks (e.g., urban regions) or with high speeds (e.g., during off-peak periods), the strategy may automatically set a short τ_s in order to maintain high tracking accuracy. In contrast, when vehicles travel in regions with sparse road networks (e.g., rural regions) or with low speeds (e.g., during peak periods), the strategy may automatically set a long τ_s in order to save communication costs while maintaining high tracking accuracy. We leave it as an interesting future research direction.

VII. Conclusion

Mobile-map-free tracking and managing of moving objects' trajectories are of great importance to network-constrained moving objects management and are also crucial for a great number of applications such as traffic flow statistical analysis and network-based moving pattern mining. However, existing methods, such as EMODs and NMODs, cannot provide high tracking accuracy or support mobile-map-free tracking. In this paper, we propose the NMTMOD model to precisely track and efficiently manage the network-matched trajectories of moving objects while requiring neither mobile maps nor network-matching at the moving object side. In particular, compared with EMODs, the NMTMOD is able to provide higher tracking precision with similar communication cost; and compared with NMODs, the NMTMOD achieves similar accuracy tracking precision and storage requirement while being mobile-map free. Thus, the NMTMOD is a mobile-map-free approach for tracking and managing moving objects' trajectories with high precession, which is able to support a wide variety of traffic related applications.

In the future work, we plan to investigate traffic pattern analysis and data mining techniques based on the NMTMOD. It is also of interest to explore how to dynamically and automatically choose important parameters such as τ_s.

References

[1] R. H. Güting *et al.*, "A foundation for representing and querying moving objects," *ACM Trans. Database Syst.*, vol. 25, no. 1, pp. 1–42, Mar. 2000.
[2] C. Guo, C. S. Jensen, and B. Yang, "Towards total traffic awareness," *SIGMOD Record*, vol. 43, no. 3, pp. 18–23, Sep. 2014.
[3] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and querying moving objects," in *Proc. ICDE*, 1997, pp. 422–432.
[4] O. Wolfson and H. Yin, "Accuracy and resource consumption in tracking and location prediction," in *Proc. SSTD*, 2003, pp. 325–343.
[5] J. Su, H. Xu, and O. H. Ibarra, "Moving objects: Logical relationships and queries," in *Proc. SSTD*, 2001, pp. 3–19.
[6] Z. Ding and X. Zhou, "Location update strategies for network-constrained moving objects," in *Proc. DASFAA*, 2008, pp. 644–652.
[7] R. H. Güting, T. de Almeida, and Z. Ding, "Modeling and querying moving objects in networks," *VLDB J.*, vol. 15, no. 2, pp. 165–190, Jun. 2006.
[8] Z. Ding and R. H. Güting, "Managing moving objects on dynamic transportation networks," in *Proc. 16th Int. Conf. Sci. Statist. Database Manag.*, Washington, DC, USA, 2004, pp. 287–296.
[9] L. Speičvcys, C. S. Jensen, and A. Kligys, "Computational data modeling for network-constrained moving objects," in *Proc. 11th ACM Int. Symp. Adv. Geograph. Inf. Syst.*, New York, NY, USA, 2003, pp. 118–125.
[10] V. T. De Almeida and R. H. Güting, "Indexing the trajectories of moving objects in networks," *Geoinformatica*, vol. 9, no. 1, pp. 33–60, Mar. 2005.
[11] E. Frentzos, "Indexing objects moving on fixed networks," in *Proc. 8th Int. Symp. Spatial Temporal Databases*, Santorini Island, Greece, 2003, pp. 289–305.
[12] A. Civilis, C. S. Jensen, and S. Pakalnis, "Techniques for efficient road-network-based tracking of moving objects," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 5, pp. 698–712, May 2005.
[13] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *Proc. KDD*, 2007, pp. 330–339.
[14] T. Nakata and J.-I. Takeuchi, "Mining traffic data from probe-car system for travel time prediction," in *Proc. KDD*, 2004, pp. 817–822.
[15] M. Sarvi *et al.*, "A methodology to identify traffic condition using intelligent probe vehicles," in *Proc. 10th ITS World Congr. Madrid*, Madrid, Spain, 2003, pp. 17–21.
[16] D. Tiesyte and C. S. Jensen, "Efficient cost-based tracking of scheduled vehicle journeys," in *Proc. 9th Int. Conf. MDM*, 2008, pp. 9–16.
[17] Y. Lou *et al.*, "Map-matching for low-sampling-rate GPS trajectories," in *Proc. GIS*, Seattle, WA, USA, 2009, pp. 352–361.
[18] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transp. Res. C, Emerging Technol.*, vol. 8, no. 1–6, pp. 91–108, Feb.–Dec. 2000.
[19] C. Wenk, R. Salas, and D. Pfoser, "Addressing the need for map-matching speed: Localizing globalb curve-matching algorithms," in *Proc. 18th Int. Conf. SSDBM*, Washington, DC, USA, 2006, pp. 379–388.
[20] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness" in *Proc. GIS*, Seattle, WA, USA, 2009, pp. 336–343.
[21] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data" in *Proc. VLDB*, Trondheim, Norway, 2005, pp. 853–864.
[22] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Proc. ICDE*, Washington, DC, USA, 2012, pp. 1144–1155.

[23] O. Pink and B. Hummel, "A statistical approach to map matching using road network geometry, topology and vehicular motion constraints," in *Proc. 11th Int. IEEE Conf. ITSCE*, 2008, pp. 862–867.
[24] J.-S. Pyo, D.-H. Shin, and T.-K. Sung, "Development of a map matching method using the multiple hypothesis technique," in *Proc. IEEE Intell. Transp. Syst.*, Oakland, CA, USA, 2001, pp. 23–27.
[25] F. Abdallah, G. Nassreddine, and T. Denoeux, "A multiple-hypothesis map-matching method suitable for weighted and box-shaped state estimation for localization," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1495–1510, Dec. 2011.
[26] F. Wang, "Scanning the issue and beyond: Real-time social transportation with online social signals," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 909–914, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1109/TITS.2014.2323531
[27] M. E. Fouladvand and A. H. Darooneh, "Statistical analysis of floating-car data: An empirical study," *Eur. Phys. J. B—Condensed Matter Complex Syst.*, vol. 47, no. 2, pp. 319–328, Sep. 2005.
[28] A. Torday and A.-G. Dumont, "Parameters influencing probe vehicle based travel time estimation accuracy," in *Proc. 4th STCR Swiss Transp. Res.*, 2004, pp. 1–15.
[29] B. Yang, M. Kaul, and C. S. Jensen, "Using incomplete information for complete weight annotation of road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1267–1279, May 2014.
[30] B. Yang, C. Guo, and C. S. Jensen, "Travel cost inference from sparse, spatio-temporally correlated time series using markov models," *PVLDB*, vol. 6, no. 9, pp. 769–780, Jul. 2013.
[31] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *Proc. IEEE ICDE*, 2014, pp. 136–147.
[32] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul, "Ecomark: Evaluating models of vehicular environmental impact," in *Proc. GIS*, 2012, pp. 269–278.
[33] C. Guo, B. Yang, O. Andersen, C. S. Jensen, and K. Torp, "Ecomark 2.0: Empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data," *GeoInformatica*, 33 pp., doi:10.1007/s10707-014-0221-7.
[34] B. Yang, C. Guo, C. S. Jensen, and Y. Ma, "Towards Personalized, Context-Aware Routing," pp. 1–22, Tech. Rep. [Online]. Available: http://people.cs.aau.dk/~byang/papers/final.pdf
[35] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *Proc. ICDE*, 2015, pp. 1–12.
[36] C. Guo, B. Yang, O. Andersen, C. S. Jensen, and K. Torp, "Ecosky: Reducing vehicular environmental impact through eco-routing," in *Proc. IEEE ICDE*, 2015, pp. 301–304.
[37] O. Andersen, C. S. Jensen, K. Torp, and B. Yang, "Ecotour: Reducing the environmental footprint of vehicles using eco-routes," in *Proc. MDM*, 2013, pp. 338–340.
[38] Z. Ding and K. Deng, "Collecting and managing network-matched trajectories of moving objects in databases," in *Proc. Database Expert Syst. Appl.*, Toulouse, France, 2011, pp. 270–279.

Zhiming Ding received the Bachelor's degree from Wuhan University, Wuhan, China, in 1989; the Master's degree from Beijing University of Technology, Beijing, China, in 1996; and the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2002.

From 1989 to 1993, he worked with the Institute of Scientific and Technical Information, Ministry of Communications of China; from 1996 to 1999, with Sinochem Group; from 2002 to 2004, with FernUniversität, Hagen, Germany; and from 2004 to 2014, with the Institute of Software, Chinese Academy of Sciences. Since August 2014, he has been with the College of Computer Science, Beijing University of Technology, Beijing, where he is currently a Professor. He is the author of three books and about 110 papers in academic journals and conferences and a Holder of five invention patents. His main research interests include database systems, mobile and spatiotemporal data management, intelligent transportation systems, sensor data management, and information retrieval.

Bin Yang received the Ph.D. degree from Fudan University, Shanghai, China, in 2010.

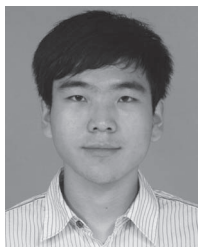
From 2010 and 2011 he was with Max Planck Institute for Informatics, Saarbrücken, Germany. From 2011 to 2014 he was with Aarhus University, Aarhus, Denmark. He is currently an Assistant Professor with Aalborg University, Aalborg, Denmark. His research interests include data management and data analytics.

Dr. Yang has been on program committees of several database conferences and was an Invited Reviewer for several database journals, including *ICDE*, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and *The VLDB Journal*.

Ralf Hartmut Güting received the Diploma and the Dr.rer.nat. degrees from University of Dortmund, Dortmund, Germany, in 1980 and 1983, respectively.

In 1987 he was a Professor with University of Dortmund. Since 1989 he has been a Full Professor of computer science with University of Hagen, Hagen, Germany. He is the author of three books and around 90 journal and conference articles. His group has built prototypes of extensible and spatiotemporal database systems, the Gral system, and the SECONDO system. From 1981 to 1984 his main research area was computational geometry. After a one-year stay at the IBM Almaden Research Center in 1985, extensible and spatial database systems became his major research interests. His current research interests include spatiotemporal or moving-object databases.

Dr. Güting was an Associate Editor for *Association for Computing Machinery Transactions on Database Systems*, an Editor for the *The VLDB Journal*, and a member of the Editorial Board of *Geoinformatica*.



Yaguang Li is currently working toward the Master's degree with the Institute of Software, Chinese Academy of Sciences, Beijing, China. His main research interests include database systems, moving-object databases, and Internet of Things.