IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 27, 2018, accepted November 7, 2018, date of publication November 14, 2018,
date of current version December 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2881039

Discovering Urban Traffic Congestion Propagation Patterns With Taxi Trajectory Data

ZHENHUA CHEN[1], YONGJIAN YANG[1], LIPING HUANG[1], EN WANG[1], AND DAWEI LI[2]
[1]Department of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Department of Computer Science, Montclair State University, Montclair, NJ 07043, USA

Corresponding author: En Wang (wangen0310@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772230 and Grant 61702215, in part by the Science and Technology Development Project of Jilin Province under Grant 20160204021GX, and in part by the Special Foundation Project for Industrial Innovation of Jilin Province under Grant 2017C032-1.

ABSTRACT Traffic congestion has gradually become a focal issue in people's daily life. When the traffic flow on a road segment exceeds its actual capacity, congestion takes place. During rush hours, a congested road segment must carry heavy loads for a long time and is very likely to spread traffic congestion to this road's adjacent segments via the spatial structure of the road. The new infected road segments continue propagating congestion in the same way. In this paper, we attempt to model the congestion propagation phenomenon with a space-temporal congestion subgraph (STCS). To this end, we detect each segment regardless of whether it is congested during consecutive time intervals and build the connection of two segments in terms of their spatio-temporal properties. Due to the sparseness of the trajectory data, two strategies of filling missing congestion edges from both temporal and spatial viewpoints are also proposed. Since STCSes are constructed from the same time interval over different days, we design a specific algorithm to discover the frequent congestion subgraphs. Finally, we evaluate the solution on Shanghai taxicab data and the corresponding road network. The experiment shows that the frequent congestion subgraph can reveal an urban congestion propagation pattern.

INDEX TERMS Congestion propagation, frequent subgraphs, trajectory data processing.

I. INTRODUCTION

The rapid growth in the number of vehicles and people's various travel demands have made urban traffic congestion increasingly severe. Every day, people have to suffer from serious traffic jam during rush hours, which leads to the loss of time and heavy air pollution. At first, congestion usually takes place on a few road segments instead of the majority of roads. In case the congestion is maintained for a long time, the current segment would probably infect its neighbours through the spatial connection between them (e.g., some vehicles move into a neighboring road). Then the new infected roads still have a chance to diffuse the congestion to their adjacent roads. This process can continue until the traffic volume starts to decrease. In some extreme cases, such as bad weather, sports event, or natural disaster, this phenomenon becomes more obvious and congestion can be spread over a large range of the road network.

Although there are some researches focusing on urban traffic congestion pattern [1]–[5], these studies place emphasis on spatial correlations between adjacent congested roads over a single time interval instead of consecutive time intervals, which leads to the lack of a useful solution that dynamically reflects the evolution of urban congestion. For example, Rempe *et al.* [2] proposed an algorithm of finding urban congestion clusters and investigated correlations between congestion clusters. Therefore, it is significant to model the process of transformation between appearance and disappearance on a road segment and contagion from congested road segments to their neighbours as a specific propagation behavior of traffic congestion. This research field is rarely tackled. To our best knowledge, only [6] has investigated the process of traffic congestion propagation. A congestion tree is built to describe the propagation pattern, but the tree structure is not sophisticated enough to reveal the spreading dynamic, thus a more powerful model of revealing urban traffic congestion pattern is urgently needed. To this end, we design a dynamic Space-Temporal Congestion Subgraph (STCS) to delineate the latent pattern of urban traffic propagation. This raises the following research challenges:

- Due to the lack of real historical data, we need to extract traffic condition on each road segment with the help of massive trajectory data.

- The time of transformation between existence and nonexistence for congestion on every road segment varies from each other. In the meantime, the duration when each congested segment spends spreading the congestion to its neighbours is also different.
- Each road segment has its respective characteristics, such as length, the number of lanes, maximum speed, and so on, while spreaders of the traffic propagation in complex networks do not have their own unique attributes (i.e., the only difference between these spreaders is their positions in the network) [7]–[10].

To tackle these challenges, we implement a Map-Matching algorithm for high-sampling-rate taxi trajectory data to compute the average speed along one or two directions on a road segment during different time intervals. Congestion on the segment is detected in terms of the index TSI [11]. We propose a directed graph named Space-Temporal Congestion Subgraph (STCS) to demonstrate the spread dynamic.

The main contributions of this paper are briefly summarized as follows:

- Inspired by the trajectory data sparseness, two strategies from both temporal and spatial perspectives are designed for filling missing congestion edges, respectively.
- We model the traffic congestion propagation as the STCS and illustrate the algorithms to dynamically construct the specific subgraph over contiguous time intervals.
- We evaluate our solution to discover frequent STCS through the taxicab trajectory data and Shanghai road network. The frequent subgraphs reveal the congestion propagation pattern in Shanghai.

The remainder of the paper is organized as follows: preliminaries are introduced in Section II. The Map-Matching algorithm is demonstrated in Section III. We explain congestion detection in Section IV. The strategies of filling missing edges are presented in Section V. In Section VI, we detail how to construct the Space-Temporal Congestion Subgraph (STCS). The method of discovering frequent STCSes is illustrated in Section VII. We evaluate our methods based on real trajectory data in Section VIII. The related work are summarized in Section IX. In Section X, we conclude our research.

II. PRELIMINARY

In this section, we give the definitions throughout the paper to avoid possible confusion.

Definition 1: A GPS point is collected by a Positioning System at regular intervals. It herein refers to a quintuple which conclude its timestamp, longitude, latitude, instantaneous velocity, and instantaneous direction.

Definition 2: A GPS trajectory is a GPS point sequence ordered by timestamps. The duration of a trajectory is the interval between the first point's timestamp and the last ones. These points in a trajectory should be consecutive in their timestamps and no point within the interval can be missed.

Definition 3: A road segment is a directed edge corresponding to one real road on a map. These edges are classified into two categories: one-way and double-way. The car can travel along the former only in a single direction and along the latter in both directions. Every edge is assigned a unique ID, and the attributes of a segment contain its location, length, road grade, the number of lanes and name of this segment.

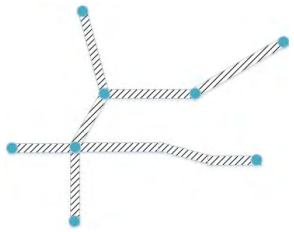


FIGURE 1. Road structure.

Definition 4: An intersection is a terminal of one certain road segment or one common end of several segments. Every intersection is associated with its own ID and location.

Definition 5: A road network is a directed graph $G(V, E)$, which is comprised of the set of road segments E and the according set of intersections V.

Definition 6: A path is a segment sequence where two arbitrary, adjacent segments are connected, i.e., one car can traverse the segments in the sequence from beginning to end one by one.

Definition 7: A subgraph is defined in the following passage. Given a directed graph $G'(V', E')$, $V' \subseteq V, E' \subseteq E$, for $\forall e_i, e_j \in E'$, one of the constraints needs to be satisfied:

1) In the graph G', there must at least be one path from e_i to e_j or it's inverse.
2) There is another edge, $e_k \in E'$ so that both the path from e_i to e_k and the path from e_j to e_k exist. (Every segment that appears in the two paths pertains to E').

Such graph like G' is a subgraph of the road network $G(V, E)$.

Fig. 1 shows the original roads just like those that appear on ordinary maps. From Fig. 1, we only distinguish whether two neighboring roads share the same terminal rather than at least a path can be formed from one road to the other. On the basis of the structure of roads, the road network can be modeled as shown in Fig. 2. Road segments are divided into two kinds: One-way (rendered in orange) and Double-way (rendered in purple). The direction of a segment is represented in the form of an intersection pair. For example, the direction of road segment e_0 is (v_0,v_1) and for the segment e_4, its direction is (v_4,v_5) and (v_5,v_4). Though v_1 is not only e_0's end but also e_1's end, the sequence (e_0, e_1) is not a path, but (e_0,e_3,e_5), (e_2,e_1,e_3,e_5) and so on are paths. $V' = \{v_0, v_1, v_2, v_5, v_7\}$ and $E' = \{e_0, e_1, e_3, e_5\}$ constitute a subgraph G' of the whole road network. Although no path can be found between the segment e_0 and e_1, both (e_0,e_3) and (e_1,e_3) are paths at the same time.

III. MAP-MATCHING

When we use graph models (road segments as vertices, intersections as edges or road segments as edges, intersections

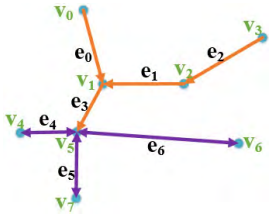


FIGURE 2. Road network.

Algorithm 1 Map-Matching

Input: a trajectory $Tr(P_1, P_2, ..., P_n)$, the road network $G(V, E)$, a distance threshold r

Output: the matched path $Pa(e_1, e_2, ..., e_m)$

1: initialize p as an empty point geometry and Pa as an empty list
2: **for** i in 1:n **do**
3: $p.X = p_i.LON$
4: $p.Y = p_i.LAT$
5: candidate = projectionOperation(p,G,r)
6: **if** candidate is not empty **then**
7: select the nearest road segment e from the candidate generation
8: query the corresponding intersection set of segment e
9: **if** the size of intersection set equals 2 **then**
10: distinguish the moving direction of p_i on current segment
11: put the directional e into the list Pa
12: **return** Pa;

as vertices) to analyze the urban road network, the statistical traffic volume or flow rate is usually allocated to every segment as respective weight. However, the raw GPS log is not able to be directly utilized for traffic statistic data. This is why Map-Matching is always taken as an indispensable preprocessing.

Map-Matching is inevitable as a fundamental preprocessing in the trajectory-based analysis and applications. Caused by the sampling errors and frequency of terminal devices, the GPS data with relevant longitude, latitude, timestamp, and other information is not completely accurate. Therefore, we need to map these observed GPS points onto exact corresponding road segments on a real road network. This procedure of establishing the mapping between original GPS points and roads is called Map-Matching. In this paper, we make use of the taxi GPS trajectory data and the road network of Shanghai, China. Thereinto, the time interval of trajectory data is 10s. According to the sampling frequency, our Map-Matching process is designed to deal with a high sampling rate of GPS data. It is no doubt that the longer the time interval of GPS data sampling is, the sparser the points become, the more steps of matching process are taken, the more difficult Map-Matching is, and thereby the usability of trajectory data itself get worse.

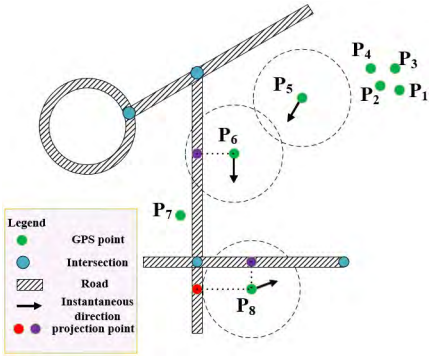


FIGURE 3. Map-Matching schematic diagram.

The detailed procedure of a Map-Matching algorithm is illustrated in Algorithm 1. First, the program loads current point and then project the objective GPS point into its adjacent candidate road segments within a circle of radius r, respectively (line 5). If this candidate generation is empty (p_5 in Fig. 3), the point would be neglected. In this case, we think that the car has no effect on the traffic condition of the roads around it. Otherwise, select the nearest road segment e from the candidate generation as the matched edge (line 7), e.g., there are two segments in the candidate set of the point p_8 in Fig. 3, with one mapping point inside each of them (painted in purple and red separately). From the two segments, we choose the one with the nearest mapping point. Besides the nearest road, we still need to know the direction of the car's movement from one terminal to the other (the order of two intersections) according to the instantaneous direction of the current point (line 10). The instantaneous direction ranges from 0° to 359.9° and this range is divided into 4 subranges: [45°,135°), [135°,225°), [225°,315°), and [315°,45°). The instantaneous direction which falls in the four different subranges reveals respectively the eastward, southward, westward, and northward motion trend. If the instantaneous direction of a point is within [315°,45°), then an eastward movement tendency is presented, so it is reasonable to judge that the car travels from the western intersection to the other. For the road segments with only one intersection (the circular road in Fig. 3), we are not able to distinguish the motion direction of a point, because no matter which direction the car travel towards, the intersection pair are all the same. The time cost of the Map-Matching algorithm is $O(nk)$, where k denotes the maximum size of all candidate generations.

IV. CONGESTION DETECTION

After the process of the aforementioned Map-Matching, all available GPS points are projected into the corresponding road segments. Then we can receive the information of the road network's traffic conditions at that time with the help of the mass trajectory data.

To detect whether the drivers suffer from traffic congestion on a road segment along one direction or both during a specified time interval, the Traffic State Index (TSI) is

adopted as the evaluation criteria [11], which is proposed by Shanghai urban and rural construction and the Transportation Development Research Institute. Basic TSI, defined as (2), is a relative value that evaluates the traffic conditions of urban roads within a certain period of time.

$$TSI = \frac{v_f - v_i}{v_f}. \tag{1}$$

$$v_i = \frac{1}{n}\sum \frac{dist(p_{(k,i,last)}, p_{(k,i,first)})}{p_{(k,i,last).t} - p_{(k,i,first).t}} \tag{2}$$

In (2), v_f represents the free flow speed of a segment i,i.e., the average velocity of the segment bearing a light load. v_i denotes the actual average speed at which cars travel along this road during a given time interval. v_i is computed by (2), where $p(k, i, last)$ and $p(k, i, first)$ are the last mapping point and the first mapping point lying on the segment i of the car k during the given period, respectively. The function $dist$ outputs the spacial distance between the two points. $P_{(k,i,first/last).t}$ is the timestamp of this mapping point. v_i is just the average speed of those cars which pass by the segment i within that time interval by chance. n is not allowed to be less than a threshold, because if n is very small, the result from only a few cars cannot reflect the actual average speed well.

The TSI quantifies the congestion degree of the road network and it ranges from 0 to 1. The greater the index is, the more crowded the drivers feel on the segment. The segment is regarded as congested when its TSI is beyond the fixed value. However, the traffic condition of the entire road network varies dynamically over time. It is impossible to get all traffic data of every moment for the study of urban road congestion. In addition, if a congestion happens on a segment, it would last for a period of time. Hence, as shown in Fig. 4, we divide a given time interval into $(n-1)$ equal-size pieces in terms of a unit time such as 5 mins. Next, all trajectories within the time interval are extracted to compute the average speeds of every segment during different time pieces. In each time piece, we estimate whether one segment is congested according to its average speed and TSI. As a result, the original continuous time interval is replaced by some discrete slices.

Road segments are classified into two categories: one-way and double-way. For the former, the orientation of congestion, which occurs on it, is consistent with the segment's direction. But for double-way road segments, it is a bit more complicated. In most cases, a congestion on the double-way segment is usually along one direction of the segment instead of both. It is also possible that both directions of the segment are congested in the meantime. In addition, the road segment is static without changing over time, while the congestion on the segment fluctuates dynamically. For instance, there is a double-way segment e with two intersections v_1 and v_2. A driver suffers from traffic jam in the morning along the orientation $(v_1 \rightarrow v_2)$ and again experiences traffic congestion in the direction $(v_2 \rightarrow v_1)$ during the afternoon rush hour unfortunately. Regardless of the jam in the morning or the second one, the segment e is constant all the time while the directions of the two traffic congestions are different. Thus the segment reflects the connectivity of the road network, while the congestion demonstrates the real-time traffic conditions. On account of the distinction between congestion and segments, congestion edge is defined below.

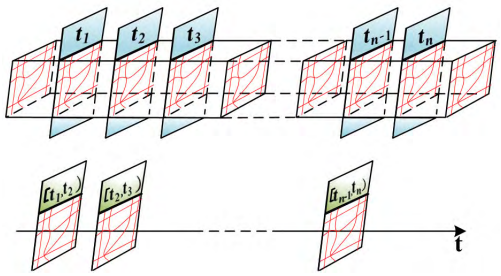


FIGURE 4. The road network in each time piece.

Definition 8: Congestion edge. If a congestion takes place on a road segment during a specific time piece t, then a congestion edge is formed. In case only one direction of the bidirectional segment is congested, the 1st rule is applied. Otherwise, the 2nd rule is complied with.

1) Remove the direction in contrast with the Congestion's direction from the road segment, the left unidirectional segment is the congestion edge.
2) Congestion edge is equivalent to road segment (Definition 3).

V. FILL MISSING CONGESTION EDGES

The daily trajectories are derived from more than 10, 000 local taxis in Shanghai. However, compared with the tremendous quantity of the road segments and overall motor vehicles in Shanghai, the trajectories are still relatively sparse. Taxis mainly appear in urban commercial districts and run along primary roads. Besides, only those segments whose traffic flow exceeds a threshold value during a period are determined whether it is congested. Caused by this, congestion edges detected from these trajectories aggregate in the core regions of this city. On the other hand, there are still some missing road segments even in the core areas. For the former, enlarging the volume of trajectories is the only solution to solve it. To tackle the latter, we design two strategies of filling missing congestion edges from both temporal and spatial perspectives, respectively.

A. TEMPORAL VIEW

From a temporal viewpoint, we pay close attention to how the congestion on the same road segment changes over time pieces. Given three contiguous time pieces, there are two congestion edges from the first and third time piece separately. Furthermore, the two congestion edges with an identical direction occur on the same road segment, and if no congestion edge on that segment can be found out within the middle time piece, then a new congestion edge consistent with those two would be inserted into the second time piece

Algorithm 2 Fill Edges From Temporal View

Input: Congestion edges set at each time piece $Layer(s_1, s_2, \ldots, s_t)$

Output: the new congestion edges set at each time piece $Layer'(s'_1, s'_2, \ldots, s'_t)$

1: $i = 2$
2: **while** $i < t$ **do**
3: initialize *miss* as an empty set
4: $common = s_{i-1} \cap s_{i+1}$
5: $miss = common - s_i$
6: **if** *miss* is not empty **then**
7: $s'_i = s_i \cup miss$
8: **else**
9: $s'_i = s_i$
10: $i = i + 1$
11: $s'_1 = s_1, s'_t = s_t$
12: **return** $Layer'(s'_1, s'_2, \ldots, s'_t)$

FIGURE 5. Filling missing congestion edges from spatial view.

(as shown in Algorithm 2). This is because the congestion on a road segment usually lasts for a certain time and is not likely to transform between existence and nonexistence frequently during contiguous time pieces. If we assume that there are at most m congestion edges at one time piece, the time complexity of algorithm 2 is $O(tm^2)$.

B. SPATIAL VIEW

From the spatial standpoint, we focus on the spatial distribution of congestion edges at each time piece. Given three road segments e_i, e_j and e_k, they can constitute a path $Pa(e_i, e_j, e_k)$. At a certain time piece, two congestion edges e'_i and e'_k are formed on the segments e_i and e_k respectively. In addition, there is no congestion edge on the segment e_j. If these conditions are satisfied, we aim to judge whether a new congestion edge e^*_j can be inserted on segment e_j so that (e'_i, e^*_j, e'_k) can be a path. If not, no new congestion edge would be added. In particular, e^*_j would be bidirectional if and only if e'_i, e'_k and e_j are all bidirectional. For example, as shown in Fig. 5, only pink congestion edges at current time piece are found through the procedure of congestion detection and the two yellow ones are filled afterwards. There are no congestion edges on the other light grey road segments (the directions of all road segments are ignored and the directions of congestion edges are displayed in form of arrows). In reality, congestion always happens on one or more road segments at first and then its or their neighbours would be infected after some time. That is why our strategy of filling congestion edges from a spatial view is necessary. But if the new congestion edge to be filled is extremely long, we would compare it with its adjacent congestion edges in terms of length to determine whether this congestion edge can be added or not. Algorithm 3 illustrates the process in detail. The complexity of algorithm 3 is $O(tkm)$, where k denotes the maximum size of the candidate segment set and m is the maximum number of congestion edges among the t time pieces.

Algorithm 3 Fill Edges From Spacial View

Input: the congestion edges set at each time piece $Layer(s_1, s_2, \ldots, s_t)$, the road network $G(V, E)$

Output: the new congestion edges set at each time piece $Layer'(s'_1, s'_2, \ldots, s'_t)$

1: **for** i in 1:t **do**
2: select from E the candidate segment set *canSeg* each of which shares at least one identical intersection with the congestion edges from s_i
3: **for** j in *canSeg* **do**
4: $origin, end = j.direction[1:2]$
5: $originSet = startFrom(s_i, G)$, $endSet = goTo(s_i, G)$
6: initialize an empty congestion edge *nEdge*
7: **if**($origin \in endSet, end \in originSet$) **then**
8: add *j.ID* into *nEdge*
9: add the direction ($origin \rightarrow end$) into *nEdge*
10: **if** j is double-way **then**
11: $origin, end = end, origin$
12: execute line 7-10 again
13: **if** *nEdge* is not empty **then**
14: add *nEdge* into s_i
15: $s'_i = s_i$
16: **return** $Layer'(s'_1, s'_2, \ldots, s'_t)$

VI. CONSTRUCTING SPACE-TEMPORAL CONGESTION SUBGRAPH

In this section, we demonstrate how to build the Space-Temporal Congestion Subgraph (STCS). STCS is defined as below. Different from the arbitrary subgraph of the road network, STCS is comprised of a group of congestion edges. These congestion edges are formed successively during several contiguous time pieces and are connected to each other in space. Initially, congestion occurs only on a few road segments rather than the majority of them. After some time, if the congestion on these segments is still not eased, other segments adjacent to those ones would be affected and are very likely to become newly congested segments. In an extreme case, the traffic congestion can be propagated over a huge extent of the road network. Thus, STCS is constructed to clearly describe this contagion phenomenon.

Algorithm 4 Construct STCS AdjTable

Input: the congestion edges set at each time piece $Layer(s_1, s_2, \ldots, s_t)$, the road network $G(V, E)$, and the time lag threshold T

Output: the adjacency table $adjT$ of all STCSes

```
adjList = list(Layer(s_1, s_2, ..., s_t))
i = T + 1
while i <= t do
    newLayer = mergeLayers(adjList[(i − T) : (i)])
    newLayer = connectNewLayer(newLayer, T, G)
    adjList[i] = newLayer
    i+ = T
if i not equal t then
    newLayer = mergeLayers(adjList[i ::])
    newLayer = connectNewLayer(newLayer, T, G)
    adjList[t] = newLayer
adjT = adjList[t]
return adjT
```

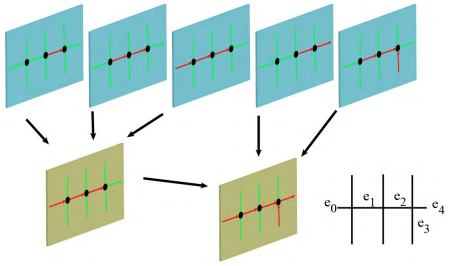


FIGURE 6. The process of constructing a STCS.

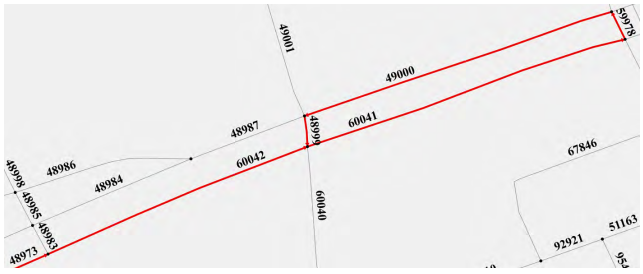


FIGURE 7. Ring structure in STCSes.

Through the process of congestion detection, we get the lists of congestion edges at each time piece. Then we apply two strategies to fill those missing edges. After the two steps, we can start to construct STCSes over all time pieces. All STCSes are stored in the same adjacency table. Therefore, the key to tackling this problem is how to build such an adjacency table. As presented in Algorithm 4, it is mainly comprised of two stages: combination and connection. The combination refers to merging those congestion edges within current sliding window as a new time piece. Because congestion on a segment often maintains for several time pieces, one congestion edge is formed separately at each of these time pieces. In this case, these repeated edges are merged into a new one, but their occurrence time (time piece number) are recorded in ascending order by the new one. The length of a sliding window is $T + 1$. On the connection stage, we judge whether two arbitrary edges within a current sliding window are connected according to their occurrence time based on Definition 9, particularly if a congestion edge is produced on the combination stage, its occurrence time has multiple values and we select the largest one,i.e., the last time piece at which the congestion happens. The sliding window moves T steps forward once until the end. The time complexity of algorithm 4 is $O(tm^2)$, where m is the maximum number of congestion edges at one time piece.

For example, there are 5 time pieces in Fig. 6 and T is set to 2. At each time piece, congestion edges are painted as red arrows and the green lines represent uncongested road segments (the directions of segments are neglected). The initial sliding window contains the first three time pieces. The congestion on the road segment e_2 lasts throughout the sliding window, so three repeated edges are replaced by a new one. After the combination stage, we connect two arbitrary adjacent edges whose time lag between them is not beyond the threshold T. Then the sliding window produces a new time piece and we place it at the beginning of the remaining time pieces and move 2 time pieces forward to enter next iteration.

Definition 9: Space-Temporal Congestion Subgraph (STCS). There is a directed graph $G_c(V_c, E_c)$ where V_c is the set of intersections and E_c is the congestion edge set over consecutive time pieces. For $\forall e, e' \in E_c$, the two edges are connected in STCS if they are connected in space and $|e.t - e'.t| <= T$. Given the road network $G(V, E)$, $V_c \subseteq V$, the corresponding road segments E_s of E_c, $E_s \subseteq V$, if the graph $G_s(V_c, E_s)$ is a subgraph of $G(V, E)$, then $G_c(V_c, E_c)$ is a STCS.

In general, we always tend to destroy the ring structure by removing certain edges in a graph; however, the situation where a ring structure exists in a STCS is allowed to arise. As shown in Fig. 7, 6 congestion edges constitute a STCS and four of them form a ring structure. This ring structure explains that the 4 edges in the ring influence each other. Especially, the edge on segment 48999 and another edge on segment 59978 are not detected directly in terms of TSI. Since the two road segments are too short, it is very difficult to catch enough GPS points for congestion detection. However they are still added into this STCS via filling missing edges, otherwise the current STCS would break up into 2 STCSes.

VII. DISCOVER FREQUENT CONGESTION SUBGRAPH

Although the congestion on the road network varies dynamically over time pieces, there still is a periodical pattern. The inherent pattern refers to the frequent STCSes over the same time pieces every day. From a STCS, we can make out how the congestion on initial one or a few road segments propagates across the corresponding adjacent segments iteratively. For common itemsets, well-developed algorithms such

Algorithm 5 Find Frequent STCSes

Input: the STCS sets within same time pieces during different days $sg(g_1, g_2, \ldots, g_k)$, and the support threshold δ

Output: the frequent STCS set $sg'(g'_1, g'_2, \ldots, g'_p)$, g'_i denotes a frequent STCS

```
initialize an empty frequent STCS set freq
freq = FPgrowth(sg,δ)
delete redundant result sets from freq
for ∀x, y ∈ freq do
    if x ∩ y ≠ ∅ then
        z = x ∪ y
        delete x and y from freq
        put z into freq
sg' = freq
for i in sg' do
    if i constitutes z STCSes then
        break up i into sg'_i1, sg'_i2,...,sg'_iz
        put sg'_i1, sg'_i2, ... ,sg'_iz into sg'
return sg'
```

as Apriori [12] and FP-growth [13] are quite suitable, but for STCS sets, these algorithms are not able to be utilized directly for the following two reasons. First, there is not any dependency relationship between common itemsets, whereas the edges belonging to the identical STCS set constitute a subgraph according to spatio-temporal connections. Second, these algorithms do not guarantee that congestion edges that pertain to the same result set compose a STCS. Therefore, we design several additional procedures on the basis of the FP-growth (as illustrated in algorithm 5).

For those result sets outputted directly by FP-growth where the congestion edges are not able to compose an integrated STCS, they can be divided into 2 cases. One is that its superset also belongs to the result sets, in other words, another STCS is found frequent, which contains all congestion edges in current result set. Thus, this result set is redundant and we remove such result sets. The other case is that a result set's superset do not exist in the result sets, but it has intersection with several result sets of the same type. This means that an originally large STCS is decomposed into some smaller and unconnected parts after FP-growth. Caused by the trajectories data sparseness, certain congestion edges cannot be formed frequently during most days. Thus, we combine these result sets together to recover the original STCS. If the new union set is still not connected entirely, then we break up the set into several corresponding connected parts.

VIII. EXPERIMENT

In this section, we evaluate our algorithms based on real road network and trajectory data. The experiments are performed on the geospatial processing platform Arcgis, of which Arcmap provides the visualized analysis for traffic data and Arcgis server can deal with large scale trajectory data.

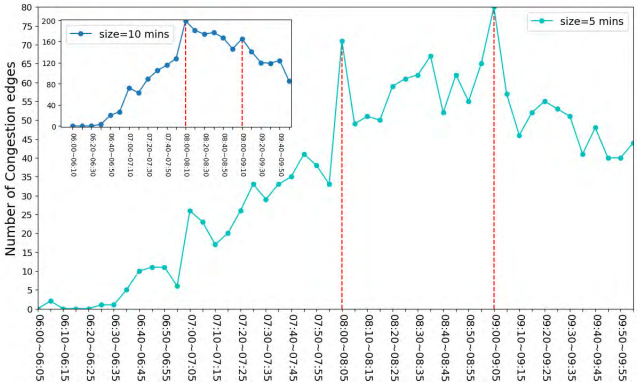


FIGURE 8. The number of congestion edges at each time piece.

A. DATA

We utilize the road network of Shanghai with 79, 148 intersections and 113, 765 road segments among which 25.49% are one-way. The trajectories are mainly comprised of over 12, 000 taxis from 8am to 9am during workdays in April, 2015.

B. EXPERIMENT ON CONGESTION DETECTION

The threshold for n in (2) is set as 10, thus only those segments whose traffic volume is not smaller than 10 are considered regardless of whether or not they are congested. The threshold for TSI is 0.70 [11]. Fig. 8 shows the number of congestion edges at various time pieces from 6am to 10am on Apr.1,2015. Whether the size of one time piece is 5 minutes or 10 minutes, they have the same variation tendency. Before 8am, only a few segments are congested and the growth of the value is relatively slow. These congested road segments disperse on the road network and most of them are not able to propagate the congestion to their adjacent segments in a short time to form a local congested area. On the contrary, there is an intense increase when it approaches 8am and the value maintains a high level from 8am to 9am. During this time interval, congested segments are distributed closely, it is more likely for them to infect their neighbours so that a large region filled with congested segments can be shaped. Therefore, we mainly study the congestion propagation over time pieces between 8am and 9am.

C. EXPERIMENT ON FILLING MISSING CONGESTION EDGES

To evaluate the effectiveness of two strategies of filling missing congestion edges, we compare 4 means from the combination of two strategies with each other. As shown in Fig. 9, if we do not fill any congestion edges into original data, the number of edges over time pieces fluctuates strongly and maintains a low level in the meantime. Only applying either of the two strategies can improve this situation effectively. No matter which one is adopted, they all keep the value on a quite similar level. The last manner that combines the two strategies is most effective and makes the change of the

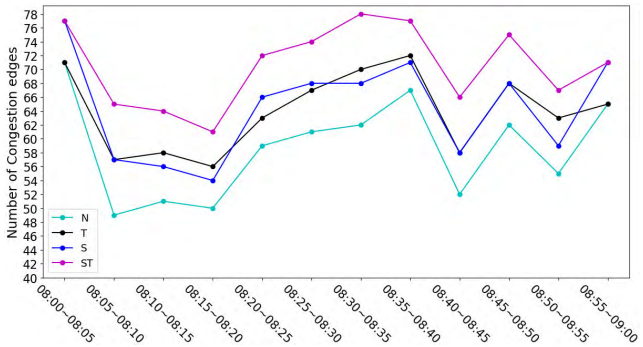


FIGURE 9. Different ways of filling missing congestion edges (N: none missing edges are filled. T: add congestion edges from temporal viewpoint. S: recover missing congestion edges from spatial view. ST: combine the two strategies together).

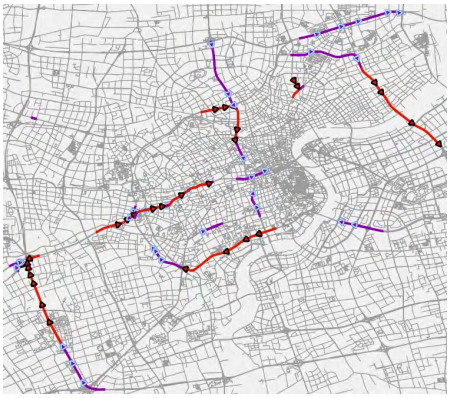


FIGURE 10. STCSes with at least 4 edges constructed from time pieces between 8am and 9am on Apr.1,2015.

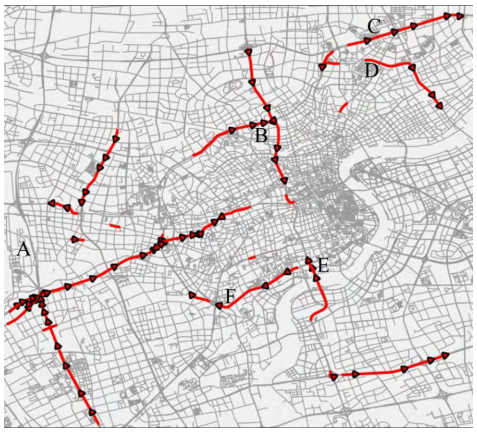
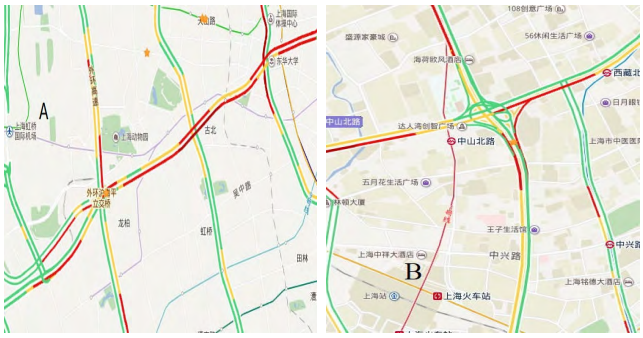


FIGURE 11. Frequent STCSes between 8am and 9am in the workdays of April 2015. (A: Hongqiao Airport, B: Shanghai Station, C: Fudan University, D: Tongji University, E: World Exposition Museum, and F: Shanghai Stadium).



(a) (b)

FIGURE 12. Snapshots of traffic condition in Shanghai road network at 8am on Apr.9,2018 from Baidu Map. (congested roads are rendered in red or dark red). (a) Congestion near A (Hongqiao Airport). (b) Congestion near B (Shanghai Station).

number over time smoother. In addition to the comparison on the number of congestion edges, we pay more attention to these filled edges' influence on the construction of STCS. In Fig. 10, the congestion edges rendered in red are received directly through congestion detection and the left labeled in purple are added via filling missing edges. We can see that the effect of these filled edges is manifest in two aspects. One hand is that some STCSes would become smaller (fewer edges) or even disappear without the filled edges because the STCS would break up into several smaller STCSes. On the other hand, filled edges make these large-size STCSes close to each other in space to delineate the latent traffic congestion propagation pattern better.

D. EXPERIMENT ON FREQUENT CONGESTION SUBGRAPH

The time lag T is set at 5 and the support threshold δ equals 7 (one third of overall weekdays). Each time piece's size is 5 minutes. Frequent STCSes between 8am and 9am during workdays in April 2015 are shown in Fig. 11. There are 8 large-size congestion subgraphs with at least 4 congestion edges, of which the biggest one contains 43 edges. Most large-size frequent subgraphs appear on the ring expressway or elevated road and the rest of frequent STCSes exist on general primary road segments. Besides, large-size frequent subgraphs are mainly distributed near some POI (e.g., the biggest subgraph is adjacent to the Hongqiao Airport). Three of four small-size STCSes (two edges at most) are close to a subway station. Because the entire time interval spans the morning rush hours, the directions of most edges are toward the core area of the city. We can see that large-size STCSes are not very far away from each other. Caused by unusual events such as bad weather, business promotions and disasters, local congested areas can lead to the cascading failure of the whole network. Fig. 12 gives two snapshots of congested roads at 8am on Apr.9,2018 from Baidu Map. The congested roads in Fig. 12 are consistent with a portion of edges belonging to the corresponding frequent subgraphs in Fig. 11. The same conclusion holds true for the other subgraphs in Fig. 11.

Fig. 13 presents the frequent congestion subtrees through the method proposed by [6]. It is obvious that our approach outperforms the algorithm in [6] by comparing the subgraphs in Fig. 11 with the subtrees in Fig. 13. More specifically, there are 49 subgraphs in Fig. 11 and only 27 subtrees are

FIGURE 13. Frequent congestion subtrees between 8am and 9am in the workdays of April 2015. (A: Hongqiao Airport, B: Shanghai Station).

constructed in Fig. 13. This means that our model can discover more frequent urban congestion propagation behaviors. Besides the strength in numbers of substructures, our method also has the advantage of discovering large-size substructures. 8 large-size congestion subgraphs with at least 4 congestion edges exist in Fig. 11 while only 3 large-size congestion subtrees are found in Fig. 13. Especially, the large-size congestion subgraph near B in Fig. 11 shrinks into a small-size congestion subtree with only 2 congestion edges in Fig. 13. The largest frequent congestion subgraph below A in Fig. 11 is decomposed into two corresponding congestion subtrees in Fig. 13 because of the absence of certain congestion edges. Such situation reveals that the approach introduced in [6] can not catch the complete traffic congestion propagation process very well. In essence, a congestion subtree is a special case of congestion subgraph. Therefore, our approach is more effective in discovering urban traffic congestion propagation pattern comparing with the method proposed in [6].

IX. RELATED WORK

A. TRAJECTORY DATA PROCESSING

Map-Matching algorithms are divided into two kinds according to the GPS points' sampling frequency. For the low-sampling-rate trajectory data, Lou *et al.* [14] devised a global algorithm named ST-Matching. ST-Matching takes the speed constraints into account. On the basis of ST-Matching, Yuan *et al.* [15] proposed a voting-based matching (IVMM) to consider the mutual influence between points. Due to the complexities of the two algorithms, they are not suitable for large-scale trajectory data. There has been much research done on taxicab trajectory data. In [16]–[18], the trajectory is utilized to identify traffic anomalies via detecting the routing behavior. Liu *et al.* [19] adopted the community detection algorithms in taxi data to discover the travel of citizens. Huang *et al.* [20] constructed the TrajGraph as a visual analytics way to compare the various centralities of different road segments. Zhuang *et al.* [21] built the knowledge graph with the history trajectories to forecast people's interest in various places of a city.

B. URBAN TRAFFIC CONGESTION

In [6], the travel time on segments are obtained by the sensors deployed at each intersection. The congestion is determined in terms of a certain percentile of all historical travel time. A congestion tree is constructed to demonstrate the congestion propagation. However, the tree structure is not good enough to reflect the actual contagion of traffic congestion. Lv *et al.* [1] developed a congestion detection system with the help of mobile phones' cellular signal. Rempe *et al.* [2], Huang *et al.* [22], and Wang *et al.* [23] introduced the method of discovering congestion clusters in the road network and investigated the correlations between these clusters. Wang *et al.* [24] studied the driver sources to understand the usage of road segments. Hong *et al.* [25] modeled the trajectory data as Spatio-Temporal Graph(STG) to find traffic anomalies in the city(block holes and volcanos). Wen *et al.* [5] proposed the Flow-based PageRank(FBPR) algorithm to study the traffic demand of road segments and discover congestion areas. In [26], an integrated computing method combing Bayesian information criterion and maximum likelihood estimation was proposed to obtain important human mobility patterns.

C. COMPLEX NETWORK PROPAGATION

The linear threshold model, cascade model, and percolation model are usually employed to investigate the diffusion dynamics on the complex network (e.g. messages, contagion and so on propagate on the social network) [7]–[9], [27]. Daqing *et al.* [28] defined traffic congestion as a behavior of cascading failure and found that long-range correlations of failures in space follow a power law decay. In [29], the relationships between local traffic flow and global flow are focused on and the critical bottlenecks in the network can be found out after the process of traffic percolation. Zhao *et al.* [30] investigated the spreading dynamics of failures with the cascading overload model.

D. FREQUENT PATTERN MINING

Apriori [12] and FP-growth [31] are usual algorithms for discovering frequent itemsets from large-scale data sets. FP-growth only needs to scan the database twice and is quicker than Apriori. However, the two algorithms cannot be applied directly to deal with graph structures without any modification while GSPAN [32] and SPIN [13] are designed for frequent graphs. The kernel of these algorithms is to solve the subgraph isomorphism problem and thus the complexity is rather high. On the other hand, because the structure of the road network is always constant, it is easy to distinguish two subgraphs regardless of whether they are the same.

X. CONCLUSION

In this paper, we investigate traffic congestion propagation based on the Shanghai taxi trajectory data. We design the

Map-Matching algorithm for high-sampling-rate GPS points. Congested road segments are detected through the index TSI. Because of the trajectory data sparseness, two strategies of filling missing congestion edges from temporal and spatial respectively are demonstrated. A dynamic graph structure named STCS built with congestion edges is proposed to describe the congestion phenomenon. According to the spreading dynamic of the traffic congestion, some post-processing steps are added on the basis of FP-growth to discover the frequent STCSes.

REFERENCES

[1] M. Lv, L. Chen, X. Wu, and G. Chen, "A road congestion detection system using undedicated mobile phones," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3060–3072, Dec. 2015.
[2] F. Rempe, G. Huber, and K. Bogenberger, "Spatio-temporal congestion patterns in urban traffic networks," *Transp. Res. Procedia*, vol. 15, pp. 513–524, Jan. 2016.
[3] T. Anwar, C. Liu, H. L. Vu, M. S. Islam, and T. Sellis, "Capturing the spatiotemporal evolution in road traffic networks," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 8, pp. 1426–1439, Aug. 2018.
[4] C. Guo, D. Li, G. Zhang, and M. Zhai, "Real-time path planning in urban area via VANET-assisted traffic information sharing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 5635–5649, Jul. 2018.
[5] T.-H. Wen, W.-C.-B. Chin, and P.-C. Lai, "Understanding the topological characteristics and flow complexity of urban traffic congestion," *Phys. A, Stat. Mech. Appl.*, vol. 473, pp. 166–177, May 2017.
[6] H. Nguyen, W. Liu, and F. Chen, "Discovering congestion propagation patterns in spatio-temporal traffic data," *IEEE Trans. Big Data*, vol. 3, no. 2, pp. 169–180, Jun. 2017.
[7] S. Pei, F. Morone, and H. A. Makse. (2017). "Theories for influencer identification in complex networks." [Online]. Available: https://arxiv.org/abs/1707.01594
[8] P. Holme. (2017). "Probing empirical contact networks by simulation of spreading dynamics." [Online]. Available: https://arxiv.org/abs/1706.09095
[9] F. Morone and H. A. Makse, "Influence maximization in complex networks through optimal percolation," *Nature*, vol. 527, no. 7579, p. 544, 2015.
[10] C. Zhu, V. C. M. Leung, J. J. P. C. Rodrigues, L. Shu, L. Wang, and H. Zhou, "Social sensor cloud: Framework, greenness, issues, and outlook," *IEEE Netw.*, vol. 32, no. 5, pp. 100–105, Sep./Oct. 2018.
[11] *Shanghai Transportation*. Accessed: Apr. 2018. [Online]. Available: http://www.jtcx.sh.cn/zhishu/jiedu.html
[12] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. Int. Conf. Very Large Data Bases*, 1994, pp. 487–499.
[13] J. Huan, W. Wang, J. Prins, and J. Yang, "SPIN: Mining maximal frequent subgraphs from graph databases," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 581–586.
[14] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," in *Proc. ACM SIGSPATIAL Int. Symp. Adv. Geographic Inf. Syst. (ACM-GIS)*, Seattle, WA, USA, Nov. 2009, pp. 352–361.
[15] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *Proc. 11th Int. Conf. Mobile Data Manage.*, May 2010, pp. 517–520.
[16] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi, "Crowd sensing of traffic anomalies based on human mobility and social media," in *Proc. ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2013, pp. 344–353.
[17] Z. Wang *et al.*, "Privacy-preserving crowd-sourced statistical data publishing with an untrusted server," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2018.2861765.
[18] H. Zhou, H. Wang, X. Chen, X. Li, and S. Xu, "Data offloading techniques through vehicular ad hoc networks: A survey," *IEEE Access*, to be published, doi: 10.1109/ACCESS.2018.2878552.
[19] X. Liu, L. Gong, Y. Liu, and Y. Gong, "Revealing travel patterns and city structure with taxi trip data," *J. Transp. Geogr.*, vol. 43, pp. 78–90, Feb. 2015.
[20] X. Huang, Y. Zhao, J. Yang, C. Zhang, C. Ma, and X. Ye, "Trajgraph: A graph-based visual analytics approach to studying urban network centralities using taxi trajectory data," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 1, pp. 160–169, Jan. 2016.
[21] C. Zhuang, N. J. Yuan, R. Song, X. Xie, and Q. Ma, "Understanding people lifestyles: Construction of urban movement knowledge graph from GPS trajectory," in *Proc. IJCAI*, 2017, pp. 3616–3623.
[22] C. Huang, Y. Chen, S. Xu, and H. Zhou, "The vehicular social network (VSN)-based sharing of downloaded geo data using the credit-based clustering scheme," *IEEE Access*, to be published, doi: 10.1109/ACCESS.2018.2873905.
[23] Z. Wang *et al.*, "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2018.2861393.
[24] P. Wang, T. Hunter, A. M. Bayen, K. Schechtner, and M. C. González, "Understanding road usage patterns in urban areas," *Sci. Rep.*, vol. 2, Dec. 2012, Art. no. 1001.
[25] L. Hong, Y. Zheng, D. Yung, J. Shang, and L. Zou, "Detecting urban black holes based on human mobility data," in *Proc. SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2015, Art. no. . 35.
[26] F. Xia, J. Wang, X. Kong, Z. Wang, J. Li, and C. Liu, "Exploring human mobility patterns in urban scenarios: A trajectory data perspective," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 142–149, Mar. 2018.
[27] H. Zhang, Y. Qi, H. Zhou, J. Zhang, and J. Sun, "Testing and defending methods against DOS Attack in state estimation," *Asian J. Control*, vol. 19, no. 3, pp. 1295–1305, 2017.
[28] L. Daqing, J. Yinan, K. Rui, and S. Havlin, "Spatial correlation analysis of cascading failures: Congestions and blackouts," *Sci. Rep.*, vol. 4, Jun. 2014, Art. no. 5381.
[29] D. Li *et al.*, "Percolation transition in dynamical traffic network with evolving critical bottlenecks," *Proc. Nat. Acad. Sci. USA*, vol. 112, no. 3, pp. 669–672, 2015.
[30] J. Zhao, D. Li, H. Sanhedrai, R. Cohen, and S. Havlin, "Spatio-temporal propagation of cascading overload failures in spatially embedded networks," *Nature Commun.*, vol. 7, Jan. 2016, Art. no. 10094.
[31] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining Knowl. Discovery*, vol. 8, no. 1, pp. 53–87, 2004.
[32] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2002, pp. 721–724.

ZHENHUA CHEN received the B.E. degree in software engineering from Jilin University, Changchun, Jilin, China, in 2016, where he is currently pursuing the degree in computer system architecture. His current research interests include urban computing.

YONGJIAN YANG received the B.E. degree in automatization from the Jilin University of Technology, Changchun, Jilin, China, in 1983, the M.E. degree in computer communication from the Beijing University of Posts and Telecommunications, Beijing, China, in 1991, and the Ph.D. degree in software and theory of computer from Jilin University, Changchun, in 2005. He is currently a Professor and a Ph.D. Supervisor at Jilin University, the Vice Dean of the Software College, Jilin University, the Director of the Key Laboratory under the Ministry of Information Industry, the Standing Director of the Communication Academy, and a member of the Computer Science Academy of Jilin Province. His research interests include network intelligence management, wireless mobile communication and services, and wireless mobile communication.

LIPING HUANG received the master's degree from the College of Computer Science and Technology, Jilin University, Changchun, China, in 2011, where she is currently pursuing the Ph.D. degree. Her current research interests include trajectory computing, data mining, machine learning, urban computing, complex networks, and traffic data analysis.

EN WANG received the B.E. degree in software engineering, the M.E. degree in computer science and technology, and the Ph.D. degree in computer science and technology from Jilin University, Changchun, in 2011, 2013, and 2016, respectively. He is currently an Associate Professor with the Department of Computer Science and Technology, Jilin University. He is also a Visiting Scholar with the Department of Computer and Information Sciences, Temple University, Philadelphia. His current research interests include the efficient utilization of network resources, scheduling and drop strategy in terms of buffer-management and energy-efficient communication between human-carried devices, and mobile crowdsensing.

DAWEI LI received the bachelor's degree (Advanced Class) from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, Hubei, China, and the Ph.D. degree from the Department of Computer and Information Sciences, Temple University, in 2016. He is currently an Assistant Professor with the Department of Computer Science, Montclair State University. His research interests include energy-aware task scheduling on multi-cores/multiprocessors, data center networks, cloud computing, and big data processing.