Predicting Traffic Congestion Propagation Patterns: A Propagation Graph Approach

Haoyi Xiong
The University of Iowa
haoyi-xiong@uiowa.edu

Amin Vahedian
The University of Iowa
amin-vahediankhezerlou@uiowa.edu

Xun Zhou
The University of Iowa
xun-zhou@uiowa.edu

Yanhua Li
Worcester Polytechnic Institute
yli15@wpi.edu

Jun Luo
Machine Intelligence Center, Lenovo Group Limited
jluo1@lenovo.com

ABSTRACT

A traffic congestion in a road network may propagate to upstream road segments. Such a congestion propagation may make a series of connected road segments congested in the near future. Given a spatial-temporal network and congested road segments in current time, the aim of predicting traffic congestion propagation pattern is to predict where those congestion will propagate to. This can provide users (e.g. city officials) with valuable information on how congestion will propagate in the near future to help mitigating emerging congestions. However, it is challenging to predict in real-time due to complex propagation process between roads and high computational intensity caused by large dataset. Recent studies have been focusing on finding frequent or most likely congestion propagation patterns in historical data. In contrast, this research will address the problem of predicting congestion propagation patterns in the near future. We predict the footprint of congestion propagation as Propagation Graphs (Pro-Graphs) where the root of each Pro-Graph is a set of congested roads propagating congestion to nearby roads. We propose an efficient algorithm called PPI_Fast to achieve this prediction. Our experiments on real-word dataset from Shenzhen, China shows that the PPI_Fast is able to predict near future propagations with AUC of 0.75 and improves the running time of the baseline algorithm. Two case studies have been done to show our work can find meaningful patterns.

CCS CONCEPTS

• **Information systems → Geographic information systems**;

KEYWORDS

Traffic Congestion, Congestion Propagation, Spatio-Temporal data Mining.

IWCTS'18, November 6, 2018, Seattle, WA, USA

ACM ISBN 978-1-4503-6037-1/18/11…$15.00
https://doi.org/10.1145/3283207.3283213

ACM Reference Format:
Haoyi Xiong, Amin Vahedian, Xun Zhou, Yanhua Li, and Jun Luo. 2018. Predicting Traffic Congestion Propagation Patterns: A Propagation Graph Approach. In *11th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS'18), November 6, 2018, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3283207.3283213

1 INTRODUCTION

Traffic congestion is an ongoing challenge in most urban settings. Frequent congestion can cause significant green-house gas emissions, low transportation efficiency, and a lower quality of life for urban residents. Traffic congestion often propagates among road segments in an urban road network, causing an expanding wave of delay. Predicting such propagation patterns is important, because it can provide citizens and city officials with valuable information on what road segments will be impacted in the near future (i.e. minutes or hours in a day, e.g. 1 hour), given the current congested road segments. This information is valuable in routing services as well as for public safety, to mitigate emerging congestion problems via real-time response and add in structural change planning. Past studies on traffic congestions have used various approaches, such as expert system [5], cell transmission model [18], and shock-wave model [6], based on simulated data or small amount of real-world observation data. To account for various situations in real-world, a large number of real-world observations is needed. Thanks to recent widely distributed sensors (e.g. GPS), large real-world vehicle trajectory data is available for studies using data-driven approaches [9, 10, 14, 16, 25, 27].

Recent data-driven studies on traffic congestion propagation have proposed approaches to find frequent [21] or most likely [14] propagation footprints in historical dataset. Instead of detecting patterns in the past, we predict patterns in the near future. A couple of challenges are related to our work. The first is the complex behavior of traffic congestion as it propagates across a road network. A congestion that occurs on a road segment may or may not propagate to the upstream segments depending on the traffic demand along the direction of travel. Therefore, it is challenging to predict which upstream segments will be impacted by a particular congestion in the future. The second challenge is computational intensity. To be useful in a large urban network, the prediction of congestion propagation must occur quickly enough to support in real time.

We adopt a data-driven approach to solve the problem of predicting traffic congestion patterns in the near future, by using large real-world vehicle trajectory data and identifying the most probable propagations from existing congestions. We propose the concept of Propagation Graph (Pro-Graph) to model the near future footprints of traffic propagation. We predict Pro-Graphs for every existing congestion at current time, using the empirical probabilities of propagation, which are obtained from historical data. Specifically, our contributions are:

- A novel concept called Pro-Graph to model the near future propagation footprints of congested road segments.
- An algorithm PPI_Fast to build Pro-Graphs in real-time.

We evaluate our proposed solution using a real world dataset of GPS records collected from taxis in Shenzhen, China. We present two case studies that demonstrate the effectiveness of the Pro-Graph concept. In addition, we show that our design that leads to the PPI_Fast Algorithm improves running time compared to the brute-force baseline.

The rest of the paper is organized as follows. Section 2 will discuss related work. Section 3 will introduce our concept and computational problem. Section 4 will illustrate our solutions, and section 5 will show the evaluation of our algorithm performance. Section 6 will conclude our work.

2 RELATED WORK

Traffic congestion has been intensively studied. D'Andrea et al. [5] developed an expert system to detect traffic congestions on every road segments given current and past recent traffic speed. Gidofalvi [7] proposed a scalable method to detect congested traffic flow in a grid framework. Anwar et al. [4] proposed a framework using a spectral-clustering based method to monitor connected congested road sets. An et al. [3] developed a customized DBSCAN algorithm to detect and analyze clusters of grids that are repeatedly congested. While these works studied traffic congestion in different settings, traffic congestion propagation between road segments is not well addressed by them.

Related works on **propagation patterns** has focused mainly on finding frequent or the most likely propagation footprints **in the past**. Liu et al. [16] proposed the STOTree algorithm and a frequentSubtree algorithm to find frequent propagation tree of abnormal traffic flow between regions. Wang et al. [26] used this work to develop a visual interface to explore traffic congestion and propagation patterns. Nguyen et al. [21] reduced the computational time in [16] by proposing a STCTree algorithm and an Apriori-based algorithm to detect frequent congestion propagation trees. Liang et al. [14] proposed a Baysian Model to infer the most likely propagation graph among congested road segments without assuming connectivity between road segments during propagation. Li et al. [13] developed an Elastic Net optimization problem to quantify casual strength between pollution measurement sites and then detected frequent pollution propagation patterns to locate pollution source. Lee et al. [12] proposed a workflow to find congestion propagation between congested areas and used two heuristic methods to find traffic bottlenecks. Although there have been various approaches to study propagation footprints, past propagation patterns were the focus of all these projects.

The other set of research related to our work is the **prediction of near future traffic condition**. Liang et al. [15] proposed two models to predict next time-step traffic volume on a single road segment to identify its congestion given current inflow, outflow and traffic volume on it and adjacent upstream road segments. Kong et al. [11] proposed a Support Vector Regression based method to predict next time-step traffic speed and volume and then estimate the congestion state. Ma et al. [20] proposed a deep learning approach by using a Restricted Boltzmann Machine and a Recurrent Neural Network to predict traffic congestions for all road segments in next time-step. All these efforts predicted traffic congestion status in one or multiple road segments in next time-step, but the propagation of traffic congestion is not predicted. Fei et al. [6] predicted propagation speed and boundary based on shockwave model for a single congestion given an observed traffic incident. However, there is no prediction of propagation footprint in [6] and this work only focus on traffic incident.

Given all the above work, to best of our knowledge, predicting footprints of traffic congestion propagation in the near future of a road network has not been addressed. Since the outputs of our work and related work are different, we are not going to compare our algorithm with that of the related studies.

3 CONCEPTS AND DEFINITIONS

We define a spatial temporal network $G_T = G \times T$, where G is a road network $G = (V, E)$ and T is a time span composed of a set of consecutive time slots $\{t_p | t_{p+1} = t_p + 1, p = 0, \ldots, n\}$. G_T is composed of a series of snapshots of G, each of which is associated with a $t_p \in T$. G is a directed connected graph. E is composed of a set of road segments r_i, each of which has a source node $r_i.s$ and a destination node $r_i.d$. r_j connects to r_i if and only if $r_i.s = r_j.d$. V is composed of all $r_i.s$ and $r_i.d$ for all r_i in E. Each time slot $t_p \in T$ is an integer representing a time interval (e.g. 8:00 am - 8:05 am). Given G and T, the set of edges of G_T is $\{(r_i, t_p) | r_i \in E, t_p \in T\}$, and the set of vertices of G_T is $\{(v_i, t_p) | v_i \in V, t_p \in T\}$. Each edge (r_j, t_p) connects to another (r_i, t_q) if and only if $r_i.s = r_j.d$ and $t_p = t_q$.

Each (r_i, t_p) has a traffic speed spd_{r_i, t_p} and each $r_i \in E$ has a free flow speed $r_i.spd_f$. The traffic speed spd_{r_i, t_p} is the average vehicle speeds of all vehicles traveling on r_i during t_p. We obtained each vehicle speed at r_i during t_p by following the work of [5], which is the average of all instant speed observations (e.g. GPS speed 81.6km/h at 8:02:36AM) of each vehicle on r_i during t_p. The free flow speed $r_i.spd_f$ of r_i is the average speed of traffic on r_i as traffic volume approaches to zero [24]. Given the traffic speed of spd_{r_i, t_p} for (r_i, t_p) and the corresponding free flow speed $r_i.spd_f$, the speed reduction index [17, 23] can be used to estimate the congestion status of (r_i, t_p). Therefore, we define traffic congestion at (r_i, t_p) as:

Definition 3.1. **Traffic congestion:** (r_i, t_p) is congested if $1 - \frac{spd_{r_i, t_p}}{r_i.spd_f} \geq 0.5$.

We follow the convention in [17], which defines a congestion at times when the average speed falls below 50% of the free flow speed. If r_i is congested at t_p, then its upstream segments connecting to r_i may become congested at $t_p + 1$, leading congestion to propagate

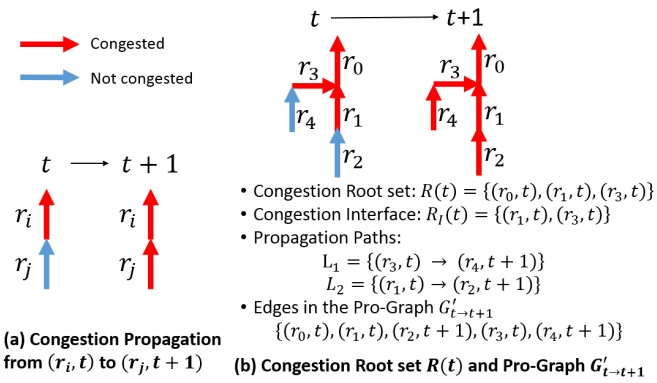


Figure 1: Example of congestion propagation, congestion root set and Pro-Graph

along the opposite direction of traffic flow. We define congestion propagation as follows:

Definition 3.2. **Congestion propagation:** Traffic congestion at (r_i, t_p) propagates to $(r_j, t_p + 1)$, if $r_i.s = r_j.d$, (r_j, t_p) is not congested, and $(r_j, t_p + 1)$ is congested . This propagation happens with a probability $p_{r_i \rightarrow r_j}$.

An example of congestion propagation is shown in Figure 1 (a). A congestion propagation happens if the connected upstream segment of a congested segment becomes congested in the next time slot. Since traffic congestion propagations can happen sequentially (i.e. starting from one road segment to upstream road segments along a path), we define the propagation path as follows:

Definition 3.3. **Propagation path** starting at (r_0, t_p) is a sequence $L = \{(r_0, t_p) \rightarrow (r_1, t_p + 1) \rightarrow \cdots \rightarrow (r_m, t_p + m)\}$, such that $\forall (r_i, t_p + i) \in L$ and $\forall (r_{i+1}, t_p + i + 1) \in L$, traffic congestion at $(r_i, t_p + i)$ propagates to $(r_{i+1}, t_p + i + 1)$. The start of the path $L.start = (r_0, t_p)$ and the time span of the path $L.ts = [t_p, t_p + m]$.

As the name suggests in Definition 3.3, the propagation path is defined to model the path of individual propagations, originating from a single road segment at a certain time slot. In fact, congestion propagation can branch into multiple paths at any stage. We treat each path from the origin to the end of each path as separate propagation paths.

Based on definition 3.2, traffic congestion can only propagate when there is already a congestion. This congestion can exist in one or multiple connected road segments, such as a connected subgraph formed by multiple congested segments. Those congested segments can trigger congestion propagations that extend in multiple directions. Thus we define the following;

Definition 3.4. **Congestion root set** at t_p are a set $R(t_p) = \{(r_i, t_p) | \ r_i \in E, (r_i, t_p)$ is congested$\}$ and they form a connected subgraph of G_T. There is a subset $R_I(t_p) \subseteq R(t_p)$ called **congestion interface**, such that $R_I(t_p) = \{(r_i, t_p) | \ (r_i, t_p) \in R(t_p), (r_i, t_p).in\_link \not\subset R(t_p)\}$, where $(r_i, t_p).in\_link = \{(r_j, t_q) | r_j.d = r_i.s, t_q = t_p\}$.

An example of congestion root set and corresponding interface is shown in Figure 1 (b). Congestion root set models the congested portion of a road network that is going to impact other portions in the future. The congestion interface of a congestion root set are the road segments through which the propagation to non-congested segments can happen. In other words, each congestion root set impacts its surrounding areas through propagation paths originating from its congestion interface. We model this impact using the following definition:

Definition 3.5. **Propagation graph (Pro-Graph)** $G'_{t_p \rightarrow t_q}$ is a subset of G_T. The edges of $G'_{t_p \rightarrow t_q}$ is $\{(r_i, t_s) | \ (r_i, t_s) \in R(t_p) \cup L_s.edges\}$, where $L_s = \{L_u | \ L_u.start \in R_I(t_p), L_u.ts = [t_p, t_e]$ *where* $t_e \in (t_p, t_q]\}$. The vertices of $G'_{t_p \rightarrow t_q}$ is all the vertices associated with the edges in it.

An example of Pro-Graph is also shown in Figure 1 (b). Each Pro-Graph $G'_{t_p \rightarrow t_q}$ from t_p to t_q shows the congestion propagation footprints in multiple directions via a set of propagation paths L_s. Each path $L_u \in L_s$ originates from the congestion interface $R_I(t_p)$ of the congestion root set $R(t_p)$ at the starting time slot t_p, ending at a certain time slot $t_e \in (t_p, t_q]$. Our goal is to predict all Pro-Graphs in the near future time slots to find out such propagation footprints. Thus we formally state the problem as follows:

- Inputs:
  - A connected road network $G = (V, E)$
  - A historical time span $T_h = [t_s, t_e]$.
  - Current time slot t_c ($t_c > t_e$).
  - Instant Speed observations at E during T_h and t_c.
  - Path propagation probability threshold γ.
- Outputs:
  - Predict all the Pro-Graphs in the near future time slots.
- Objective:
  - increase accuracy
  - Reduce computational cost
- Constraints
  - All distance are measured in network distance

4 COMPUTATIONAL SOLUTION

In this section we present our computational solution to the stated problem. We propose a method to calculate free flow speed for each segment and then compute propagation probabilities among road segments. Then we build the predicted Pro-Graphs using these probabilities. First, we propose a brute-force algorithm as a baseline. Then, we propose the efficient PPI_Fast algorithm.

4.1 Estimating the Free Flow Speeds and Congestions

According to the definition of free flow speed in [24], obtaining accurate free flow speed requires field work and this becomes difficult to achieve in a large network. We decide to estimate free flow speed of each road segment $r_i \in E$ based on its empirical vehicle speed distribution. Here we use instant vehicle speed observations (e.g. 100.1km/h at 10:02:30AM) as it fully demonstrates how vehicle speed varies along r_i during observation period.

We studied the empirical vehicle speed distribution of different road segments to determine a way of estimating free flow speed. As Figure 2 shown, although there are many records showing vehicles driving in reduced speed or being completely stopped, the majority

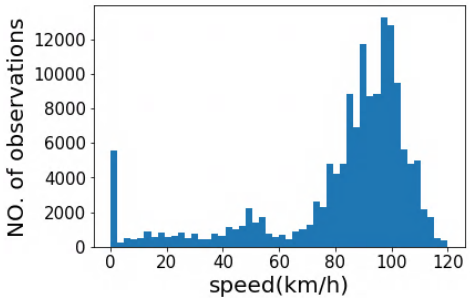


Figure 2: Vehicle instant GPS speed distribution along a segment of Expressway G4 in Shenzhen, China

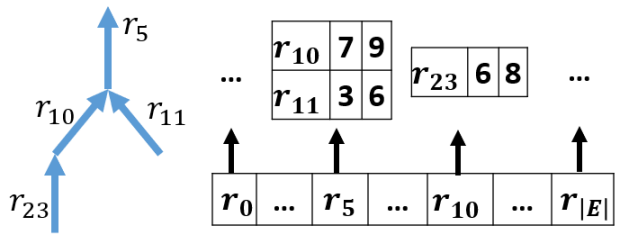


Figure 3: The proposed index structure for propagation probabilities (left side is an example network subset showing all segments connected to r_5 and r_{10}, and right side is the corresponding index structure for this)

of the records shows vehicles driving around 100km/h (62.1miles/h) on this expressway segment. It is shown that drivers prefer to drive fast. In fact, such a skewed speed distribution pattern is commonly found in many road segments. Given such speed distribution pattern and definition of free flow speed, We exclude top-k percentile observations and use the maximum of the rest observations to represent the free flow speed in a road. Excluding top-k percentile speed observation is to reduce the impact caused by special events (e.g. speeding and emergency).

For road segments which do not enough speed observations for free flow speed estimation, we set the number of unique visited vehicles as a threshold. If a road segment does not meet this threshold, we assume it will not be congested for all time slots. There are two reasons for this: it is unreliable to estimate free flow speed given very small number of speed observations, and such road segments are very unlikely to be congested due to small number of visited vehicles. Algorithm 1 shows how we compute congestion for all road segments at each time slot. We follow section 3 to compute road speed spd_{r_i,t_p} and definition 3.1 to estimate congestion. We assume no congestion on a road segment if it does not have speed observations.

Algorithm 1: Procedure get_congestions.

Input: $G = (V, E)$, Historical period T_h, Road speed spd_{r_i,t_p}, Road free flow speed $r_i.spd_f$
Output: Congestion Matrix C

```
C = Boolean Matrix of size |G.E| × |T_h| initialized to False
for t_p ∈ T_h do
    for r_i ∈ G.E do
        if (spd_{r_i,t_p} ≠ null) ∩ (r_i.spd_f ≠ null) then
            if spd_{r_i,t_p} / r_i.spd_f ≤ 0.5 then
                C[r_i][t] = True
return C
```

4.2 Calculating Propagation Probabilities

We use historical observations to calculate the propagation probabilities. To predict Pro-Graphs, two types of probabilities need to be calculated. The first is the probabilities between connected road segments, which are presented in Definition 3.2. The second is the probabilities of propagation paths presented in Definition 3.3. This type of probability measures how likely a congestion happening in r_0 at t_p will propagate to an upstream segment r_i at $t_p + i$, regardless of the connectivity between r_0 and r_i. The first type of probability is needed to calculate the second one.

Based on definition 3.2, congestion in segment r_i at t_p may propagate to its adjacent upstream segment r_j at $t_p + 1$, if r_i is congested at t_p, and r_j **becomes** congested at $t_p + 1$, i.e. r_j was not congested at t_p. The probability of this propagation is defined as $p_{r_i \to r_j}$. We calculate this using the following equation:

$$p_{r_i \to r_j} = \frac{|\{t_p \in T_h | Cng(r_i, t_p), Cng(r_j, t_p + 1), !Cng(r_j, t_p)\}|}{|\{t_p \in T_h | Cng(r_i, t_p), !Cng(r_j, t_p)\}|} \quad (1)$$

Where $Cng(r_i, t)$ means r_i is congested at t_p. Equation 1 is the conditional probability of r_j becoming congested at $t_p + 1$, given that it was not congested at t_p and r_i was congested at t_p. In other words, The propagation probability of r_i to r_j is the ratio of number of times r_i propagated to r_j to the number of times r_i could propagate to r_j.

To calculate the probabilities of propagation paths and predict Pro-Graphs in the future, we need to calculate Equation 1 for all the pairs of road segments. We propose an index structure to accomplish this called Propagation Probability Index (PPI). Figure 3 shows the proposed index structure. The first level of the index is an array of size $|E|$. Element i of the array corresponds to road segment r_i in G and points to a list of connected upstream road segments, to which r_i has propagated to in the historical data at least once. The size of this list is bounded by the in-degree of r_i (i.e. number of connected upstream road segments). Each element of the list stores the number of times r_i propagated to the corresponding road segment, as well as the number of times r_i could propagate to it. In other words, the list stores the nominator and the denominator of Equation 1. In the example of Figure 3, r_5 has propagated to r_{11} 3 times. In total r_5 had the chance to propagate to r_{11} 6 times. Thus the probability $p_{r_5 \to r_{11}} = \frac{1}{2} = 0.5$. Therefore, PPI can be built by going through all the road segments at all time slots in a given time interval and checking for propagation conditions. *Multiple time intervals* (e.g. morning and evening rush hours) can be given

to build different PPIs to address **temporal variety** of congestion propagations.

To calculate the probability of a propagation path defined in Definition 3.3), we use the Markov property of congestion propagation. In such paths the probability of propagation for segment r_j depends on its adjacent downstream segment r_{j-1} being congested, i.e. traffic congestion must have already propagated to r_{j-1}. Thus the probability of the propagation path L is the probability of all the propagations happening along L. We use the following equation to obtain the propagation path probability:

$$p(L) = \prod_{i=0}^{|L|-1} p_{r_i \to r_{i+1}} \tag{2}$$

We will use PPIs trained from historical data to calculate the propagation path probabilities via equation 2. Then we will use those propagation path probabilities to build Pro-Graphs.

Algorithm 2: Predict Pro-Graphs: a Brute-Force Algorithm.

Input: $G = (V, E)$, Historical period T_h, Current time slot t_c, Congestion Matrix C, Path propagation probability threshold γ
Output: List of future Pro-Graphs $\{M_{t_c}\}$

```
1  M_{t_c} = []; C_{t_c} = []
2  PPI = Empty array of size |G.E|
3  for r_i ∈ G.E do
4      N=get_inlink(r_i)
5      for r_j ∈ N do
6          for t_p ∈ T_h do
7              if C[r_i][t_p] and !C[r_j][t_p] then
8                  PPI[r_i][r_j].denom++
9                  if C[r_j][t_p + 1] then
10                     PPI[r_i][r_j].nom++
11 for r_i ∈ G.E do
12     if is_congested(r_i, t_c) then
13         C_{t_c}.push(r_i)
14 R_{t_c} = get_root_sets(C_{t_c})
15 M_{t_c} = get_proG(R_{t_c}, PPI, γ)
16 return M_{t_c}
```

4.3 Baseline: A Brute-Force Algorithm to Predict Future Pro-Graphs

To predict the future Pro-Graphs, first we need to build the PPI. Although multiple PPIs can be built to address temporal variety of propagation patterns, for simplicity of demonstration, we show the algorithm of building one PPI. To do this, for each road segment, we check the condition of its neighbors to which congestion can propagate, at current time slot and next time slot and update the values in the PPI. This procedure is demonstrated in Lines 3-10 of Algorithm 2. Once the PPI is built, we obtain a list of congested

Algorithm 3: Procedure get_root_sets.

Input: List of Congested Road Segments C_{t_c}
Output: List Congestion Root Sets R_{t_c}

```
1  R_{t_c} =[]
2  for r_i ∈ C_{t_c} do
3      N=get_inlink(r_i)
4      R_merge = []
5      for R_pre ∈ R_{t_c} do
6          N' = get_all_inlink(R_pre)
7          if r_i ∈ N' or N ∩ R_pre ≠ ∅ then
8              R_merge.push(R_pre)
9      R_new = {r_i}
10     for R_pre ∈ R_merge do
11         R_new = R_new ∪ R_pre
12         R_{t_c}.pop(R_pre)
13     R_{t_c}.push(R_new)
14 return R_{t_c}
```

Algorithm 4: Procedure get_proG.

Input: List of Congestion Root Sets R_{t_c}, PPI, Path propagation probability threshold γ
Output: List of future Pro-Graphs M_{t_c}

```
1  M_{t_c} = []
2  for R ∈ R_{t_c} do
3      I =.get_interface(R)
4      PG = empty Pro-Graph
5      PG.root_set = R; PG.paths = []
6      paths = []
7      for r_i ∈ I do
8          p = {r_i}
9          p.prob = 1
10         paths.push(p)
11     while paths.length > 0 do
12         next_paths = []
13         for p ∈ paths do
14             r_i = get_last_segment(p)
15             N = get_inlink(r_i)
16             for r_j ∈ N do
17                 if p.prob × PPI[r_i][r_j] ≥ γ then
18                     new_p = copy(p)
19                     new_p.push(r_j)
20                     new_p.prob = p.prob × PPI[r_i][r_j]
21                     next_paths.push(new_p)
22                 else
23                     PG.paths.push(p)
24         paths = next_paths
25     M_{t_c}.push(PG)
26 return M_{t_c}
```

road segments in current time slot, which is presented in line 11-13. The function *is_congested* works similar as line 4-6 in Algorithm 1. Then Algorithm 3 and Algorithm 4 are called in sequence to predict Pro-Graphs.

Algorithm 3 obtains all the dominate (i.e. not a subset to another) congestion root sets at current given time t_c. For each congested road segment, the algorithm obtains its upstream connected segments (line 3). Also, for each detected root set, the algorithm obtains all the connected upstream segments (line 6). If either the congested road segment or the congestion root set is found to overlap with the other's connected upstream segments, the congestion root set will be marked for merging (line 7-8). Once all such root sets are found, we merge all related congestion root sets with the segment to form a larger root set (lines 9-12).

Once all congestion root sets are obtained for the current time slot, Algorithm 4 will predict Pro-Graphs. For every congestion root set, we first initialize a Pro-Graph object and related paths which starts from the congested segment in the interface of the root set (lines 3-10). Then, for each path, we find its last segment and determine whether or not to extend the propagation path to each of this segment neighbors (lines 13-16). If this extension is allowed given threshold (γ), we create a new path for next extension (lines 17-21). Otherwise, we add this path to the Pro-Graph object for output (line 22-23). Once there is no additional path can be extended, a Pro-Graph is complete (line 25). The algorithm ends once all Pro-Graphs are built for all given root sets.

4.4 PPI_Fast: A Time-Efficient Algorithm to Predict Future Pro-Graphs

The steps of building the PPI in the brute-force algorithm (Algorithm 2) is naive and inefficient. In lines 3-10, the algorithm goes through all the time-slots in the historical period for all pairs of connected road segments. This is a very inefficient procedure and should be avoided whenever possible, especially when the input congestion state matrix derived from Algorithm 1 is very sparse (i.e. many road segment is not congested in many time slots). We believe this stage of the algorithm can be designed to be more efficient by employing smarter strategies. Our proposed algorithm PPI_Fast employs one such strategy. The idea behind PPI_Fast is that we can use the information of the congestion matrix from Algorithm 1 to speed up the process of building PPI in lines 3-10 off Algorithm 2.

To accomplish this, we use an additional matrix to keep track of the next congestion in every road segment. As Figure 4 shown, given a matrix C showing the congestion state of all road segments and all time slots, we build an additional matrix D that records the time slot of the next congestion for all segments not congested in current time slot. Assuming $r_0.s = r_1.d$, the brute-force algorithm has to goes through each the time slots from 0 to 4 to update the values in PPI for r_0 and r_1. However, given the values associated with r_0 presented in D, time slots 1 and 2 can be skipped since r_0 is not congested. Using the time slot values stored in D to jump only between congestion time slots in C effectively reduces the total number of operations. The PPI_Fast Algorithm adopts this strategy to build PPI. The effectiveness of this strategy increases as the congestion matrix C become sparser.

Algorithm 5: Predict Pro-Graphs: the PPI_Fast Algorithm.

Input: $G = (V, E)$, Historical period T_h, Current time t_c, Congestion Matrix C, Path propagation probability threshold γ

Output: List of future Pro-Graphs $\{M_{t_c}\}$

```
1  M_tc = []; C_tc = []
2  D = Integer Matrix of size |G.E| × |T_h| initialized to inf
3  PPI = Empty array of size |G.E|
4  for r_i ∈ G.E do
5      e_ind = 0
6      for t_p ∈ T_h do
7          if C[r_i][t_p] then
8              for i ∈[e_ind, t_p] do
9                  D[r_i][i] = t_p
10             e_ind=t_p + 1
11 for r_i ∈ G.E do
12     N=get_inlink(r_i)
13     for r_j ∈ N do
14         for t_p ∈ T_h do
15             if t < D[r_i][t_p] then
16                 t = D[r_i][t_p]
17             if C[r_i][t_p] and !C[r_j][t_p] then
18                 PPI[r_i][r_j].denom++
19                 if C[r_j][t_p + 1] then
20                     PPI[r_i][r_j].nom++
21 Same as lines 11-15 in Algorithm 2
22 return Mtc
```

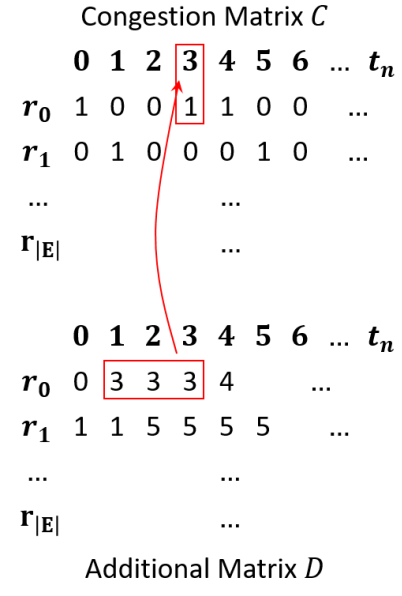


Figure 4: indexing strategy of PPI_Fast

Algorithm 5 documents the PPI_Fast Algorithm. The additional matrix D is created (line 2) and is updated each time a congestion is identified (line 4-10). For every pair of connected road segments, while iterating through each time slot, the algorithm jumps to the time slot stored at the same indexed-location of the matrix D if it is larger than the current time slot (line 11-16). This ensures the time slot to be processed in following lines is a congestion time slot. Then the PPI will be updated (line 19-22).

5 EVALUATIONS

In this section we validate our proposed concepts and methods by presenting two real case studies and rigorous experiments. First we introduce the dataset and settings.

5.1 Dataset and Settings

We use a real-world dataset of taxi GPS records. The dataset was recorded in Shenzhen, China in November 2014 using around 19 thousand taxis. Each record contains a vehicle ID, the geographical coordinates, current speed and current time. This information is recorded for every taxi in 30-second intervals. We used methods from [8] and [19] to match the geographical coordinates of each record into a road segment. The road network of Shenzhen, China is obtained from Open Street Map [22]. We use the top three level road segments: "Motorway", "Trunk" and "Primary". We exclude the lower level road segments from our analysis as many road segments in these have no observations to allow for a calculation of road speed at many time slots. This results in a road network with 6570 road segments. Figure 5 shows the road network we used in this study.

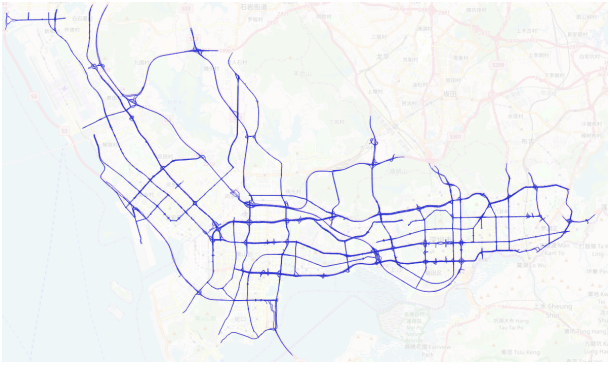


Figure 5: The studied road network.

We partition time into 5-minute time slots. Every record in the dataset is matched to a road segment and time slot. We only include time slots between 06:00 AM to midnight on each day, as we do not observe traffic congestions outside this period in the dataset. For estimation of free flow speed, top-5 percentile speed observations are excluded and then we take the maximum of the rest GPS speed observations as free flow speed. We use data from day 1 to day 23 to estimate free flow speed.

To calculate congestion propagation probabilities, we also use the data from day 1 to day 23. *We build 6 PPIs to account for different propagation patterns during different period.* Specifically, for weekdays, we build 4 PPIs: 6:00AM-10:AM, 10:00AM - 3:00PM. 3:00PM - 8PM, and 8PM - 12PM. For weekends, we build a PPI for the entire day of Saturday and of Sunday separately. The period from day 24 to day 30 is used for evaluation of our methods. We set the propagation path probability threshold to 0.01 as default.

5.2 Evaluation Mesures

We use two measures to evaluate the prediction performance of the proposed methods. The first measure is called Receiver Operating Characteristic (ROC) curve. This curve illustrates the classification ability of a binary classifier, as the discriminating threshold varies. In case of methods, the occurrence of every congestion propagation is a binary classification. The x-axis of the ROC curve is the False Positive Rate (FPR), which is calculated using the following equation:

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

Where FP is the number of falsely classified instances which are positive in truth. TN is the number of correctly classified negative instances. Value of FPR ranges between 0 and 1. Value of 0 means no mistakes in positive classifications. Value of 1 means complete failure in negative classifications. The y-axis of the ROC curve is the True Positive Rate (TPR), which is calculated using the following equation:

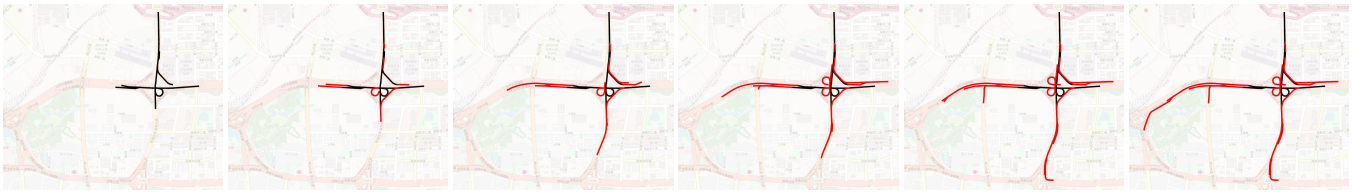
$$TPR = \frac{TP}{TP + FN} \tag{4}$$

Where TP is the number of correctly classified negative instances. FN is the number of falsely classified instances that are negative in truth. The value of TPR also ranges between 0 and 1. Value of 0 means complete failure in positive classifications, while value of 1 means no mistakes in negative classifications. The discriminating threshold controls a trade-off between FPR and TPR. That is to say, it increases one, while decreasing the other. The line $TPR = FPR$ is regarded as the ROC curve of the random guess, in case of binary classification. The more the ROC curve sits above this line, the more predictive power the classifier has. This allows for an intuitive visual examination of classification results.

Although the ROC curve was originally developed for visual examination of performance, its desirability (i.e. visualized performance) can be quantified, too. Intuitively, the higher the ROC curve is above the random-guess line, the better the classification performance is. Therefore, the Area Under the Curve (AUC) has become a common measure of classification performance. The AUC of a perfect classifier is 1. In this section, we use both measures to evaluate the performance of methods.

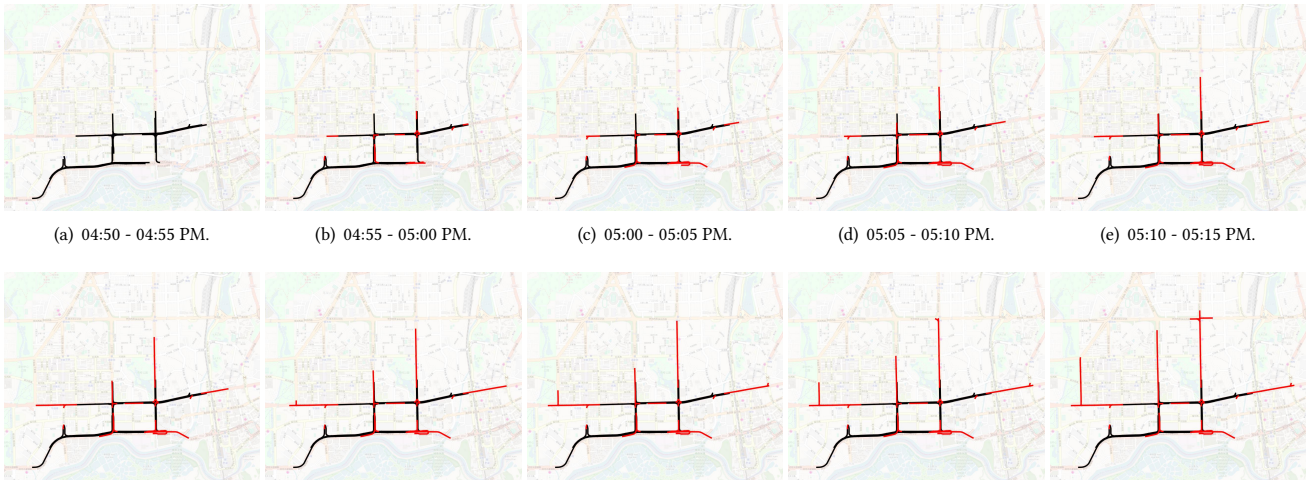
5.3 Case Studies

In this section, we present two case studies to demonstrate the effectiveness of the proposed methods. After running the algorithm for the entire testing period, the algorithm predicted a congestion propagation pattern starting from 7:40 AM on Monday the 24th at the intersection of Expressways G107 and X256. Figure 6 shows how the pattern is predicted to propagate in each near future time slot. The algorithm keeps predicting future propagations until the



(a) 07:40 - 07:45 AM. (b) 07:45 - 07:50 AM. (c) 07:50 - 07:55 AM. (d) 07:55 - 08:00 AM. (e) 08:00 - 08:05 AM. (f) 08:05 - 08:10 AM.

Figure 6: First case study of congestion propagation (best viewed in color).



(a) 04:50 - 04:55 PM. (b) 04:55 - 05:00 PM. (c) 05:00 - 05:05 PM. (d) 05:05 - 05:10 PM. (e) 05:10 - 05:15 PM.

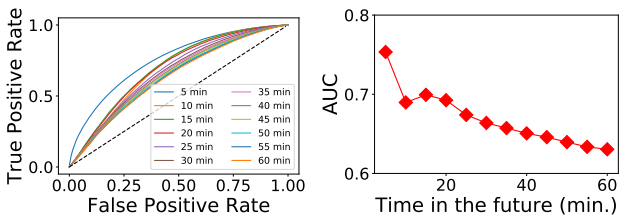
(f) 05:15 - 05:20 PM. (g) 05:20 - 05:25 PM. (h) 05:25 - 05:30 PM. (i) 05:30 - 05:35 PM. (j) 05:35 - 05:40 PM.

(k) 05:40 - 05:45 PM. (l) 05:45 - 05:50 PM. (m) 05:50 - 05:55 PM.

Figure 7: Second case study of congestion propagation (best viewed in color).

probabilities of propagation paths fall below the threshold. In this case the algorithm outputs 5 time slots in the future. The black lines are the congested root set, from which the propagation starts. The red lines are the road segments that the congestion propagates to. After this prediction, we looked into public records and found news reports [1] confirming that the congested root set and the propagation paths were congested around this period. One factor contributing to the congestion and propagation in this area is the business and industrial district surrounding it, which attracts large traffic volumes. In the north of this area, the expressway G4 that serves as a major road for traveling between Shenzhen and other cities also contributes to the congestion and propagations in this area by attracting traffic from south.

Also, the algorithm predicted large congestion propagation pattern starting from 4:50 PM on Monday the 24th, the congestion root set of which covered parts of Expressway S550 and Expressway Shen Nan Zhong Lu. Figure 7 shows how the pattern was predicted to propagate over 13 time slots. After this prediction, we looked into public records and found news reports [2] suggesting that the large congested root set and the propagation paths were congested around this location and time period. The congestion and propagations in this area are mainly because large traffic demand in both directions (going west and east) of Shen Nan Zhong Lu and in the west-to-east direction of Expressway S550. This also causes congestion to propagate to the north as there is a significant volume of traffic coming from north that turns west or east in Shen Nan Zhong Lu. The large commercial district at the northwest of this



(a) ROC curve

(b) AUC of ROC curve for the future time slots (1 hour)

Figure 8: ROC curve and AUC measure for future time slots.

area is also a factor contributing to the congestion and propagation patterns.

5.4 Experiments

We evaluate the performance of our proposed methods in terms of prediction accuracy and running time efficiency.

In the first experiment, we evaluate the prediction accuracy of the predicted propagation patterns. We use the actual propagations as the ground truth which occurs when a true congestion satisfies the congestion propagation definition (i.e. Definition 3.2). In order to evaluate our algorithm performance for longer future time period instead of a couple of time slots, we decrease the default propagation path probability to 10^{-32} in this experiment. Figure 8 shows the ROC curve and the AUC measures for predictions in future time slots. The predictions of 12 time slots (1 hour) in the future was evaluated. Figure 8 (a) shows that prediction power declines as with time. Figure 8 (b) shows the AUC for each of the 12 time slots in the future. It starts with 0.75 in the immediate next time slot and only drops to 0.63 after 1 hour.

In the next three experiments, we evaluate the running time efficiency of our proposed algorithm PPI_Fast versus the brute-force method. Among those experiments, we find most of computational time is occupied by building PPI. Predicting Pro-Graphs for a given current time slot only costs at most 2 seconds. Given the entire road network and 5 minute time slot, our proposed algorithm PPI_Fast is able to make prediction in about 2 minutes.

In the second experiment, we evaluate the running time by varying the network size (i.e. number of road segments in the road network). We vary the road network size from 690 to 6570 to measure the impact on running time. Figure 9 shows that the design decision in Section 4 improves computational time as the network size increases. The running time is approximately reduced by half. This is because our design decision reduces number of operations on each road segment. The figure shows that the proposed algorithm significantly increases the running time performance.

In the third experiment, we evaluate the running time by varying the length of the time slot, by setting it from 5 minutes to 50 minutes. Figure 10 show the running of the entire algorithm. Again, the figure shows that the design decision in Section 4 improves the running time. As the length of time slot increase, the improvement on running time decrease. This is because increased length of time slot decreases the sparsity of congestion matrix used to build PPI.

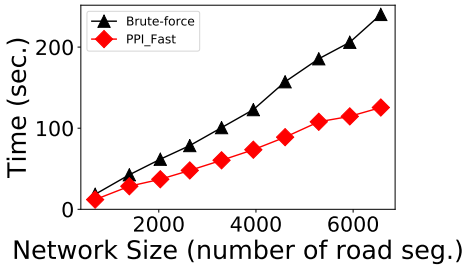


Figure 9: Running time with varying network size.

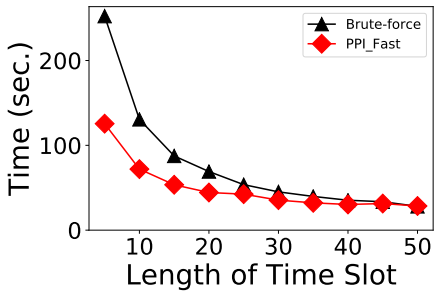


Figure 10: Running time with varying length of time slot.

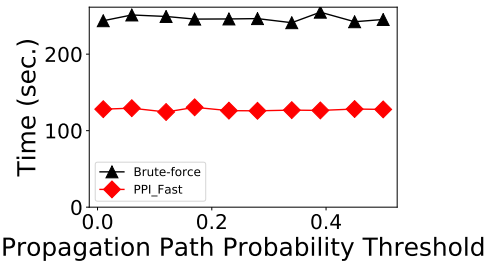


Figure 11: Running time with varying path propagation probability threshold

In the last experiment, we evaluate the running time by varying the propagation path probability threshold. We vary the probability from 0.01 to 0.5. Figure 11 also shows that the design decision in Section 4 significantly improves the running time by reducing it to half approximately, This figure also shows that algorithm performance does not vary with propagation path probability threshold. This is because the threshold only affect the computational time of prediction and the computational time of prediction is much less than that of building PPI.

Overall, our proposed algorithm PPI_Fast significantly improves computational performance, comparing to the baseline brute-force algorithm. Assuming the maximum number of congestion per road segment is c_m ($c_m \ll |E|$), and the maximum in-degree of each road segment $r_i \in E$ is d, the computational complexity of building a PPI in PPI_Fast is $O(c_m \times |E| \times |T_h| \times d)$, while the computational complexity of that in brute-force algorithm is $O(|E|^2 \times |T_h| \times d)$.

6 CONCLUSIONS

In this paper we address the problem of predicting traffic congestion propagation patterns. The predicted information can provide valuable information to citizens and city planners about how congestion will evolve in the future. Related works did not consider predicting the footprint of congestion propagations in the future. This paper formulate the footprint of a congestion propagation as Pro-Graph. An algorithm is proposed to discover all Pro-Graphs given current time and spatial-temporal field. Case studies show that the proposed algorithm have the ability to capture real-world congestions. Experimental evaluations show that the PPI_Fast algorithm has better performance than the baseline brute-force algorithm. In the future, we plan to (1) improve the algorithm to achieve better performance, and (2) use simulation to evaluate our method in different scenarios

7 ACKNOWLEDGMENTS

Xun Zhou and Amin Vahedian are supported in part by NSF grant IIS-1566386. Yanhua Li is partly supported by NSF grant CNS-1657350, CMMI-1831140, and an industrial grant from DiDiChuxing Research.

REFERENCES

[1] 2014. One Map Showing how congested Shenzhen are. http://jt.sz.bendibao.com/news/2014422/611997.htm. Accessed: 2018-06-12.
[2] 2015. How to avoid Shenzhen most congested roads? http://app.myzaker.com/news/article.php?pk=550e73797f52e92b3d0002c1. Accessed: 2018-06-12.
[3] Shi An, Haiqiang Yang, Jian Wang, Na Cui, and Jianxun Cui. 2016. Mining urban recurrent congestion evolution patterns from GPS-equipped vehicle mobility data. *Information Sciences* 373 (2016), 515–526.
[4] Tarique Anwar, Hai L Vu, Chengfei Liu, and Serge P Hoogendoorn. 2016. Temporal tracking of congested partitions in dynamic urban road networks. *Transportation Research Record: Journal of the Transportation Research Board* 2595 (2016), 88–97.
[5] Eleonora D'Andrea and Francesco Marcelloni. 2017. Detection of traffic congestion and incidents from GPS trace analysis. *Expert Systems with Applications* 73 (2017), 43–56.
[6] Wenpeng Fei, Guohua Song, Jinrui Zang, Yong Gao, Jianping Sun, and Lei Yu. 2016. Framework model for time-variant propagation speed and congestion boundary by incident on expressways. *IET Intelligent Transport Systems* 11, 1 (2016), 10–17.
[7] Győző Gidófalvi. 2015. Scalable selective traffic congestion notification. In *Proceedings of the Fourth ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*. ACM, 40–49.
[8] Joshua S Greenfeld. 2002. Matching GPS observations to locations on a digital map. In *81th annual meeting of the transportation research board*, Vol. 1. 164–173.
[9] Liang Hong, Yu Zheng, Duncan Yung, Jingbo Shang, and Lei Zou. 2015. Detecting urban black holes based on human mobility data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 35.
[10] Amin Vahedian Khezerlou, Xun Zhou, Lufan Li, Zubair Shafiq, Alex X Liu, and Fan Zhang. 2017. A traffic flow approach to early detection of gathering events: Comprehensive results. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 6 (2017), 74.
[11] Xiangjie Kong, Zhenzhen Xu, Guojiang Shen, Jinzhong Wang, Qiuyuan Yang, and Benshi Zhang. 2016. Urban traffic congestion estimation and prediction based on floating car trajectory data. *Future Generation Computer Systems* 61 (2016), 97–107.
[12] Wei-Hsun Lee, Shian-Shyong Tseng, Jin-Lih Shieh, and Hsiao-Han Chen. 2011. Discovering traffic bottlenecks in an urban network by spatiotemporal data mining on location-based services. *IEEE Transactions on Intelligent Transportation Systems* 12, 4 (2011), 1047–1056.
[13] Xiucheng Li, Yun Cheng, Gao Cong, and Lisi Chen. 2017. Discovering Pollution Sources and Propagation Patterns in Urban Area. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1863–1872.
[14] Yuxuan Liang, Zhongyuan Jiang, and Yu Zheng. 2017. Inferring Traffic Cascading Patterns. (2017).
[15] Zilu Liang and Yasushi Wakahara. 2014. Real-time urban traffic amount prediction models for dynamic route guidance systems. *EURASIP Journal on Wireless Communications and Networking* 2014, 1 (2014), 85.
[16] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. 2011. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1010–1018.
[17] Tim Lomax, Shawn Turner, Gordon Shunk, and Herbert S Levinson. 1997. *Quantifying Congestion, Volume 1: Final Report*. National Academy Press, Washington, D.C.
[18] JianCheng Long, ZiYou Gao, HuaLing Ren, and AiPing Lian. 2008. Urban traffic congestion propagation and bottleneck identification. *Science in China Series F: Information Sciences* 51, 7 (2008), 948.
[19] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 352–361.
[20] Xiaolei Ma, Haiyang Yu, Yunpeng Wang, and Yinhai Wang. 2015. Large-scale transportation network congestion evolution prediction using deep learning theory. *PloS one* 10, 3 (2015), e0119044.
[21] Hoang Nguyen, Wei Liu, and Fang Chen. 2017. Discovering congestion propagation patterns in spatio-temporal traffic data. *IEEE Transactions on Big Data* 3, 2 (2017), 169–180.
[22] OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.
[23] Amudapuram Mohan Rao and Kalaga Ramachandra Rao. 2012. Measuring Urban Traffic Congestion-A Review. *International Journal for Traffic & Transport Engineering* 2, 4 (2012).
[24] National Reseach Council (U.S.). 2010. *Highway capacity manual, 2010*. Transportation Research Board, Washington, D.C.
[25] Amin Vahedian, Xun Zhou, Ling Tong, Yanhua Li, and Jun Luo. 2017. Forecasting gathering events through continuous destination prediction on big trajectory data. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 34.
[26] Zuchao Wang, Min Lu, Xiaoru Yuan, Junping Zhang, and Huub Van De Wetering. 2013. Visual traffic jam analysis based on trajectory data. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2159–2168.
[27] Xun Zhou, Amin Vahedian Khezerlou, Alex Liu, Zubair Shafiq, and Fan Zhang. 2016. A traffic flow approach to early detection of gathering events. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 4.