Real-time Traffic Jam Detection and Congestion Reduction Using Streaming Graph Analytics

Zainab Abbas*, Paolo Sottovia†, Mohamad Al Hajj Hassan†, Daniele Foroni†, Stefano Bortoli†
*KTH Royal Institute of Technology, Stockholm, Sweden
†Huawei Munich Research Centre, Munich, Germany
*zainabab@kth.se, †{paolo.sottovia, mohamad.alhajjhassan, daniele.foroni, stefano.bortoli}@huawei.com

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9378068

***Abstract*—Traffic congestion is a problem in day to day life, especially in big cities. Various traffic control infrastructure systems have been deployed to monitor and improve the flow of traffic across cities. Real-time congestion detection can serve for many useful purposes that include sending warnings to drivers approaching the congested area and daily route planning. Most of the existing congestion detection solutions combine historical data with continuous sensor readings and rely on data collected from multiple sensors deployed on the road, measuring the speed of vehicles. While in our work we present a framework that works in a pure streaming setting where historic data is not available before processing. The traffic data streams, possibly unbounded, arrive in real-time. Moreover, the data used in our case is collected only from sensors placed on the intersections of the road. Therefore, we investigate in creating a real-time congestion detection and reduction solution, that works on traffic streams without any prior knowledge. The goal of our work is 1) to detect traffic jams in real-time, and 2) to reduce the congestion in the traffic jam areas.**

In this work, we present a real-time traffic jam detection and congestion reduction framework: 1) We propose a directed weighted graph representation of the traffic infrastructure network for capturing dependencies between sensor data to measure traffic congestion; 2) We present online traffic jam detection and congestion reduction techniques built on a modern stream processing system, i.e., Apache Flink; 3) We develop dynamic traffic light policies for controlling traffic in congested areas to reduce the travel time of vehicles. Our experimental results indicate that we are able to detect traffic jams in real-time and deploy new traffic light policies which result in 27% less travel time at the best and 8% less travel time on average compared to the travel time with default traffic light policies. Our scalability results show that our system is able to handle high-intensity streaming data with high throughput and low latency.

***Index Terms*—streaming graphs, scalable, congestion, traffic jams, real-time**

I. Introduction

With the plethora of vehicles used to commute every day, traffic congestion has become a common sight. It is important to monitor traffic flows to prevent congestion to avoid a multitude of problems. Some of these problems include: increase in fuel consumption and pollution [1], decrease in economy [2] and traffic safety that is caused by a speed variance between cars in the congested region compared to cars moving freely [3], and harmful effects on the mental and physical health of people [4], [5].

Mitigating congestion is thus an essential task of a traffic control system. Moreover, there are real-time requirements of modern traffic control systems that require a traffic monitoring and congestion control mechanism with low latency. In this work, we investigate on real-time traffic jam detection and congestion reduction over traffic streams. Real-time congestion detection can help in sending safety warnings to drivers approaching the congested region to avoid accidents, to do daily route planning, and to deploy various policies for mitigating congestion. Once congestion is detected in real-time, congestion mitigation can be done by setting up speed limits for the vehicles approaching the congested region and by controlling the traffic lights for limiting incoming traffic towards the congested region. Therefore, an online traffic stream processing based solution is necessary in order to efficiently mitigate traffic congestion by measuring the current traffic conditions.

Most of the existing congestion detection algorithms [6]–[8] use historic data and are thus suitable to work offline because in a pure streaming setup no prior knowledge about the stream is available. An online system is required to process unbounded streams, making congestion detection and mitigation challenging. One example of such offline technique is [6], which uses link journey times of vehicles to detect congestion. It requires historic information on past link journey times to detect non-recurrent congestions. Hence, it is not efficient for real-time use.

In order to detect traffic jams caused by congestion in real-time, we represent the traffic infrastructure network in the form of a directed weighted graph to capture correlations between traffic sensors based on the dependencies between their generated data streams. An example of such a dependency is that a vehicle detected by one sensor will also be detected by another sensor a few moments later in the traffic flow direction. Another example of a dependency is a traffic queue moving in the opposite direction of the traffic flow during the traffic jam that causes slow down of cars approaching the end of the queue, resulting in dependency between readings of the sensors placed in the opposite direction of the traffic flow. These dependencies are taken into account by us in measuring traffic flow variables that are used for the detection of traffic jams and tracking of traffic jams' propagation.

Zainab conducted this work partly during her internship at the Huawei Munich Research Centre, Munich, Germany.

In this work, we use traffic flow theory [9] combined with graph analytics to detect traffic jams in the streaming traffic data. Next, we develop a streaming graph-based algorithm to find correlated traffic jams in the network. We also propose a congestion mitigation/reduction mechanism by dynamically changing traffic light policies to control the traffic flow moving towards the congested region.

The main contributions of our work are as follows.

- We represent the traffic infrastructure network in the form of a directed weighted graph to capture the spatial and temporal dependencies of the traffic streaming data. We use our proposed graph representation to measure traffic flow variables that are valuable for traffic jam detections.
- We propose to offer an end-to-end traffic control framework based on Apache Flink [10]. Our system comprises of 1) an online traffic jam detection mechanism for detecting jams on streaming data collected from traffic sensors, and 2) a congestion reduction mechanism based on streaming graph analytics for reducing the effect of congestion in the congested area. In our proposed congestion reduction approach, we identify correlated traffic jams and essential parts in the road network on which new traffic light policies are deployed for congestion control.
- We develop dynamic traffic light policies based on our congestion reduction mechanism that helps in mitigating the impact of congestion by reducing the travel time of cars during traffic jams.
- We have evaluated our traffic jam detection and congestion reduction techniques using both real-life and synthetic datasets to test the performance and scalability of our system.

Main Findings: Our results indicate that application of dynamic traffic light policies in the congested regions and its neighboring regions yields less travel time of vehicles. Furthermore, our traffic jam detection and congestion reduction system gives high throughput and low latency by keeping minimal state in memory thus enabling high-speed processing of large scale streaming data. From these results, we believe that considering the real-time requirement of traffic optimization, using our proposed framework, a simple consumer machine would allow us to monitor and to help mitigate congestion in large urban areas.

Structure: The remainder of the paper is structured as follows. We present the preliminaries in Section II. Section III gives the overview of our framework and presents the traffic jam detection mechanism, Section IV describes congestion reduction mechanism, followed by Section V explaining the evaluation experiments and results. Finally, the related work is presented in Section VI and conclusion and future work in Section VII.

II. Background

In this section, we provide the necessary background by introducing the fundamental traffic flow theory used to detect congestion. We also give an overview of streaming graph-based analytics used in implementing our real-time congestion detection policies.

A. Traffic Flow Theory

Traffic flow theory is the study of vehicles' behaviour on road; it helps to explain the vehicle flow and the interaction of vehicles with each other. Mainly three traffic variables are used to explain the vehicles' movements on road, namely [9]: 1) traffic flow q (number of vehicles per unit time) that is the number of vehicles passing a particular point on road, 2) density k (number of vehicles per unit distance) that is the concentration of vehicles on road, and 3) speed v (distance covered per unit time). The three variables are related as:

$$q = k \times v \tag{1}$$

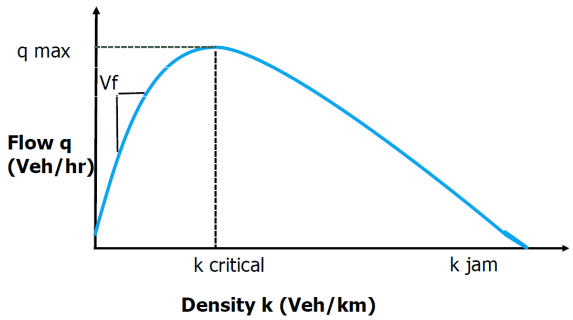


Fig. 1: The fundamental curve of road traffic flow

The fundamental traffic flow theory diagram, as shown in Fig.1, gives us a useful relation between the three traffic flow theory variables. At the start of the curve, the flow q of cars increases along with the density k; during this phase the vehicles move with free-flow speed V_f, represented by a positive slope on the curve. When q increases further the density k reaches its critical value $k_{critical}$. At this point, the flow is maximum, i.e., q_{max}. Beyond $k_{critical}$ the vehicles' movements become restricted because their concentration is increasing on the road, thus we see a decrease in the speed with a negative slope. k_{jam} indicates the traffic jam density, at this point the speed is very low due to congestion.

B. Traffic Congestion

Traffic congestion is a state of the traffic on the road in which the vehicles cannot move freely on road, their movements are restricted, i.e., vehicles cannot easily overtake each other, change lane, or move at high speed. Congestion can be caused by a poorly designed road infrastructure that is unable to meet the traffic demand. It can also be caused by external factors, such as rainy weather, accidents, and road repair work. On the contrary to congestion, "free-flow" state is the one in which vehicles can easily overtake, change lane and increase speed [11], [12].

Congestion Detection: The fundamental traffic flow curve can be used to find important measures, that include, maximum free-flow q_{max}, free-flow speed V_f and critical density $k_{critical}$

which are used to differentiate between the free-flow traffic and the congested traffic. Fig. 2 represents the empirical fundamental traffic flow diagram generated using one of our real-life datasets. It shows the classical traffic flow curve behaviour. For estimating a congestion threshold, a line (shown in red) is drawn from the empirically computed q_{max} point to the origin. All the sensor readings on the left side of the line represent free-flow traffic and all the sensor readings on the right side represent congested traffic.

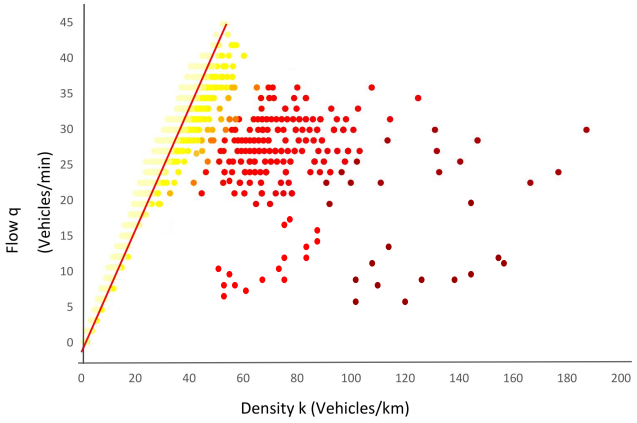


Fig. 2: Empirical fundamental traffic flow diagram

The magnitude of congestion can be determined depending on the distance from the congestion threshold line. The points farther from the line indicate traffic getting more and more congested. Different congestion levels are shown by different shades of red. Dark red color indicates high congestion. High congestion is the state in which traffic jams appear with jam density k_{jam}. Traffic density is an important measure to determine traffic jams on road. We use density in our work as an indicator of traffic jams.

C. Streaming Graph-Based Analytics

Graphs are very useful to represent relationships between entities. They are being popularly used to represent social media network data, molecular structures and complex road networks etc. With the increase in the size of graph data being generated, various distributed graph processing systems emerged to process huge graphs. Most popular of them is Apache Giraph [13]. However, these systems are not enough to address the real-time requirements of modern applications.

Stream processing is gaining importance due to its ability to process a huge volume of data in real-time. Processing graph data in real-time is challenging because it combines graph processing complexities with streaming. We built our system using Apache Flink [10] to handle a large volume of traffic sensor streams. Flink is a modern stream processing system that provides good performance guarantees in terms of high throughput and low latency. We also use Gelly-Streaming [14], an open-source library built on top of Flink for online processing of graphs.

III. Traffic Jam Detection

In this section, we present an overview of our traffic jam detection and congestion reduction system. Next, we explain the graph representation of the dataset used in our work for computing various traffic metrics over the traffic streaming data and we explain how it is used for traffic jam detection. In the end, we present the streaming graph processing based algorithm to find connected traffic jams from the traffic stream.

A. Overview of Traffic Jam Detection and Congestion Reduction System

Our traffic jam detection and congestion reduction system, shown in Fig. 3, is built using Apache Flink [10] and Kafka [15]. First, the input stream from various cameras placed on road intersections is fed to the system. A Flink job then 1) processes the camera data stream along with the road infrastructure information to compute traffic metrics, such as, average speed and density of vehicles; 2) detects traffic jams in the streams using the computed traffic metrics; 3) identifies the traffic jams that are connected; 4) detects the paths containing traffic jams in the traffic graph and their neighboring paths from which the traffic is incoming to the congested paths. These paths' information, i.e., paths with traffic jam and their neighboring paths, is written on the Kafka topics as output. This data helps in creating traffic light policies for congestion reduction in the congested region. We will explain the congestion reduction mechanism in detail in the next section.

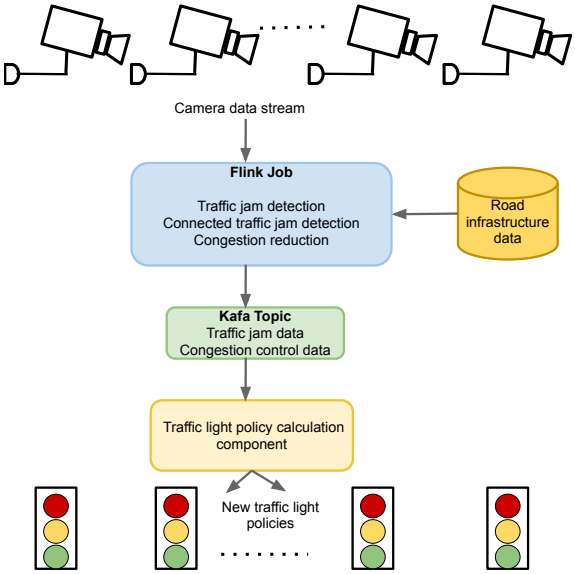


Fig. 3: Traffic jam detection and congestion control system

B. Graph Representation of Traffic DataSet

The traffic dataset used in our work is taken from a region of Shenzhen city in China containing traffic cameras deployed across various intersections of the roads in the city. Each intersection contains at least one camera in every direction,

which detect the number of vehicles passing that particular part of the intersection. The camera data source is sending traffic data stream per second to the system, making it a high-intensity stream. To detect traffic jams and to find connected traffic jams, the input data is represented in the form of a graph. A graph is useful to represent relationships between various entities in a network. In our case, the relationship exists between various camera sensors that are deployed across the same intersection in the network or are connected with a path in the traffic network graph.

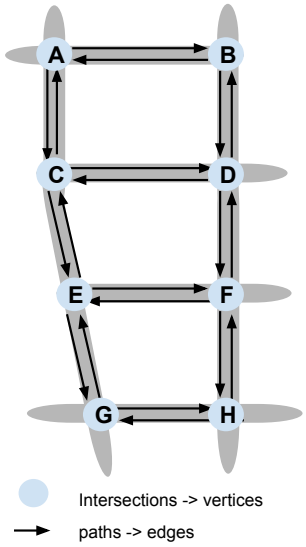
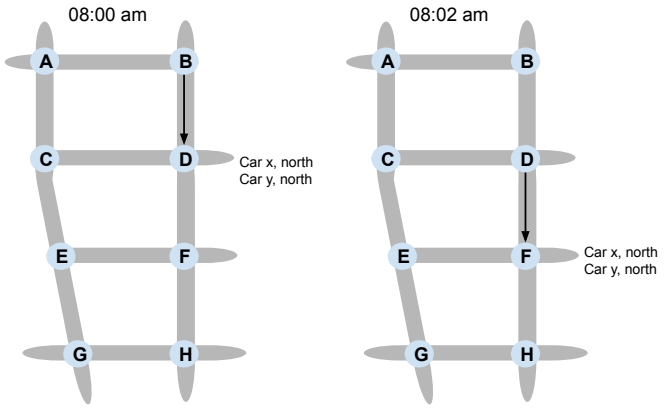


Fig. 4: Graph representation of traffic data

Fig. 4 presents the road network of one of the regions from a city in China, which contains eight intersections, namely, A, B, C, D, E, F, G and H. Each intersection has at least one camera in every direction depending upon the number of directions from which the vehicles are passing these intersections. In order to create a graph of this network, we represented the camera sensors placed over intersections as vertices of the graph and possible paths that vehicles can take between these intersections as directed edges of the graph. The traffic graph is created using the road infrastructure data information. The edges of the graph are then labelled dynamically using the camera data stream, which contains the sensor ID, the timestamp, the number of vehicles passing the sensor in the direction of the directed edge and their number plates.

C. Traffic Jam Detection

We detect traffic jams on the streams by computing the traffic density, that is the concentration of vehicles on the segment of the road. The stream we receive, per second, from cameras contains the number of vehicles that pass the sensor, i.e., the traffic flow, and their number plates. First, we use this information to compute the average speed of the vehicles that are detected by the sensors, then we use the flow and speed values to compute the density of vehicles for detecting traffic jams using equation 1.



(a) vehicles detected at 08:00 am (b) vehicles detected at 08:02 am

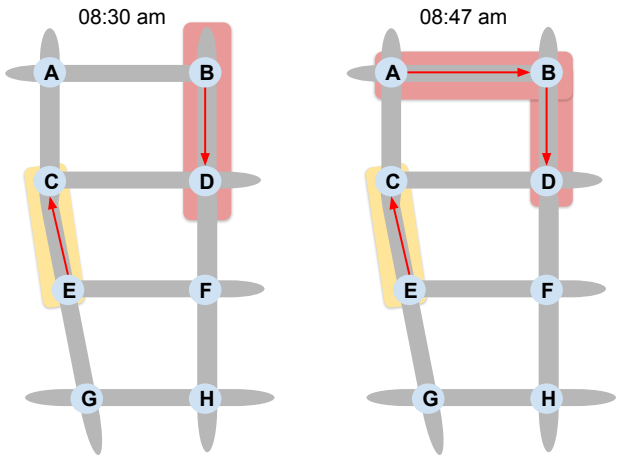
Fig. 5: Vehicles detected across various intersections on the road

We explain our approach to compute traffic density with a simple scenario given in Fig. 5, where we assume, for simplicity, that the distance between each intersection connected is 0.5 km. Fig. 5a shows that two cars, with number plates Car x and Car y, were detected at intersection D coming from the north direction. Later at 08:02 am in Fig. 5b, Car x and Car y are detected at intersection F. In order to compute the average speed of cars at the intersections we keep one-hop records of the vehicles that cross the intersections. In the given scenario, Car x and Car y made one-hop from intersection D to F coming from the north direction in 2 min (08:00 am to 08:02 am). The average speed of vehicles crossing the sensor detecting vehicles from the north on intersection F at 08:02 am is computed using the travel time of Car x and Car y coming from D to F and the distance between D to F. In this case, the average speed is $\approx$ 15 km/h. This average speed value v is then used along with the aggregated traffic flow q value, that is aggregated per minute, to estimate the traffic density at the path from D to F using equation 1. Similarly, traffic density values are computed over the stream at various intersections using the one-hop speed of vehicles at a particular time interval. We only keep one-hop trip information for speed computation of vehicles, as soon as the speed is computed we replace this information with the next-hop data to keep the state in memory minimal. Based on our empirical results, the traffic density $\geq$ 140 veh/km [16] indicates a traffic jam on the road. We use this threshold to identify the paths in the graph containing traffic jams.

D. Connected Traffic Jams

Once we detect the paths containing traffic jams in the graph, we use a streaming graph-based algorithm to track the propagation of traffic jams across the network and to find the traffic jams that are connected. Fig. 6a shows two paths in the graph, i.e, BD, and EC, labelled as congested (red edges). These paths indicate the presence of traffic jams across them at 08:30 am. Fig. 6b shows another path AB labeled as congested

at 08:47 am. This path is connected to the previous congested path BD, thus the traffic across them are part of the same traffic jam. We adapted the connected components algorithm [17] over streaming graphs to find the connected traffic jams in our graph of traffic streams and track their propagation in time.



(a) traffic jam at 08:30 am (b) traffic jam at 08:47 am

Fig. 6: Traffic jams detected across the network

Input: Incoming stream of congested edges (x, y, w)
Output: Traffic Jam IDs for each vertex x and y
begin
  foreach *edge* (x, y, w) **do**
    foreach $j \in J$ **do**
      if *Set* j *contains both* x *and* y **then**
        `/* Both vertices have been seen before */`
        return
      end
      else if x *and* y *are in different sets* **then**
        merge the two sets and set the id of the new set to minimum of both set ids
      end
      else
        `/* Only one of the vertices has been seen before */`
        if x is in a set, add y to the same set, and vice versa
      end
    end
    `/* x and y vertices have not been assigned before */`
    create new set with id `min(x.id, y.id)` and assign x and y to it
  end
  Return J
end

Algorithm 1: Pseudo-code for detecting connected traffic jams

Algorithm 1 contains the pseudo-code of the connected traffic jam algorithm for graph streams. A graph stream is created in the form of streaming edges [18], where each edge (x, y) has two-end vertices x and y. In our case, the end-vertices represent the intersection IDs and we create a weighted edge, (x, y, w), with the weight w equal to the traffic density value on the destination vertex. For example, for the path BD (in Fig. 6), assuming it has a density 150 veh/km at 08:30 am at D, an edge representing this information will be created as (B,D,150). After creating these edges, they are first filtered based on our traffic density threshold, i.e., 140 veh/km. These filtered edges are denoted as congested edges in the pseudo-code. For each incoming edge in the stream, the end vertices x and y are assigned to a set j with $j.id$, where id indicates the traffic jam id and j represents a set containing vertices that belong to the traffic jam with the assigned id. The end vertices are assigned based on the given rules: 1) If all of the sets, $j \in J$, does not contain x and y, then create a new set j with id that is minimum of $x.id$ and $y.id$. 2) If either of x and y and present in j, then add the other vertex to the same j. 3) If both x and y are present in the same j, do nothing. 4) If both x and y and present in different sets then merge the two sets and set id to the minimum value of both set ids.

In the case of the aforementioned scenario in Fig. 6, our algorithm will result in giving two groups of traffic jams, i.e., group 1 (B, D, A) containing intersection IDs for paths AB and BD, and group 2 (E, C) containing intersection IDs for the path EC. Once we detected the connected traffic jams, they are used in the congestion reduction policies that we discuss next.

IV. Congestion Reduction

In this section, we present our congestion reduction scheme. We first explain the identification of paths in the traffic network that are selected for deploying new traffic light policies and then explain the changes we make to the traffic lights along these paths in the road network.

A. Selection of Paths for Deploying New Traffic Light Policies

For reducing the effect of traffic jams, once they are detected in the traffic network, we identify the links that are essential to reduce the overall travel time of the cars in the congested area. The aim of identifying these links is 1) to control the traffic that is moving towards the congested region to avoid further congestion in that region and, 2) to reduce the travel time of cars that are in the congested region. Fig. 7 shows two traffic jams detected at 08:47 am. Path AB and BD are part of the first traffic jam and the EC is part of the second traffic jam. The traffic moving towards the first traffic jam is coming from path AC (highlighted in green) and the traffic moving towards the second traffic jam is coming from paths FE and GE (highlighted in green). In order to find these paths highlighted in green, we do one-hop backward propagation in the graph from the intersections that are part of traffic jams (highlighted in red and yellow) and get the ids of the sensors located

along these backward propagation paths (highlighted in green). Policies for traffic lights placed on these backward propagation paths along with the paths that are in the congested region are then changed to reduce the overall effect the congestion.

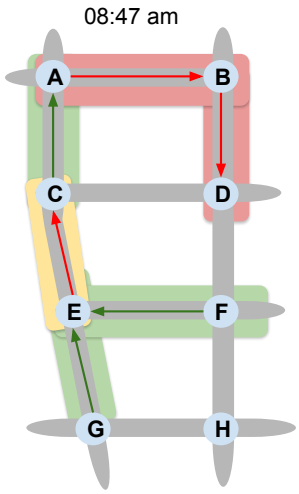


Fig. 7: Paths in traffic network containing vehicles moving towards the congested regions

B. Traffic Light Policies

The traffic light policies are built to control the red and green signal time of traffic lights. Typically the traffic light policies are fixed and do not react dynamically to the changes in traffic. In our work, we dynamically change the traffic light policies to control the green signal time of the traffic lights placed on roads under congestion, marked in red and yellow in Fig. 7, and the roads containing traffic moving towards the congested area marked in green in Fig. 7. We change the green signal time either by increasing it with bonus addition or by decreasing it with bonus removal. Adding a bonus gives more green time for the vehicles to pass the intersection. Alternately, subtracting a bonus gives less green time for the cars to pass the intersection. A bonus is added to the traffic lights green signal time in the congested region to allow more cars to pass the intersection to reduce the concentration of vehicles stuck in a traffic jam. A bonus is subtracted from the green signal time of traffic lights placed on roads containing traffic moving towards the congested region to allow fewer cars to move towards the congested regions. The bonus is computed using the given formula:

$$bonus \;=\; ((k_{node} - k_{jam}) * bonus \;\; factor)/100 \quad (2)$$

Here, $bonus\ factor$ here is 0.3 times the green time, where green time is the time during which the traffic signal is green. k_{node} is the traffic density value at the node in the graph which selected for bonus application, and k_{jam} is the density value threshold during traffic jam set to 140 veh/km based on empirical results and theoretically suggested traffic jam density threshold [16].

TABLE I: Datasets with their attributes used in experiments

Attributes	Region 1	Grid
# intersections	8	225
# sensors	28	900
# records	2,419,200	137,721,600
size of storage	1GB	50GB

V. Experimental Evaluation

In this section, first, we explain the experimental setup and datasets used in our work to evaluate the performance of our proposed traffic jam detection and congestion control framework. Then, we present the experimental results of our work. The aim of our experiments is to measure, 1) the performance of our system in terms of reducing the effect of congestion on roads, and 2) the scalability of system in terms of handling high-intensity streams from a large number of camera detectors.

Datasets: Datasets used in our experiments consist of both real-life and synthetic datasets. Information about the datasets used in our experiments is given in Table I. The first dataset, which we refer as Region 1, is a real-life dataset taken from a region of one metropolitan city, namely Shenzhen, in China. This dataset is collected from various traffic sensors deployed across intersections of the roads. The type of sensors from which data is collected is cameras. The cameras send per second information about the vehicles crossing the intersection. This per second stream consists of a timestamp, number of vehicles crossing, their number plates and the direction from which they arrive. Region 1 consists of 8 road intersections containing 28 number of traffic sensors, where each intersection has a camera in every direction. Region 1 dataset consists of a total of $\approx$ 2 million records that are collected over a period of 24 hrs. The records consist of vehicle detections sent per second by the sensors. The second dataset, which is bigger, is used to test the scalability of the system. It is a synthetically generated Grid dataset using SUMO simulator tool [19] based on the traffic behaviour of our real traffic datasets. The road network for this large dataset is a squared grid with 225 intersections. Each intersection is connected to its neighboring intersections with two lanes (one per direction) of 400 meters length. Grid consists of a total of 900 sensors generating per second streaming data. This makes more than 137 million data points in the data for a period of 12 hrs. The size of this dataset is around 50 GB.

Experimental Setup: We performed our experiments on a physical on-premises machine. Its specs are `Intel (R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz`, Linux OS and 32 GB of RAM. We used Apache Flink v1.10.0 with a local cluster consisting of one job manager and one task manager consisting of 8 parallel task slots. Furthermore, we used Kafka v2.12. The traffic jam detection and congestion control framework is written in Java 8.

Metrics: We measure the performance of our system in terms of reducing the effect of congestion on road in Section V-A. We compute the travel time of cars after applying

changes to the traffic light polices once the traffic jam is detected. We compare this travel time to the base-line travel time of cars with default traffic light policies. Next, we measure the scalability of our framework, in Section V-B, by computing the throughput, i.e, the number of records processed by the system per second with an increasing traffic stream, and the latency, i.e, the time to process one record by the system with an increasing traffic stream.

A. Congestion Reduction

In this section, we explore the effect of the bonuses that are applied to the traffic lights dynamically as a part of our congestion reduction policies after detecting congestion. We generated three different disruption scenarios in the simulation to capture the effect on travel times of cars. The scenarios were created by blocking two links in the graph of Region 1 for 10 min, 20 min and 25 min. Fig. 8 is a snapshot taken from the SUMO simulator tool that contains two cars shown in red creating disruption on the road network.

In the disruption scenarios, we measure the average travel time of cars during their trips after applying the congestion reduction policies, denoted as TTR, and compare it to the average travel time of cars during the trips without the congestion reduction policies, denoted as TTC. The normal travel time of cars on the same trips with no disruption is denoted as TTN.

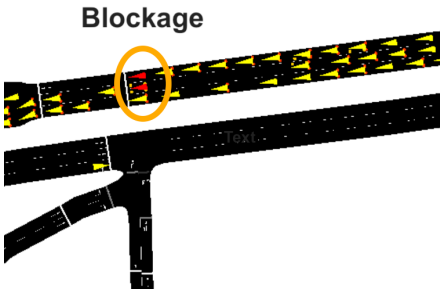


Fig. 8: Disruption created using two blocking vehicles shown in red

1) 10 min disruption: Fig.9 shows the average trip times of vehicles with (TTR) and without (TTC) congestion reduction policies after a 10 min disruption is created. TTN is the average travel time without disruption. A traffic jam is detected after 08:30 am. Overall TTR is lower compared to TTC, resulting in fewer travel times of all cars with the application of congestion reduction policies. During the traffic jam interval from 08:30 onwards till 09:10, the average travel time of all cars is reduced by $\approx$ 15%, at the best at 08:50. After the traffic jam interval, 09:10 onwards, the reduction in travel time is even more as TTR is lower compared to TTC. The average time reduced after 09:10 is $\approx$ 27%, at the best. Overall during disruption, vehicles have $\approx$ 8% less travel time after bonuses are applied to traffic lights during congestion reduction, compared to travel time with default traffic light policies.

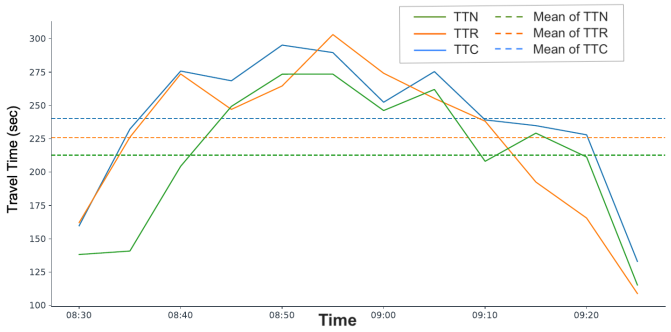


Fig. 9: Average travel time of vehicles with 10 min disruption

2) 20 min disruption: Fig.10 shows the average trip times of vehicles with (TTR) and without (TTC) applying congestion reduction policies on traffic lights after a traffic jam is detected around 08:30 am for the 20 min disruption scenario. Overall cars take less travel time with congestion reduction policies since TTR is lower compared to TTC. During the traffic jam interval, i.e., from 08:30 onwards till 09:10, the average travel time of all cars is reduced by $\approx$ 15%, at the best at 09:10. After 09:10, TTR continues to be lower than TTC. The average travel time reduced after 09:10 is $\approx$ 18%, at the best. Overall all cars take 5% less travel time after applying congestion reduction policies compared to travel time with default policies.

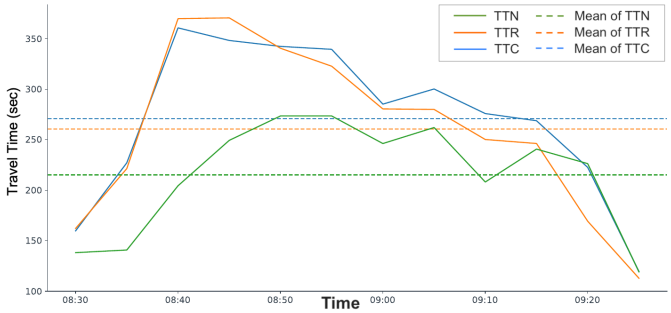


Fig. 10: Average travel time of vehicles with 20 min disruption

3) 25 min disruption: Fig.11 plots the average trip times of vehicles with (TTR) and without (TTC) applying congestion reduction policies to the traffic lights for the 25 min disruption scenario. A traffic jam is detected after 08:30. Similar to 10 min and 20 min disruption cases, overall TTR is lower compared to TTC indicating fewer travel times of all cars with the application of congestion reduction policies. On average all cars take $\approx$ 5% less travel time, and at the best $\approx$ 22% less travel time, after congestion reduction policies are applied to the traffic lights, compared to travel time with default policies.

Bonus Application: For understanding the behaviour of TTR, we plot the bonuses computed during the aforementioned disruption in Fig.9. With reference to the Fig.7, we created a disruption for the 10 min scenario on the edge EC. In order to

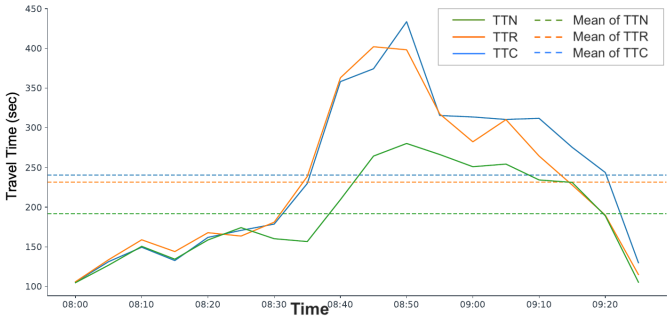


Fig. 11: Average travel time of vehicles with 25 min disruption

allow more cars to pass during the congestion, we give more green time to the traffic light placed at the intersection C. The bonuses added to the traffic light at the intersection C are given in Fig.12. Furthermore, we reduce the green signal time of the traffic lights placed at the intersection E to allow fewer cars to move towards the intersection C. Fig.13 shows the bonus time reduced from the green time of the traffic lights (two in this case) on intersection E controlling the traffic moving towards C. This bonus application helps in mitigating the effect of congestion overall by allowing fewer cars to move towards C and more cars to get out of the congestion at C.

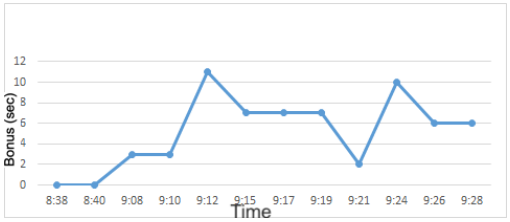


Fig. 12: Bonuses applied at intersection C

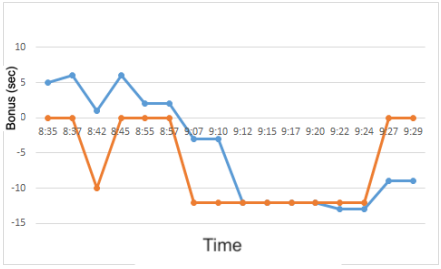
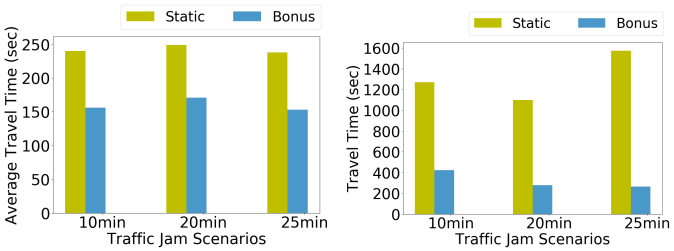


Fig. 13: Bonuses applied at intersection E

Comparison: We compare the congestion results for all three disruption scenarios by measuring the travel time of vehicles that were only in the congested region during the disruption scenario, i.e., the region where the traffic jam is detected. We take into account the vehicles for which the travel time was reduced after applying bonuses to the traffic lights. Fig. 14a plots average travel times of the aforementioned vehicles with bonuses applied to traffic lights and the average travel times of these vehicles with the default static traffic light plans. For 10 min disruption scenario, the average travel time of 4242 vehicles is $\approx$ 35% less with bonuses applied to traffic lights compared to the default static policies. Similarly, for 20 min disruption, the average travel time of 4163 vehicles is $\approx$ 31% less with bonuses. Lastly, for 25 min disruptions, the average travel time of 6663 vehicles is $\approx$ 36% less with bonuses applied to traffic lights compared to static plans.



(a) Average travel time of cars in congested region for different disruption scenarios

(b) Maximum travel time gain in the congested region for different disruption scenarios

Fig. 14: Comparison of travel time gains for 10 min, 20 min and 30 min disruption

Fig. 14b shows the travel time of cars during the three aforementioned disruption scenarios that had the most travel time gain. Travel time gain is the difference in trip time during static traffic light policies from the trip time during bonus based policies. Our results for the best single vehicle trips show that the vehicle takes almost 3$\times$ less travel time for 10 min, 4$\times$ less travel time for 20 min, and more than 6$\times$ less travel time for 25 min disruption scenario when bonuses are applied to traffic lights compared to the default traffic light policies.

B. Scalability

We now evaluate the scalability of our traffic jam detection and congestion reduction system by measuring the throughput and latency. The input dataset used for the scalability test is the Grid dataset comprising of total 137,721,600 records. In order to process the dataset in a distributed manner, we used a Taskmanager in Flink with 8 parallel task slots. It took a total of $\approx$ 40 min to process the complete dataset. We measured the throughput and latency of our system during the complete processing job of the dataset.

1) Throughput: Fig. 15 shows the throughput of our system on the y-axis in terms of records processed per second and the x-axis shows the percentage of stream processed from the total dataset. Throughput of the system slightly increases with the increase in processing of the data stream. Throughput curve starts with around 56000 records/sec when 10% data is processed and goes up till 57056 records/sec at 70% processed stream. Finally, the throughput is the highest, i.e., 57104 records/sec, when the complete stream is processed. The throughput curve shows that our system is capable of achieving high throughput during the processing of the stream.

2) Latency: Fig. 16 shows the latency of our system on the y-axis in terms of milliseconds and the x-axis shows the percentage of stream processed from the total dataset. Our plot

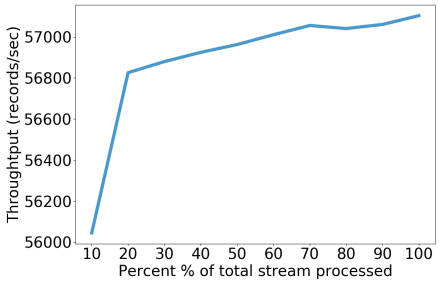


Fig. 15: Throughput of the system

depicts that latency of the system, in general, decreases with the increase in the percentage of the processed stream. Latency slightly increases at 30% processing of data, then goes low again at 60% processing of data. The throughput curve shows that our system has a very low latency during the processing of a big data stream.

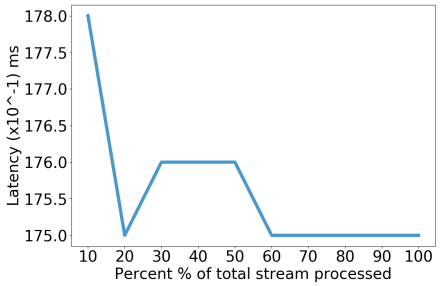


Fig. 16: Latency of the system

Findings: Our results indicate that 1) applying dynamic traffic light policies in the congested regions and its neighboring regions yields less travel time of cars. The average travel time of all cars is reduced at the best by 27% in 10 min, 18% in 20 min and 22% in 25 min disruption scenario and, 2) our traffic jam detection and congestion reduction system gives high throughput and low latency by keeping minimal state in memory yielding high-speed processing of large scale streaming data. From these results, we can deduce that considering the real-time requirement of traffic optimization, using our proposed framework, a simple consumer machine would allow us to monitor and help mitigating congestion in large urban areas.

VI. Related Work

Several interesting research has been done on traffic monitoring, more specifically for congestion detection. One of the most popular and widely used congestion detection systems is Google Maps [20]. It uses probe vehicles and data collected from cellphones with GPS to monitor traffic. However, this method is based on massive data collection and raises privacy concerns according to the General Data Protection Regulation (GDPR) laws [21].

Existing work done on congestion detection by Anbaroglu et al. [6], [7] detects congestion using link journey times of cars. Soylemezgiller et. al [8] proposes a road pricing model for reducing congestion on the road. These approaches make use of historic data that includes past link journey times and other past statistics of the road at a specific hour of the day. In a pure streaming system, this historic data is not available, making these approaches not suitable to be implemented for online processing.

Other techniques include vision-based congestion detection [22]–[25]. These techniques require computationally expensive pre-processing steps that include feature extraction, background subtraction etc., thus making them unsuitable for real-time congestion detection system with low latency requirements over large scale streaming data. Recently various studies [26]–[28] have been done on using vehicular ad hoc networks (VANETs) for local congestion information propagation in real-time. In this approach, a vehicle collects its surrounding information about speed, position etc, and sends messages to surrounding vehicles. There are several problems with VANETs based techniques that still need to be addressed. Firstly, information disseminating on urban roads is challenging due to their complex topology. Secondly, cooperation among vehicles is not very effective making congestion detection imprecise and challenging in real-time. Moreover, we use fixed-point dataset in our work, VANETs based approaches cannot work on our dataset. Besides this, several neural network-based techniques [29]–[33] are being explored to detect and predict congestion. Since they all require historic data for training, we are unable to use them in our work for building a stream processing based traffic control system.

Another activity related to congestion detection is incident detection. An incident happens due to congestion on unusual times. INGRID [34] and RAID [35], to name a few, are systems developed for incident detection. Both these systems make use of inductive loop detectors. In order to monitor traffic with induction loops, multiple of them need to be installed on the roads for accuracy of traffic metrics measured. Furthermore, they are expensive to install, which makes them not economic to use in large cities. Our congestion detection mechanism works only with a camera installed on the intersections of the road. Which are less in number and cheaper compared to induction loops.

VII. Conclusion and Future Work

In this work, we address the problem of traffic congestion detection and mitigation using real-life data collected from a region in one of the largest metropolis cities of China. We proposed an end-to-end framework built on top of a modern stream processing system, i.e., Apache Flink. Flink is used because of its high throughput and low-latency guarantees. Our framework comprises of two mechanisms: 1) an online traffic jam detection mechanism for detecting jams on streaming data collected from traffic sensors, and 2) a congestion reduction mechanism based on streaming graph analytics for reducing the effect of congestion in the congested area. In our proposed congestion reduction approach, we identify correlated traffic

jams and essential parts in the road network on which new traffic light policies are deployed for congestion control. With the proposed framework, we are not only able to detect traffic jams in real-time, but we also apply dynamic traffic light policies that yield less travel time of vehicles in the congested region. Moreover, our system is capable of processing high-intensity and large scale traffic stream with low-latency and high throughput because the memory state is kept minimum.

We believe that our work, which is based on a real-life case study, is effective in giving the desired performance and scalability in traffic congestion detection and mitigation. This work can help in monitoring and reducing congestion considering the real-time requirement of traffic optimization in a large urban area.

For future work, we consider investigating more into dynamic traffic light policy adaption. The policy adaption and removal strategies can be deeply investigated to bring more effective results.

ACKNOWLEDGMENT

We would like to thank Cristian Axenie and Alexander Wieder from the Huawei Munich Research Centre, and Stella Maropaki from NTNU, Norway, for their valuable feedback and guidance. Furthermore, Zainab is funded by the Erasmus Mundus Joint Doctorate program in Distributed Computing (EACEA of the European Commission under FPA 2012-0030).

REFERENCES

[1] M. Barth and K. Boriboonsomsin, "Real-world carbon dioxide impacts of traffic congestion," *Transportation Research Record*, vol. 2058, no. 1, pp. 163–171, 2008.

[2] P. Goodwin, "The economic costs of road traffic congestion," 2004.

[3] U. States., *Vehicle- and infrastructure-based technology for the prevention of rear-end collisions [electronic resource]*. National Transportation Safety Board Washington, D.C, 2001.

[4] D. Stokols, R. W. Novaco, J. Stokols, and J. Campbell, "Traffic congestion, Type A behavior, and stress." *Journal of Applied Psychology*, vol. 63, no. 4, pp. 467–480, 1978.

[5] J. Currie and R. Walker, "Traffic Congestion and Infant Health: Evidence from E-ZPass," *American Economic Journal: Applied Economics*, vol. 3, no. 1, pp. 65–90, Jan. 2011.

[6] B. Anbaroglu, B. Heydecker, and T. Cheng, "Spatio-temporal clustering for non-recurrent traffic congestion detection on urban road networks," *Transportation Research Part C: Emerging Technologies*, vol. 48, pp. 47–65, Nov. 2014.

[7] B. Anbaroglu, T. Cheng, and B. Heydecker, "Non-recurrent traffic congestion detection on heterogeneous urban road networks," *Transportmetrica A: Transport Science*, vol. 11, pp. 1–33, 09 2015.

[8] F. Soylemezgiller, M. Kuscu, and D. Kilinc, "A traffic congestion avoidance algorithm with dynamic road pricing for smart cities," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2013, pp. 2571–2575.

[9] M. J. Lighthill and G. B. Whitham, "On kinematic waves ii. a theory of traffic flow on long crowded roads," *Proc. R. Soc. Lond. A*, vol. 229, no. 1178, pp. 317–345, 1955.

[10] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink™: Stream and batch processing in a single engine," *IEEE Data Eng. Bull.*, vol. 38, pp. 28–38, 2015.

[11] B. S. Kerner, *The physics of traffic: empirical freeway pattern features, engineering applications, and theory*. Berlin: Springer, 2010.

[12] H. Rehborn and J. Palmer, "Asda/foto based on kerner's three-phase traffic theory in north rhine-westphalia and its integration into vehicles," in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 186–191.

[13] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan, "One trillion edges: Graph processing at facebook-scale," *Proc. VLDB Endow.*, vol. 8, no. 12, p. 1804–1815, Aug. 2015. [Online]. Available: https://doi.org/10.14778/2824032.2824077

[14] J. D. Bali, "Streaming graph analytics framework design," Master's thesis, KTH, School of Information and Communication Technology (ICT), 2015.

[15] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, 2011, pp. 1–7.

[16] V. L. Knoop and W. Daamen, "Automatic fitting procedure for the fundamental diagram," *Transportmetrica B: Transport Dynamics*, vol. 5, no. 2, pp. 129–144, 2017.

[17] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang, "Graph distances in the data-stream model," *SIAM Journal on Computing*, vol. 38, no. 5, pp. 1709–1727, 2008.

[18] Z. Abbas, V. Kalavri, P. Carbone, and V. Vlassov, "Streaming graph partitioning: an experimental study," *Proceedings of the VLDB Endowment*, vol. 11, no. 11, pp. 1590–1603, 2018.

[19] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner, "Microscopic traffic simulation using sumo," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2575–2582.

[20] "The bright side of sitting in traffic: Crowdsourcing road congestion data." [Online]. Available: https://googleblog.blogspot.com/2009/08/bright-side-of-sitting-in-traffic.html

[21] "General data protection regulation (eu)," Tech. Rep., 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN

[22] F. Porikli and Xiaokun Li, "Traffic congestion estimation using hmm models without vehicle tracking," in *IEEE Intelligent Vehicles Symposium, 2004*, 2004, pp. 188–193.

[23] G. Di Leo, A. Pietrosanto, and P. Sommella, "Metrological performance of traffic detection systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 9, pp. 3199–3206, 2009.

[24] F. Mehboob, M. Abbas, and R. Jiang, "Traffic event detection from road surveillance vide os based on fuzzy logic," in *2016 SAI Computing Conference (SAI)*, 2016, pp. 188–194.

[25] Q. Wang, J. Wan, and Y. Yuan, "Locality constraint distance metric learning for traffic congestion detection," *Pattern Recogn.*, vol. 75, no. C, p. 272–281, Mar. 2018. [Online]. Available: https://doi.org/10.1016/j.patcog.2017.03.030

[26] S. A. Vaqar and O. Basir, "Traffic pattern detection in a partially deployed vehicular ad hoc network of vehicles," *IEEE Wireless Communications*, vol. 16, 2009.

[27] R. Bauza and J. Gozálvez, "Traffic congestion detection in large-scale scenarios using vehicle-to-vehicle communications," *Journal of Network and Computer Applications*, vol. 36, no. 5, pp. 1295–1307, 2013.

[28] L. Zhang, D. Gao, W. Zhao, and H.-C. Chao, "A multilevel information fusion approach for road congestion detection in vanets," *Mathematical and Computer Modelling*, vol. 58, no. 5-6, pp. 1206–1221, 2013.

[29] X. Yu, S. Xiong, Y. He, W. E. Wong, and Y. Zhao, "Research on campus traffic congestion detection using bp neural network and markov model," *Journal of information security and applications*, vol. 31, pp. 54–60, 2016.

[30] X. Cheng, W. Lin, E. Liu, and D. Gu, "Highway traffic incident detection based on bpnn," *Procedia Engineering*, vol. 7, pp. 482–489, 2010.

[31] B. P. L. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing. ISIMP 2001 (IEEE Cat. No. 01EX489)*. IEEE, 2001, pp. 158–161.

[32] Z. Abbas, A. Al-Shishtawy, S. Girdzijauskas, and V. Vlassov, "Short-term traffic prediction using long short-term memory neural networks," in *2018 IEEE International Congress on Big Data (BigData Congress)*, 2018, pp. 57–65.

[33] Z. Abbas, J. R. Ivarsson, A. Al-Shishtawy, and V. Vlassov, "Scaling deep learning models for large spatial time-series forecasting," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1587–1594.

[34] D. Bowers, R. Bretherton, and G. Bowen, "The astrid/ingrid incident detection system for urban areas," Tech. Rep., 1995.

[35] T. Cherrett, B. Waterson, and M. McDonald, "Remote automatic incident detection using inductive loops," in *Proceedings of the Institution of Civil Engineers-Transport*, vol. 158, no. 3. Thomas Telford Ltd, 2005, pp. 149–155.